Module 8

Naming

Naming system

- A DCS supports several types of objects such as processes, files, I/O devices, mail boxes and processor nodes
- The Naming facility of DCS Enables users and programs to assign character-string names to objects and subsequently use these names to refer to these objects
- The Locating facility Maps an object’s name to the object’s location in a Distributed System
- Together they form a **naming system** that provides the users with an abstraction of an object that hides details of how & where an object is located in the network
- Naming system plays an important role in achieving the goal of location transparency in a DCS
- It facilitates transparent migration and replication of objects and object sharing

Desirable Features of a Naming System

1. Location transparency: means that the name of an object should not reveal any hint as to the physical location of the object
   - i.e., An object's name should be independent of the physical connectivity of topology of the system or the current location of the object
2. Location independency: For performance, reliability, availability, and security reasons, DS provides the facility of object migration that allows the movement of the objects dynamically among various nodes of a system
   - Location independency means name of an object need not be changed when the object's location changes
   - A user should be able to access an object by its same name, irrespective of the node from where he or she access it (also known as user mobility)

Desirable Features of a Naming System (Cont'd)

- The requirement of location independency calls for a global naming facility .
- A location-independent naming system should support a dynamic mapping scheme so that it can map the same object name to different locations at two different instances of time
- Hence location independency is a stronger property than location transparency

3. Scalability: DCS vary in size from few nodes to many hundreds of nodes

- They are open systems and their size changes dynamically

Desirable Features of a Naming System (Cont'd)

- Hence, it is impossible to have an a priori idea about how large the set of names to be dealt with is liable to get
- Hence the naming system should be capable of adapting to this dynamically changing size of the DCS that leads to the change in the size of the name space
- i.e., A change in the system scale should not require any change in the naming or locating mechanisms

4. Uniform naming convention: In many existing systems, different ways of naming objects, called naming conventions are used for naming different types of objects

- e.g., Filenames typically differ from user names and process names
- A good naming system should use the same naming convention across all types of objects in the system

Desirable Features of a Naming System (Cont'd)

5. Multiple user-defined name for same object: For a shared object, it is desirable that different users of the object can use their own convenient names for accessing it
   - Should support multiple user defined names to the same object
   - It should be possible for a user to change or delete his or her name for the object without affecting those of other users
6. Group naming: A naming system should support many different objects to be identified by the common group name
   - Such a facility is useful to support broadcast facility, to group objects for group communication, for conferencing and so on
7. Meaningful names: A name can be simply any character string identifying some object

Desirable Features of a Naming System (Cont'd)

- However, Users prefer meaningful names rather than jumble of character string as it is easy to remember and use
- Hence a good naming system should support at least two levels of object identifiers, one convenient for human users and one convenient for machines

8. **Performance**: The most important performance measurement of a naming system is the amount of time required to map an object's name to its attributes such as location

- In a distributed environment, this performance is dominated by the number of messages exchanged during name mapping operation
- Hence, a naming system should be efficient in the sense that the number of messages exchanged in a name mapping operation should be as minimum as possible

Desirable Features of a Naming System (Cont'd)

9. **Fault tolerance**: A naming system should be capable of tolerating, faults that occur due to failure of a communication link or a node
   - i.e., The naming system should continue functioning, may be in degraded form, in the event of these failures
   - The degradation can be in performance, functionality or both
10. **Replication transparency**: In a distributed system, the replicas of an object are created to improve performance and reliability
    - The naming system should support the use of multiple copies of the same object in a user transparent manner; i.e., user need not be aware of the existence of multiple copies of an object in use

Desirable Features of a Naming System (Cont'd)

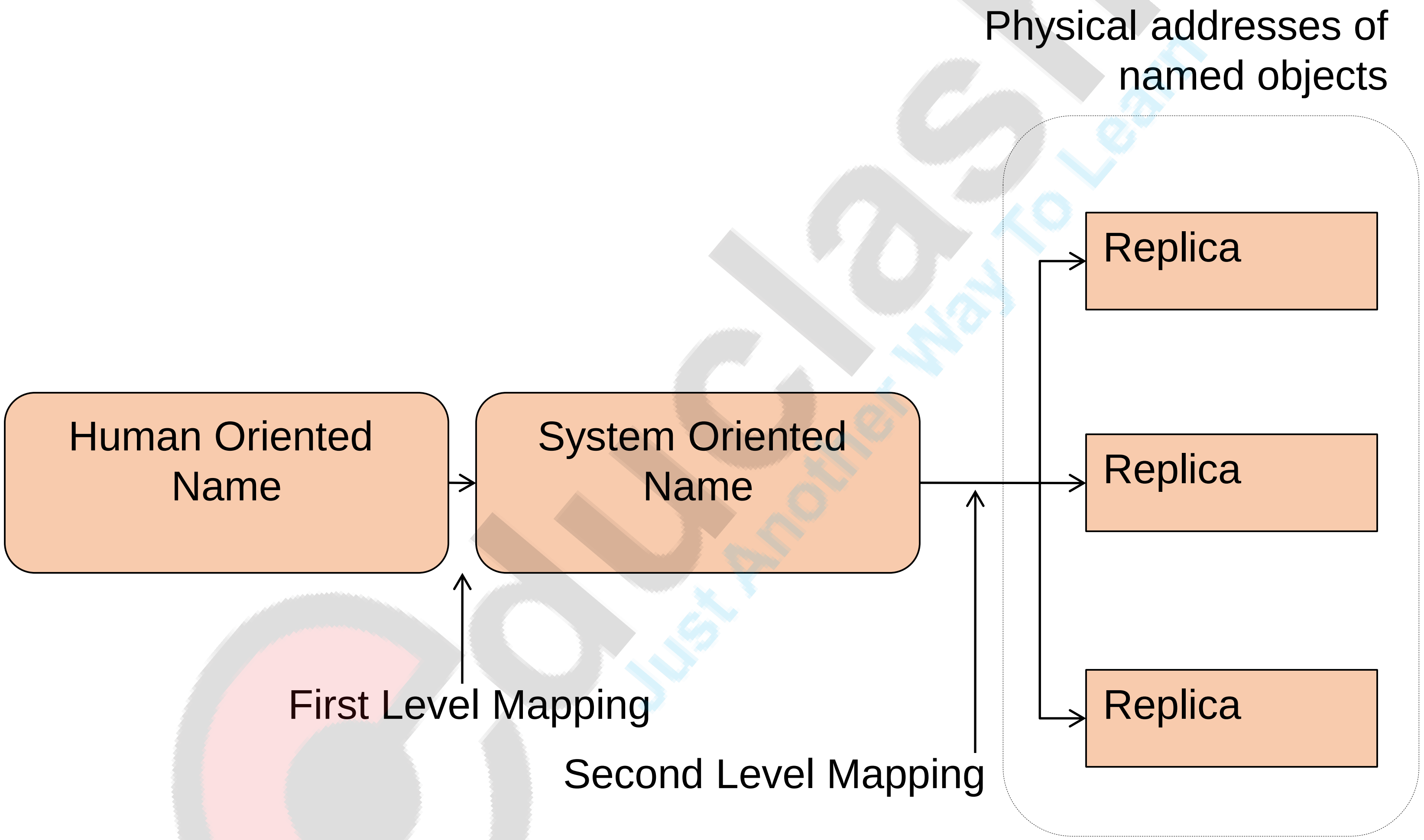
11. **Locating the nearest replica**: When a naming system supports the use of multiple copies of the same object, it is important that the object locating mechanism of the naming system should always supply the nearest replica of the desired object
    - The efficiency of the object accessing operation will be affected if the object locating mechanism does not take this point into consideration
12. **Locating all replicas**: In addition to locating the nearest replica of an object, it is also important from a reliability point of view that all replicas of a desired object be located by the object locating mechanism.
    - Another replica can be used if nearest is inaccessible due to any reason, such as, communication link failure.

Fundamental Terminologies and Concepts

- **Name (Identifier)**
  - String composed of a set of symbols chosen from finite alphabet set. A logical object that identifies a physical object to which it is bound to.

- **Human oriented names (High level names)**
  - A character string that is meaningful to its users.
  - Different users can define their own names for shared objects.
  - Independent of physical location or structure of objects.
  - Not unique for an object & variable in length.

- **System-oriented names (Low level names)**
  - Uniquely identify every object in entire system
  - Bit patterns of fixed size automatically generated by system that can be easily manipulated and stored by machines


Physical addresses of named objects
Human Oriented Name
System Oriented Name
Replica
Replica
Replica
First Level Mapping
Second Level Mapping

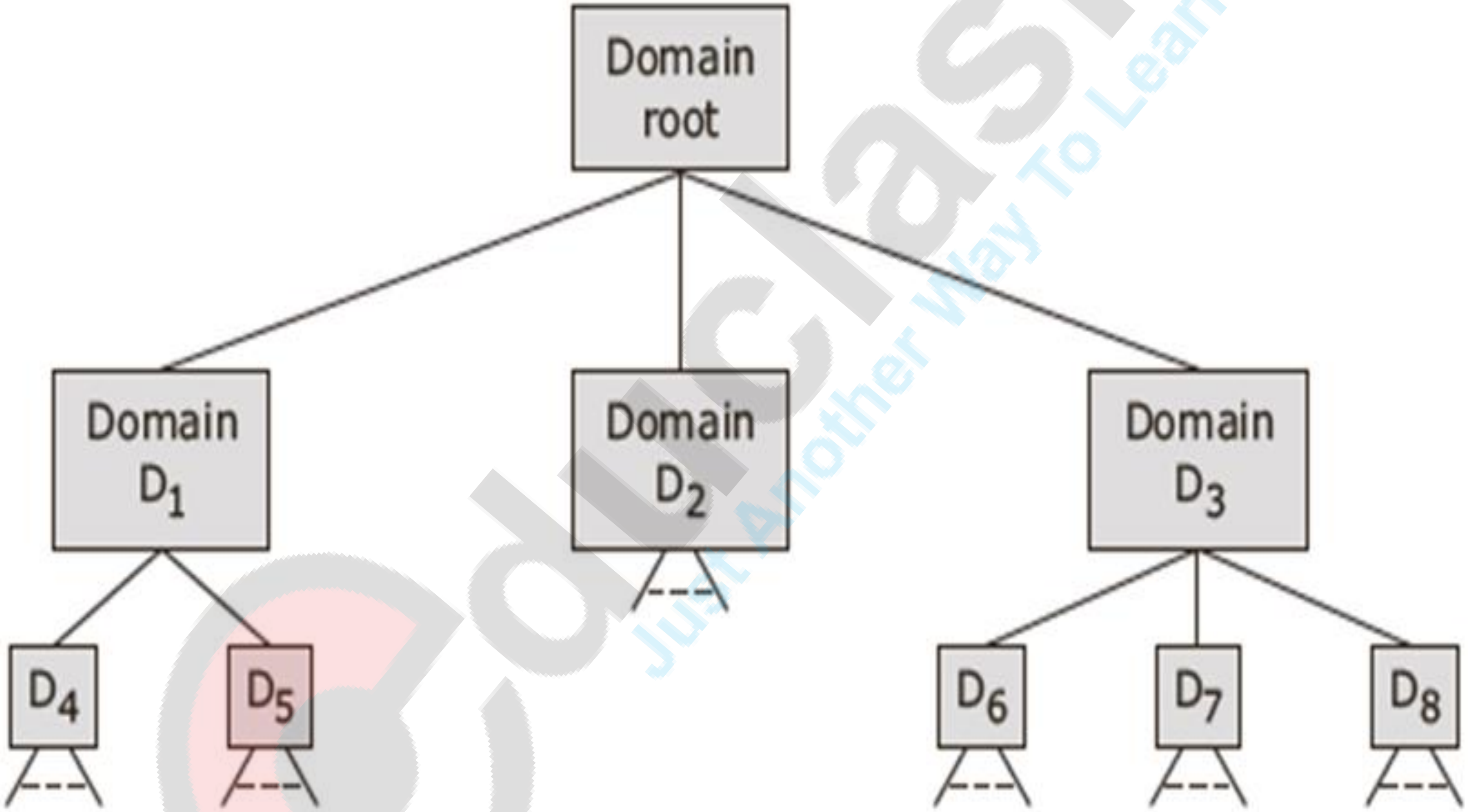
- **Name space**
  - Set of names complying with a given naming convention is said to form a name space.
  - An abstract container providing context for the names it holds and allowing disambiguation of items with same names residing in different namespaces.
  - Names in a namespace have to be unique.

