X.500 Directory Service at the University of Michigan

Tim Howes
University of Michigan
5/27/93

Overview

- The University of Michigan Environment
- Data and procedures
- Security and Privacy
- Campus access methods
- Lightweight Directory Access Protocol
- DUA/DSA Development
- WHOIS++
- North American Directory Forum

The University of Michigan

- Large research university ($2 billion/year budget)
- 17 schools and colleges
- Three main campuses (200 buildings on 2600 acres in Ann Arbor)
- U-M Hospital / medical center complex
- ~50,000 students
- ~25,000 faculty and staff

The University of Michigan Computing Environment

- 3 IBM ES/9000-class mainframes (hospital, administrative, academic)
- ~18,000 desktop workstations (Mac, PC, UNIX) relatively few "high-powered" workstations
- Fiber backbone network (mostly Cisco FDDI)
- Computer Aided Engineering Network (CAEN) provides service to the College of Engineering
- Rest of campus served by the Information Technology Division

X.500 Data at U-M

- Just under 80,000 entries. One for each student, faculty and staff-member at U-M
- Monthly bulk updates from the University's administrative databases (Personnel & Registrar)
- We get: name, address and phone data, job title and appointment information, and university ID
- Data merging is done by an off-line process called "munge". Munge, Munge II, Son of Munge, Attack of the Killer Munge, ...
- Quite complicated (~4500 lines of C code)

X.500 Data at U-M (con't)

- We also get bulk loads of other attributes from system administrators around campus (email, fax, etc.).
- Campus-wide logins are handled by a separate program called uniqname. uniqname maintains an automatic real-time link with X.500's uid attribute.
- Right now, the link between the administrative DBs and X.500 is one-way (they feed us). We are working to make this a two-way link (we feed changes back to them).

X.500 Data at U-M - Changes

- **We allow users to change most fields in their entry**
- **They cannot change:**

  - **commonName, surname: for email security reasons**
  - **universityID: this is how we identify people in munge**
  - **uniqname (uid): maintained by the uniqname server**
  - **ACL: to keep them out of trouble**
  - **various internal flags**

- **Everything else falls into one of two categories:**

  - **Things that come from the administrative source**

  - **Things that users add themselves**

X.500 Data at U-M - More Changes

- For things that come from the administrative source

  - Problem: what happens when the next bulk update happens?

  - We make users choose whether they want their entry to be overwritten each month or not. They do this by setting a flag (noBatchUpdates) in their entry.

  - User agents we distribute have support for this built in and will ask the user when they change one of the affected fields.

- For other attributes, users are free to add/delete/change them as they please.

X.500 Security at U-M

- Authentication - We have a campus-wide Kerberos authentication service, so we wanted to be able to use that with X.500
- Solution - We modified QUIPU to support Kerberos through EXTERNAL BIND credentials.
- Kerberos support handles multiple realms through the use of a kerberosName attribute stored with the entry. This attribute makes the association between an X.500 entry and a Kerberos principal.
- LDAP also modified to support Kerberos.
- Shortly after we did this, we had about 27,000 entries with kerberosNames

X.500 Access Control at U-M

- We provide two kinds of access control: individual entry ACLs and bulk (search/list) ACLs
- Entry ACLs are configured to protect certain attributes (e.g. universityID), and to give specific access to some special users/programs (e.g. uniqname, ITD accounts office).
- Search/list ACLs are used to make it harder to get bulk downloads of data from X.500
- Using these ACLs, we prevent searching by certain attributes (e.g. userPassword), and restrict the number of results a search can return, and restrict the kinds of searches that can be done.

Search ACLs

- Objective: prevent trawling of X.500
- Basic syntax:

<who> <access> <what> <scope> <limit> <partial> <minsublen>

<who> specifies to whom the access applies

<access> is either search or nosearch

<what> is a set of attributes to which access is granted or denied

<scope> is singlelevel, subtree, or baseobject

<limit> limit on number of matches returned

<partial> says whether partial results are to be returned if the limit is exceeded

<minsublen> specifies the minimum substring length that must be supplied in a query, or the minimum common prefix for a range query.

Search and List ACLs - example

- Suppose we wanted to limit the number of matches returned from a search to 10, and did not want to return any matches if this limit was exceeded. Furthermore, to prevent people from doing things like cn=a*, cn=b*, we want to impose a minimum substring length of 3. Also, don't let people search by userPassword.

```
sacl= others # nosearch # userPassword # subtree$singlelevel
sacl= others # search # default # subtree$singlelevel # 10 #  nopartialresults # 3
```

- A search hitting more than 10 matches returns 0 matches

- Searches like cn=a* are refused
- Searches like cn<=b & cn>=a are refused
- Searches like userPassword=foo are refused

X.500 Privacy at U-M

- Our administrative data sources (Personnel & Registrar) have a policy that campus addresses are public. Home addresses can be marked as "do not publish".
- This policy exists for the printed faculty/staff/student directory.
- We adopted the same policy for our X.500 data.
- In addition, if a person really wants to be removed from X.500, we will remove them.
- Problem: People do not see electronic publishing the same way they do paper publishing.

