Imperial College London

Computer Networks and Distributed Systems

Part 2.5 – Name and Directory Services

Course 527 – Spring Term 2017-2018

Emil Lupu

e.c.lupu@imperial.ac.uk

Contents

Naming Concepts

Names

Names are strings used to identify objects such as people, computers, processes, files, communication links.

Textual names:

- Identify services, people etc. a.person@doc.ic.ac.uk, swan.doc.ic.ac.uk
- Human readable

Numeric addresses:

- 146.169.2.20
- Location dependent

Object identifiers:

- Pure names, usually numeric, large context eg 5021334679
- Never reuse ➔ include timestamp component
- Location independent
- Used to compare identities
- Example unique identifiers?

Naming Concepts

Route:

- Directed path from source to destination
- gummo!swan!mta-relay!csgate!xyz

Group names identify a set of objects

- *Multicast* address: single address which maps to a *group* of objects
- *Broadcast* address: single address which maps to *all* objects in a context (eg nodes on a LAN)

There is no real distinction between a name, address and route – they are all names within different namespaces which are resolved in different ways

Name Space

A set of homogeneous contexts with a syntax for specifying names and rules for resolving names e.g. right to left or left to right. A name may specify a path through a namespace. e.g. file system names.

Name context: a set of unique names which can be resolved to particular objects ie names must be unique within a single context (eg directory)

Namespace defines the potential set of valid names. The namespace name may be used as prefix to name to indicate to the system which name service to use for name resolution

Name Space Examples

Unix file system: contexts = directories (hierarchic)

Internet host names: contexts = domains

IP addresses contexts = subnets, host address

Ethernet address: single context (flat namespace)

Telephone no: country, area, number

Name Space

Naming authority may control the assignment of names to objects within a namespace or within a context e.g.

- university or department assigning login names
- X500 Administration domain
- Xerox assigning Ethernet numbers

Objects may be registered more than once within a namespace.

Alias: alternative name for an object which maps directly to the object. E.g., ln creates hard links in a file system
Symbolic name: an alternative name which maps to a path name in the name space. This path name has to be resolved to locate the object. E.g. ln –s creates a symbolic link, which is a path.

Hierarchic names

A sequence of name tokens resolved in different contexts
Syntax : name token (text string) + delimiter eg "/" or "."
e.g. /var/log/system.log

Or through explicit identification of context as in X500 e.g.
C=uk, O=ac, OUN=ic, LN=doc, CN=mss

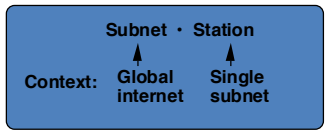
The structure has semantic significance i.e. it reflects organisational structure or physical location and the name changes if object migrates

Names may be absolute – relative to a root e.g.:
/homes/ecl1/public_html/index.html
or relative to a current context e.g.
~/public_html/index.html

Distributed authorities control local contexts

Examples

- Topology based network address



- Unix File systems
- Email addresses - domain name systems
  a.person@doc.ic.ac.uk
- X500

Flat Names

Single global context and naming authority for all names

- e.g. National Security numbers, object identifier, computer serial number, Ethernet address

No semantic information in name

- Difficult to resolve: which directory to use and how to access directory
- Easy to create - local information

How would you create a unique flat object ID?

Name Binding

An association between a name and an object.

Name is bound to a set of attributes of an object, one of which may be an address. Other attributes may include type information, quality of service information for services; OS, memory, Ethernet address for workstations.

Name may be a placeholder which can be bound to different objects at different times e.g. named ports pointer, RPC interface ref

Sometimes we can translate from one namespace to another if there is a binding between names in the different namespaces e.g. IP address to Ethernet address

Directory Service

A directory service translates (resolves) names for objects into a set of attributes (one of which is usually an address) i.e. lookup

Complex searches based on attribute values or ranges

- ***Yellow pages***: attribute values returns a list of objects
- ***Browsing*** when user cannot specify query

Basic Operations: insert, modify or delete entries.

