Distributed Name Services

A distinct service that is used by client processes to obtain attributes such as addresses of resources or objects when given their names

Introducing Name Services

- Naming is fundamental concept in distributed systems design
- Naming facilitates communication and resource sharing
- Names are used to refer to a wide variety of resources such as computers, services, remote objects and files, as well as to users

Names, Addresses, Attributes

- The term "identifier" is sometimes used to refer to names that are interpreted only by programs
- Identifiers are chosen for the efficiency with which they can be looked up and stored by software
- ***Pure name*** - uninterpreted bit patterns; they have to be looked up before they are of any use
- ***Non-pure name*** - contains information about the object that they name, including location information

Addresses

- An object's address: a value that identifies the location of the object rather than the object itself
- Addresses are the opposite of pure names
- Addresses are efficient for accessing object
- For example: e-mail addresses are organization or ISP specific; change organization or ISP, and you need to change your e-mail address

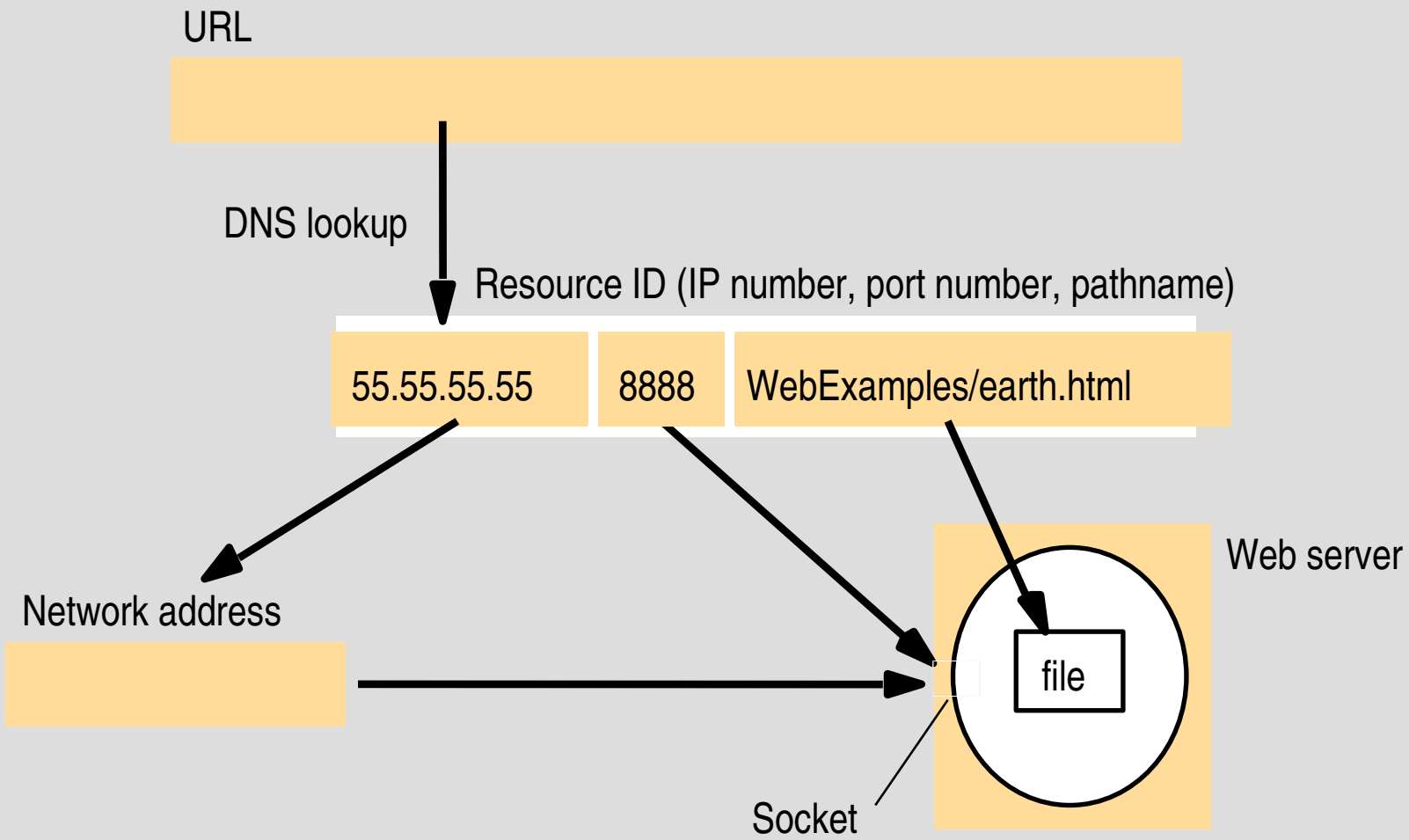
Name Resolution

- A name is resolved when it is translated into data about the named resource
- The association is called a "binding"
- Names are - generally - bound to the attributes of the named objects
- An attribute is the value of a property associated with an object (with an object's address usually the key attribute)

