Distributed Systems - CORBA

Sara Kadry

National College of Ireland

sara.kadry@ncirl.ie

Acknowledgements

Slides material are taken from the following source:
Dr.Dominic Carr
National College of Ireland
dominic.carr@ncirl.ie

Section 1

Naming in Distributed Systems

Naming in a Distributed System I

Naming System

The naming system is one of the most important components of a distributed system because **it enables services and objects to be identified and accessed in a uniform manner**.

Names

A name is a string of bits or characters which refers to some entity.

Entity

An entity is a resource such as a host, printer, file, a process running on a host, a user etc. We perform operations of entities at access points, the name of an access point **is an address**.

Naming in a Distributed System II

Address

- An access point is named by an address
- An entity address is the address of the entity's access point
- There can be multiple access points per entity, like in a content distribution network (CDN).
- The entity's access points may change

Identifier

A name which uniquesly identifies an entity e.g. a URI. This refers to at most one entity

Naming in a Distributed System III

We can distinguish between Human-oriented and system-oriented naming:

System-Oriented Names

- Represented in machine readable form
- Usually 32-bit or 64-bit strings
- Structured or unstructured
- Easy to store, to manipulate, to compare
- NOT easy to remember
- ...consequently, hard for humans to use
- Example: inode (0x00245d99900s)

Naming in a Distributed System IV

Human-oriented Names

- Variable length character strings
- Usually structured
- Often many human-oriented names can map onto a single system-oriented name
- Easy to remember and distinguish between
- Harder for the machine to process
- Example: pathnames (/Users/dominiccarr/slides)
- Example URL (http://www.ncirl.ie/learning)

Name Spaces I

- A name space is a labeled, directed graph wit type types of node.
- A leaf node represents a named entity and has the property that it has no outgoing edges.
- A directory node that has a number of outgoing edges, each labelled with a name, and has an identifier.
- n0 is the root node, if a path starts at the root node we say that it is an absolute path, otherwise it is relative. '/' is used as a separator between labels and as a representation for n0.