Manage internal, possibly replicated tables & propagate

updates to replicas

Cache results of queries

Access control for both reads and updates

Directory Service Goals

Location independence - same name wherever it is used

Migration transparency - object can move and still be located without changing its name

Predictability - uniquely identify named objects

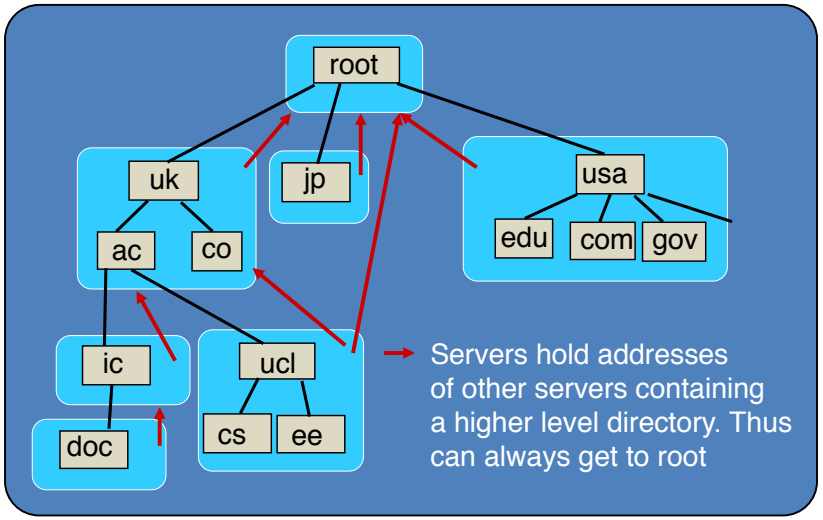
Availability - at least commensurate with objects named

Higher availability required for key information may require replication of information. Strong or weak consistency between replicas?

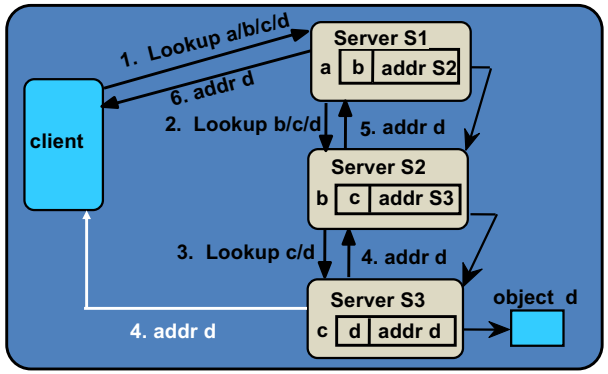
Decentralisation of administration - no central authority

Scalability - cater for very large systems by federating or merging name spaces

Distributed Name Servers



Recursive Lookup

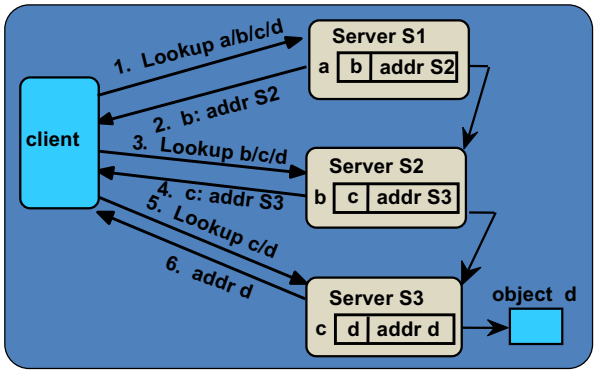


Cannot block S1 during time taken to resolve complete path name threads within server to do lookup

Increase loading and complexity of server.

If message passing used, Server 3 can reply directly to client.

Iterative Lookup Using Referrals



Simpler servers – off load work (referrals) to clients, but there are multiple clients anyway.

Used with a name server agent in client to do lookup

Replication

Replicate directories for performance & availability

Updates:

- Write to single master then responsible for propagating updates to all replicas
- Write to any replica – and later merge updates. Time stamp updates and latest update wins if updates occur in different servers

Lookups: Try any local server – it can always get higher

Caching: Clients typically cache names & addresses of recently used objects.

Replica Consistency

Strong Consistency:

- Lock all copies while updating so no lookups while updating
- Do you block update if a copy is unreachable? This is impractical.

Weak Consistency:

- Eventually propagate updates to all copies but entries may be out of date in some copies.
- Address can be considered a hint, which may be incorrect – detected at invocation time.

What to do if address is incorrect?

Internet Domain Name Service (DNS)

Used mainly for host names and email addresses.

