NAMING in DISTRIBUTED SYSTEMS

References [Col94], [Gos91], [Tan95]

NAMING SERVICE in DISTRIBUTED SYSTEMS

NAMES, ADDRESSES and ROUTES

- **Components of name service**
- **Objectives of the name service**
- **Properties of a name service**
- **Names**
- **User names**
- **System names**
- **Routes**
- **Addresses and mapping**

NAMING FACILITY

- **Structural components**
- **Functional components**
- **Name distribution**
- **Name resolution**

NAMING SERVICE in DISTRIBUTED SYSTEMS

NAMING SERVICE = facility which enables clients to manage objects, it provides the following services to distributed system:

1. to assign names to objects, and subsequently,

2. to use these names to refer those objects,

3. to provide information about those objects

4. to locate objects given only their names ,

5. to discover how to access those objects

where

object = any entity in a distributed system that requires a name

for example processes, mailboxes, nodes, I/O devices, files

- naming service affects the following entities and services in a distributed system:

1. sharing and replication of objects and resources

2. inclusion and relocation of objects and resources

3. distributed operating system

4. application view of object and resource access

- naming service enables other services to be identified and accessed i.e. it is "master" service for other servers and clients,

- users access the name server in order to access indirectly to all other services,

- name servers use a hierarchy of identifiers in order to manage objects

- problems of the name service

1. names for communication is not enough because communicating processes execute on different nodes, it requires the knowledge of their current location

2. choosing names and their structures for all objects

3. mapping between different levels of identifiers

- naming service in distributed systems contains the following basic components:

1. name

2. address

3. routes

4. ports

NAMES, ADDRESSES and ROUTES

Components of name service

name /identifier/ **= string of symbols that identifies an object i.e. it specifies what a process seeks or what it wants to access**

- names are chosen by users of distributed systems

- types of names:

1. user name

- human readable text string

2. system name

- binary numbers or digital identifiers

3. ports

-

- names are mapped onto addresses

address **= specifies where this object is i.e. object's address is the name of the object it is bound to**

- addresses refer directly to objects and are assigned by the system

- addresses are mapped onto routes

route **= defines how to get to the object**

Objectives of the name service

the naming facility

1. should be independent of the connectivity and topology of the systems, and the current location of the object handled by the naming facility

2. should be a global space of objects

3. should have at least two levels: system-level and user-level; and there should be mapping between them

4. generates unique identifiers that are structured in order to provide distributed name management

5. should minimise messages and delays generated by them to initialise and map identifiers

6. should support relocation of objects to provide high performance

7. should support the of multiple copies of the same object and user should not aware that multiple copies of an object is in use

8. should provide not only one-to-one mapping, but also a one-to-many mapping

9. should allow many different objects to share the same identifier and should support broadcast and multicast (group) addressing,

Properties of a name service

1. generality

- naming must name entities in different applications and different environment

2. multiple definitions

- naming should handle different names of the same entity in different environment

3. distributability

- naming database may be distributed amongst the networks nodes

4. user-friendliness

-

5. application oriented

- identifiers are used for a wide variety of purposes for example: locating, referencing, scheduling

6. mapping

- identifiers exist in different forms at all levels of a distributed system that require mapping between different levels of identifiers

7. appearance

- identifiers appear in different forms and there are some identifiers which are unique within a global context, others are unique only within a local context

8. designation of objects

- there are several methods how to designate objects for example: by address, by content with value, by source, by route

Names

name **denotes or identifies the object to which it is bound**

selects a particular object from among a set of objects

Attributes of name

1. types of names:

- ambiguous/unambiguous name

- unambiguous name refers to at most one object i.e. the same name cannot be used by different clients to refer to different objects

- ambiguity implies one-to-many relationship between names and objects

- unique/non-unique name

- unique name represents only one object in a distributed system

- several non-unique names may identify the same object

- group/individual name

- group name denotes a defined set of objects

- individual name identifies a single object

- global/local name

- global name is interpreted in a consistent manner by all clients and all services

