Names in Distributed Systems

* Unique Identifiers UIDs e.g. 128 bits

  are never reused
  refer to - nothing
  - the same thing at all times

  UIDs 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: jmb@cl.cam.ac.uk
    jeanb@lcs.mit.edu
    puccini.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/cam/cl etc

Unique names

uniqueness is achievable by using
- a hierarchical name
- a bit pattern

Issues

* a problem with pure names:
  - how do you know that an object does not exist?
  - how may a global search be avoided?

* a problem with impure names
  - how to restructure the namespace
    e.g. when objects move about
    when companies restructure

A familiar example: DNS

The Internet Domain Name Service

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

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

In practice, the objects named are
- computers
- mail hosts *named objects*
- domains *directories - resolution contexts*

top level names are:
com, edu, gov, mil, net, org, int, uk, fr, ...

co ac

examples of domain names are: uk.ac
uk.ac.cam
uk.ac.cl

management is delegated for sub-domains
e.g. cl is managed locally by the domain manager
- names added to a local file

queries are of the form:

computer -> IP address

mail-host -> < host, preference, (IP address)> list

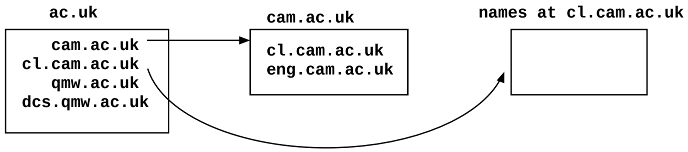
.....others

DNS name servers (note the large scale)

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

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

* directory address: IP address, well-known port



* for availability and good response,
directories are replicated

optimisations
* resolved queries are cached as naming data tends to be stable

* queries and responses may be batched

Naming Services in Distributed Systems

in general - provide clients with attributes of (values associated with) 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. /a/b/c/d
puccini.cl.cam.ac.uk
128 bits

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 (as we have seen for DNS)

name resolution or binding

obtaining a value which allows an object to be used

e.g. file-service? → name service
IP-address, port# ← name service

LATE BINDING is considered GOOD PRACTICE

programs should contain names, not addresses

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

attributes stored by a name service

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	name
directory?	list of hosts

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

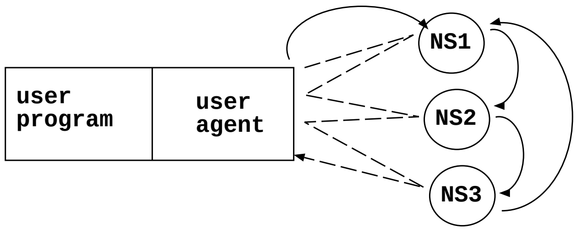
as we saw for DNS, directories are likely to be replicated for scalability, fault-tolerance, efficiency

directory names will resolve to a list of hosts possibly with their addresses to avoid further lookup

attribute-based lookup may be offered -
a YELLOW PAGES style of service for object discovery
e.g. X.500

TOO MUCH information might be dangerous -
used by hackers

architecture of name resolution



user agent starts off with a root address
in the name service
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 and iterate through ac then cam then cl

the UA will cache resilved names as hints for future use

---

alternative: any name server will take a name, resolve it and return the required attribute

Examples of name services

Grapevine

Xerox PARC early 80's
two-level naming hierarchy   name@registry
birrell@pa
primarily for email, but also gave (primitive) authentication (password as attribute), access control,

any GV server would take a request from the GV-user agent

ref Birrell, Lampson, Needham, Schroeder

Clearinghouse

ISO standard based on an extension of Grapevine, developed at Xerox PARC

three-level hierarchy

DNS - see above   internet domain naming service

GNS - see below

developed for DEC at SRC, Lampson et al 1986

full hierarchical naming
support for namespace reconfiguration

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 guaranteed?

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

it is essential to have fast access to naming data
- so 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

   OK - info on users and machines
   NOTOK - distribution lists

   does the design rely on low update traffic?

2. we detect obsolete naming data when it doesn't work

   OK - users
   NOTOK - 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

lookup (args) ⟶ either value
or not know at (time of last update)

telephone company analogy

* partitioned, distributed naming database

* given a name, (Yudel Luke) which directory to use?

  (don't know where to lookup pure names)

* call (#) -> unobtainable
  we detect out-of-date names, call directory enquiries
  cache until new directory comes

* frequency of update

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

* caching numbers in a diary (hints)
  call (#) -> unobtainable
  call directory enquiries or ask another friend

* can't find an entry for e.g. Phillips company

  check spelling - Philips
  look under S.S. rather than Social Services

  need higher-level tools

Long-term consistency must be ensured

requirement (that can't be tested):

if updates stopped there would be consistency
when all updates had propagated

* 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 version after this consistency check
  GNS declare a new "epoch" after such a check

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

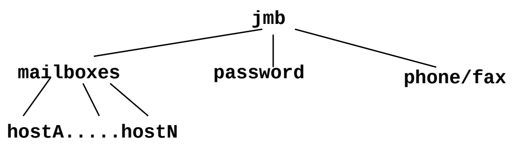
  arbitrary number of names and administrative domains

Names

define two-dimensional (2D) names of the form

< directory name, value name >

where value names may be a tree such as



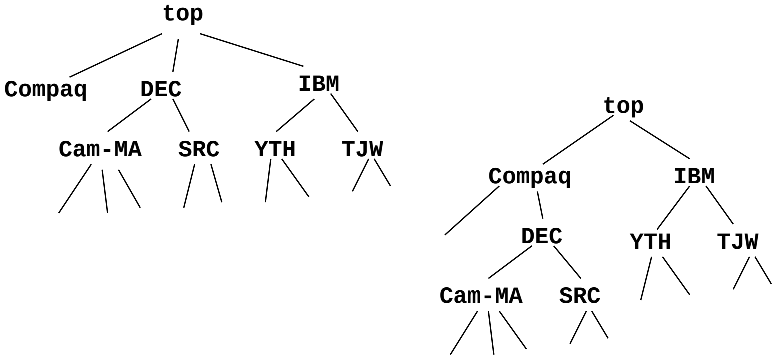
the GNS directory structure is
- hierarchical
- every directory has a UID, a directory identifier (DI)

A full name is any name starting with a DI

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

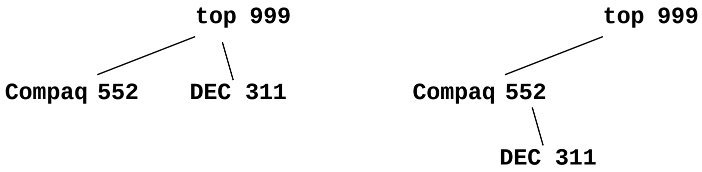
GNS continued

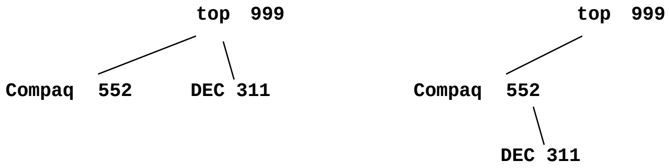
If the directory hierarchy is reconfigured a directory may still be found via its DI and names starting with that DI will not change, if the reconfiguration is above that DI.



old names still work
below the DEC directory

support is needed to locate a directory from its DI as well as the usual location of directories by pathname lookup





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