Extensible number of fields and each fields is variable length text delimited by "."

eg a.smith@doc.ic.ac.uk dse-mail.doc.ic.ac.uk
schmidt@atm.aeg.kn.daimlerbenz.com

Top level: USA domains + other countries
com, edu, gov, mil, org, fr, uk, jp, kr, de

Second Level
e.g. in UK – co, ac, mod

Third level
Institutions or companies e.g. ic, lancs, bt

Additional Levels depend on institutions.

DNS Implementation

Resource records hold

- Domain name for which record applies
- Time to live: initial validity time for cached entries
- Type of record
  e.g. IP address, mail server for domain, name server, alias names, host description or text information
- Value fields: number, domain name or ASCII string

Replicated and partitioned information - update master.

Secondary servers download data periodically from master and can cache entries from any other servers.

Secondary servers always hold address of one or more masters a numbers of levels above

Higher level directories do not change frequently and are cached by many servers.

X500 Directory Service

A object-oriented database in which objects represent people, services, servers routers etc.

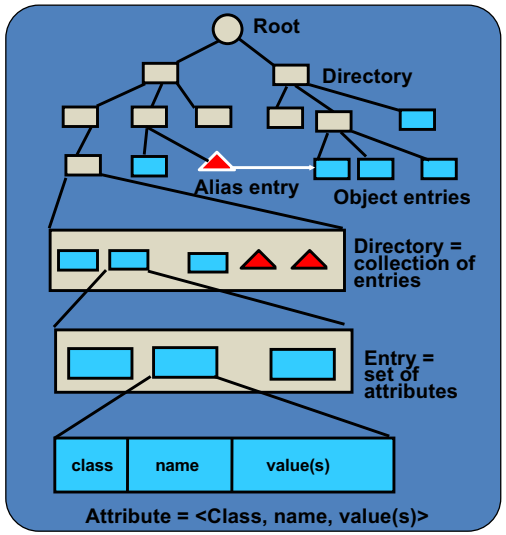
Each entry is a set of attributes for an object, one can be tagged as distinguished and is used to identify the object.

Parts of the information in the Global directory can be stored and controlled by independent administrations

Differs from normal database:

- Object-oriented, attribute-based information
- Optimised for reads rather than updates ie static information
- Distributed & some directories may be highly replicated
- Hierarchical naming of objects
- Potentially large (>1M) numbers of objects
- No transaction support

X500 Information Model



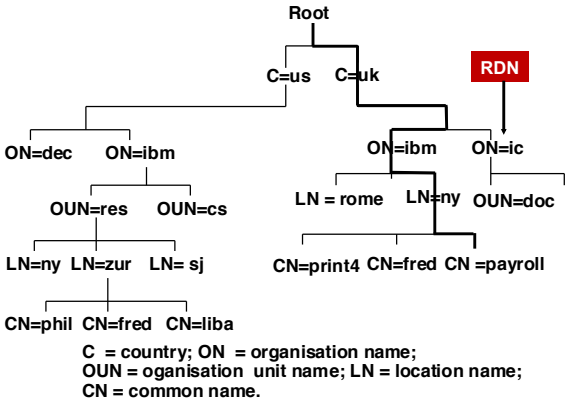
Attributes: eg name, email, phone, fax and room number for a person, sets - members of a group

Alias: alternative name for object + symbolic link to object

Collective Attributes: apply to a set of entries or subtree eg switchboard phone no.

Operational Attributes: set by administrators or system eg date & time of modification, access control information & roles

Directory Information Tree (DIT)



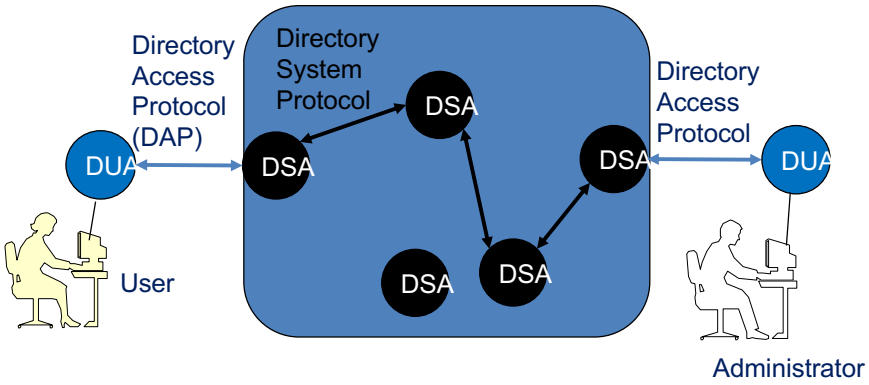
Relative Distinguished Name (RDN): name within a context e.g. ON=ic