Name Spaces II

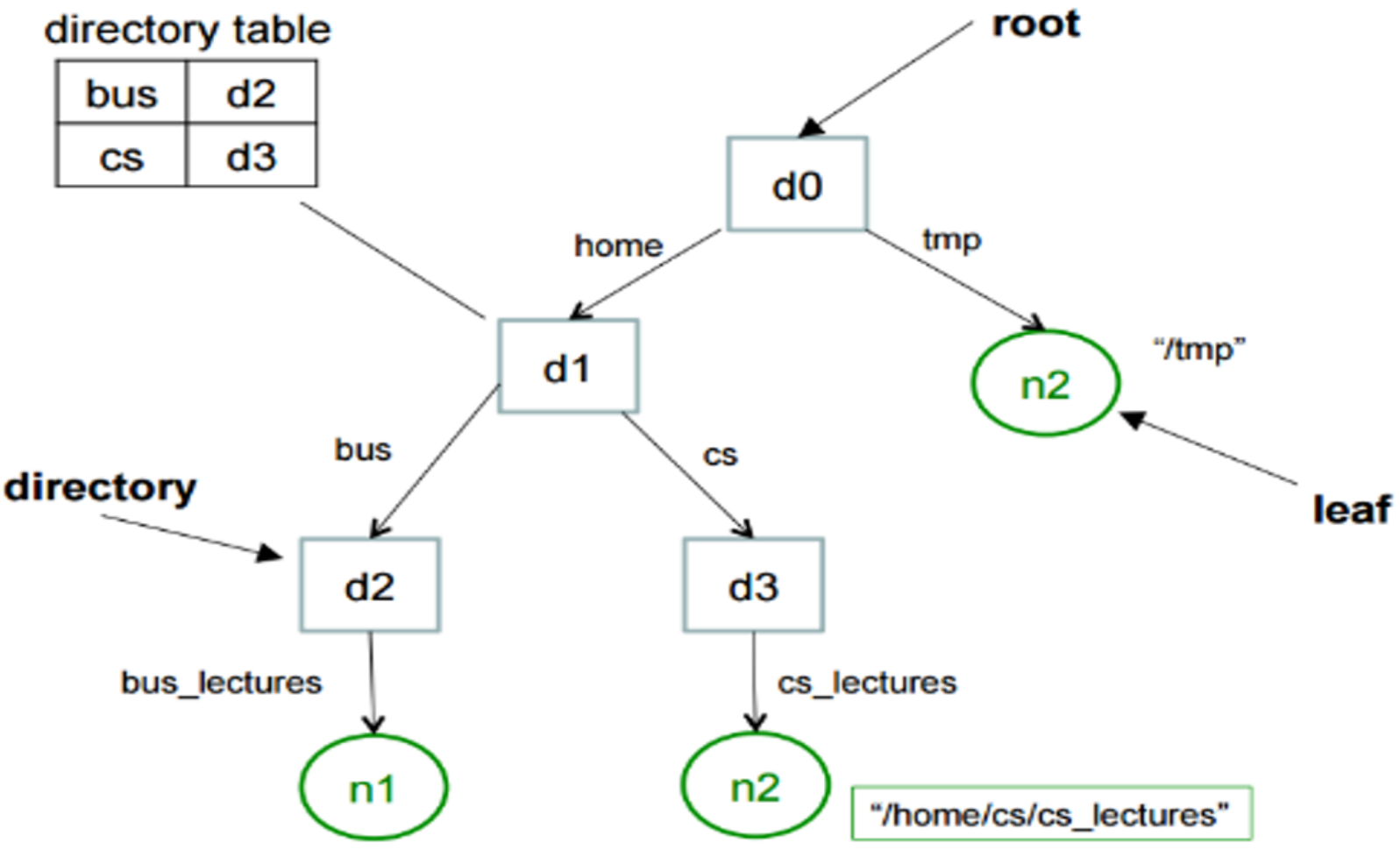

Figure: Example of a name space

Name Spaces III

Path name is a sequence of edge labels from one node to another.

- Absolute such as /home/cs/cs_lectures
- Relative (to where we are located) such as cs/cs_lectures

Aliasing

- Name spaces sometime support multiple names (aliases) for an entity.
- In other words an entity can be reached by multiple path names
- A CDN like Amazon CloudFront behaves in this way where distributed replicas of a file can be accessed from different locations

Two types of alias

- Hard Link (two or more links to an entity)
- Soft Link (Leaf node holds a pathname to another node, referred to as a symbolic link in Unix)

