Names in Distributed Systems

*** Unique Identifiers (UIDs)** e.g. 128 bits

are never reused

refer to either: nothing, or: the same thing at all times

UIDs (may) achieve location-independence: the named object can be moved

*** pure and impure names (Needham)**

*** pure names**

- the name itself yields no information e.g. UID
- contains no location information
- commits system to nothing
- can only be used to compare with other similar bit-patterns e.g. in table look-up

*** impure names**

examples: email addresses
foo.cl.cam.ac.uk
host-ID, object-ID (unless used ONLY to *generate* UIDs)
disc-pack-ID, object-ID

- the name yields information
- commits the system to maintaining the context in which the name is to be resolved
  e.g. the directory hierarchy uk/ac/cam/cl

Unique names

uniqueness is achievable by using

- a hierarchical name: scope of uniqueness is level in hierarchy

- a bit pattern: flat, system-wide uniqueness

Issues

* a problem with pure names:

  - where to look them up?

  - how do you know that an object does not exist?
    how may a global search be avoided?

  - how to engineer uniqueness?

* a problem with impure names

  - how to restructure the namespace
    e.g. when objects move about
    when companies restructure

examples of names - note requirement for unique identification

Health Service ID - every citizen at birth (but hospitals still use local names)

UK National Insurance - on employment

US Social Security - on employment

Passport
Driving licence

Professional Societies: ACM, IEEE, BCS
Charities: National Trust, RSPB, ...
Services: RAC, AA, AAA (US)

Credit cards
Bank accounts
Utilities: gas/electricity/water/phone customer numbers

Loyalty schemes: Airlines - frequent flyer, hotels, shops

Database key - must be unique. e.g. "David Evans" could be a poor choice

---

look for structure, explicit or implicit

is allocation centralised or distributed?

what is the resolution context?

Telephone company analogy (wired service)

* geographically partitioned, distributed naming database

  electronic is current, paper is an official cache

* given a name, (Yudel Luke) or (Yudel Luke, 3 Acacia Drive) which directory to use?

  don't know where to lookup pure names

* call (#) -> unobtainable, # came from official cache

  we detect out-of-date values, call directory enquiries
  cache until new directory comes

* caching numbers in a personal address book (an unofficial cache)

  call (#) -> unobtainable, report fault if use often
  call directory enquiries, or ask another contact (may have moved, or changed provider)

* frequency of update (some years ago)

  e.g. Cambridge area: 1,000,000 entries
  5,000 updates per week

* can't find an entry

  e.g. Phillips company - check spelling: Philips

  e.g. look under S.S. rather than Social Services

BT offer a web service *www.thephonebook.com* (name and address -> #)
- only offers exact search e.g.Phillips, Cambridge - doesn't suggest Philips
need higher-level tools - "***do you mean Philips?***" increasingly use search engines to augment directories

in general - provide clients with values of attributes of named objects

name space

the collection of valid names recognised by a name service

a precise specification is required, giving the structure of names

e.g. ISBN-10: 1-234567-89-1 namespace identifier: namespace-specific string
/a/b/c/d filing system, variable length, hierarchical
puccini.cl.cam.ac.uk DNS machine name, see later for DNS
e.g. Mach OS 128-bit port name (system-wide UID)

naming domain

a name space for which there exists a single overall administrative authority
for assigning names within it
this authority may delegate name assignment for nested sub-domains (see below for Internet DNS)

name resolution or binding

obtaining a value which allows an object to be used

LATE BINDING is considered GOOD PRACTICE

programs should contain names, not addresses e.g. a machine may fail and a service restarted on another.
Your local agent may cache resolved names for subsequent use + may expire values based on timestamp.
Cached values aren't embedded in programs and are always used at one's own risk.
If they don't work you have (your agent has) to look the name up again.

for a large-scale system, name resolution is an iterative process which requires navigation among name servers (N-8)

Names, attributes and values stored by a name service

	object type	attribute list
example:	user	login name, mailbox host(s)
	computer	architecture, OS, network-address, owner
	service	network-address, version#, protocol
	group	list of names of members
	alias	canonical name
	directory?	list of hosts holding the directory

directories may be held as a separate structure
rather than as just a type of name as here

* directories are likely to be **replicated** for scalability, fault-tolerance, efficiency

* directory names will resolve to a list of hosts, as above, with their addresses to avoid further lookup

* attribute-based (inverse) lookup may be offered -
a YELLOW PAGES style of service for object discovery e.g. X.500, LDAP

TOO MUCH information might be dangerous - could reveal structure

A useful structure for names

query:
object type, object-name, attribute-name -> attribute-value

machine, foo.cl.cam.ac.uk, location -> IP address
user, some-user-name, public-key -> PK-bit-pattern
etc ....