X.500 Privacy at U-M - futures

- **We need to do better than an all or nothing approach**

- **Problem: Much of our access is via anonymous services (finger, gopher, email, etc.)**

- **Possibilities:**

  - **Allow users to specify local (U-M) access only**

  - **Allow users to specify which service(s) may access their entry (e.g. email yes, finger no)**

  - **Allow users complete access to their entry's ACL**

- **Notification: we should ask permission of users, rather than forgiveness.**

X.500 Setup at U-M

- Currently we run 2 QUIPU DSAs
  - Woolly Monkey holds all of our people data
  - Mighty Joe Young holds "experimental" stuff and handles go500gw traffic
- Both DSAs run on a Sun 4/470 with 192 MB of RAM and 2 1GB IPI disks
- This machine also runs the umich.edu mail service, finger service, gopher to X.500 gateways, rcpt500, LDAP servers and DIXIE servers.

X.500 Usage at U-M

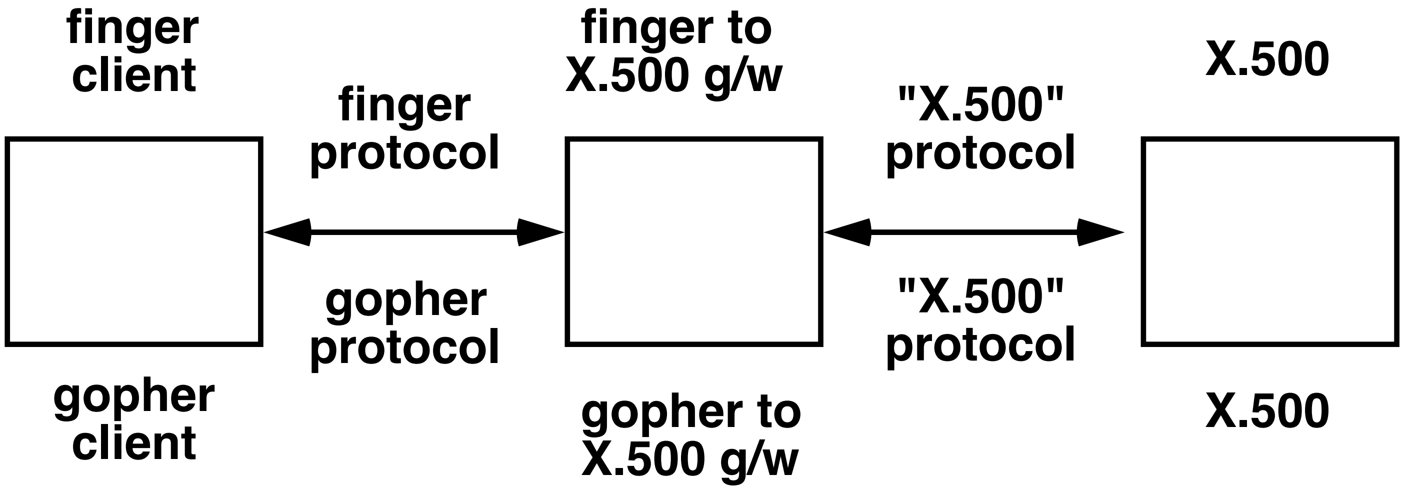
- Both DSAs combined handle about 30,000 operations per day
- Most of these are search operations because most of the access is through LDAP
- Some "housekeeping" processes we run occasionally cause over 150,000 operations per day
- Relatively few modify operations, typically < 100 per day
- The biggest users of the service are our gopher to X.500 gateways, followed by our email service, finger service, interactive DUA access, and sundry other services

X.500 Access Methods

- Finger - finger name@umich.edu
- Gopher - through 2 gopher to X.500 gateways
- rcpt500 - email query responder
- mail500 - mail service to name@umich.edu
- um-x500/ud - simple command-line DUA
- maX.500 - Macintosh DUA
- others - other DUAs, licensed, cardkey, uniqname

X.500 Access Methods - Approach

- General approach: allow users to use the clients they have on their desks (finger, gopher, ...) without them having to run anything special.



finger

- Speaks finger on one side, X.500 (LDAP) on the other.
- Returns white-pages information or a list of multiple matches

rcpt500

- E-Mail query responder
- Send a subject or body like "whois tim howes"
- rcpt500 searches X.500 and returns white-pages information or a list of matches.

Gopher to X.500 Gateway

- Two gateways:

  - go500 searches a fixed subtree
  - go500gw provides general access

- go500 is used primarily by people on our campus. It gets about 1000 queries/day

- go500gw is used by people all over. It gets about 5000 queries/day.

- Problems:

  - Need to tell leaves from non-leaves
  - Error handling
  - General search access

Gopher to X.500 Gateway Example - root

```
               Internet Gopher Information Client v1.01

                        Gopher to X.500 Gateway

         1.  About the Gopher to X.500 Gateway.
         2.  Search root <?>
         3.  Internet (organization)/
         4.  Europe (locality)/
         5.  Austria (country)/
         6.  Australia (country)/
          ...
        16. France (country)/
        17. Great Britain (country)/
          ...
--> 36. United States of America (country)/

Press ? for Help, q to Quit, u to go up a menu        Page: 1/1
```

Gopher to X.500 Gateway Example - us

```
                 Internet Gopher Information Client v1.01

                  United States of America (country)

      1.  Read United States of America entry.
  --> 2.  Search United States of America <?>
      3.  Advanced Decision Systems (organization).
      4.  ALCOA (organization)/
      5.  Anterior Technology (organization)/
      6.  Apple Computer, Inc. (organization)/
      7.  ATT (organization)/
      8.  Auburn University (organization)/
      9.  Bellcore (organization)/
      10. Bucknell University (organization)/

  ...

 Index word(s) to search for: umich
```