- **Flat name space**
  - Names are character strings exhibiting no structure. Primitive or flat names.
  - Suitable for small namespace

- **Partitioned name space**
  - Name space is partitioned into disjoint classes (domains).
  - Names defined in a domain must be unique within that domain.
  - Name structure reflects physical or organizational associations
  - Simple / Compound names
  - Ex. Hierarchical name space – Telephone System, IP Addresses
  - Number of levels may be fixed or arbitrary in hierarchical name space

- **Name server**
  - Manages namespace
  - Process that maintains information about named objects and provide facilities that enables users to access that information.
  - Binds object name to its location
  - Several name servers present in distributed systems with each server having information of a small subset or set of objects.
  - Authoritative name server
  - Partitioned namespaces easier to manage

- **Name agent**
  - Communication between clients & name servers happens via name agents.
  - Maintains knowledge of existing name servers.
  - Private name agent
    - Works for a single client and is structured as a set of subroutines linked to a client program.
  - Shared name agent
    - Structured as a part of the operating system kernel with system calls to invoke the naming service operations.



Name servers in hierarchical namespace

- **Context**
  - The environment in which name is valid.
  - Represent division of name space along natural geographical, organizational or functional boundaries.
  - Qualified name (context, name pair) uniquely identifies an object.
  - Useful for partitioning the name space.
  - In a partitioned namespace, each domain corresponds to context of namespace.

- **Name Resolution**
  - Process of mapping an object's name to object's properties, such as its location.
  - To resolve a name, client contacts its name agent, which contacts a known name server, which may in turn contact other name servers.
  - In partitioned namespace, name resolution mechanism travels a resolution chain from one context to another until the authoritative name servers of named object are encountered.

- **Abbreviation/ Alias**
  - Users can specify their own short-form substitutes for compound qualified names called abbreviations.
  - More than one simple name may be bound to same qualified name within a given context.(synonyms)

- **Absolute name**
  - An absolute name begins at the root context of the name space tree and follows a path down to the specified object.

- **Relative names**
  - It defines a path from the current context to the specified object.

- **Generic and multicast name**
  - Support one-to-many binding.
  - Naming system allow a simple name to be bound to a set of qualified names.
    - **Generic naming facility** - Name is mapped to any one of the set of objects to which it is bound.
    - **Group or multicast naming facility** - Name is mapped to all the members of the set of objects to which it is bound.

- **Descriptive/ Attribute Based Name**
  - An object is identified by a set of attributes or properties that describe the object & uniquely identify it among the group of objects in the name space.
  - All attributes together refer to a single object.
  - Ex. User = ?, Creation date = ?, file type = ?

- **Source – Routing Name**
  - Mirrors the structure of the underlying physical network. Eg, host1/host2/host3/file 1

System-Oriented Names

- Large integer or bit strings.
- Unique identifier as do not change during their lifetime and are never reused.
- All System-Oriented Names are of same size. Allows naming of all objects uniformly.
- Suitable for effective handling by machine as they are uniform and normally shorter than human-oriented names.
- Used for security related purpose as they are hard to guess.
- Automatically generated by the system.
- Can be structured or unstructured.

A single field of large integers		Unstructured name
Node identifier	Local unique identifier	Structured name

Generating System-Oriented Names

- Centralized approach
  - Used for unstructured name.
  - Standard uniform global identifier generated for each object in the system by a centralized global unique identifier generator.
  - Implementation easy but poor efficiency and reliability.
  - Single global unique identifier generator bottleneck for system.

- Distributed approach
  - Used for structured identifiers.
  - Uses hierarchical concatenation strategy – object identifier is domain identifier & unique local identifier (node-id, timestamp), (server-ID, server specific unique identifier).
  - Disadvantages
    - Non-uniform global identifiers in heterogeneous environment.
    - Node or server boundaries explicitly visible.
- Generating Unique Identifiers In Event of Crashes
  - Using a clock that operates across failures
  - Using two or more levels of storage

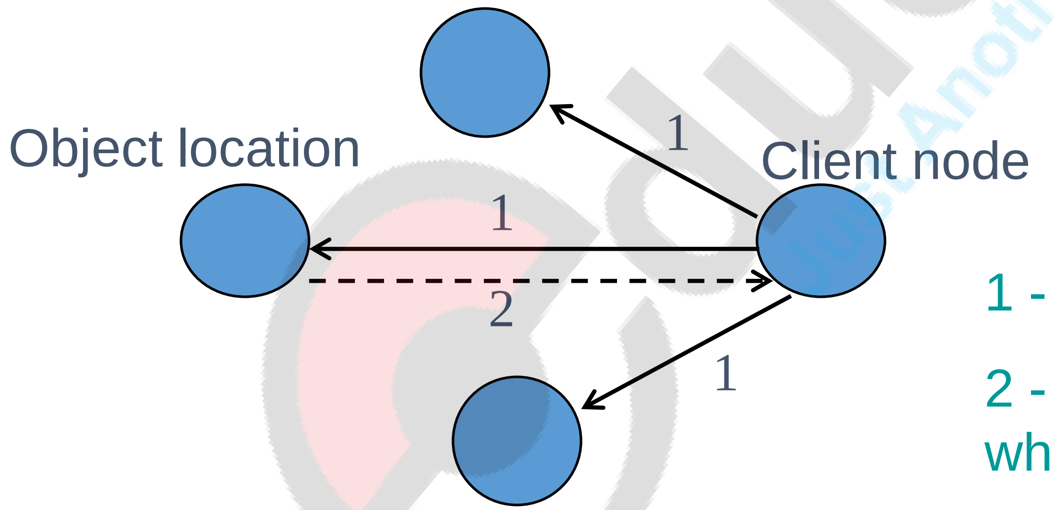
Object Locating Mechanism

- Process of mapping an object's system oriented unique identifier (UID) to the replica locations of the object.

- Several object locating mechanism are:
  - Broadcasting
  - Expanding ring broadcast
  - Encoding location of objects within its UID
  - Searching creator node first and then broadcasting
  - Using forward location pointer
  - Using hint cache and broadcasting

Broadcasting

- An object is located by broadcasting a request for the object from a client node.
- Request processed by all nodes & nodes currently having object reply back.
- Supplies all replica locations of target object.
- Have high degree of reliability but poor efficiency & scalability.
- Suitable for small networks.

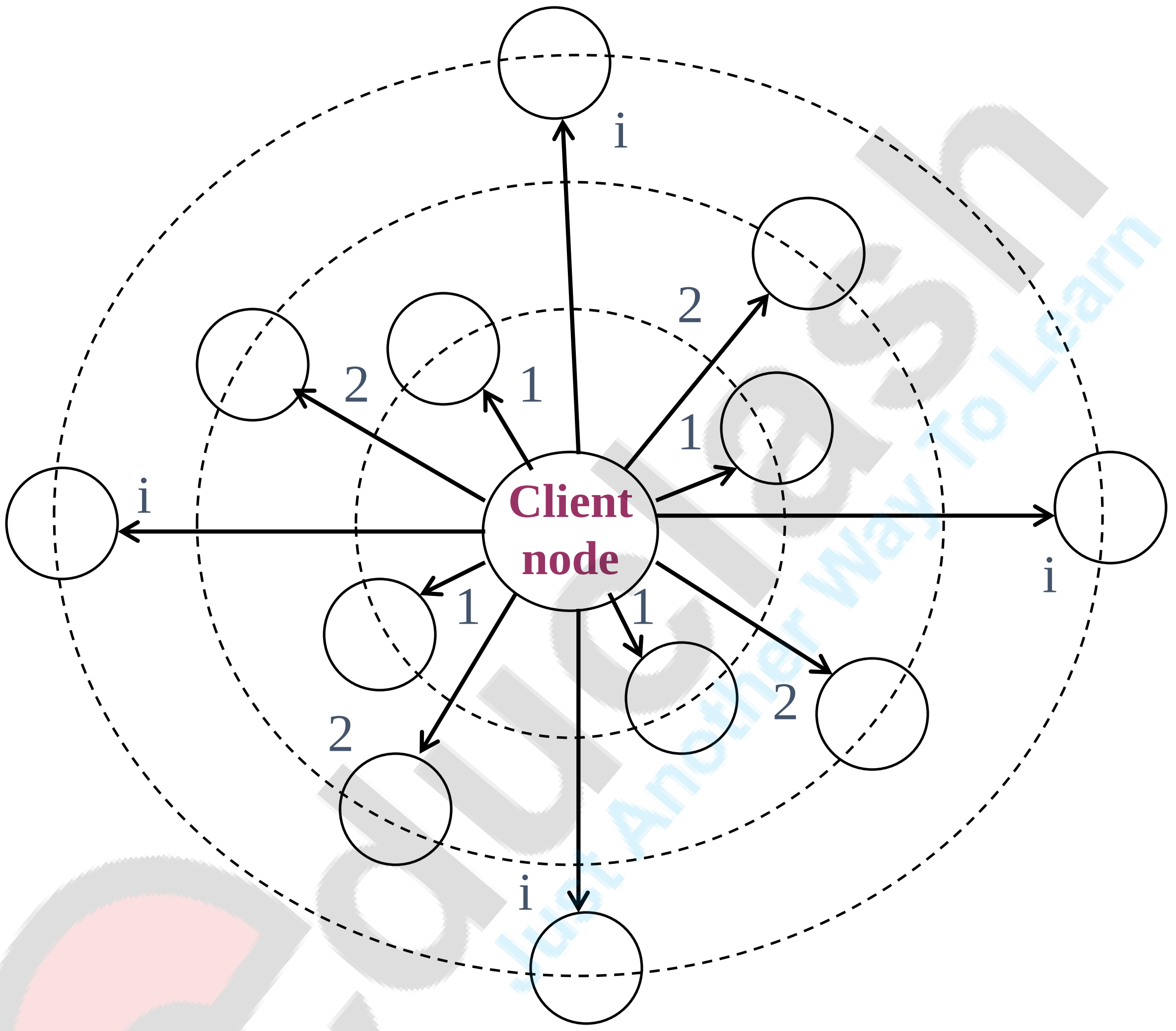


1 - Broadcast request msg

2 - Reply from node on which the object is located

Expanding Ring Broadcast

- Employed in an internetwork that consists of LANs connected by gateways.
- Increasingly distant LANs are systematically searched.
- Hop - A hop corresponds to number of gateways in between
- Ring - A ring is a set of LANs a certain distance away from processor. $Ring_i[A]$ is set of LANs 'i' hop away
- Beginning with i =0, request message is broadcast to all LAN's in $Ring_i[A]$. If response received, search ends or else i is incremented by 1.
- It does not necessarily supply all replica locations, but supplies nearest replica location.



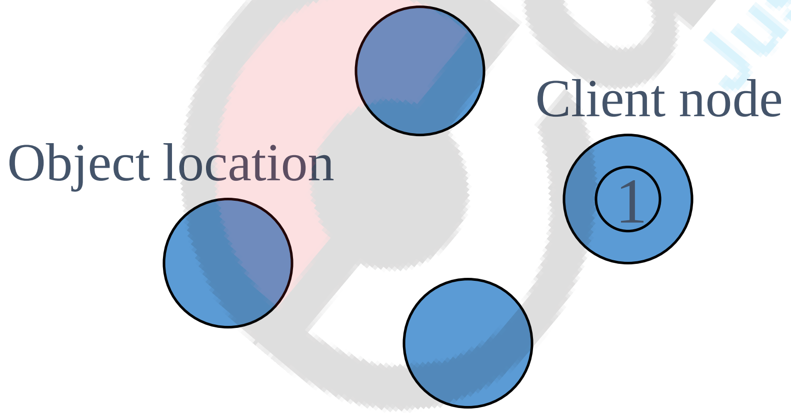
1 - Searching nodes at 0 hop distance.

2 - Searching nodes at 1 hop distance if search of 0 hop fails.

i - Searching nodes at i hops distance if searches up to i -1 hops fail.