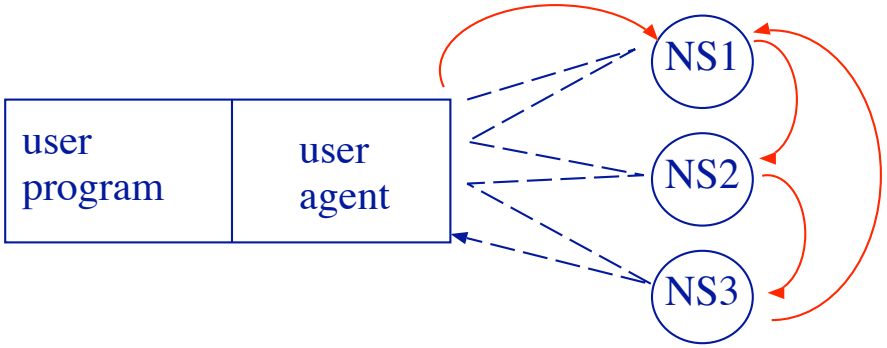
checking:
object type, object-name, attribute-name, attribute-value -> yes/no

ACL, filename, some-user-name, write-access -> yes/no
(is the user on the ACL with the permissions requested? e.g. Xerox PARC Grapevine)

attribute-based lookup may be offered (yellow-pages style):
object-type, attribute-value -> list of object names having that attribute value

machine, OS version# -> list of machines

Architecture of name resolution



user agent starts off with the root address of the name service
or some well-known sub-tree root:
e.g. the location of the uk directory for agents in the UK

to resolve cl.cam.ac.uk, the UA can start from the uk directory
then ac then cam then cl

the UA (and directories) will cache resolved names as hints for future use

---

alternative: any name server will take a name,
resolve it and return the required attribute
client can sometimes choose e.g. select "recursive" in DNS

in practice, there are engineering optimisations: local caches are tried first,
directories may be able to indicate whether they offer recursive evaluation, etc.

Examples of name services

DNS - see below N-10 to N-13

Grapevine - see below N-17

Xerox PARC early 80's
- two-level naming hierarchy name@registry
  birrell@pa
- primarily for email, but also gave (primitive) authentication (password as attribute) and access control
- any GV server would take a request from a GV-user agent

ref: Birrell, Levin, Needham, Schroeder, Comm. ACM 25(1) April 82

Clearinghouse

ISO standard based on an extension of Grapevine, developed at Xerox PARC
- three-level naming hierarchy

GNS - see below N-18 to N-20

developed for DEC at SRC, Lampson et al. 1986
- full hierarchical naming
- support for namespace reconfiguration

X.500 - see below N-21

Example: DNS - the Internet Domain Name System

Before 1987 the whole naming database was held centrally and copied to selected servers periodically

The Internet had become too large for this approach and a hierarchical scheme was needed
(Mockapetris 1987)

What does it name? in practice, the objects named are:

* computers
* servers such as mail hosts *named objects* * domains *directories - resolution contexts*
* servers providing other services

Definition of names

hierarchy of components or labels (total max 255 chars)
highest level of hierarchy is last component
label: max 63 chars, case insensitive, restrictions on characters (but arguments over relaxing these)

final label of a fully qualified name (a TLD) can be :