Gopher to X.500 Gateway Example - umich

```
                Internet Gopher Information Client v1.01

                 Search United States of America: umich

       1.  Read University of Michigan entry.
-->    2.  Search University of Michigan <?>
       3.  Experimental (organizationalUnit)/
       4.  Groups (organizationalUnit)/
       5.  Miscellaneous Servers (organizationalUnit)/
       6.  People (organizationalUnit)/
       7.  Manager (person).
       8.  Postmaster (organizationalRole).

Index word(s) to search for: babs jensen
```

Gopher to X.500 Gateway Example - Babs

```
                Internet Gopher Information Client v1.01

Search University of Michigan: babs jensen

 -->  1.  Babs Jensen (person).

Press ? for Help, q to Quit, u to go up a menu  Page: 1/1
```

Gopher to X.500 Gateway Example - Babs info

```
"Babs Jensen, Information Technology Division, Faculty and Staff, People, University of Michigan, US"
 cn
              Babs Jensen 1
              Babs Jensen
              Babs J Jensen
 sn
              Jensen
 title
              Research Systems Unix Group Manager
 postalAddress
              535 W. William
              Ann Arbor, MI 48103
 telephoneNumber
              +1 313 747-4454
 facsimileTelephoneNumber
              +1 313 764 5140
 uid
              bjensen
 drink
              water
 homePhone
              +1 313 930-2936
 homePostalAddress
              494 Waymarket Dr.
              Ann Arbor, MI 48103
 multiLineDescription
              Manager of the rsdd unix project

Press <RETURN> to continue, <m> to mail, <s> to save, or <p> to print:
```

mail500 - E-Mail and group support

- Problem:

  a) Allow users to receive mail as name@umich.edu

  b) Replace our current mainframe-based directory system that allows users to create/delete/join/resign from groups.

- First solution handled a) but not b)

  - We modified sendmail to look in X.500 the same place it does alias look-ups.

  - Handled these cases: multiple matches, matches without an e-mail address, matches with an e-mail address.

mail500 - group support

- Second solution was to write a mailer that knows about X.500 and groups. This solution solves both problems and is embodied in the fabulous mail500 program.

- Also good because it requires no changes to sendmail. Only the config file needs to be changed to call mail500.

- Group support requirements:

  - Users must be able to create/delete groups
  - Users must be able to join/resign from groups if the group owner so desires
  - Users must not be able to interfere with other users umich.edu mail by creating groups.

mail500 - more group support

- New objectClass: rfc822MailGroup
  - Must contain: cn
  - May contain: owner, rfc822Mailbox, member, rfc822ErrorsTo, errorsTo, rfc822RequestsTo, requestsTo, joinable, + description, address, etc.
- The idea was to create a hybrid object that could handle people both in and out of X.500.

mail500 - more group support - joinable groups

- Problem: How can users join and resign from groups?

  - QUIPU ACL mechanism is not general enough to do this
  - We don't want to have to force these changes to go through a special privileged program

- Solution: Define a new attribute type users can add to their entry called memberOfGroup

  - If a user wants to join the group "cn=a group, c=US" they modify their entry to include the attribute:

    memberOfGroup= cn=a group, c=US

mail500 - more group support - more joinable groups

- mail500 takes care of the rest
- When mail arrives to a group, mail500 sends to all members explicitly listed in the group as either members or rfc822Mailbox members.
- If the joinable flag is TRUE, it searches for other entries with memberOfGroup=<DN of the group> and sends to them as well.
- Problem: How does mail500 know where to search?
- Right now, this is compiled in. In future, the rfc822MailGroup mail be extended to include a searchForMembersInSubtree attribute.

mail500 stats

- Currently about 3000 invocations/day of mail500
- It's been in service for about 2 months.
- Group support has not yet been unleashed on our users. We are waiting for publicity and user agent support. ETA 1 month.
- Interesting ACL aside:

  - We wanted to allow the owner of a group to be able to change the group, but without listing all owners explicitly in the ACL
  - We extended the QUIPU ACL to have another access selector, namely some DN-valued attribute in the entry

  e.g. acl= attribute # owner # modify # entry

Miscellaneous Access

- Several other services on the U-M campus make use of the X.500 database.

- Cardkey - an automated system that provides access to campus computing sites to authorized users. It uses X.500 to verify that a person (as identified by a card-swipe machine) is affiliated with U-M.

- Licensed - a software license server for the Macintosh. It also uses X.500 to validate U-M affiliation.

- Uniqname - uses a real-time link to X.500 to keep the uid attribute up to date in users' entries.

- IFS Account creation - students use this application to automatically create an IFS account for themselves. The application uses X.500 to validate U-M affiliation.

DUA Development - ud

- Problem: We needed a lowest common denominator DUA to which we could give everybody access.
- Solution: We (Bryan Beecher) developed ud, modelled after a mainframe directory program in wide use on the U-M campus.
- Advantages: Simple, portable, lightweight, easy to add features.
- Disadvantages: Not intuitive, mainframe program it is modelled after is not the best, hamstrung by the lowest common denominator requirement.
- Still, it is widely used on our campus (but not elsewhere).

DUA Development - maX.500

- Problem: We needed a native-environment DUA for our large population of campus Macintosh users.
- Solution: We (Mark Smith) developed maX.500, a nice friendly Macintosh DUA.
- Advantages: It's highly functional, intuitive, very Mac-like, supports pictures, sound, etc.
- Disadvantages: None really, it's great!
- Widely used on our campus and elsewhere. Runs over LDAP.
- Version 2.0 about to be released.

