Distributed Systems

COMP90015 2018 SM1

Name Services

Lectures by Aaron Harwood

Names, addresses and other attributes

Names are used to refer to a wide variety of resources such as computers, services, remote objects and files, as well as to users. A name is needed to request a computer system to act upon a specific resource chosen out of many.

Processes need to be able to name resources in order to share them. Users need to be able to name each other in order to (privately or directly) communicate.

In some cases a description that includes *attributes* of the resource can be used to uniquely identify it. In some cases a client requires a service but does not have a preference for a particular entity to provide that service.

An *address* is an attribute of an object. An address cannot be used as a name because the object may change its address.

Filenames such as `/etc/passwd` and URLs such as `http://www.cdk4.net/` are examples of *human-readable* names. An *identifier* is a name that is usually not human-readable, such as remote object references and NFS file handles. Identifiers can be more efficiently stored and processed by software.

A *pure name* contains no information about the object itself. Whereas a non pure name contains some information about the object; usually some kind of address information.

A pure name must be looked up, to obtain an address, before the named resource can be accessed. E.g., names of people are pure names.

An extreme example of a non pure name is an address. E.g. a user's email address is often used as a name for the user. However if the user changes their email address then the "name" is no longer correct.

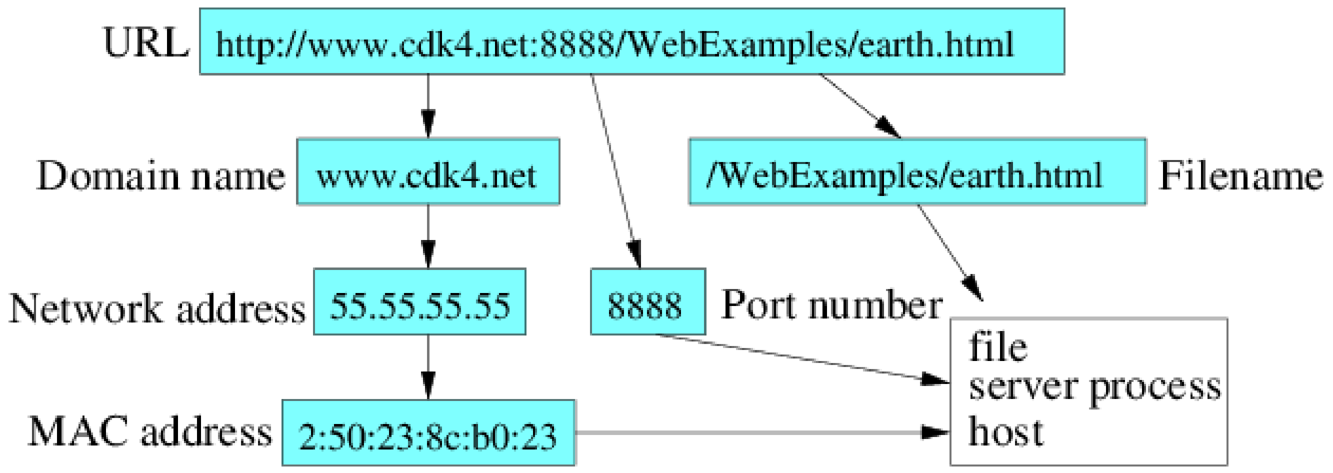
A name is *resolved* when it is translated into data about the named resource or object.

The association between a name and an object is called a *binding*.

In general, names are bound to attributes of the named objects and a common attribute is the address of the object.

- Domain Name System (DNS) maps human readable domain names to IP addresses and other kinds of attributes such as the type of host (e.g. mail server) and time for which the DNS entry remains valid.
- The X500 directory service can be used to map a person's name onto attributes including their email address and telephone number.

Example name



Names and services

Many names have meaning only to the service that creates the name, i.e. they have only *local significance*. When a service allocates a resource it also generates a unique name and a client to the service needs to supply the name in order to access the associated resource.

In some cases the client can specify the desired name for a new resource to the service. E.g., users will specify what email (user) name they want to use. The service needs to ensure that the names are locally unique. Along with a unique domain name, the email address is unique. There may be one or more hosts that maintain the service; with e.g. the domain name mapping to the host that is least loaded at the time of access.

Services sometimes need to cooperate in order to have name consistency. E.g., one attribute of a file is the owner which is a unique ID on the server. A login name resolves to a unique ID for that user. For the owner to be recognized by a client in NFS the client must ensure that user login names resolve to the same unique IDs as on the server.