3+ letter code: type of hosting organisation
edu, gov, mil are still US-based, others e.g. com, net, org, int, jobs can be anywhere
2+ letter code: area of registrar, defined by ISO 3166 e.g. uk, fr, ie, de, ...
arpa: for inverse lookup (e.g. 20.0.232.128.in-addr.arpa)

mit.edu
cl.cam.ac.uk
cs.tcd.ie
tu-darmstadt.de

examples of domain names:

final 2-letter label doesn't imply country of location of host, just where registered
e.g. www.yahoo.co.uk has been in Germany

computers using DNS are grouped into ***zones***, e.g. uk, cam
within a zone, management of sub-domains is delegated
e.g. cl is managed locally by the domain manager - names added to a local file

primary name server (***authority***) for a zone is the computer that holds the master list for the zone
usually there will be secondary servers, holding replicas, for the zone

queries can relate to individual hosts or zones/domains, examples:

A computer name -> IPv4 address
AAAA computer name -> IPv6 address
MX mail-host (domain) -> < host, preference, IP address > list
includes mail hosts for detached computers
NS DNS servers for a domain -> < host, IP address, ... > list

Unix examples
/etc/hosts holds IP addresses of hosts in local domain

```
$ /usr/bin/nslookup
> set q=A
> cosi.cl.cam.ac.uk
Address:  128.232.8.110
> www.cl.cam.ac.uk
Address:  128.232.0.20
>set q=MX
>cl.cam.ac.uk
mail exchanger = 5  mx.cl.cam.ac.uk
Address: 128.232. ...
```

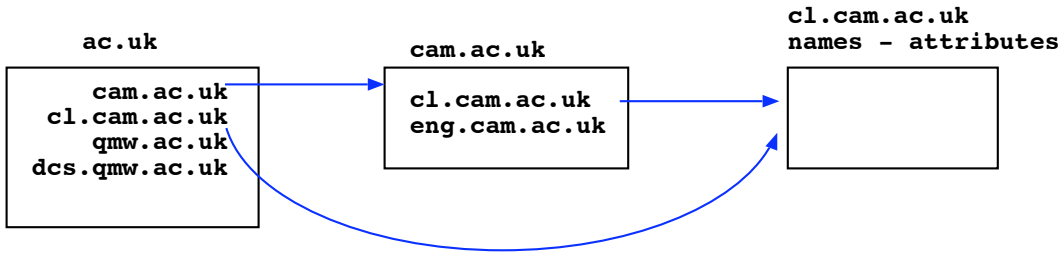
```
$ /usr/bin/nslookup
> set q=NS
> cl.cam.ac.uk
Server:     128.232.1.2
Address:    128.232.1.2#53
cl.cam.ac.uk nameserver=resolv1.cl.cam.ac.uk.
cl.cam.ac.uk nameserver=resolv6.cl.cam.ac.uk.
cl.cam.ac.uk nameserver=resolv2.cl.cam.ac.uk.
cl.cam.ac.uk nameserver=resolv3.cl.cam.ac.uk.
cl.cam.ac.uk nameserver=resolv0.cl.cam.ac.uk.

> set q=A
> resolv1.cl.cam.ac.uk
 ...
Address = 128.232.1.2
```

DNS name servers (note the large scale)

* the domain database is partitioned into directories which form a distributed namespace

  e.g. ac.uk may be held on the computer nsl.cs.ucl.ac.uk

* can lookup DNS directory address for a domain: IP address, well-known port
  need a starting point for name resolution



frequently used so have
a redundant direct link (engineering issue)

* directories are replicated for availability and good response
  (authoritative name server for domain is distinguished (weak consistency))

optimisations

* resolved queries are cached (by user agent and at directories) as naming data tends to be stable
  if not in directory, consult cache. values returned with a TTL (time to live)

* queries and responses may be batched into composite query messages

DNS future issues

1. mobile/roaming devices attach anywhere worldwide

2. mobile, wireless ad-hoc networks, groups form (MANETS)
   (i) some node may act as an internet gateway
   single-hop or multi-hop connection to it
   (ii) may be detached from Internet, may be prepared to share services

approaches
first need protocols - mobile IP: mip6 IETF working group

for 1 above: device can contact local DNS server
local and home can cooperate
can you be monitored while roaming? - privacy?

for 2(i) above - any node can connect, as in 1, via the gateway,
provided gateway has appropriate code

for 2(ii) above: any node may broadcast offering to act as DNS server
asuming it has server code and others have client code
the group can then advertise services to clients

Back to general name service issues: Replication and Consistency

directories are replicated for scalability etc....
how should propagation of updates between replicas be managed?