DUA Development - futures

- We need an MS-Windows DUA. Work has started on this using LDAP which has been ported to be a DLL using the WinSock API

- We need an X-Windows DUA. Existing ones are not that great, or don't do just what we want. I like some things about fredtcl, but it's read-only.

- We also need to integrate X.500 with the various LAN directories springing up around campus.

    - AOCE
    - Novell
    - QuickMail
    - CC:Mail
    - etc.

DUA Development - AOCE Integration

- We have a project started to use LDAP to give Apple's Open Collaboration Environment X.500 access.
- AOCE defines the concept of a CSAM (Catalog Service Access Module), which is basically a back-end API to the directory service.
- At U-M, we (Gavin Eadie) have been working to write a CSAM that speaks AOCE on one side and LDAP on the other.
- This was demoed highly successfully at the recent Apple World-Wide Developers Conference
- We would like to do similar things for Novell and friends.

DSA Development

- We have been involved in a number of QUIPU development projects, mostly to help it scale up to the size of our environment:

  - Changed internal structures from lists to AVL trees
  - Introduced gdbm files instead of text EDBs to make modifies faster
  - Added in-core AVL-based indexing to speed searches
  - Implemented search/list ACLs
  - Added Kerberos support
  - Extended ACL mechanism

- Currently working on a disk-based version based on the Generic Directory API (GDA) by Steve Kille and me

DSA Development - disk based version

- Entries and indexes are all disk-based.
- The indexing scheme uses gdbm files to store substrings and soundex codes of attribute values.
- These indexes map to a DNID (a long) that identifies an entry.
- Example: sn=Pays gets stored in the files cn.sub.gdbm:

^PA	..., 1234, ...
PAY	..., 1234, ...
AYS	..., 1234, ...
YS$	..., 1234, ...

- And in cn.approx.gdbm: PS 1234

DSA Development - using the indexes

- Now if a query comes in for sn=Pays, we look up ^PA, PAY, AYS, YS$ in the index and intersect them to get the result.
- Approximate queries just require one look-up.
- This method can handle exact, approximate, and all substring queries.
- Surprisingly fast using gdbm.
- Problem: What about range queries? sn<=Pays or sn>=Pays
- An ordered index would solve this (e.g. BTREE)

DSA Development - database futures

- gdbm database is currently read-only, but pretty full-featured search-wise. Modifies coming when I return to the States!
- Separate process approach
- Next up, interfacing GDA to Oracle or some other RDBMS.
- Lots of people are probably in our same situation, with organizational data stored in a RDBMS.
- Instead of doing monthly bulk updates, why not do live queries?
- Easier said than done. Lots of issues to be worked out, but it's doable.

DSA Development - other development

- DISP replication
- Need to support incremental and subtree replication
- New scheduler to do away with the current separate database approach
- Upgrading to the various '93 extensions
- Plus various other smaller, more interesting projects:

  - X.500 to gopher gateway
  - Centroids
  - Data to program hooks
  - etc.

LDAP

- LDAP = Lightweight Directory Access Protocol
- Product of the IETF OSI-DS working group
- Two documents:
  - "Lightweight Directory Access Protocol", by Yeong, Howes, and Kille
  - "The String Representation of Standard Attribute Syntaxes", by Howes, Kille, Yeong, and Robbins
- Virtually all of our X.500 access is through LDAP
- finger, gopher, rcpt500, mail500, ud, maX.500 are all LDAP clients.

More on LDAP

- The goal was to get access to X.500 without having to run an entire OSI protocol stack

- This would allow access by Macs, PCs, and UNIX machines with only TCP/IP networking

- Precursors were the Directory Assistance Serice (DAS), by Marshall Rose, and DIXIE by Howes, Beecher, and Smith

- Our goal was accomplished by:

  - Running directly over TCP
  - Using simplified BER
  - Encoding a lot of things as just strings (e.g. DNs)

LDAP Implementation

- Last year around this time we started implementing LDAP at U-M

- Version 3.0 is about to be released. 3.0b1 is available for anonymous ftp from terminator.rs.itd.umich.edu in the file x500/beta/ldap-3.0b1.tar.Z

- It includes: server, client library, various clients

LDAP Implementation - detailed description

- ldapd - ldap server based on ISODE/QUIPU
- liblber - lightweight BER encoding/decoding library. The API provides a printf/scanf-like interface, e.g.

  ```
  ber_printf( ber, "{it{s{ss}}}", msgid, COMPARE, dn,
    attr, value );
  ```

  encodes an LDAP COMPARE request.
- libldap - provides synchronous and asynchronous API access to the LDAP protocol. For example:

  ```
  ldap_search_s( ld, "o=U of M,c=US", SUBTREE,
    "sn=Howes", returnattrs, 0, &result );
  ```

LDAP Implementation - more detailed description

- libldap also provides routines to parse results that are returned, e.g.

  - ldap_first_entry()
  - ldap_next_entry()
  - ldap_first_attribute()
  - ldap_next_attribute()
  - ldap_get_values()
  - ldap_get_dn() ... etc ...

- Basic interaction:

  - ldap_open => connects to ldap server, returns handle
  - ldap_bind => connects and authenticates to X.500
  - ldap_search, etc. => do operations and parse results
  - ldap_unbind => unbind and close connection

LDAP Implementation - ldapfilter.conf file

- Problem: A DUA developer may often want to produce different search filters based on what the user types

  - Example: "Babs Jensen" => "cn=Babs Jensen"
    "74454" => "telephoneNumber=*74454"

- The ldapfilter.conf facility provides this service in a very general, application independent way.