Uniform Resource Identifiers

Uniform Resource Identifiers (URIs) are concerned with identifying resources on the Web, and other Internet resources such as electronic mailboxes.

URIs are intended to allow a generic way of specifying the identifier so as to make it easy for common software to process the identifier. This allows new types of identifiers to be readily introduced and for existing identifiers to be used by a wide variety of different software and services.

Very basically, the first part of a URI contains a short text mnemonic for the kind of resource being named, followed by a colon, e.g. `http:`, `widget:` or `tel:`. The remaining portion of the URI must be interpreted according to the URI syntax which allows for local rules concerning the resource.

A URL is a URI. URLs provide ways to locate the resource being named. They clearly suffer if the resource has since changed its name (e.g. broken links in the Web).

Uniform Resource Names (URNs) are URIs that are used as pure resource names rather than locators. An URN requires a resolution service or name service to translate the URN into an actual address to find the named resource.

The URI prefix `urn:` has been allocated for URNs, e.g. `urn:ISBN:0-201-62433-8` identifies a resource (book) by the ISBN number, but there are many kinds of different URI prefixes in use today. E.g. for referring to documents:

- `doi:10.1007/s10707-005-4887-8` where the lookup service is `http://dx.doi.org/10.1007/s10707-005-4887-8` and this resolves to `http://www.springerlink.com/content/c250mn1u2m7n5586/` and in turn this refers to a document, "Building and Querying a P2P Virtual World".

- `oai:arXiv.org:cs/0605057` where the lookup service is `http://arxiv.org/abs/cs/0605057` and this resolves to a document. In this case the `oai` or Open Archives Initiative resource name includes the resolver service domain name.

Name Services

The major operation of a name service is to resolve a name, i.e to lookup the attributes that are bound to the name.

Name management is separated from other services largely because of the openness of distributed systems, which brings the following motivations:

- *Unification*: Resources managed by different services use the same naming scheme, as in the case of URIs.
- *Integration*: To share resources that were created in different administrative domains requires naming those resources. Without a common naming service, the administrative domains may use entirely different name formats.

Name services have become more important as the size of distributed systems has grown.

Goals of the Global Name Service

- To handle an essentially arbitrary number of names and to serve an arbitrary number of administrative organizations.
- A long lifetime, over many changes to the names and the system.
- High availability, dependent services stop working if the name server is unavailable.
- Fault isolation, local failures do not cause the entire service to fail.
- Tolerance of mistrust, a large open system cannot have any component that is trusted by all of the clients in the system.

Name spaces

A *name space* defines the set of names that are valid for a given service. The service will only lookup a valid name and may or may not resolve the name (the name may be unbound). E.g., "Two" is not a valid name for a UNIX process but "2" is.

Names may have a structure that is hierarchical, like a UNIX filename, or otherwise they are flat as in a randomly chosen integer identifier.

Structured names allow the lookup procedure to be more efficient and allows the name to incorporate semantics about the resource.

Names may be unbounded in length or fixed length. Unbounded length names allow the name to be whatever is required (e.g. by the user). Fixed length names are easier to store and process. E.g. consider the differences between a name string of arbitrary length and a 32 bit identifier.

An *alias* is a name that is typically substituted by the name service for another name that is harder to remember. In other cases the alias is a common name that allows ease of access to a resource which is managed using some other name. E.g. `www.example.com` may be an alias for `fred.example.com`, and later change to `alice.example.com` for management purposes. The alias does not change though. This provides transparency.

A *naming domain* is a name space for which there exists a single overall administrative authority for assigning names within it.

Administrative authority is usually delegated by dividing a domain into sub-domains. Each sub-domain shares a common part of the overall name in the name space.

Name resolution

Name resolution is in general an iterative process. A name either resolves to a set of primitive attributes or it resolves to another name.

The use of aliases make it possible for resolution cycles to occur and the potential for the resolution process to never terminate. Two solutions to overcome this include:

- Abandon the resolution process after some number of iterations.
- Require administrators to ensure that no cycles occur.

Distribution and navigation

Any name service that stores a very large database and is used by a large population will not store all of its naming information on a single server computer.

Bottlenecks include network I/O and server reliability.

Heavily used name services can use replication to increase availability.

When name service authority is delegated then service distribution is also naturally distributed over the delegated authorities. In other words name service data is usually distributed in terms of domain ownership.