- global name covers the distributed system and is valid for a long time

- absolute/relative name

- absolute name

- relative name

2. structures of names

- name space (or name domain) is a set of names complying with a given naming convention

- names are associated with some context where a context is an environment in which a name is valid,

- in a distributed system context represents a partitioning of the name space

- primitive name

- name that does not have internal structure but identifies unambiguously a particular object

-partitioned/structured name

- a structured name that identifies an object as a series of primitive name for example: a domain, a subdomain, a sub-subdomain

-descriptive name

- is a list of attributes that are true for exactly one object where a partitioned name is a particular form of descriptive names that has a well-defined structure

3. names in temporal domain

-static name

- is a name that permanently denotes the same object

-dynamic name

- is a name assigned to an object for only a limited period of time

User names

user names = character/textual string names because users cannot cope

efficiently binary numbers i.e. with system names

components of user names

- syntactic (hierarchy)

- explicit attributes

- implicit attributes (environment)

types of user names

1. attribute-based user names

- set of attributes used by a client to describe an object (attributed-based name)

- a name server maintains a database of attributes for each of the collection of objects, and computes the set of objects described by the attributes, when it is given an attributes-based name

2. hierarchical user names

- there are three levels of naming: character-string names, object identifiers and entity identifiers

- the naming facility assigns character-string names to objects that are interpreted in a given context

- each server manages its own context and character-string names are partitioned by type using prefixes

System names

system names = digital identifiers of a fixed length, and of known form in order to support system calls

Unique computer-oriented identifiers

- large integers or bit strings that could be unstructured or structured

properties of (unstructured) unique identifiers:

location independence

- internal name of an object is separated from its location in the distributed system

absolute name

- unique identifier does not require any relocation when an object migrates

uniformity

- unique identifier allows the uniform naming of all objects

composite objects

- unique identifier allows the construction of objects - composite objects -which refer to other objects

object typing

- unique identifier allows typing of objects

creation of unique identifiers

a. hierarchical concatenation

- first, an unique identifier is created for each identification domain secondly, it is concatenated to the unique identifier for that domain

b. concatenation

- the ID of the generating node is concatenated with the value of the real-time clock

c. uniform approach

- there is a standard and uniform global identifier form and space for all resources

- global identifiers are partitioned and local identifiers are binded to global identifiers either permanently or temporally

obtaining unique local identifiers

- the local identifier generator (server) defines an unique local identifier of an object taking into account the local domain

- the local identifier generator binds the unique local identifier to its local domain taking into account its implementation

- when an object is relocated it gets a new local identifier in the new local domain to correspond to the new context

- when an object has several copies each copy has an unique local identifier within its domain

obtaining server identifiers

- the remote interprocess communication has to have identifiers for all processes in order to perform message delivery

- the interprocess communication has to map a name from the global address context to a specific local address context

- having source and destination ports, and a message the interprocess communication finds a route for the message from the source port (global identifier) to the destination port (local identifier)

locating objects

- binding identifies the location of the server and of the path between the object and the server,

- firstly, binding finds a network address to which that service is assigned

- secondly, binding finds a route from the process requesting the service to the location where the service is provided

lost objects and objects with different versions

- a lost object exists in a distributed system but it is non-accessible because no reference to its exists

- several versions of objects may exist in a distributed syst- em, users are interested only in the latest version but the same unique identifier cannot name two different objects even if they are just two different versions

<u>Protection-oriented identifiers</u>

1. capability-based identifiers

capability = unforgeable ticket specifying one or more permission mode which allow their holders to access a named object

names objects, protects them and defines operations that may be performed on these objects

is a trusted object identifier which specifies an unique object

gives a client an authority to access the relevant object and also allows objects to be shared i.e. all clients which have capabilities to a given object, can share this object

protects the object assigned to it by additional information contained in the identifier

types of capabilities:

tagged capability

- tag is a string of a few bits added to each memory cell, codifying the type of the object stored in that cell

partitioned capability

- capability is stored in special segments separately from data in order to preserve the integrity of the capabilities

sparse capability

- sparse capabilities are not distinguished from data, they are bit of strings of a defined length

encrypted signature capabilities

- a valid capability consists of two strings, an object identifier and right information