- The file contains mappings from regular expressions to search filters. The input (what the user types) is also tokenized. These tokens can be referred to in the search filter

LDAP Implementation - ldapfilter.conf file example

A bit of code lifted from our LDAP finger server:

```
LDAPFiltInfo *fi;
LDAPFiltDesc *fd;

fd = ldap_init_getfilter( "filter.conf.file.name" );

for ( fi = ldap_getfirstfilter( fd, "*finger*", buf );
   fi != NULL; fi = ldap_getnextfilter( fd ) ) {
       ldap_search_s( ld, base, SUBTREE, fi->lfi_filter,
          attrs, 0, &result );

       if ( (matches = ldap_count_entries( ld, result )) != 0 )
              break;
}
```

LDAP Implementation - ldapfilter.conf file example

```
"finger and go500 and go500gw subtree"
  "?*=?*"      " "        "%v"

  "[0-9]*"     " "        "(telephoneNumber=*%v)"

  "?*@?*"      " "        "(mail=%v)"

  "?[. _]?*"   ". _"      "(cn=%v1* %v2-)"

  "?*[. _]?"   ". _"      "(cn=%v*)"

  "?*[. _]?*"  ". _"      "(l(sn=%v1-)(cn=%v1-))"
                           "(l(sn~=%v1-)(cn~=%v1-))"

  "?*"          ". "      "(l(cn=%v1)(sn=%v1)(uid=%v1))"
                           "(l(cn=%v1)(sn~=%v1))"
```

LDAP Implementation - more ldapfilter.conf file

- This is a Good Thing for several reasons:
  - New filters/capabilities can be added at run time
  - Removes a lot of potentially redundant code from clients
  - Makes it easy to experiment with what works and what doesn't
  - Makes it possible to easily tailor behavior to a particular site

LDAP - A Shameless Plug

- LDAP is a Good Thing because:
  - Greatly reduces the machine/software requirements needed to access X.500
  - Greatly reduces program size and complexity for most applications
  - Has the potential to bring X.500 to "the masses"
  - Makes X.500 not significantly more complex than gopher and friends
- The best testimonial I can give is this fact: The last piece of non-LDAP code I wrote that accesses X.500 is the LDAP server.

LDAP - futures

- Would LDSP be a Good Thing too?
- It could potentially make the server side of "X.500" in the Internet much lighter.
- We'd need to make sure that compatibility with real X.500 is maintained.
- It should be possible to gateway between the two without losing any information.
- The OSI stack is not the whole problem, though. Part of it is that providing directory service is just plain complicated.
- All in all, a Good Thing in my opinion.

WHOIS++ - origins

- Grew out of several people's frustration with X.500
- They found the X.500 hierarchy too restrictive
- They found the X.500 implementations to be big and complicated and hard to administer
- They noticed that X.500 was "overkill" for white pages usage
- So, they came up with two ideas to solve these problems
- "They" is Chris Weider, Peter Deutsch, Alan Emtage, Simon Spero, Jim Fulton, ...

WHOIS++ - overview

- WHOIS++ basically defines two things
- First, there is the WHOIS++ query syntax and data model. It's a lot like regular WHOIS, but it allows the idea of "templates".
- A template is a lot like an X.500 object class. Data in a template are attribute/value pairs.
- There is a "person" template, an "information resource" template, etc. Individual servers can also specify their own templates. A "help" template is required.
- When a user queries a WHOIS++ server, she can restrict her query to objects of a particular template. She can also ask the server what templates it holds.

WHOIS++ - query syntax

- WHOIS++ users can search for records, possibly of a specified template type, that contain attribute=value pairs.
- Different output formats can be selected (full, abridged, handle, summary)
- As I write this, some aspects of the WHOIS++ protocol remain unspecified (non-text data, management, etc.)
- Query syntax is meant to be a superset of the old WHOIS query syntax

WHOIS++ - centroids

- Second, WHOIS++ defines the idea of "centroids"
- Problem: What if I want to find "Paul-Andre Pays", but I don't know where he is (i.e. what organization he works for)?
- In X.500, a simple-minded approach would require asking every DSA in the world (or at least in France). WHOIS has the same problem.
- Centroids provide a way to ask only those servers that might have data about Paul-Andre.
- This can cut the search time dramatically, and consume far fewer resources in answering the query.

WHOIS++ - centroids example

- Suppose there are 4 WHOIS++ servers, containing the following data in the name field of "person" templates:

Server A: Paul-Adre Pays, Jean-Paul Le Guigner
Server B: Paul-Andre Smith, Marie Pays
Server C: Jean-Paul Le Guigner, Marie Tetard
Server D: Marie Tetard

- The person template centroids from each server are:

Server A:
  - First Name: Paul-Andre, Jean-Paul
  - Last Name: Pays, Le Guigner

Server B:
  - First Name: Paul-Andre, Marie
  - Last Name: Pays, Smith

etc.