When the name service is distributed then a single server may not be able to resolve the name. The resolve request may need to propagate from one server to another, referred to as *navigation*.

The client name resolution software carries out navigation, e.g. on behalf of the application that requests for a name to be resolved. Approaches to navigation can be categorized in different ways:

- *iterative navigation* -- The client makes the request at different servers one at a time. The order of servers visited is usually in terms of domain hierarchy. Always starting at the root server would put excessive load on the root.

- *multicast navigation* -- The client multicasts the request to the group (or a subset) of name servers. Only the server that holds the named request returns a result.

- *non-recursive server-controlled navigation* -- The client sends the request to a server and the server continues on behalf of the client, as above.

- *recursive server-controlled navigation* -- The client sends the request to a server and the server sends the request to another server (if needed) recursively.

Caching

Caching is a key technique to achieving performance for name services. The binding of names to attributes including other names is observed to not frequently change in many circumstances.

The results of a name resolution request can be cached by the client and by the servers.

Caching is used to eliminate high level name servers from the navigation path and allows resolution to proceed despite some server failures.

Of course, cached data may be out of date and changes can take time to propagate through the system.

Domain Name System

The Domain Name System (DNS) is a name service design whose main naming database is used across the Internet.

Before DNS, all host names and addresses were held in a single central master file and downloaded by FTP to all computers that required them.

The problems with the original name service included:

- It did not scale to large numbers of computers.
- Local organizations wished to administer their own naming systems.
- A general name service was needed -- not one that serves only for looking up computer addresses.

DNS is designed for use in multiple implementations, each of which may have its own name space, though in practice the Internet DNS name space is the one in widespread use.

DNS naming

The DNS name space is partitioned both organizationally and according to geography, though the domain name itself does not force the named resource to be located in any particular place or location.

DNS domain names are hierarchical from right to left, with "." characters used as a separator. E.g. the original highest level domains were `com`, `edu`, `gov`, `mil`, `net`, `org` and `int`. Country domains are `au`, `us`, `uk`, etc.

Each domain authority can specify their own subdomains, e.g. the UK uses `co.uk` and `ac.uk` for companies and academic communities (edu) respectively.

DNS queries

In general applications use the DNS to resolve host names in IP addresses. E.g. a query for DNS name `www.unimelb.edu.au`:

```
 aharwood:~\$ host www.unimelb.edu.au
www.unimelb.edu.au has address 128.250.6.182
```

DNS can also be used to make requests for certain services that support a domain, a common example being the mail host for a domain. E.g. the DNS request for the mail host for domain `unimelb.edu.au`.

```
 aharwood:~\$ host -t MX unimelb.edu.au
unimelb.edu.au mail is handled by 10 antispam4.its.unimelb.edu.au.
unimelb.edu.au mail is handled by 50 muwayb.ucs.unimelb.edu.au.
unimelb.edu.au mail is handled by 10 antispam1.its.unimelb.edu.au.
unimelb.edu.au mail is handled by 10 antispam2.its.unimelb.edu.au.
unimelb.edu.au mail is handled by 10 antispam3.its.unimelb.edu.au.
```

Reverse resolution allows an IP address to be resolved into a domain name. E.g.:

```
 aharwood:~\$ host 128.250.6.182
182.6.250.128.in-addr.arpa domain name pointer www.unimelb.edu.au.
```

Host information allows information about a host to be obtained. This is usually not provided by the administrator because it constitutes a security problem. E.g.:

```
 aharwood:~\$ host -t HINFO www.unimelb.edu.au
www.unimelb.edu.au host information "None" "None"
```

Well-known service allows information about services that are run by a computer to be returned.

DNS name servers

The DNS database is distributed across a logical network of servers.

The DNS naming data are divided into *zones*. A zone contains the following data:

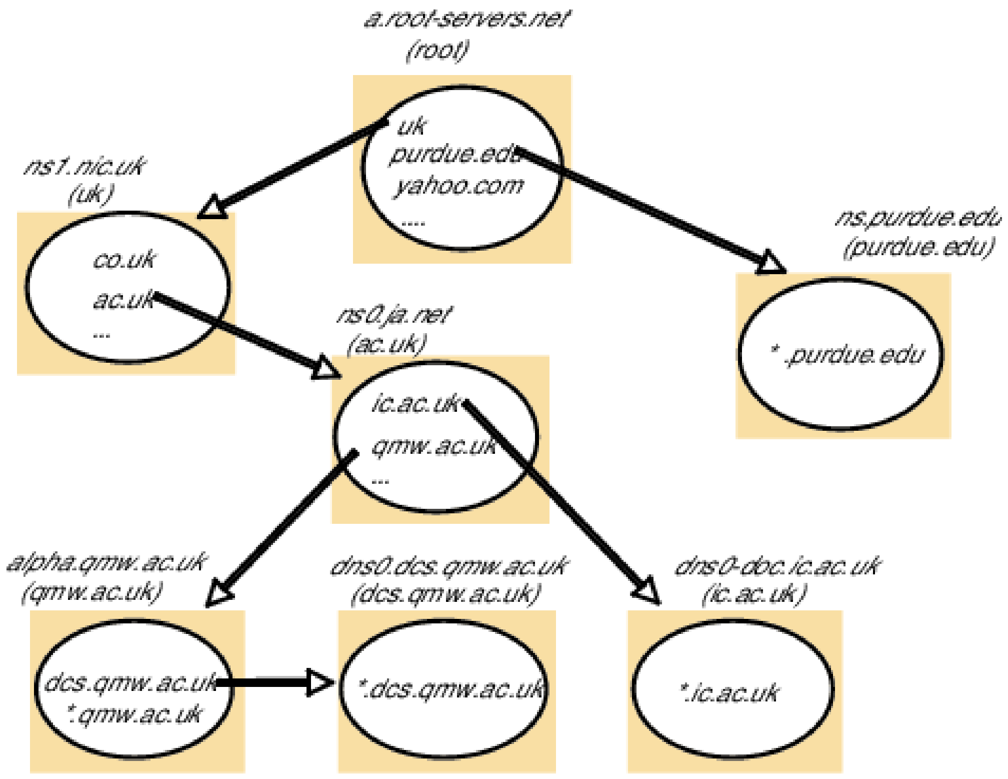
- Attribute data for names in a domain, less any sub-domains administered by lower-level authorities. E.g., a zone could contain data for `unimelb.edu.au` but not for `csse.unimelb.edu.au`.
- The names and addresses of at least two name servers that provide *authoritative* data for the zone.
- The names of the name servers that hold authoritative data for delegated sub-domains and the IP addresses of these servers.
- Zone management parameters, such as those governing the caching and replication of zone data.

The DNS architecture specifies that two name servers be provided for each domain, so that the name service can be available in the event of a single server crash.

A *primary* or *master server* reads zone data directly from a file. A *secondary server* downloads zone data from a primary server. Both of these kinds of servers are said to provide authoritative data for the zone. Secondary servers check and update their information on a regular basis, according to zone management parameters.

Any DNS server is free to cache data from other servers. A server that does so must let clients know that the data is not authoritative. Each entry in a zone has a time-to-live and a server must delete or update a cached entry after this time.

Example DNS domains



DNS resource records

Zone data are stored by name servers in files in one of several fixed types of resource record. For the Internet database these include:

- `A` -- A computer address, an IP number.
- `NS` -- A name server address, a domain name for the server.
- `CNAME` -- The canonical name for an alias, a domain name for alias.
- `SOA` -- Marks the start of data for a zone, parameters governing the zone.
- `WKS` -- A well-known service description, list of service names and protocols.
- `PTR` -- Domain name pointer (reverse lookups), a domain name.
- `HINFO` -- Host information, machine architecture and operating system.
- `MX` -- Mail exchange, list of (preference,host) pairs.
- `TXT` -- Text string, arbitrary text.

Example DNS zone file

```
\$TTL   86400 ; 24 hours could have been written as 24h or 1d
\$ORIGIN example.com.
@  1D  IN    SOA ns1.example.com.   hostmaster.example.com. (
                  2002022401 ; serial
                  3H ; refresh
                  15 ; retry
                  1w ; expire
                  3h ; minimum
                 )
       IN  NS     ns1.example.com. ; in the domain
       IN  NS     ns2.smokeyjoe.com. ; external to domain
       IN  MX  10 mail.another.com. ; external mail provider
; server host definitions
ns1    IN  A      192.168.0.1  ;name server definition
www    IN  A      192.168.0.2  ;web server definition
ftp    IN  CNAME  www.example.com.  ;ftp server definition
; non server domain hosts
bill   IN  A      192.168.0.3
fred   IN  A      192.168.0.4
```