Name Spaces IV

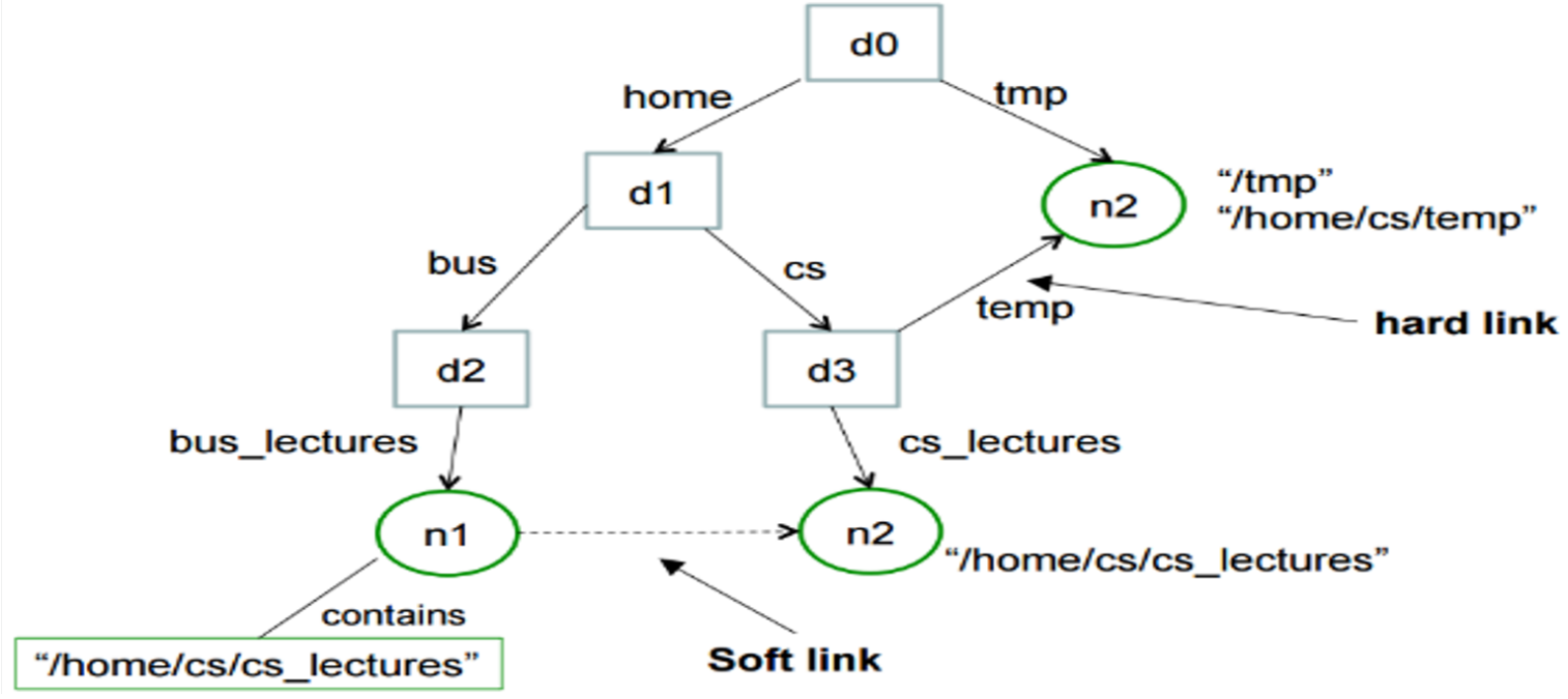

Figure: Example of aliasing

Name Spaces V

- A structured name space can be strictly hierarchical
  - Organised as a tree
  - Each node has one incoming edge only
  - Ergo there is exactly one associated (absolute) path name
- Or it can form a directed acyclic graph (DAG)
  - Each node can have more than one incoming edge
  - But no cycles are allowed in the graph

Mounting I

- Merging of two different name spaces
- A directory node stores the identifier of a directory node from a different (foreign) name space
- That directory is called a mount point
- The corresponding directory node in the foreign name space is called a mounting point, and this is typically the root node