Encoding Location of Object within its UID

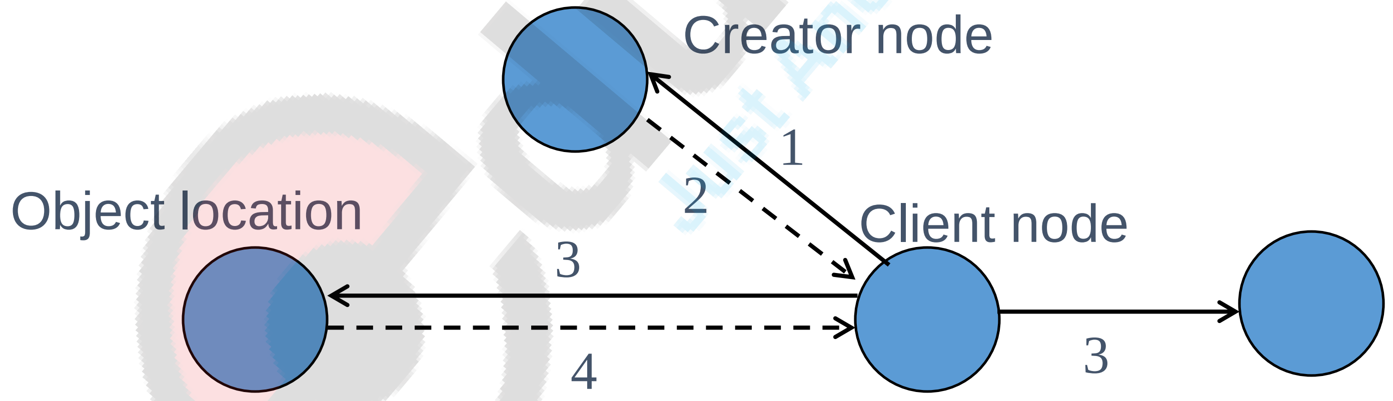
- Uses structured object identifier.
- Extracts the corresponding object's location from its UID by examining the appropriate field of the structured ID.
- Object migration & replication not allowed.



1 - Extracting object's location from its UID. No message exchange with any other node is required for locating the object.

Searching Creator Node First and then Broadcasting

- Supports object migration
- Creator node information is extracted from the UID and request is sent to that node.
- If object no longer resides on its creator node, object is located by broadcasting the request from the client node.

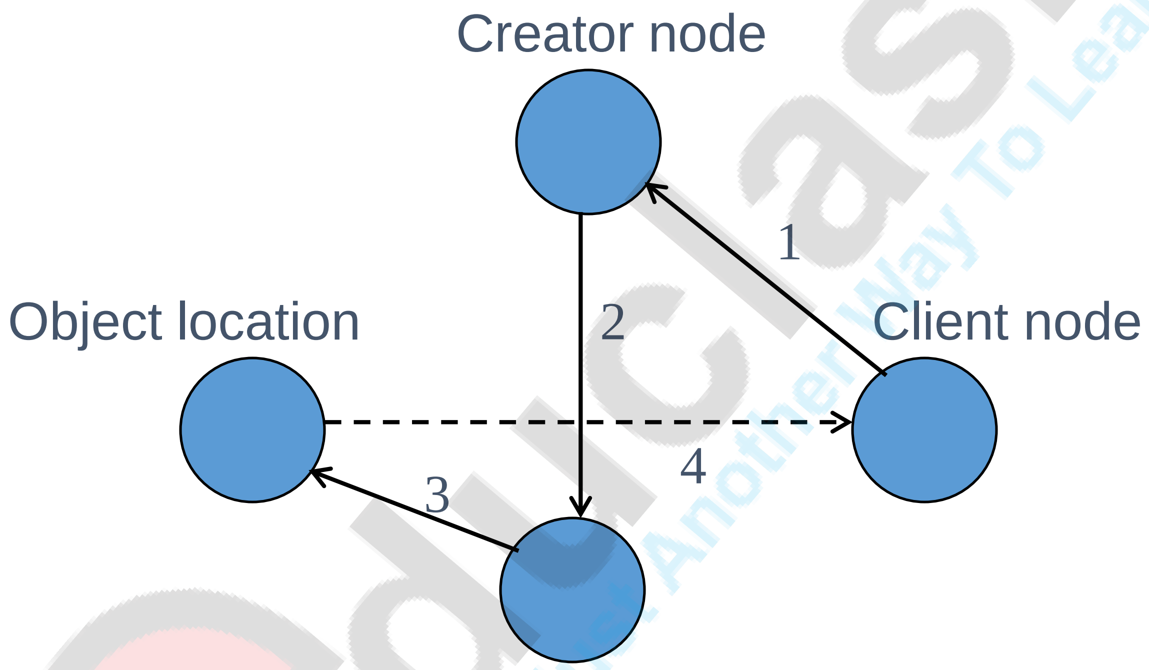


1 - Searching creator node; 2 – Negative reply from creator node

3 - Broadcast request; 4 - Reply from object's current location.

Using Forward Location Pointer

- Whenever an object is migrated from one node to another, a forward location pointer is left at its previous node.
- To locate a node system first contacts creator node and then simply follows forward pointers.
- Avoids broadcasting.
- Chain of pointers can be long or an intermediate pointer might be lost.

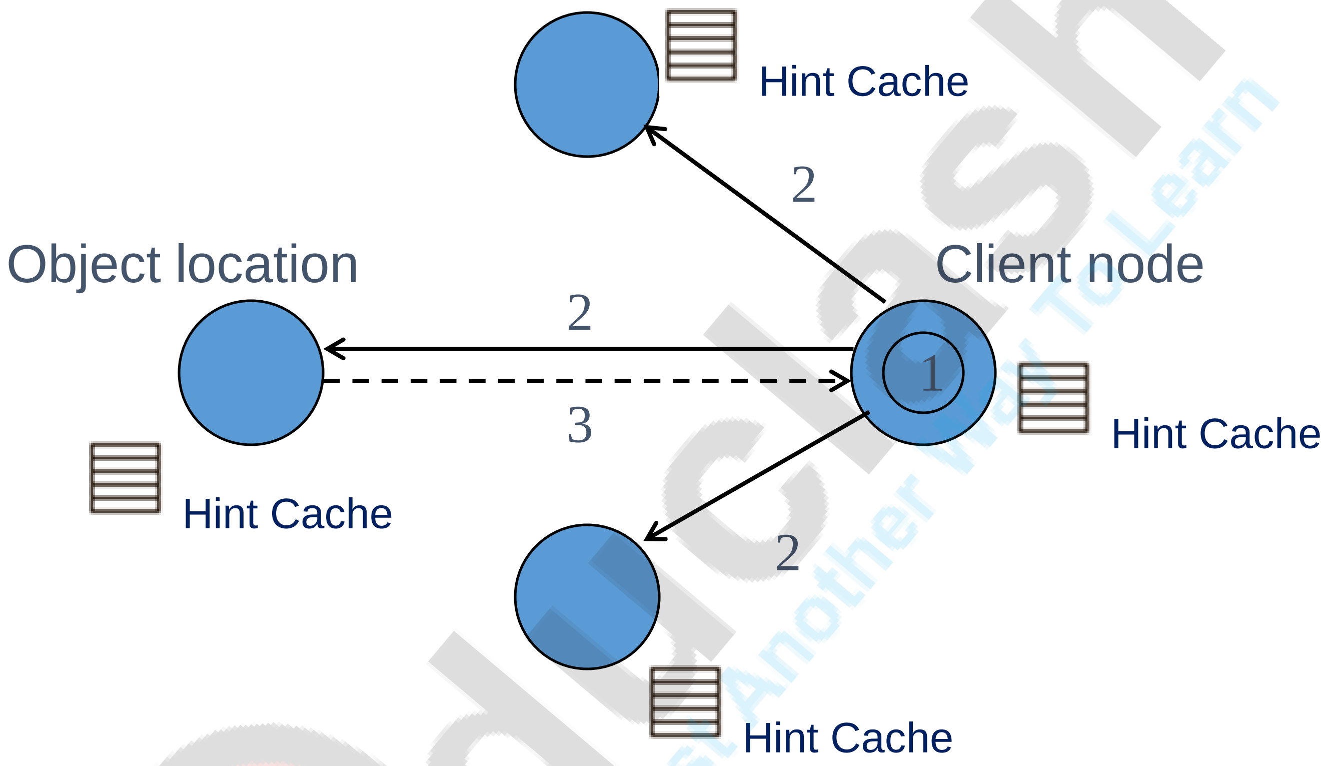


1, 2, 3 – Path of message forwarding

4 - Reply from the node on which the object is located

Using Hint Cache & Broadcasting

- Cache is maintained on each node that store (UID, last known location) pairs of recently referenced remote objects.
- Local cache is searched for the UID; If found, location information is extracted from the cache.
- Request is then send to the node specified in the extracted location information.
- If object no longer resides at that node, negative reply is send back.
- Location is then searched through broadcasting and is recorded in the client's node cache.
- Efficient if high degree of locality is exhibited.
- Supports object migration but broadcasting involved.



1 - Searching of local cache

2 - Broadcast request message

3 - Reply from the node on which the object is located.

Human Oriented Names

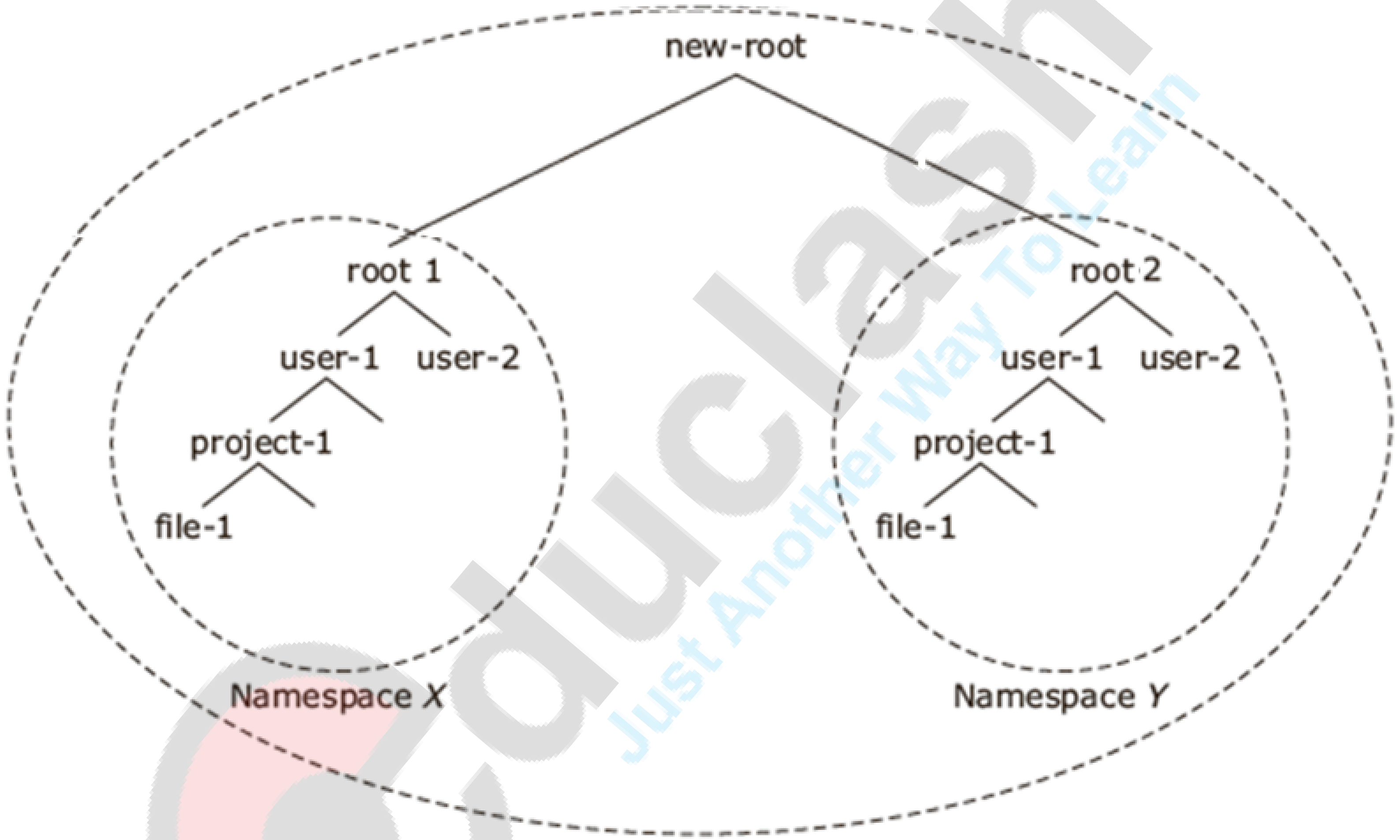
- Character strings that are meaningful to their users.
- Defined and used by users.
- Aliasing can be done.
- Names are variable in length.
- Not unique in space or time.
- Hierarchically partitioned namespace are commonly used for human-oriented objects names.