Attribute Examples

- DNS maps domain names to the attributes of a host computer - IP address, type of entry, TTL value for each host entry
- X.500 maps a person's name onto attributes (including their e-mail address and telephone number)
- CORBA's name service maps remote objects onto remote object references

Naming and URLs



Names and Services

- Many names are specific to some particular service
- A client uses such a name when requesting a service to perform an operation upon a named object or resource that it manages
- Obviously, names must be readable by and meaningful to humans
- Given the connectivity provided by the Internet, these naming requirements are potentially world-wide in scope

Uniform Resource Identifiers (URI)

- There's a need to identify resources on the web
- Important goal is to identify resources in a coherent way
- The advantage of uniformity is that it eases the process of introducing new types of identifier, as well as using existing types of identifier in new contexts, without disrupting existing usage
- Some URIs provide information to locate a resource, while some are used as pure resource names

Uniform Resource Locators (URL)

- URLs are reserved for identifiers that are resource locators
- URLs are efficient identifiers for accessing resources
- However, they suffer from the disadvantage that if a resource is deleted or if it moves, then there may be dangling links to the resource containing the old URL
- The wrong resource may be returned or no resource may be returned

Uniform Resource Names (URN)

- URNs are URIs that are used as pure resource names rather than locators
- For example, `mid:0E4FC2720-5C02-11D9-B115-000A95B55BC8@hp1.hp.com` is a URN that identifies the e-mail message containing it in its Message-Id field

Name Services

- A name service stores a collection of one or more naming contexts
- A *naming context* is a set of bindings between textual names and attributes for objects such as users, computers, services and remote objects
- **Name resolution** - resolving a name, that is, to look up attributes from a given name

Motivation for Name Services

- **Unification** - it is often convenient for resources managed by different services to use the same naming service
- **Integration** - it may become necessary to share and use name resources that were created in different administrative domains

Name Service Requirements

- Initially quite simple - *bind names to addresses*

- The interconnection of networks and the increased scale of distributed systems have produced a much larger name-mapping problem
- Grapevine was an early example of an extensible, multi-domain name service, and DEC's Global Name Service was a descendant

GNS Goals

- To handle an essentially arbitrary number of names and to serve an arbitrary number of administrative organizations
- A long lifetime
- High availability
- Fault isolation
- Tolerance of mistrust

Naming and Caching

- To provide satisfactory service, name services rely heavily upon replication and caching of naming data
- Cache consistency needs not be strictly maintained
- As updates are less frequent within an established naming service, the use of an out-of-date copy of a name translation can generally be detected by client software

Namespaces

- A namespace is a collection of all valid names recognized by a particular service
- Namespaces are generally arranged as a hierarchy
- Hierarchic namespaces are potentially infinite, so they enable a system to grow indefinitely
- An important advantage of a hierarchic namespace is that different contexts can be managed by different people

Example Namespace

- The DNS namespace has a hierarchic structure - a domain name consists of one or more strings called "name components" or "labels", separated by the "." delimiter
- DNS does not recognize a relative name, as all domain names are referred to absolutely in relation to the global root node

Naming Design Issues

- **Aliases** - allows for a convenient name to be substituted for a more complicated one (providing transparency)
- **Naming Domains** - a naming domain is a namespace for which there exists a single overall administrative authority for assigning names within it (but that is free to delegate this task, if so desirable)
- The administration of domains may be devolved to sub-domains