lookup (args) ⟶ ⟵ is the most recent value, known system-wide, guaranteed to be returned?

if system-wide consistency is guaranteed we have: - delay to update
- delay on lookup

it is essential to have fast access to naming data
- so we must relax the strong consistency requirement

this is justified because:

1. naming data doesn't change very fast, changes propagate quickly, inconsistencies will be rare
   - YES - info on users and machines...usually
   - NO - distribution lists
   - NEW/NO - mobile users and computers
   - NEW - huge number of things to be named - does the design rely on low update traffic? (like service advertisements)

2. we detect obsolete naming data when it doesn't work
   - YES - users
   - NO - distribution lists

3. if it works it doesn't matter that it's out of date

   you might have made the request a little earlier - recall uncertainties over time in DS

The crux of this problem

consistency vs availability tradeoff
have to choose availability for name services - they underlie most use of the system

what should be returned when only weak consistency is supported:

lookup (args) ⟶ either: value, version# / timestamp
⟵ or: not known at time (timestamp of last update)

examples:
service on failed machine - restart at new IP address - update directories - rare event
user changes company - coarse time grain
companies merge - coarse time grain
change of password - takes time to propagate - insecurity during propagation?
changes to ACLs and DLs - insecurity during propagation?
revocation of users' credentials - may have been used for authentication/authorisation
hot lists - must PUSH rather than PULL - must propagate fast

so - take care what name services are being used for, and how they are being used.
Perhaps active database triggers could be useful
(register interest in some change - notified of change immediately)
DNS-SD tries to do this

Long-term consistency must be ensured for correctness

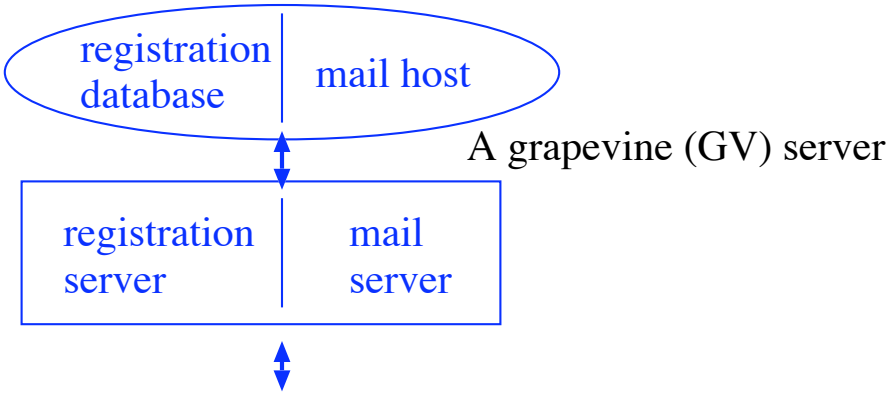
requirement:
if updates stopped there would be consistency after all updates had propagated

this cannot be tested
- we cannot guarantee there will be periods with no updates (quiescence)
- we would, in any case, need to specify failure behaviour in detail

* updates are propagated by the message transport system
  conflicting updates might arrive out of order
  need an arbitration policy e.g. based on timestamps

* typically, transmit whole directories periodically and compare them
  tag the directory with a version number after this consistency check
  e.g. GNS declares a new "epoch" after such a check

Example: Grapevine - outline



2D names name@registry

every GV server contains the gv registry which contains

registry name -> list of locations

2 types of name within a registry Note: small scale allows rapid navigation

group-name -> list of members

used for distribution lists,
access control lists

individual-name -> attributes: (password, mail-host list, ...)

problems - soon outgrew its specification:

- #servers, #clients
- huge distribution lists not foreseen
- message transport used for update messages - updates could be held up.