Mounting II

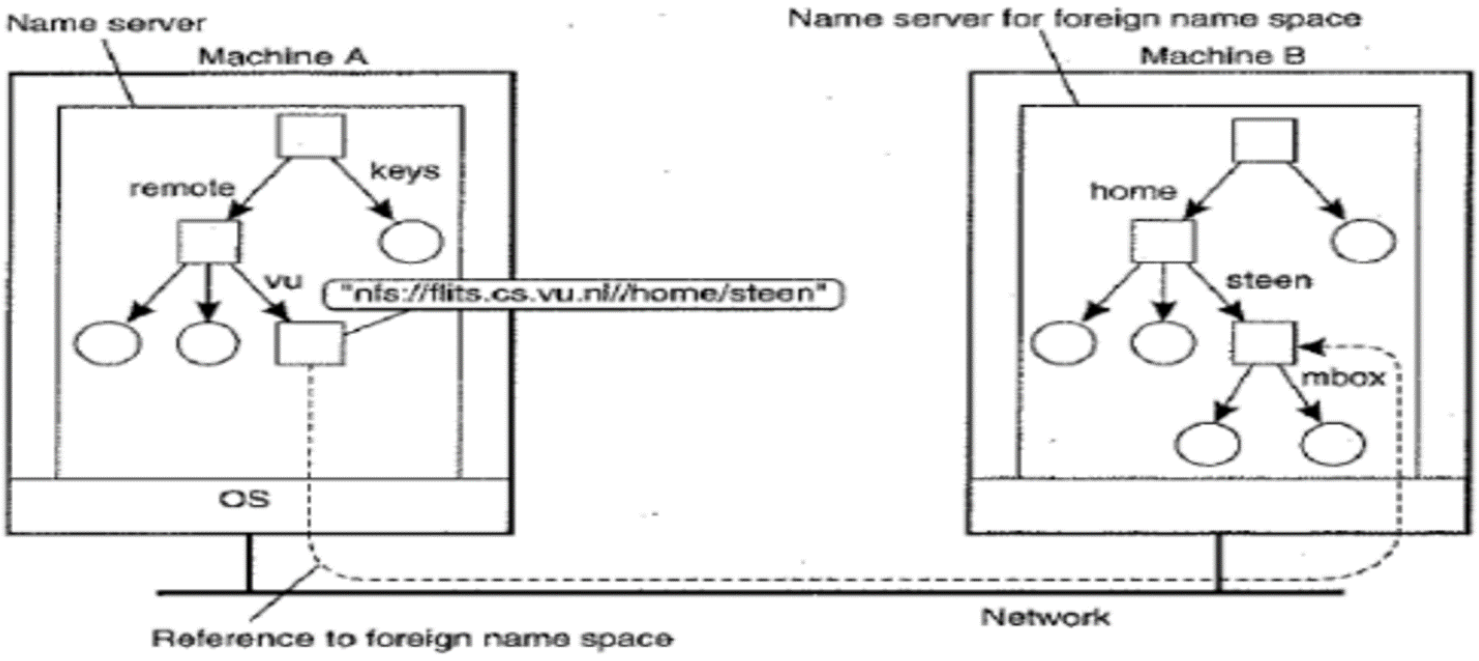

Figure: Mouting remote name spaces through a specific access protocol

Naming Services I

Naming Service

A service which allows users and processes to add or remove leaf nodes, modifying the content of nodes and lookup names

- Naming service is implemented by naming servers
- A Local Area Network requires a single name server
- In a DS it is often necessary to distribute name space implementation over multiple name servers, which are partitions into logical layers
- A well known, hugely crucial, example is the Domain Name Systems (DNS) which translates from human-oriented URLs into machine addresses (IPs).

Naming Services II

Note on RMI

The RMI registry helps to map names to objects, we have already seen this in our first tutorial class where the name "/HelloServer" was bound to the implementation of the the Hello interface. It was through this name that the client was able to access the remote service as if it were a local object.

Name Resolution I

This is the process of looking up a name to retrieve the information stored at the node referred to by that name. To do so we need to perform:

Pathname resolution

- Start node is either the root node (absolute name) or directory node (relative)
- Follow links to successive nodes by reading the directory tables
- Ends with data from(or a reference to) the last node (last element of the path name)

Name Resolution II

- Every client in a DS has access to a local name resolver
- Responsible to ensure that name resolution is carried out
- Knowing how and when to start name resolution is known as a closure mechanism.
- This is what your browser does when you enter a URL, it checks if it knows the corresponding machine address, if it doesn't it goes to ask a DNS server, which may need to ask a higher-level DNS server...