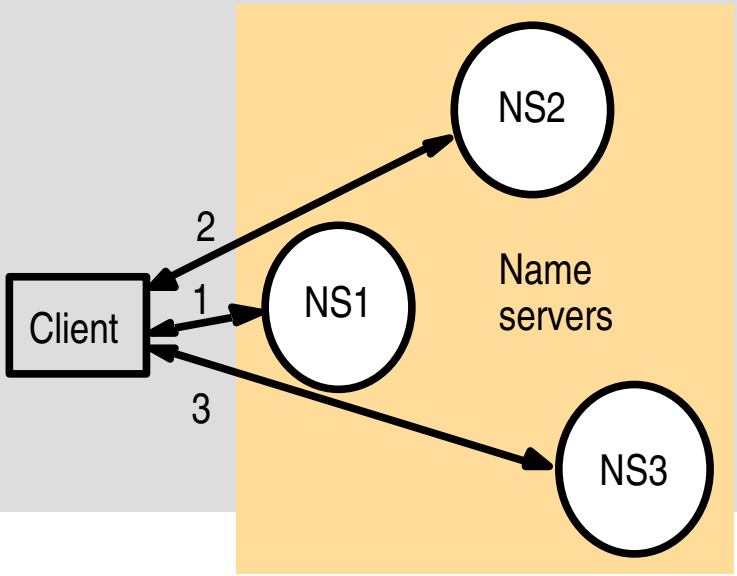
Name Resolution and Replication

- An iterative process whereby a name is repeatedly presented to naming contexts
- It either maps a given name onto a set of primitive attributes directly or it maps it onto a further naming context
- Any heavily used name services should use replication to achieve high availability, for example, the DNS database subsets are replicated in at least two failure-independent servers

Navigation

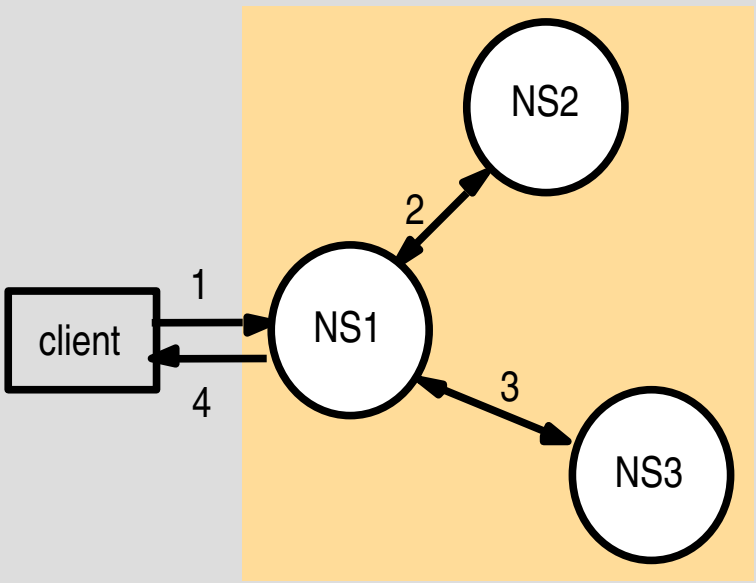
- The process of locating the naming data is called navigation
- Name resolution software carries out navigation on behalf of the client
- There's a distinction made between iterative and recursive navigation

Iterative Navigation

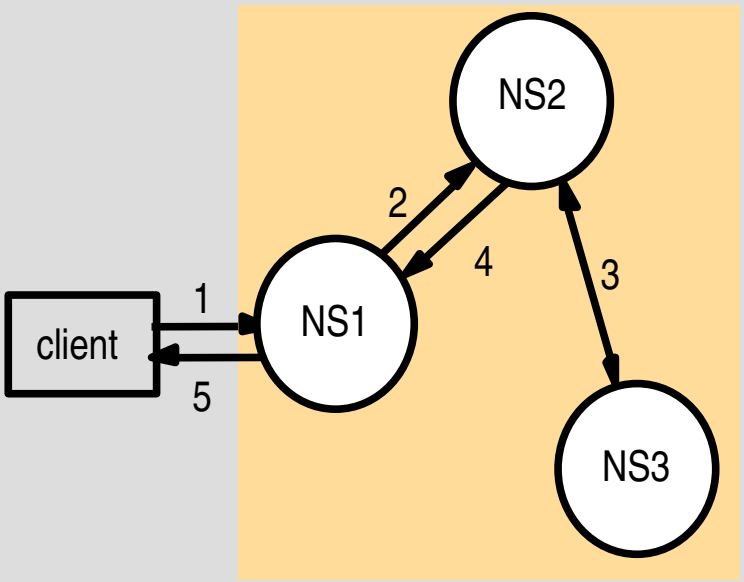


A client iteratively contacts name servers NS1–NS3 in order to resolve a name

Recursive Navigation



Non-recursive
server-controlled



Recursive
server-controlled

A name server NS1 communicates with other name servers on behalf of a client

Multicast Navigation

- With multicast navigation, a client multicasts the name to be resolved and the required object type to a group of name servers
- The server that holds the named attributes responds to the request
- If the name proves to be unbound, then the request is greeted with silence
- Some technologies include a separate server to respond to a client whenever a name is unbound