DEC's Global Name Service (GNS)

Butler Lampson 1986, Designing a Global Name Service, Proc. 5th ACM PODC, pp1-10

Aims: * long life

many changes in the organisation of the name space

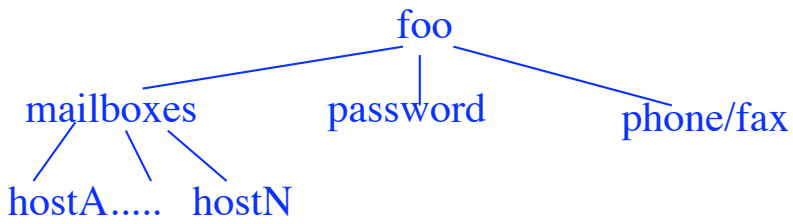
* large size

arbitrary number of names and administrative domains

Names

define two-dimensional (2D) names of the form < directory name, value name >

where value names may be a tree such as



the GNS directory structure is

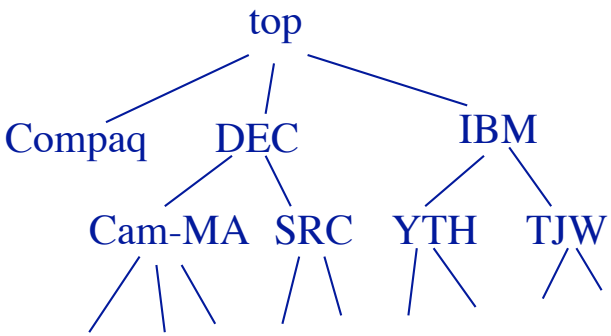
- hierarchical
- **every directory has a UID**, a directory identifier (DI)

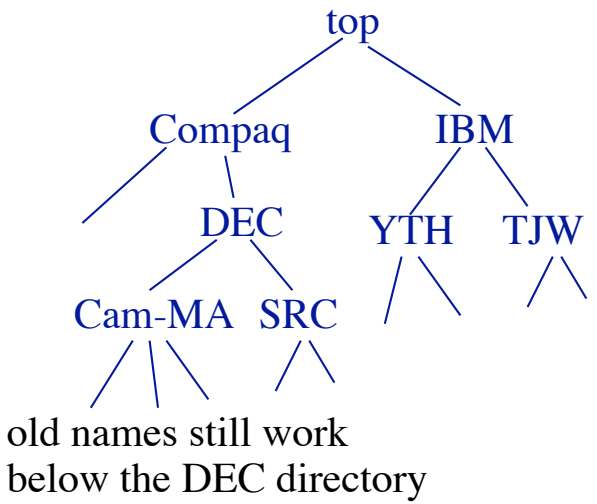
A **full name** is any name starting with a DI

* doesn't require a single root directory
* doesn't rely on the availability of some root directory

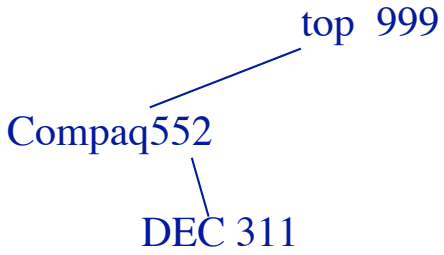
GNS continued

If the directory hierarchy is reconfigured a directory may still be found via its DI
Names starting with that DI will not change, if the reconfiguration is above that DI





Support is needed to locate a directory from its DI (*a pure name - where do we look it up?*)
as well as the usual location of directories by pathname lookup.
Top level directories provide DIs with directory names.