Constant vs. Arbitrary Number of Levels

- Hierarchically partitioned namespace can have:

- Constant number of levels
  - Simpler & easy to implement
  - Difficult to decide number of levels
  - Algorithms for name manipulation have to be changed if new levels are added later

- Arbitrary number of levels
  - Expansion easy by combining independently existing name spaces into a single name space by creating new root
  - Name that was unambiguous within its old namespace, is unambiguous in new namespace also
  - No need to change algorithms for name manipulation

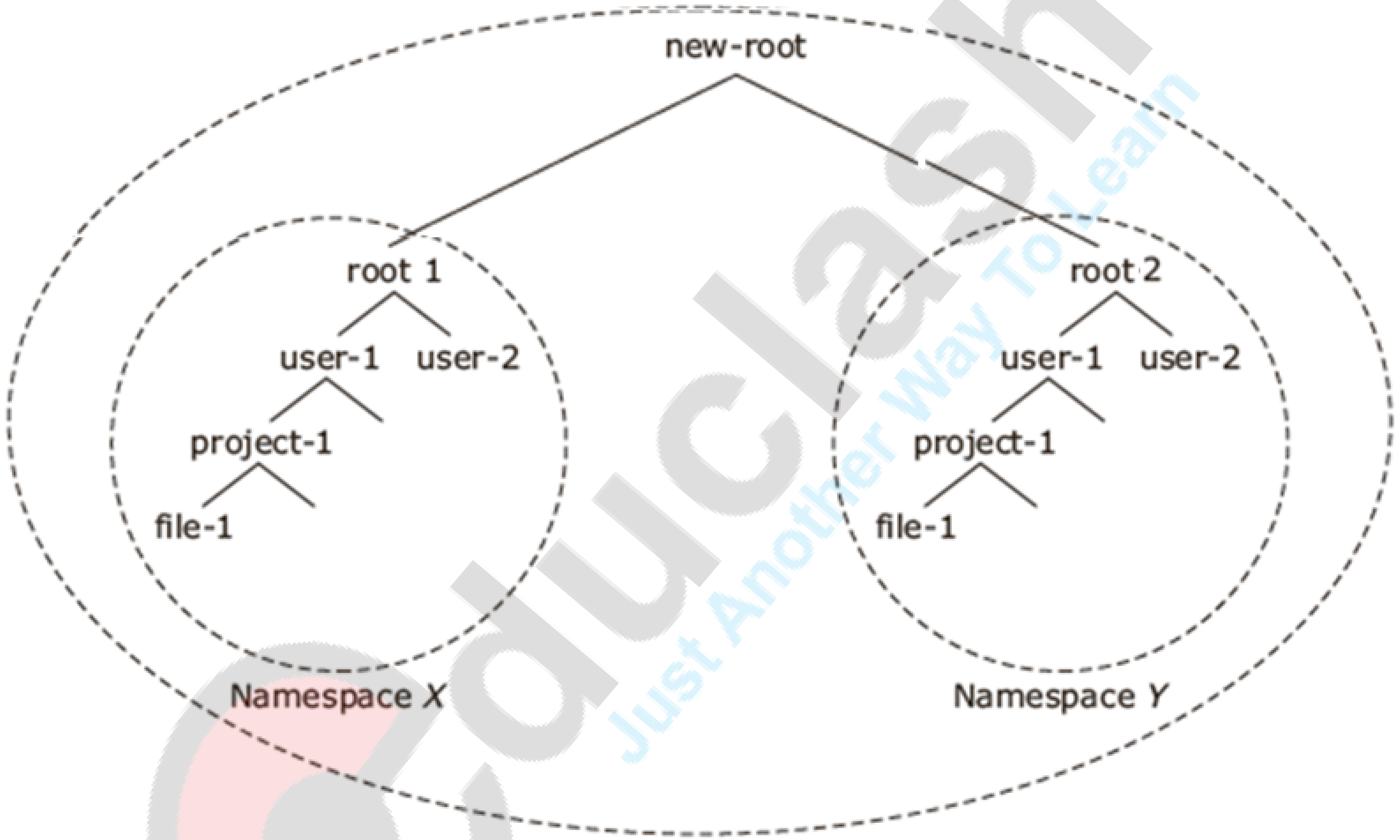


The new namespace
Combining namespaces

Human Oriented Hierarchical Naming Schemes

- Combining an object’s local name with its host name
    - Namespace comprises of several isolated namespaces
    - Unique identifier of object – host name, local name
    - Neither location transparent nor location independent
    - Object’s absolute name changes on migration

- Interlinking isolated name spaces into a single namespace
    - Isolated namespaces are joined together to form a single namespace.
    - Clients should be allowed to use names that were valid in independent namespaces without need to update them to match new namespace structure.

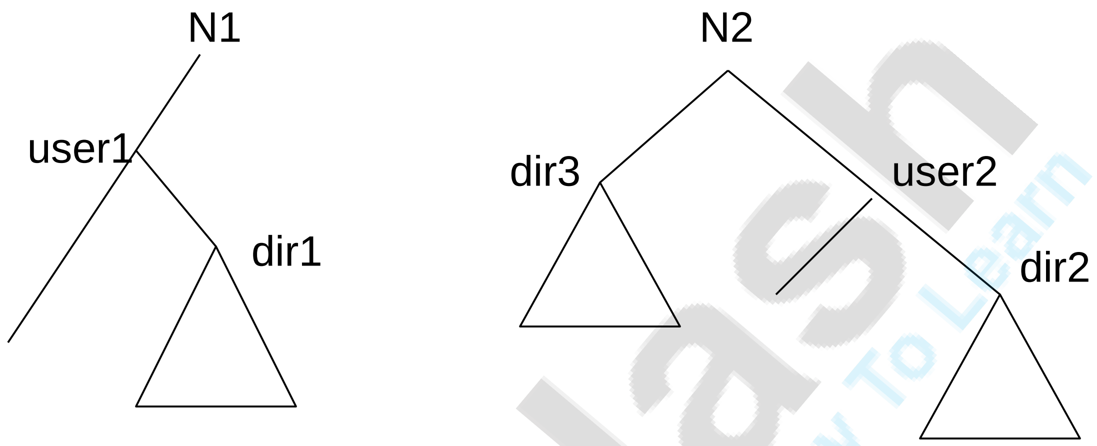


Combining namespaces

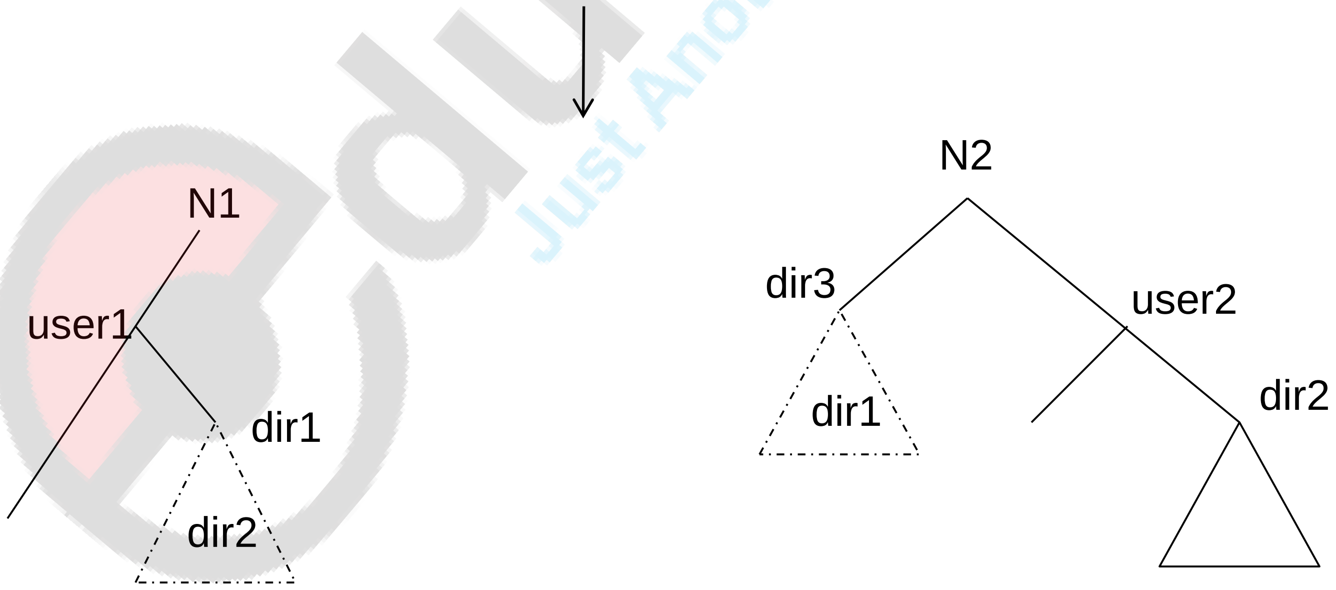
- Sharing remote name spaces on explicit request
  - Users can attach a complete remote namespace or part of it to their local namespace tree. Ex. mount protocol
  - The node that performs mount operation is called client & node whose directory is mounted is called server.
  - All directories below an exported directory automatically become accessible to client as a link is set up between mount point of client's local namespace & exported directory of server's namespace.
  - Manual / Static/ Auto mounting

- Advantages
  - Provides flexibility to attach only necessary portions of remote namespace.

- Disadvantages
  - Require management of each node's mount table
  - Directories exported by failed nodes become unavailable on its client nodes



After Mounting



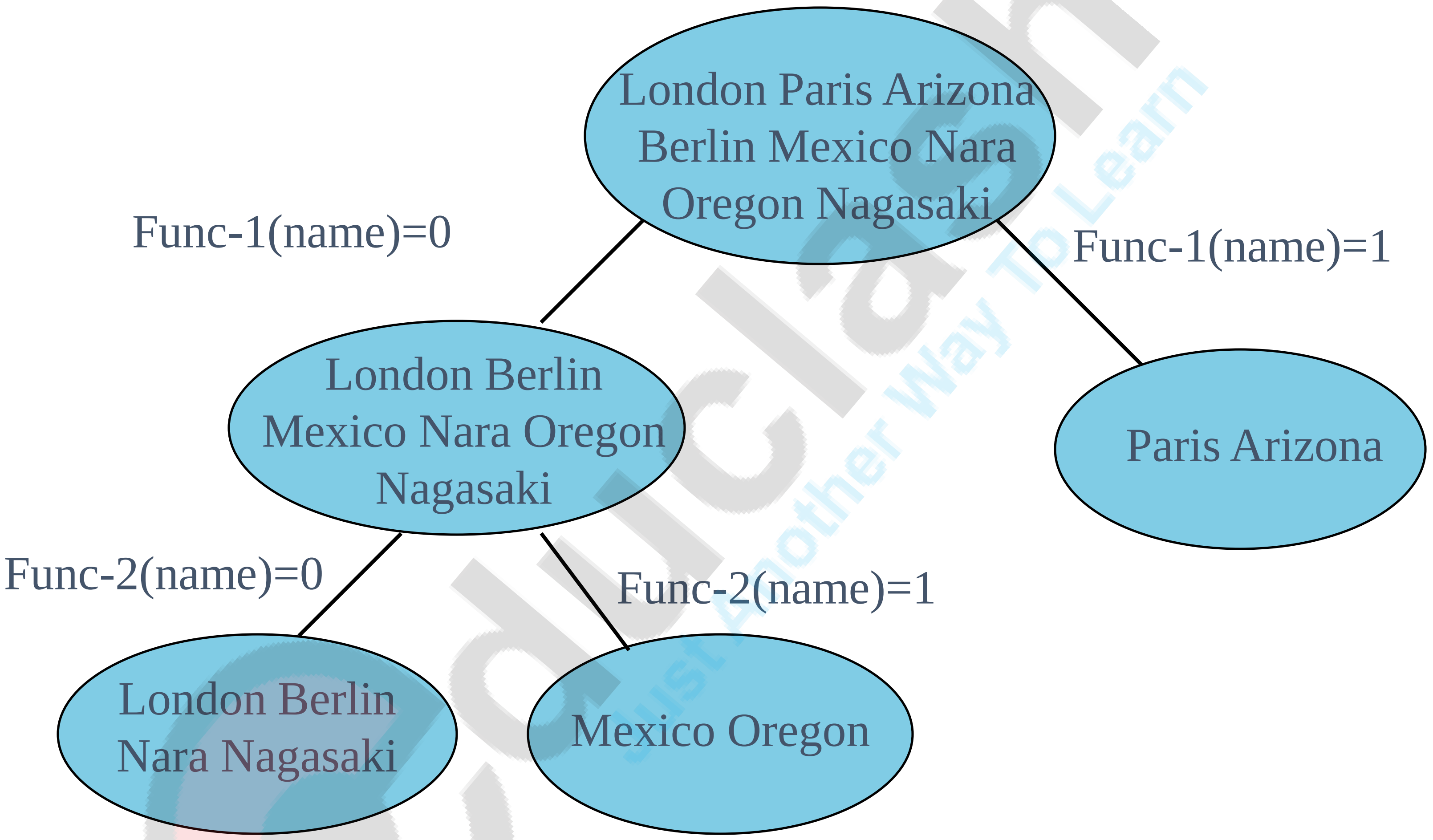
- A single global name space
  - Same namespace is visible to all users & object’s absolute name is always the same irrespective of its location.
  - Location independent and location transparent

Partitioning Namespace into Contexts

- Naming information is decentralized and replicated at many name servers each storing a copy of a portion of the complete naming information.
- How to decompose naming information database to be distributed among name servers?
- A namespace is partitioned into contexts by using clustering condition.
- A clustering condition is an expression that, when applied to a name, returns either a true or false value, depending on whether given name exists in the context.