More on Caching

- It is typical for both clients and servers to maintain a cache of the results of previous (and recent) name resolutions
- Caching is key to a name service's performance and assists in maintaining the availability of both the name service and other services despite name server crashes
- Caching by client name resolvers is particularly successful because naming data are changed relatively rarely

Case Study - The DNS

- The Domain Name System (DNS) is the main naming database used across the Internet (and is defined in RFC 1034)
- The original Internet naming scheme -- *a centrally maintained list that was downloaded to all sites each day at midnight* -- did not scale, did not allow local administrators to manage their part of the network and only matched IP names to IP addresses (when generality was required)

DNS Observations

- In principle, any type of object can be named, and the DNS architecture gives scope for a variety of implementations
- Millions of names are bound by the Internet DNS
- Any name can be resolved by any client
- Hierarchical partitioning of the name database, as well as replication and caching have allow the DNS to grow to support the current Internet

Domain Names

- The Internet DNS namespace is partitioned both organizationally and according to geography
- The "generic domains" are listed at the `http://www.iana.org` web-site
- As well as the generic domains, each country has its own TLD (.ie, .uk, .us, .fr, and so on)

DNS Queries

- The Internet DNS is primarily used for simple host name resolution and for looking up electronic mail hosts
- Reverse resolution is also possible - given an IP address, return the "canonical" domain name associated with it
- Host information can also be provided (machine type, operating system used) but, such information can be used maliciously
- Well known service addresses can also be provided via the DNS (for example: which machine runs your e-mail server?)

DNS Resource Records/Types

Record type	*Meaning*	*Main contents*
A	A computer address	IP number
NS	An authoritative name server	Domain name for server
CNAME	The canonical name for an alias	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)	Domain name
HINFO	Host information	Machine architecture and operating system
MX	Mail exchange	List of <*preference, host*> pairs
TXT	Text string	Arbitrary text