names starting from DEC: 311/SRC, birrell

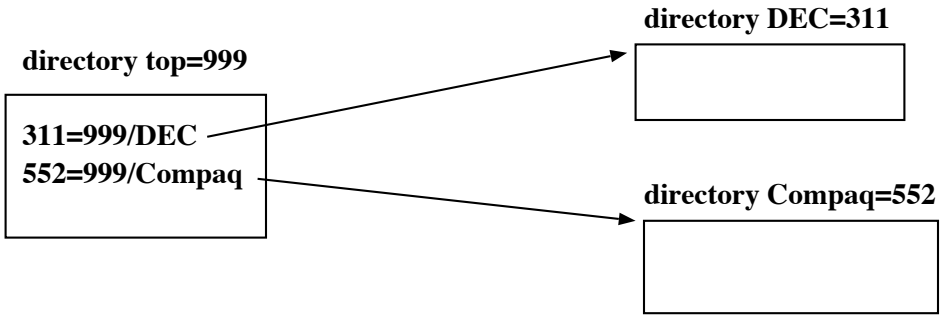
name of DEC directory: is always 311

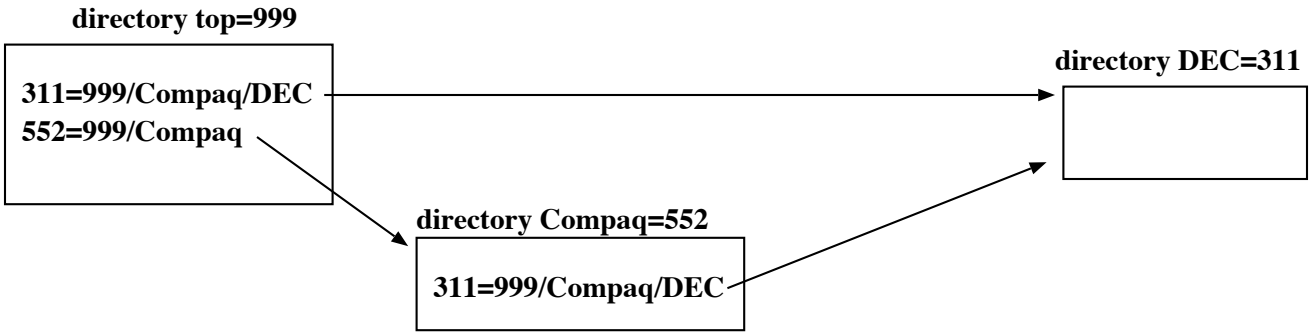
names starting above DEC

was: 999/DEC, name
now: 999/Compaq/DEC, name
was: 999/DEC/SRC, birrell
now: 999/Compaq/DEC/SRC, birrell

directory entries

- include DIs with directory names
- include pathnames from root





X.500 Directory Service (White and Yellow pages)

ISO and CCITT standard, above OSI protocol stack.
More general than a name service where names must be known precisely and are resolved to locations.

components:
DIT directory information tree
DSA directory service agent
DUA directory user agent
DAP directory access protocol

resource-consuming and difficult to use

1993 major revision including replication, access control, schema management.
But X.500 was never accepted as a generic name service.
X.509 certificates for authentication and authorisation have been successful.

Lightweight Directory Access protocol (**LDAP**) Howes, Kille, Yeong, Robbins, 1993

IETF accepted. can download free and deploy - widely used

* access protocol built on TCP/IP

* heavy use of strings, instead of ASN.1 data-types

* simplification of server and client

* current status V3

* LDUP duplication and update protocol (but see internet draft draft-zeilenga-ldup-harmful-02.txt)

Naming - summary

Naming for the Internet - see DNS

Naming for companies, world-wide - see Grapevine, GNS

Standard name services - X.500 (CCITT, ISO), X.509 for authentication, LDAP (IETF)

Naming for the web - document names are based on Internet naming:

scheme://host-name:port/pathname at host
scheme = protocol: http, ftp, local file, ...
host name = web server's DNS address, default port 80
pathname in web server's world of file containing web page

e.g. *http://www.cl.cam.ac.uk/research/.....*

Also web services...

Name Services in middleware

Object-oriented middleware

remotely accessible objects are registered with local ORB
a remote object reference is returned which may be entered in a name service
together with an associated name

e.g. CORBA Naming Service
name -> remote object reference
CORBA Trading Service
attributes -> name, remote object reference
JAVA Naming and Directory Interface (JNDI) for services

naming interface: service interface publication
service-name -> remote object reference
directory interface: attribute -> remote object reference

Message-oriented middleware

MOM evolved from the packet switching paradigm
naming and routing may be defined statically
e.g. IBM MQSeries queue names are assumed known to the application
and embody fixed routing from client to server

e.g. JMS (Java Messaging Service) can use JNDI

Event-based middleware

names are topics or event types, used by (advertisers) publishers and subscribers

topics may be assumed to be known (TIBCO)
or may be advertised (Siena, Hermes, ...)

message routing tables, for publications,
are set up from (advertisments and) subscriptions

subscription can be either topic/type or attribute/content-value based

e.g. topic hierarchy: stocks . stock-exchange-name . stock-type . stock-subtype
subscription: stocks . * . utilities . *

e.g. attribute/content:
subscription: stocks, stock-exchange-name=NY, stock-type=mining, value>$100, ...

e.g. for message type: seen (person, room)
subscription: seen (person = *, room = FN34)

may be integrated with a programming language