Name Resolution III

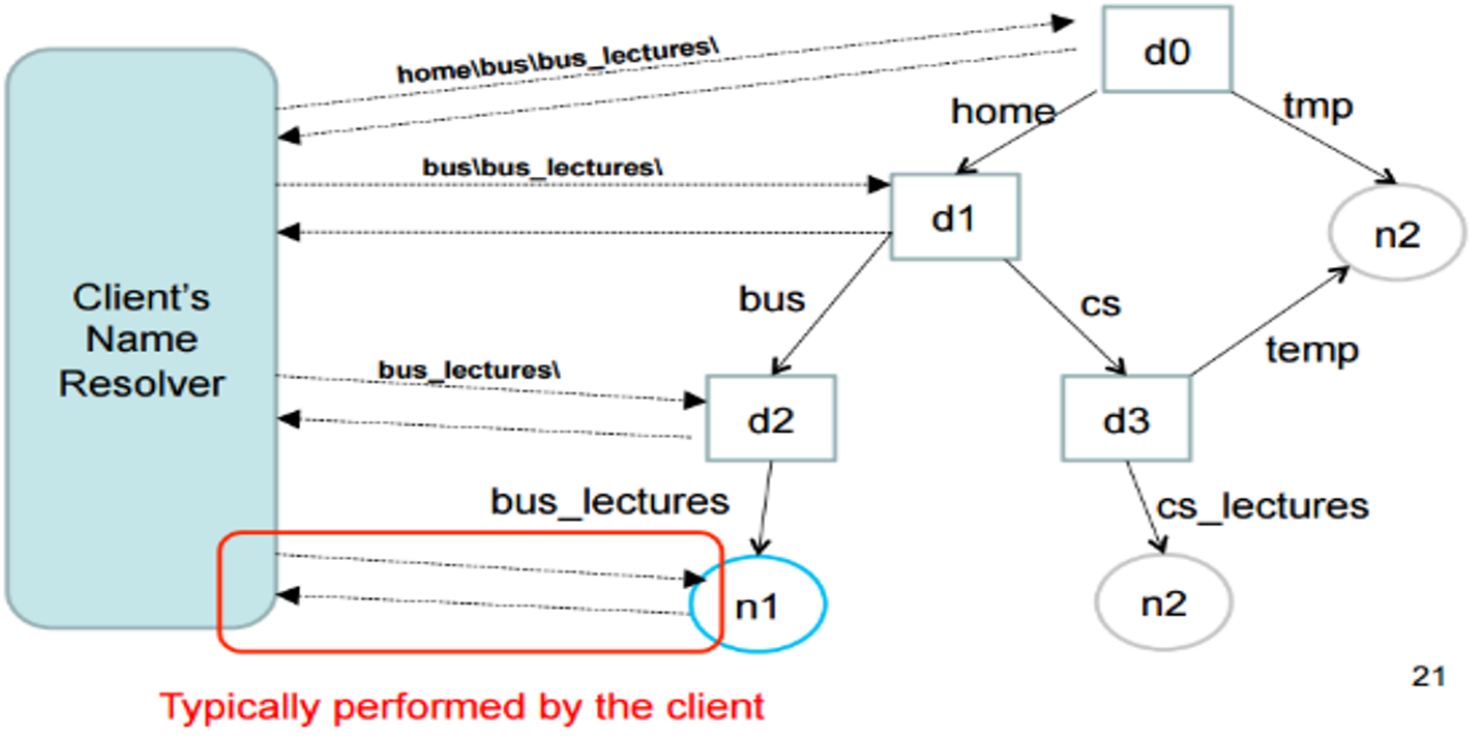

Figure: Iterative Name Resolution

Name Resolution IV

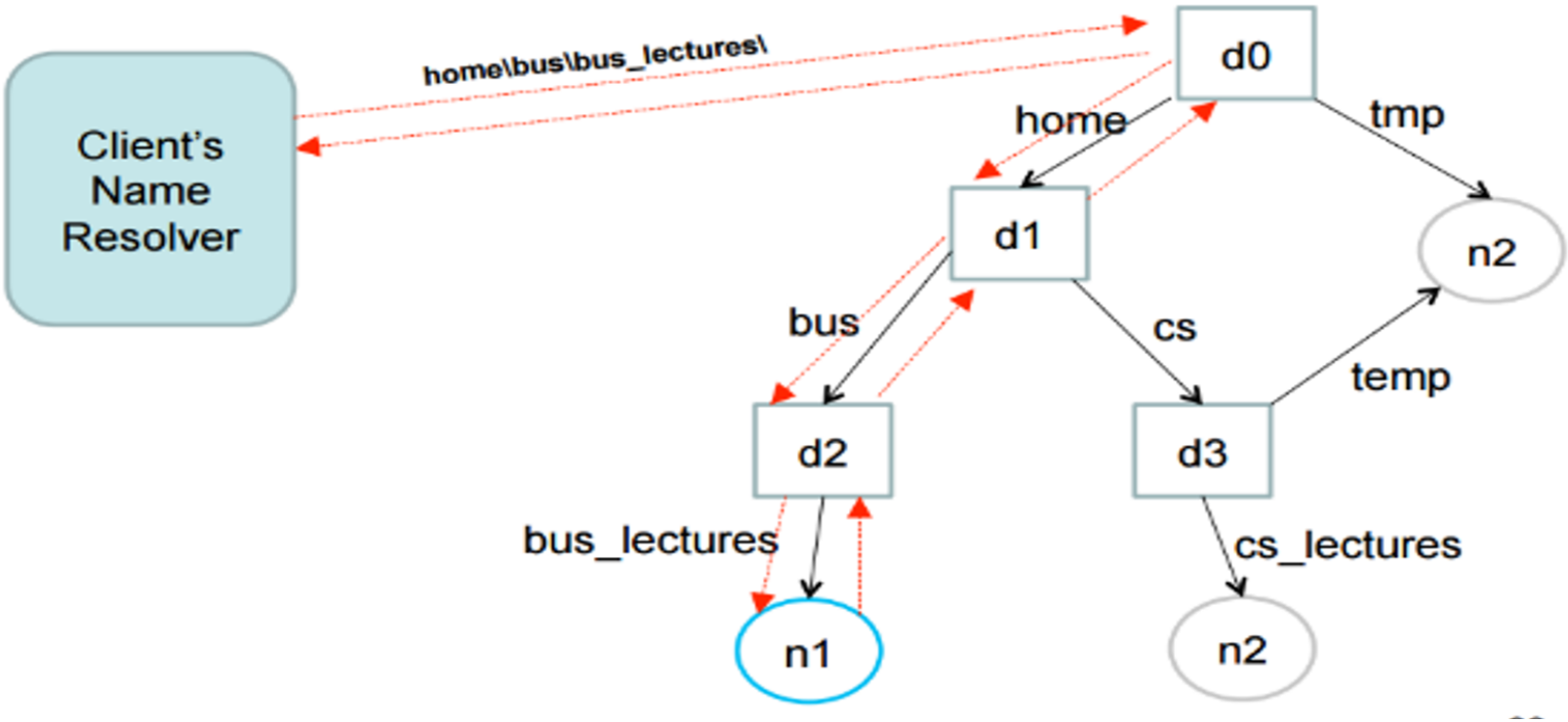

Figure: Recursive Name Resolution

Name Resolution V

Recursive

- (+) Effective caching at each name server
- (+) Reduced communication (if name servers close together)
- (+) Name servers can be protected from external access
- (-) Higher performance demand placed on servers

Iterative

- (-) Caching only at resolver (could be improved by using a local/intermediate name server)
- (-) High level of communication

Naming Service in CORBA I

CORBA Common Object Naming Service (COS)

- The Naming Service provides a mapping between a name and an object reference.
- Binding an object to a name means storing the mapping in the Naming Service
- Unbinding the name means removing the mapping
- Resolving the name means obtaining an object reference that is bound to a name
- Names can be hierarchically structured by using contexts, Contexts are similar with directories in file systems and can contain: Name bindings, and Sub-contexts

Naming Service in CORBA II

- Naming context = an object that contains a set of name bindings in which each name is unique.
- Different names can be bound to an object in the same or different contexts at the same time.
- A name is always resolved relative to a context, there are no absolute names

Naming Service in CORBA III

How do we get the object reference from the name server?

1. Server creates a servant object
2. Server registers object with the name server
3. Client requests object from nameserver (resolves name)
4. Nameserver returns reference to remote object (stub gets created)
5. Client invokes method on stub
6. Stub communicates with skeleton
7. Skeleton invokes method on remote object.

Key Point

This is quite similar to Java RMI! Why? Because this is the way things are done in almost all RPC systems. However this is a bit more low-level, whereas Java RMI would be higher-level.

CORBA Names I

- Names are composed of a sequence of simple names.
- A simple name is a (value, kind) tuple.
- The value attribute is used for resolving names.
- The kind attribute is used to store and provide information about the role of the object.

For example

```
NameComponent nc = new NameComponent("Lecture", "Context");

NameComponent path[] = {nc};
//rebind is used to avoid an error if the name is already bound
ncRef.rebind(path, helloRef);
```