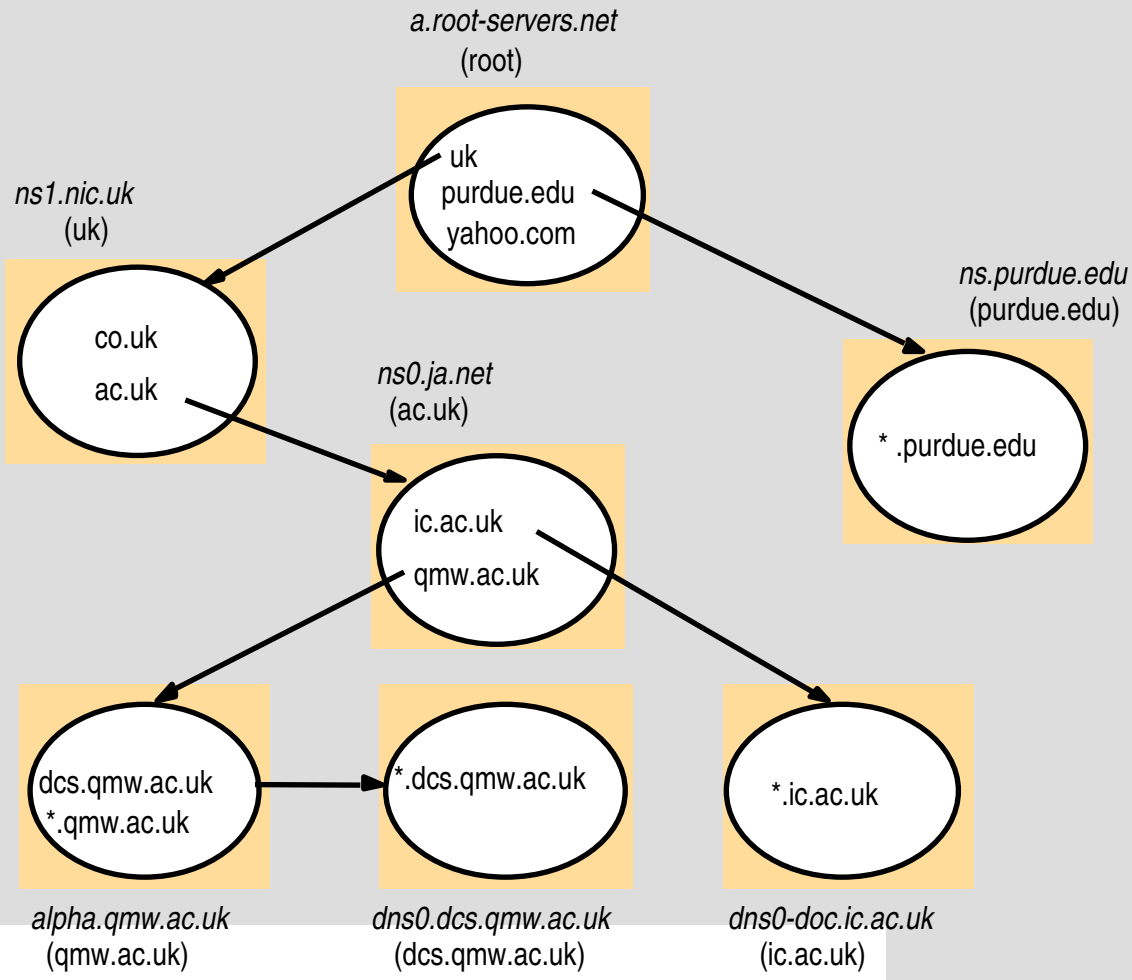
DNS Name Servers

- The problems of scale are treated by a combination of partitioning the naming database and replicating/caching parts of it close to the “points of need”
- The DNS database is distributed across a logical network of servers
- In order that naming data are available even when a single server fails, the DNS architecture specifies that each zone must be replicated authoritatively in at least two servers
- Primary (master) server - reads from the local master file
- Secondary servers - download the data from the master on a regular basis

Example DNS Name Servers

Note: Name server names are in italics, and the corresponding domains are in parentheses.

Arrows denote name server entries



Just How Busy are Root Servers?

Back in 1998, it was shown that some root servers needed to be able to respond to (or serve) about 1000 queries per second

Navigation and Query Processing

- A DNS client is called a resolver
- A simple request-reply protocol is used, typically using UDP packets on the Internet
- The resolver can be configured to contact a list of initial name servers in order of preference in case one or more are unavailable

DNS Iterations and Recursions

- The DNS architecture allows for recursive navigation as well as iterative
- The client resolver specifies which type of navigation is required when contacting a name server
- However, name servers are not bound to implement recursive navigation
- The DNS protocol allows multiple queries to be packed into the same request message, and for name servers correspondingly to send multiple replies in their response messages

Example DNS Resource Records

domain name	*time to live*	*class*	*type*	*value*
	1D	IN	NS	dns0
	1D	IN	NS	dns1
	1D	IN	NS	cancer.ucs.ed.ac.uk
	1D	IN	MX	1 mail1.qmul.ac.uk
	1D	IN	MX	2 mail2.qmul.ac.uk

domain name	*time to live*	*class*	*type*	*value*
www	1D	IN	CNAME	apricot
apricot	1D	IN	A	138.37.88.248

dcs	1D	IN	NS	dns0.dcs
dns0.dcs	1D	IN	A	138.37.88.249
dcs	1D	IN	NS	dns1.dcs
dns1.dcs	1D	IN	A	138.37.94.248
dcs	1D	IN	NS	cancer.ucs.ed.ac.uk

An Example DNS Implementation

- The *Berkeley Internet Name Domain* (BIND) is a popular implementation of the server, and is typically called "named"
- BIND allows for three categories of servers: primary, secondary and caching-only
- Caching-only servers read in from a configuration file sufficient names and addresses of authoritative servers to resolve any name
- Thereafter, they only store this data and data that they learn by resolving names for clients (that is, there's no persistent data)

Discussing DNS

- DNS achieves relatively short average response times for lookups, thanks to partitioning, replicating and caching of named data
- DNS allows naming data to become inconsistent - state data is tolerable
- The DNS does not address itself to how the staleness of addresses is detected
- Not designed to be the only name service in the Internet - it can coexist with Sun's NIS and Microsoft's ADS

DNS Challenges

- Its rigidity with respect to changes in the structure of the namespace
- Lack of the ability to customize the namespace to suit local needs
- Problems interfacing with DHCP