Algorithmic Clustering

- Names are clustered according to the value that result from applying a function to them.
- Supports structure free name distribution i.e. does not depend on number or order of components etc.
- Can be used for flat name spaces also.
- Difficult to devise proper clustering functions.

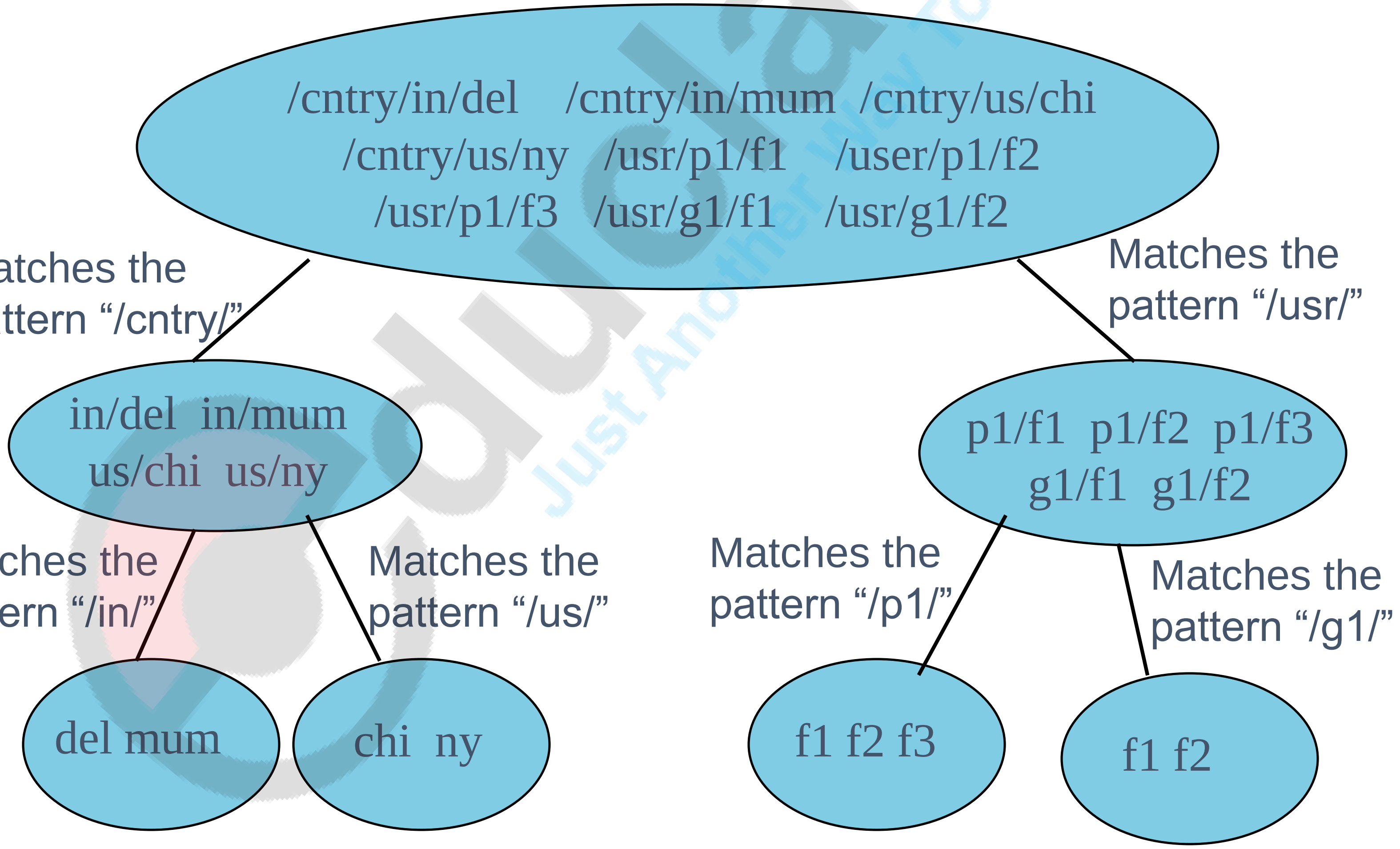

London Paris Arizona Berlin Mexico Nara Oregon Nagasaki
Func-1(name)=0
Func-1(name)=1
London Berlin Mexico Nara Oregon Nagasaki
Paris Arizona
Func-2(name)=0
Func-2(name)=1
London Berlin Nara Nagasaki
Mexico Oregon

Func-1(name) : if (total characters (name) = even)

Func-2(name) : if (total vowels (name) = even)

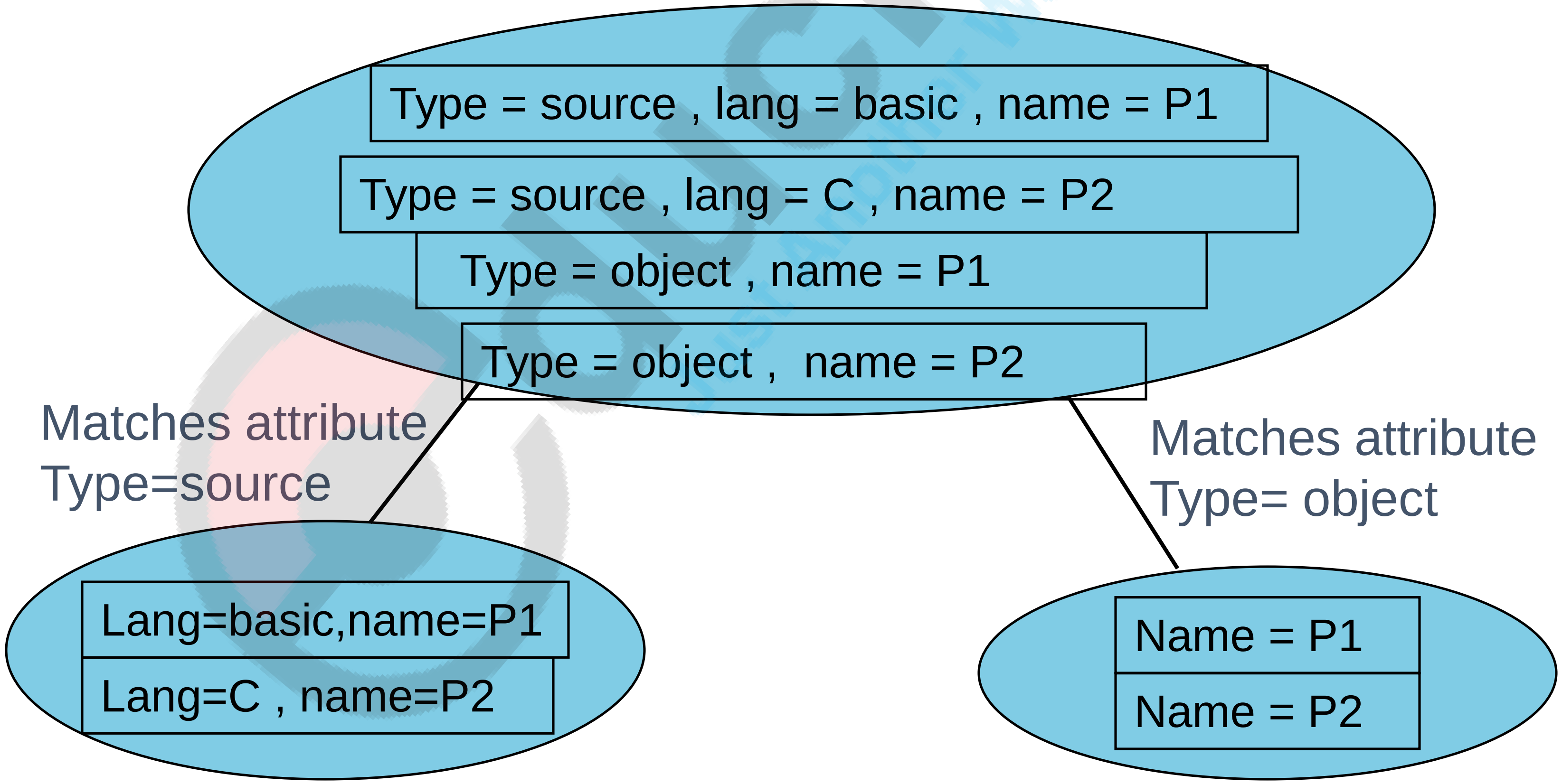
Syntactic Clustering

- Uses pattern matching techniques
- Can be used for structured or unstructured names


/cntry/in/del /cntry/in/mum /cntry/us/chi
/cntry/us/ny /usr/p1/f1 /user/p1/f2
/usr/p1/f3 /usr/g1/f1 /usr/g1/f2
Matches the pattern "/cntry/"
Matches the pattern "/usr/"
in/del in/mum
us/chi us/ny
p1/f1 p1/f2 p1/f3
g1/f1 g1/f2
Matches the pattern "/in/"
Matches the pattern "/us/"
Matches the pattern "/p1/"
Matches the pattern "/g1/"
del mum
chi ny
f1 f2 f3
f1 f2