- the valid capability is extended by adding a signature string which is computed from the contents of the main part, this extension provides sparseness of capability

- the three basic strings are encrypted by a kernel key because a user can deduce the signature function from the basic strings

- the capability is decrypted by the kernel using the secret key

Routes

route **is a list of name representing the path from source to destination to designate end-to-end communication between a source and a destination**

allows to reach destination objects using addresses

is a mapping that corresponds destination node/process to source node/process

Properties of routing

1. correctness

2. simplicity

3. robustness

4. stability

5. fairness

6. optimality

Attributes of routing

1. the place where the routing decision is made

source routing

- routing decision is made at the source node

hop-by-hop routing

- only the next part of the routing is made at each step

broadcast routing

- each message is broadcast through a network and every object receives it, but only that one that recognises its name pick up the message from the network

2. the place where the routing function is performed

centralised routing

- routing tables are computed periodically based on information gathered from other nodes

partially distributed routing

- routing tables are based on local information where nodes do not exchange routing information

distributed routing

- routing tables are based on global information changed between nodes

3. time constant of the routing information

static routing

- routing tables are not changed for a long time

dynamic routing

- routing tables are constantly are updated

Addresses and mapping

address **is an intermediate form between the name and route which is associated with the location of objects**

allows to locate a communication entity (or an object)

contrary to a name, it is attached to a communication object but not to an entity

the name directly denotes the entity, the address refers to the entity through binding

mapping **the name can be routed by two different approaches**

1. routing by name

- a name is directly mapped into a route using a mapping function

2. routing by address

- firstly, a name is mapped into an address by a directory function, secondly, the address is mapped into a route by a routing function

NAMING FACILITY

- there are two types of components: functional and structural components

Structural components

name server

- provides an instance of the name service which enables clients to name objects and share information about these objects

- should provide the following services:

- communication

- storing names and database management

- maintaining replications

- name resolution

- name service operations

agents

- location of name servers should be transparent to clients of the name service

- transparancy can be achieved through name agents which act between name servers and their clients

- name agents maintains a detailed knowledge of the name space and of existing name servers

Functional components

communication

- when a naming facility is distributed in a distributed system, communication is required among name servers and the name agents/clients, and between servers

database management

- the name service database stores and manages objects' attributes

- the database is maintained in virtual memory, record updates are performed incrementally on a disk in a log

name management

- there are two major tasks of name management: name distribution and name resolution

Name distribution

name distribution = assigns authority for managing the name space to various name servers

- distributed name service should manage the name space in a distributed system by a set of name servers using basic set of operations

- distributed name service divides the name space among the processes in the distributed system

database and replication

- the name service database, which is distributed and replicated among the name servers, stores information about all objects

- each name server stores a set of attribute tuples that contains object's name, its attribute type and value, where attribute tuples are maintained in special database objects

- more than one server may manage a given attribute in a distributed system

- replicating information about an object's name among name servers increases reliability in a distributed system

- an object with several name servers has its attributes replicated among those servers

- the replication requires the participation of several computers in name service operations in order to read and update replicated database

tuples

distributed operations

name resolution

- defines the name server for every named object using configuration data that contains a list of the name servers for every object and is stored in the name server database as attribute tuples of the special type called authority

locating a name server

- a name server is also an object that may be on any node in a distributed system thus, it requires locating a name server before accessing it

- to locate name servers only needs to find a single name server because name servers should be able to locate each other

- to perform database operation it is necessary to select the authorities using a database of server addresses stored at every server

accessing attribute tuples

- after locating and selecting a name server for a named object, it is possible to perform database operation

Name resolution

name resolution = locates the attributes of a specific object given by its name
i.e. selection of name server for a this object

types of name resolution:

1. structure-oriented name resolution

- locates the set of name servers for a named object based on the structure of the name space

2. structure-free name resolution

- locates the set of name server for a given object without relying on the

structure of the name space