Distinguished name: determined by position in tree eg C=uk, ON=ibm, LN=ny, CN=Payroll

Directory Schema

- Controls what information can be stored in directory by means of object & attribute class specifications. Uses Abstract Syntax Notation One (ASN.1)
- Predefined Standards - extendable by users
- Standard objects: Top, Alias, Country, Locality, Organization, Organization Unit, Person …. Uses object class inheritance
- Standard attributes: Name, Description, Fax number, Street address, Postal Code, Country, Owner, Member, ….. Attributes definitions include rules for matching - eg equality, ordering, substrings, ignore case

Distributed Directory



DUA = Directory User Agent i.e. client
Discover DSA, cache information

DSA = Directory System Agent i.e. server - holds partial information tree

LDAP

Lightweight Directory Access Protocol (LDAP)

Simpler access protocol than X500

- TCP/IP rather than OSI
- Omits some operations, service control and security features
- String encoding for distinguished names and data elements

Widespread use across industry to hold addresses as well as policy and network configuration information

LDAP Basic Protocol Operations

Bind/Unbind:	set up session & transfer authentication information
Search:	to find one or more objects and read their attributes
Modify:	to add, delete or replace attribute values for an object
Add:	add new entry (ie object) - can include values for attributes
Del:	delete entry
Compare:	evaluate an assertion based on an attribute of an object e.g. Is Person=sloman age >50?
Abandon:	abandons an outstanding request eg complex search
Extended:	permits extensions to directory service operations
Notifications:	unsolicited messages from server to client to signal extraordinary conditions eg failure

LDAP Search Operation

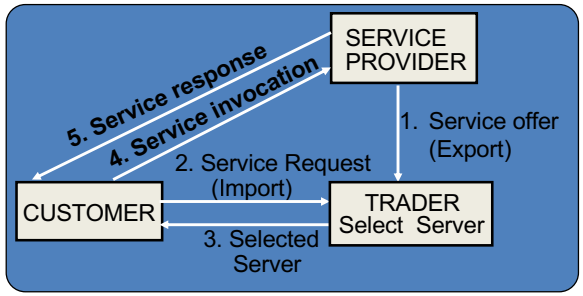
contains following parameters:

Base object:	starting directory in DIT
Scope:	base object, single level, whole subtree
Alias dereferencing	instructions
Size limit:	maximum number of entries returned in result
Time limit:	maximum time (in secs.) allowed for search
Types only:	whether search should return attribute classes, values or both
Attributes:	list of attributes to be returned
Filter:	defines conditions which must be fulfilled for search to succeed. Evaluates to True, False or Undefined. A predicate combining attributes using and, or, not, =, ≥, ≤, substrings, approximate (using phonetic algorithms eg soundex), extensions (to permit extensions to filter rules).

Trading

Many servers providing similar services may wish to offer their services in the "market place".

Trading is the activity of selecting one or more servers, from the offered services, which meet a specification of required service provided by prospective consumer. The selection is based on a service specification supplied by the service providers



Service Offers

Contains service information ***exported*** by a service provider:

- Service type name
  E.g. printer, file service, time service, compile service
- Service properties as set of <name, value> pairs
  Mandatory properties + zero or more optional ones
- Interface reference: Server identifier + Interface identifier + Server network address

Trader returns a unique offer identifier

A server may support multiple interfaces, so trading is based on interface reference rather than server identifier.

A server may export the same interface in different servers.

Service Request

The ***import*** information provided by a customer to the trader to enable the trader to match and select an offer from its repository.

The customer's specification of a required service includes:

- Required service type name
  Type management policies determine whether this can be matched with a subtype
- Constraint - property selection expression in terms of names and values combined using functions or boolean operators e.g.
  - Query (printservice, [papersize = A3 & resolution > 400] or [papersize = A4 & floor = 5])
- Desired property values to be returned with the offer

Summary

Distributed processing services are:

- Logically centralised
- Implemented as a set of co-operating servers
- Servers accessed by clients using RPC/Object invocation
- Partitioned to reflect application distribution
- Replicated for availability: weak or strong consistency?

Require access control for security

Use local service agent in client node

- Cache recently used information to improve performance: hints
- Hold knowledge of at least one server, which can be used to find other servers