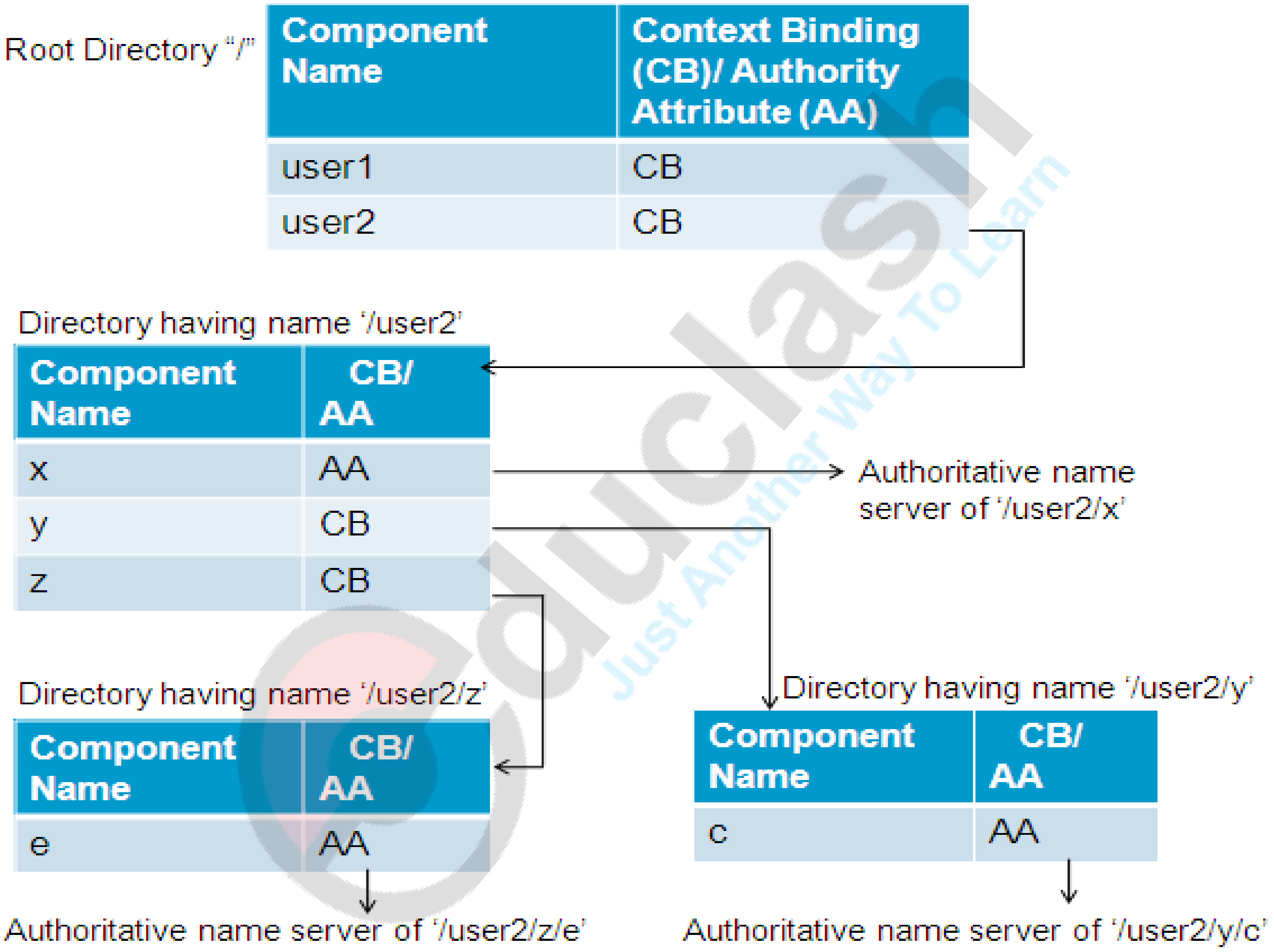
Attribute Clustering

- Names are grouped based on the attributes possessed by the names.
- All names having the same attribute (type, value) pair are placed in the same partition.


Type = source , lang = basic , name = P1
Type = source , lang = C , name = P2
Type = object , name = P1
Type = object , name = P2
Matches attribute Type=source
Matches attribute Type= object
Lang=basic,name=P1
Lang=C , name=P2
Name = P1
Name = P2

Context Binding

- Procedure - Based Strategy
  - Context binding done by procedure, which, when executed, supplies information about the next context to be consulted for the named object. Ex. Syntactic clustering.
  - Changing of context binding will require changes in clustering procedures.
- Table - Based Strategy
  - Used for implementing context bindings in hierarchical tree structured name spaces.
  - Each context is a table having component name and context binding information or authority attribute information.



Root Directory "/"

Component Name	Context Binding (CB)/ Authority Attribute (AA)
user1	CB
user2	CB

Directory having name '/user2'

Component Name	CB/ AA
x	AA
y	CB
z	CB

Authoritative name server of '/user2/x'

Directory having name '/user2/z'

Component Name	CB/ AA
e	AA

Authoritative name server of '/user2/z/e'

Directory having name '/user2/y'

Component Name	CB/ AA
c	AA

Authoritative name server of '/user2/y/c'

Distribution of Contexts and Name Resolution Mechanisms

- Name resolution is the process of mapping an object's name to the authoritative name servers of the object.

- Involves transversal of a resolution chain of contexts until the authority attribute of the named object is found.

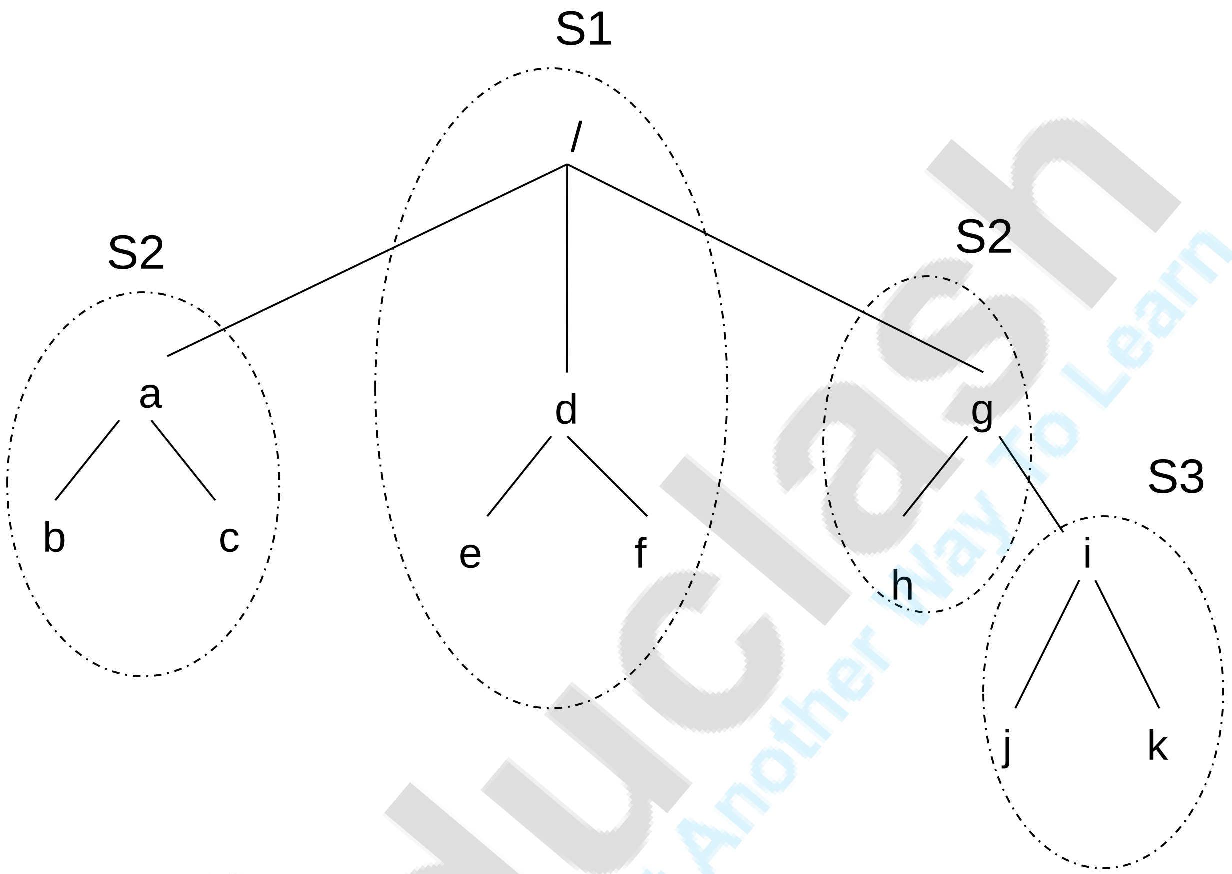
- Name resolution mechanisms:
  - Centralized approach
  - Fully replicated approach
  - Distribution based on physical structure of name space
  - Structure free distribution of contexts

- Centralized approach
  - A single name server in the entire distributed system is located at a centralized node.
  - Location of centralized server known to all nodes.
  - Simple & easy to implement
  - Not scalable & reliable

- Fully replicated approach
  - A name server is located on each node of distributed system & all contexts are replicated at every node.
  - Simple & efficient as resolution requests serviced locally.
  - Overhead in maintaining consistency.
  - Not suitable for large namespace

- Distribution based on physical structure of name space
  - Most common for hierarchical tree structured namespace
  - There are several name servers, with each server providing storage for one or more zones (domains).
  - Each client maintains a name prefix table

- Advantages
  - Matching name prefix allows a small number of table entries to cover large number of names.
  - Name prefix table helps in bypassing part of directory lookup mechanism. So performance & reliability enhances because a crash on one name server does not prevent clients from resolving names in zones on other servers.
  - On use consistency checking saves upon consistency control overheads



Name Prefix	Zone Identifier	
	Server Identifier	Specific Zone Identifier
/	S1	12
/a	S2	33
/g	S2	24
/g/i	S3	56

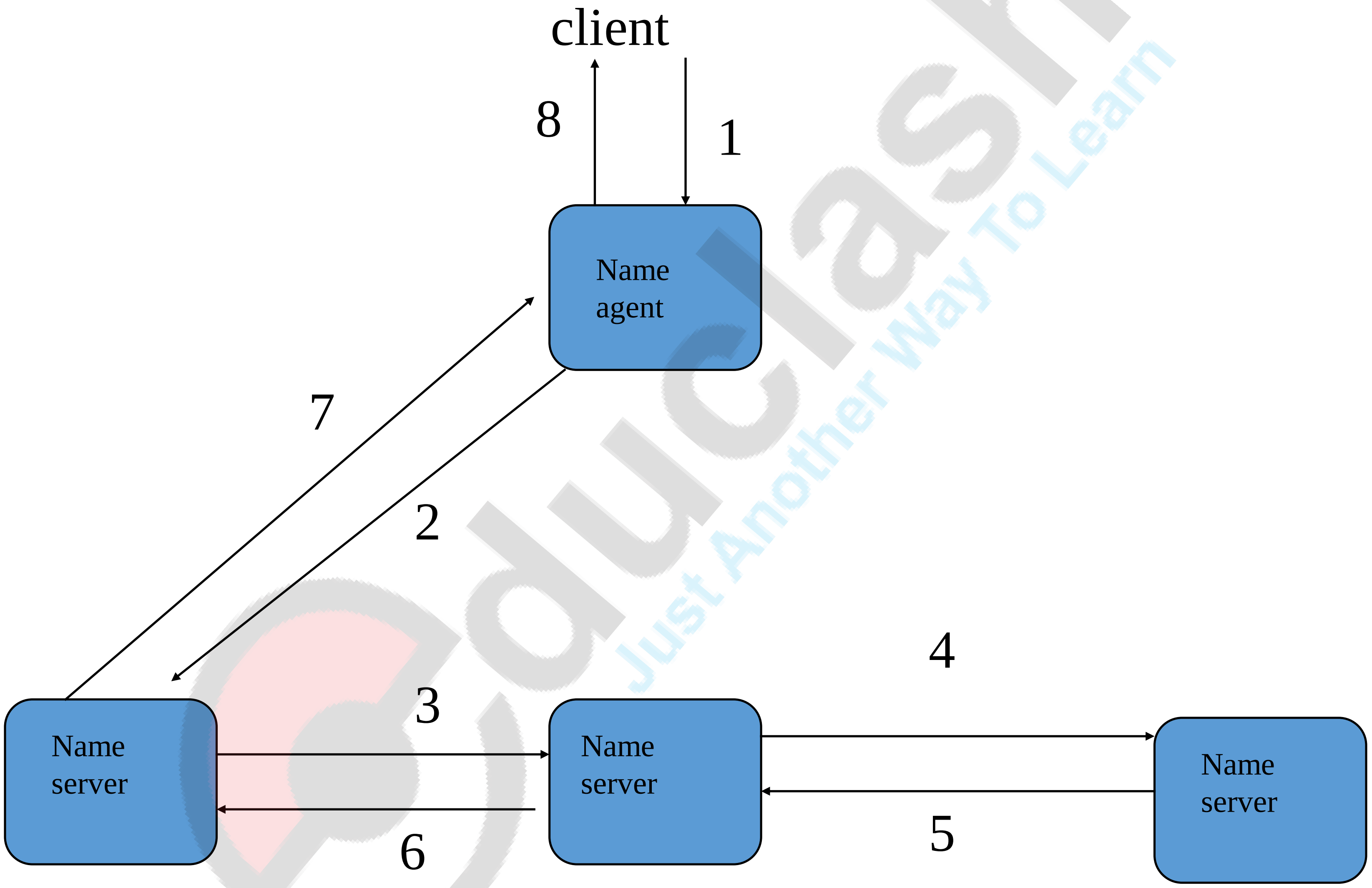
- Structure Free Distribution of Contexts
    - Name services should be able to be reconfigured if present servers become overworked or if system scale changes.
    - Reconfiguration might require changing an object's authoritative name server.
    - Use Structure Free Distribution. Context of a namespace can be freely stored/ moved at any name server independently of any other context.
    - Permits maximum flexibility.