WHOIS++ - centroids example

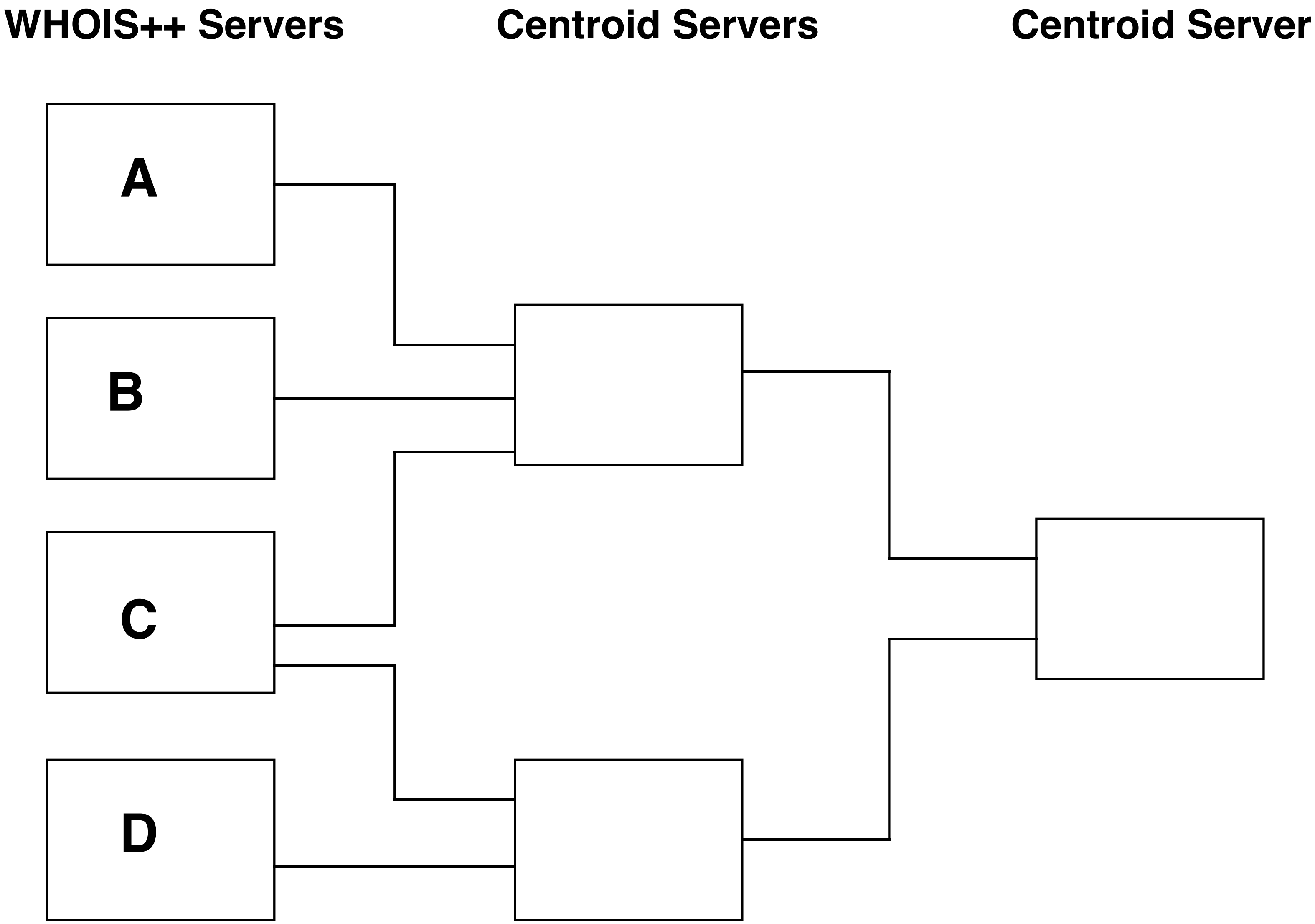
- **Now, when searching "the world" for Paul-Andre Pays, consulting the centroid tells us that we only need to bother contacting Server A and Server B, since they are the only servers holding a First Name of Paul-Andre and a Last Name of Pays.**

- **Note that in this case the centroid eliminates 2 of 4 servers immediately. It also sends us to one server on a wild goose chase, which we discover when we search that server.**

- **From an implementation perspective, a centroid provides a mapping from an attribute value to a set of servers containing that value.**

- **Centroids are a Good Thing - But they are not specific to WHOIS++. Stay tuned...**

WHOIS++ - centroids example (con't)



Centroids in X.500

- Question: Is there any reason the centroid idea cannot be applied to X.500? Answer: No.
- The X.500 referral/chaining mechanism can be used to cut down the number of DSAs to search, provided a DSA knows about centroids.
- Assume DSA X has the following information:

commonName: Paul-Andre Pays => DSA A
Jean-Paul Le Guigner => DSA A, C
Marie Tetard => DSA C, D
etc.

+ information on which subtree the centroid is from in each DSA

Centroids in X.500 - example

- If a DSA holds this information, it could provide an entry called, for example, { o=Internet, ou=Internet People }
- When a search below this object for "cn=Paul-Andre Pays" comes in, the DSA consults its centroid.
- It finds that DSA A is the only one holding data on a Paul-Andre Pays.
- The DSA can then either chain the request to DSA A, or return a referral to DSA A.
- Same effect as centroids in WHOIS++

Centroids in X.500 - maintenance

- **Problem: How do we maintain centroids in X.500?**
- **Through the WHOIS++ centroid protocol?**
- **Is there a way to do it using '93 replication?**
- **Through some other method?**
- **Offline? Online?**
- **Sometime soon I'll be trying this out...**

North American Directory Forum

- A group of public service providers planning to provide directory service (e.g. AT&T, MCI, IBM, ...)

- Working together to figure out how they can cooperate to provide the "one directory" view of X.500, while still competing while providing service.