Issues in Structure Free Distribution

- Locating Context objects during Name Resolution
  - Using metacontext
    - It contains name & authority pairs for all context objects in namespace
    - To resolve a name first metacontext is referred to obtain authority attribute of context.
    - Name is then sent for resolution to one of authoritative name servers of the context.
  - Use absolute names and start resolution at root context which is replicated at all name servers.

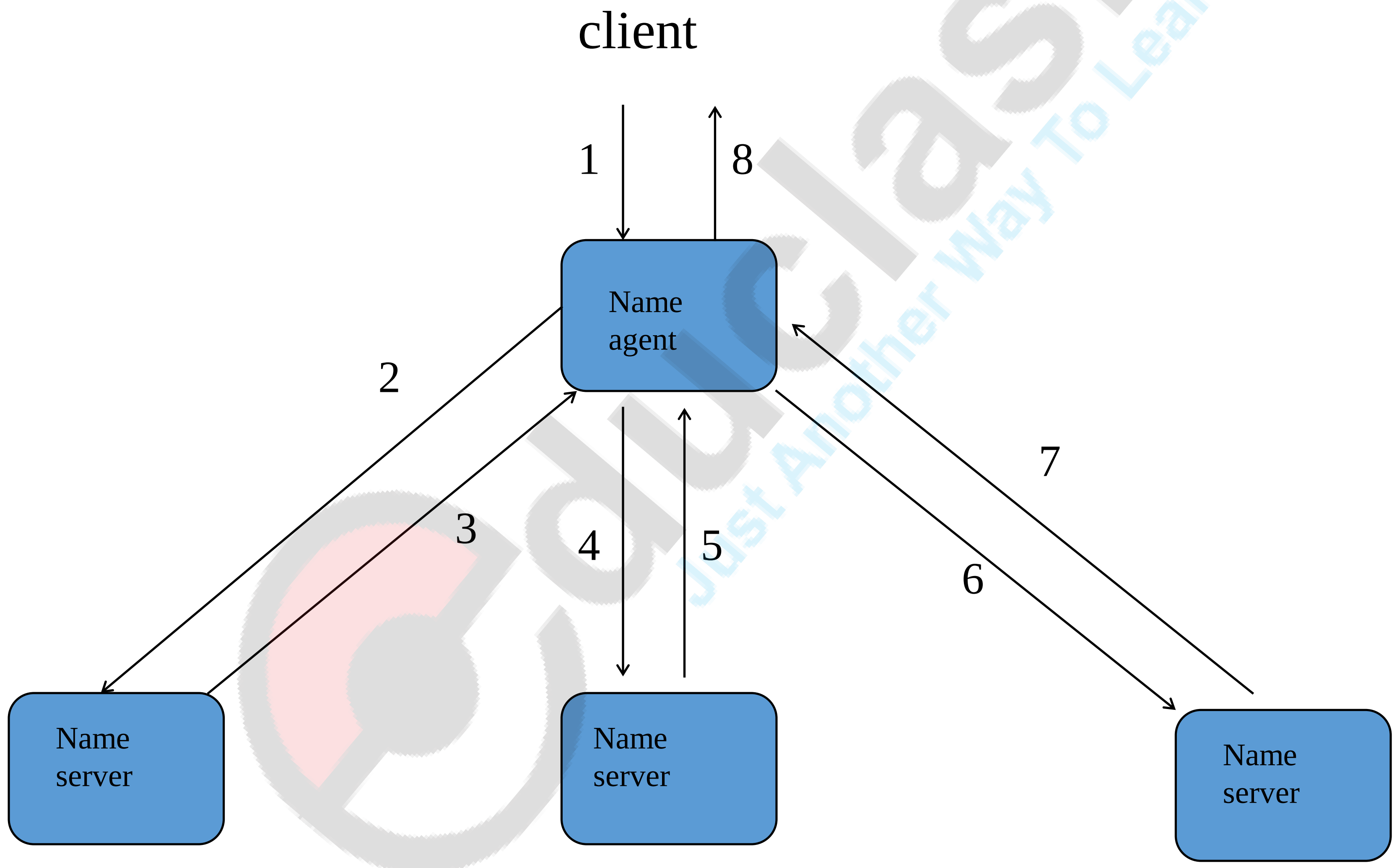
- Interacting with name servers during Name Resolution
  - Recursive
  - Iterative
  - Transitive

Recursive

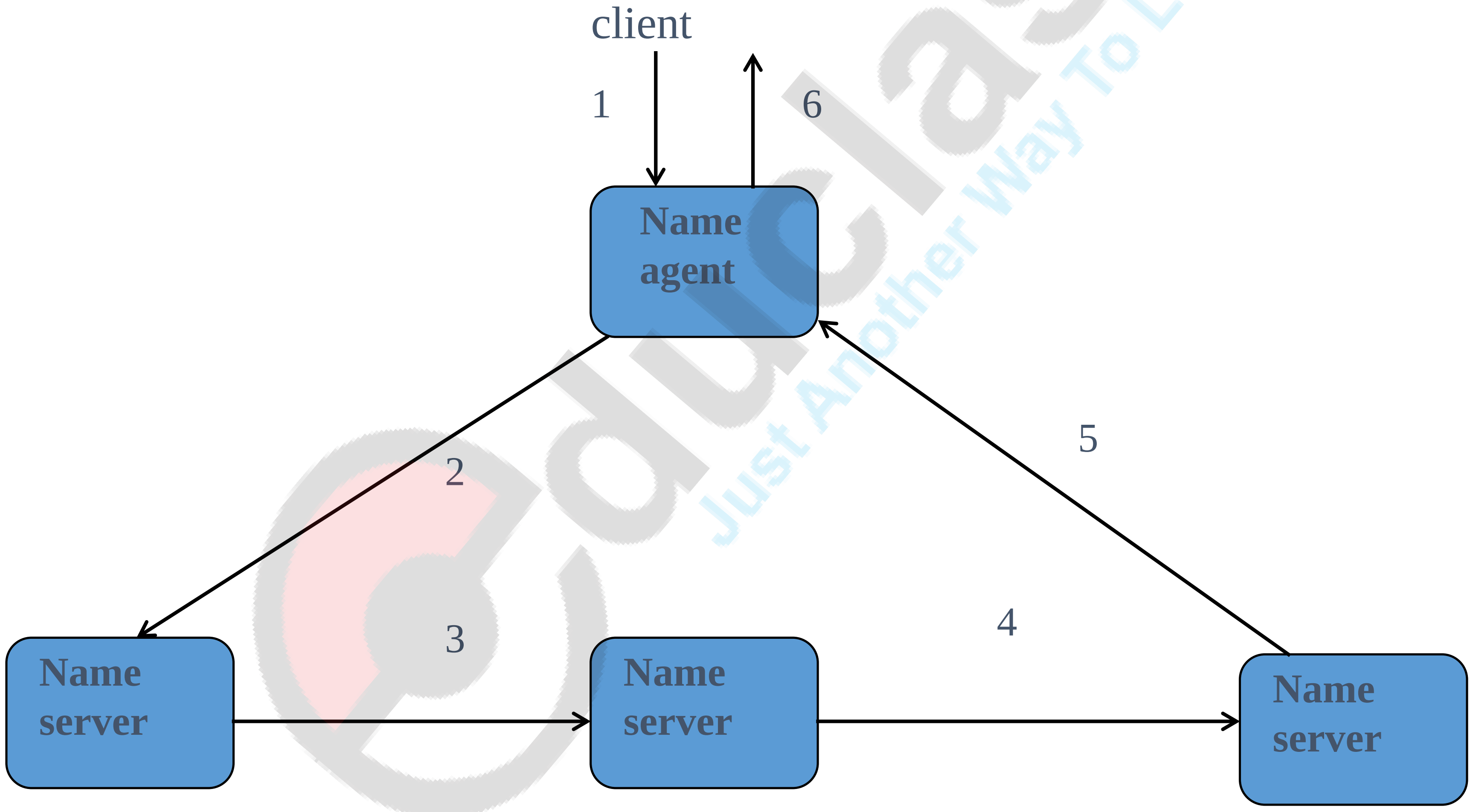


client
8
1
Name agent
7
2
Name server
3
6
Name server
4
5
Name server

Iterative


client
1
8
Name agent
2
3
4
5
6
7
Name server
Name server
Name server

Transitive


client
1
6
Name agent
2
5
3
4
Name server
Name server
Name server

Name Caches

- System overhead in name resolution operations is considerably large.
- Client caches the result of a name resolution operation for a while, rather than repeating it every time the value is needed.
- Characteristics of name service related activities:
  - High degree of locality of name lookup.
  - Slow update of name information database.
  - On-use consistency of cached information is possible.

Types of Name Caches

- Classified on basis of type of information stored in each cache entry.
- Directory cache
  - Each entry consist of a directory page.
  - All recently used directory pages that are brought to the client node during name resolution are cached for a while.
  - Entire page cached for only one useful entry. Large size cache required.
- Prefix cache
  - Used in naming systems that use the zone-based context distribution mechanism.
  - Each entry consist of a name prefix and the corresponding zone identifier.

- Full-name cache
  - Each entry consist of an object’s full pathname and the identifier and location of its authoritative name server.
  - Each entry serves as mapping information for single object.
  - Larger in size as compared to prefix caches.

Name Cache Implementation

- A cache per process
  - Separate name cache is maintained for each process in process’s own address space.
  - Fast access
  - No memory area of the OS is occupied by name caches.
  - Every new process has to create its cache from scratch.
  - Caches will have short lifetime if processes are short lived.
  - Hit ratio low due to start-up misses. Avoid by cache inheritance.
  - Naming information duplicated in several caches on same node.

- A single cache for all processes of a node
  - Single cache maintained at each node for all the processes of that node.
  - Larger in size.
  - Located in memory area of the OS, thus slower.
  - Cache information is long lived and is removed only by replacement policy.
  - Higher average hit-ratio
  - No duplication of naming information.

Multicache Consistency

- When a naming data update occurs, related name cache entries become stale and must be invalidated or updated.

- Immediate invalidate
  - All related name cache entries are immediately invalidated when a naming data update occurs. Two methods:
    - Invalidate message is broadcasted to all nodes
    - Storage node of naming data keeps a list of nodes against each data that corresponds to nodes on which data is cached. Nodes in this list are only notified about a naming update.

- On-use update
  - Client informed when it uses a stale cached data.

Naming and Security

- Naming system controls unauthorized access to both named objects & information in naming database.
- Naming access control mechanisms:
    - Object names as protection keys
    - Capabilities
    - Associating protection with name resolution path

- Object names as protection keys
  - Only the user who knows name of an object can access it.
  - Not reliable
  - Does not provide flexibility of specifying modes of access control.

- Capabilities
  - Capability (object identifier, Rights information) is a special type of object identifier that contains additional information for protection.
  - There are several capabilities for same object.
  - A client that possesses a capability of an object can access it in the modes allowed by it.
  - Capabilities are maintained by operating system & only indirectly used by the user. Thus they can not be modified by user process.

- Associating protection with name resolution path
  - Protection can be associated either with an object or with the name resolution path used to identify the object.
  - An Access Control List (ACL) is associated with an object & specifies user name & types of access allowed for each user of the object.
  - When a user requests access to an object, the operating system checks the ACL associated with that object.
  - To access an object, user must be allowed access to both directories of object's pathname & object itself.

DCE Directory Service

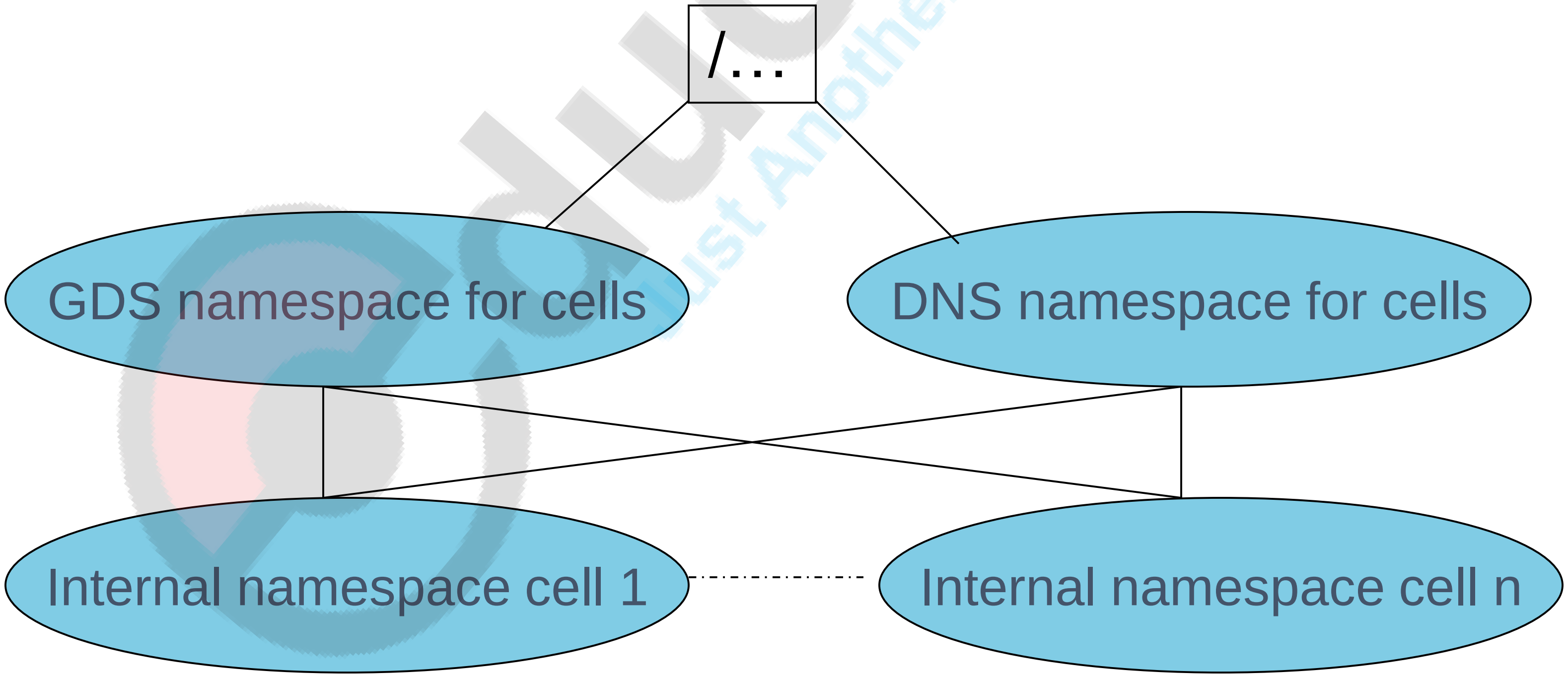
- In DCE system all users, machines, resources etc that have a common purpose & share common services are grouped into cells.

- Components of DCE Directory service:-
  - Cell Directory Service (CDS)
    - Controls naming environment used within a cell.
  - Global Directory Service (GDS)
    - Controls global naming environment outside(between) cells.

- DCE also supports standard Internet Domain Name System (DNS) for cell naming.

DCE Namespace

- Single global namespace.
- Each name is unique and has different parts
  - Prefix (Indicates whether name local to cell (/.:) or global (/…)), Cell name, Local name



- GDS namespace
  - Uses X.500 international standard for naming.
  - It uses hierarchical, attribute based naming scheme. Ex.

/country=US/OrgType=COM/OrgName=IBM/Dept=ENG/Person=Nancy/

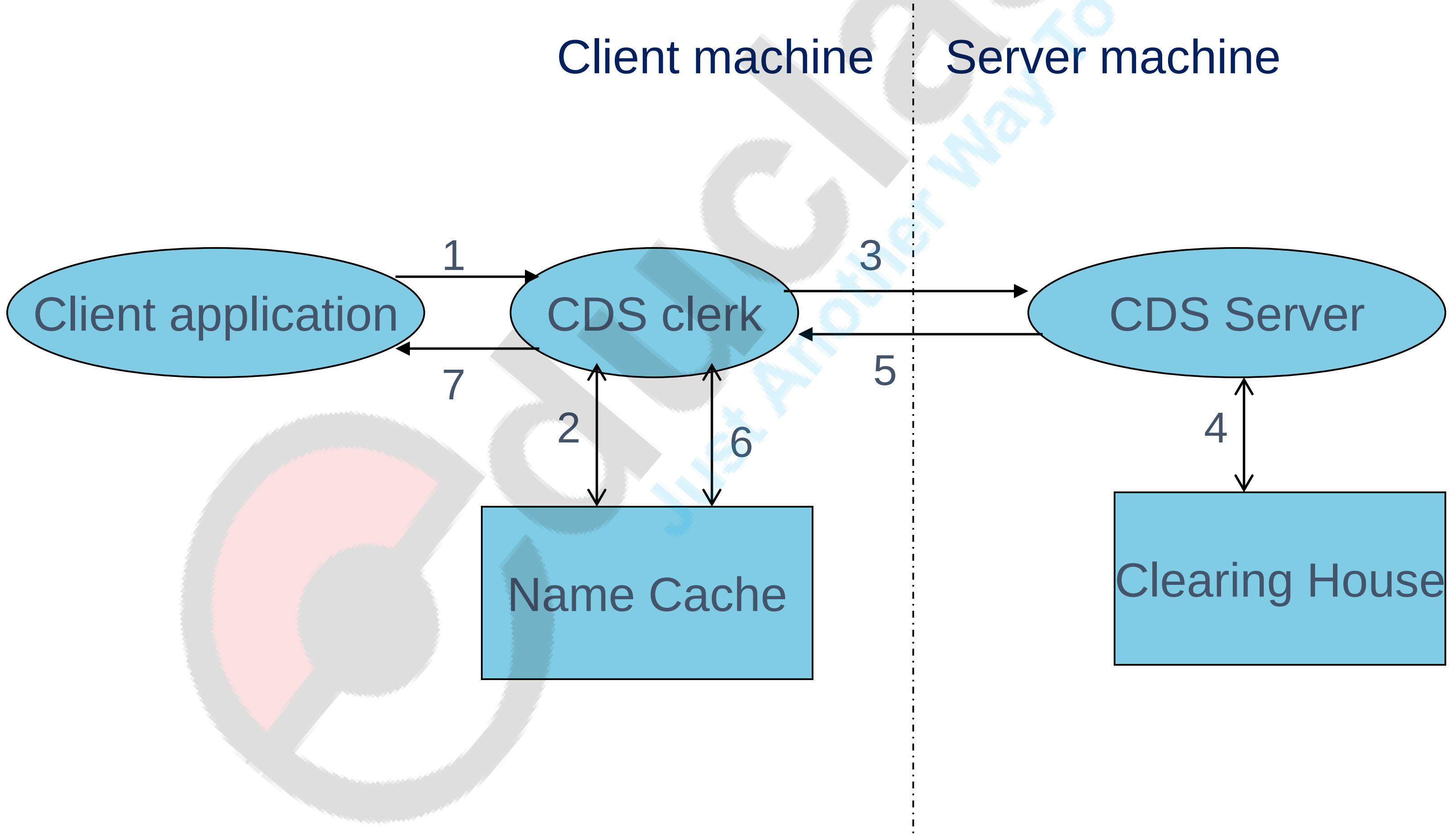
- DNS
  - Standard scheme for naming hosts & other resources on internet.
  - Uses hierarchical, tree structured namespace partitioned into domains both organizationally or geographically.
  - Name consists of strings separated by '.'
  - Highest level domain is towards right.

Intracell Naming

- All names within cell managed by Cell Directory Service (CDS)
- CDS Directories
  - Organized in hierarchical tree structure
  - Manages names & attributes of all objects within cell
  - Each directory entry consists of name, attribute, protection information.
- Can have more than one CDS server. Replication of naming information is supported.
- Unit of replication is directory.
- Each CDS server maintains one or more clearing house (collection of directories)

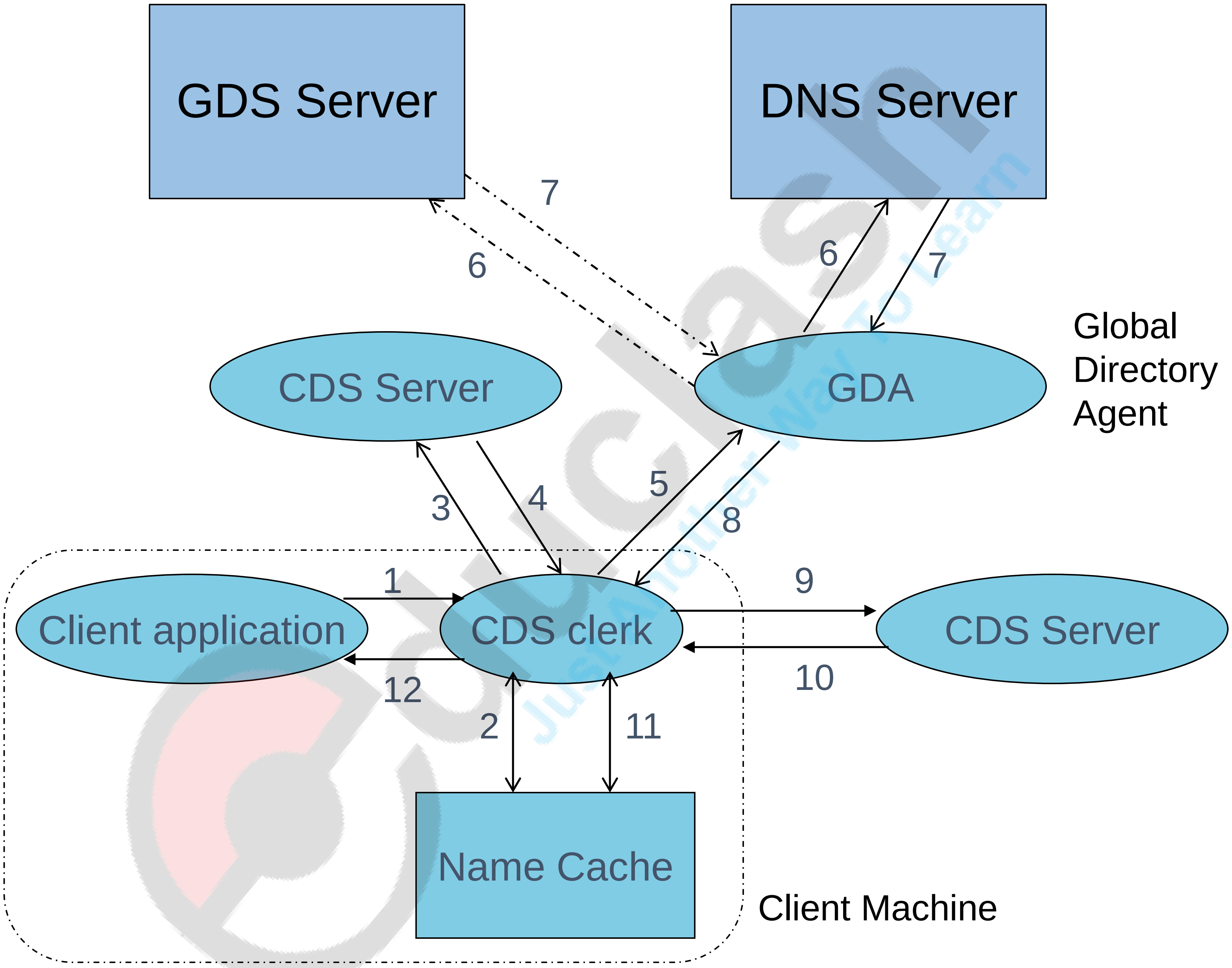
CDS Implementation

- Uses client – server model.



InterCell Naming

- CDS clerks must have a way to locate CDS servers in another cells.
- Either GDS(Global Directory Service) or DNS(Domain name system) notation might be used for naming
- Global Directory Agent (GDA) exists in any cell that needs to communicate with other cells.


GDS Server
DNS Server
7
6
6
7
Global Directory Agent
CDS Server
GDA
3
4
5
8
1
9
Client application
CDS clerk
CDS Server
12
10
2
11
Name Cache
Client Machine