- Three major contributions:

  - Naming scheme (registration vs. listing approach)

  - Naming Links (distributed entry solution)

  - Maintenance of public namespace through Central Administration for NADF (CAN)

NADF - Naming Scheme

- Uses the existing civil naming infrastructure for orgs, businesses, localities, states, provinces, etc.

- Separates the idea of registration from listing in the directory.

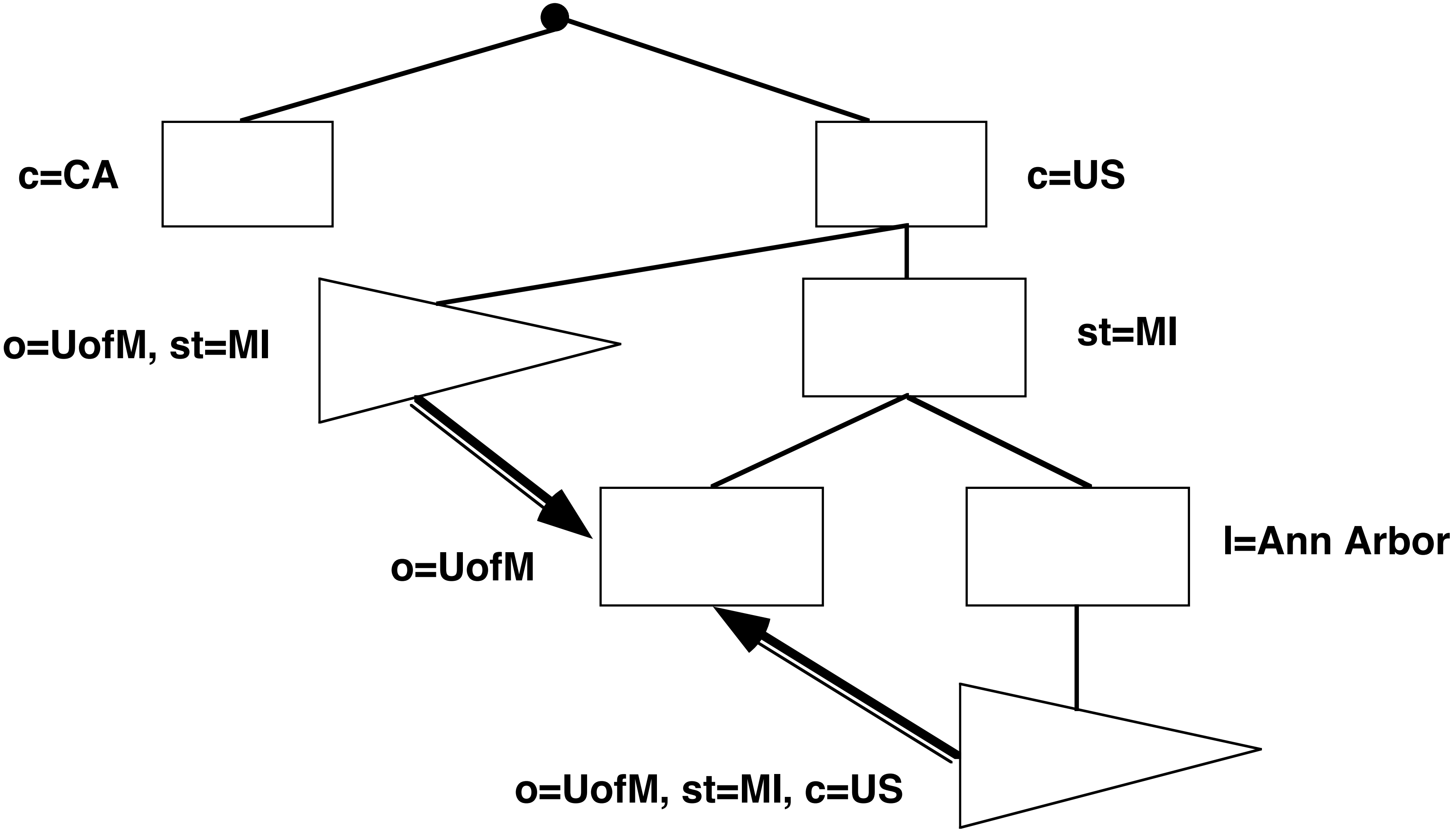
- For example, the University of Michigan is registered as doing business in the state of Michigan. Therefore, it derives its registration in the directory at this point:

  { c=US, st=Michigan, o=University of Michigan }

- Given that the civil registry prohibits any other org from having the above three RDNs, U of M can list there, or:

  { c=US, (o=University of Michigan, st=Michigan) }

NADF - Naming Scheme - example tree



c=CA

c=US

o=UofM, st=MI

st=MI

o=UofM

l=Ann Arbor

o=UofM, st=MI, c=US

NADF - Naming Links

- Problem: How can different competing service providers provide information about the same X.500 entry?

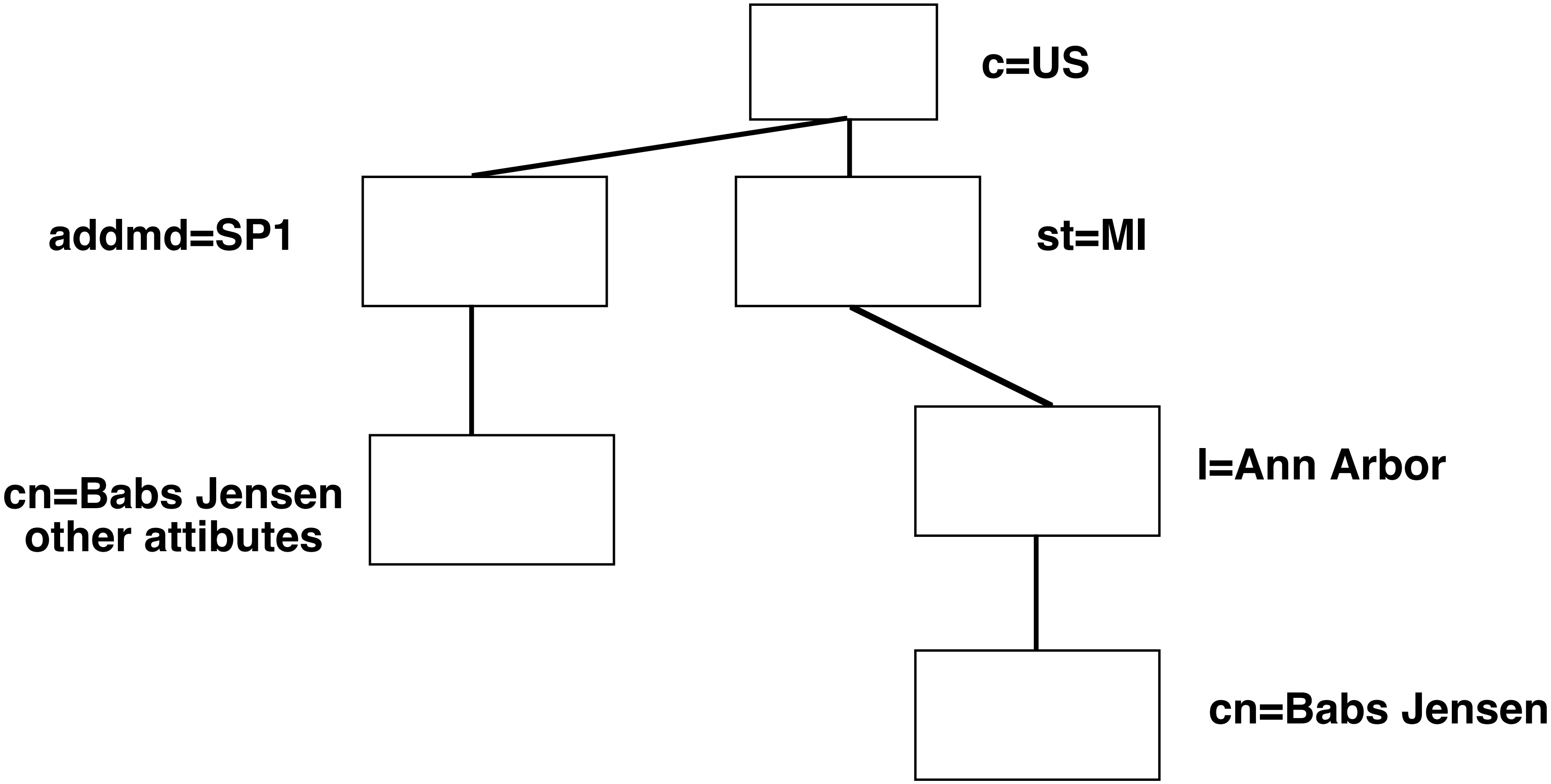
- Solution: namingLinks - The idea is that public entries contain only three kinds of information:

  - Naming information
  - Information useful in searching
  - Naming Links

- The "real" information is stored in private service provider space, accessed by following the namingLink attribute(s)

- namingLink may by multi-valued, thus allowing multiple service providers to hold data.

NADF - Naming Link - example tree



c=US

addmd=SP1

st=MI

cn=Babs Jensen
other attibutes

l=Ann Arbor

cn=Babs Jensen

namingLink= {c=US, ad=SP1, cn=Babs Jensen}
namingLink= {c=US, ad=SP2, cn=Babs Jensen}
...

NADF - More Naming Links

- Do Naming Links work? Yes.
- Performance might be a problem. Replication can solve this problem.
- DUAs have to know about namingLinks to work very well.
- Regular knowledge references are also supported (e.g. for organizations getting directory service from one provider or for PRDMDs).
- No way to tell which attributes are held by which ADDMD, so all namingLinks must be followed.

NADF - CAN

- Problem: All ADDMDs might want to add entries to the public namespace. In the absence of replication, how is this accomplished?
- Solution: A separate "neutral" authority called the CAN collects changes from ADDMDs and distributes them to other ADDMDs.
- In the current pilot, this is an offline process accomplished by some special tools (xkan by MTR) and FTP.
- Eventually, NADF may move to an online solution based on X.500 DISP.

NADF - Status

- Several documents released describing the naming scheme, naming links, CAN operation, etc.
- The NADF is currently engaged in a pilot to test out their ideas and concepts.
- Some pilot participants will be demoing at EMA in June.
- Next steps: more piloting, schema agreements, how to handle charging.
- Also, PRDMDs are now being allowed to join the NADF as non-voting members.

DUA Guidelines

- Problem: DUA developers, if left to their own devices, will produce behavior/queries that are very hard for DSAs to handle.

- Solution: Develop a guide for DUA implementors that tells them what's good and what's bad to do.

- Example 1: There is just no way to do a good job of handling a one-level or subtree search high in the tree that has the dontUseCopy flag set to TRUE.

- Example 2: Setting the searchAliases flag to TRUE on a one-level or subtree search can often cause slow behavior in the absence of replication.

- At U-M we refuse lots of queries that our DSA can't handle well