Naming

Naming Entities

- A name is a string of bits or characters that is used to refer to an entity.
  - hosts, printers, disks, files, processes, users, mailboxes, newsgroups, web pages, graphical windows, messages, network connections, etc.
- Entities can be operated on. To operate on an entity, it is necessary to have an access point.
- The name of an access point is called an address.
  - Combination of an IP address and port number.
- An entity can offer more than one access points.
- An entity may change its access points in the course of time.

Naming Entities

Disadvantages of treating addresses as a special type of name

- an entity may access point or an access point may be re-assigned to a different entity
- for entities that offer more than one access points, it is not clear which address to use as a reference.

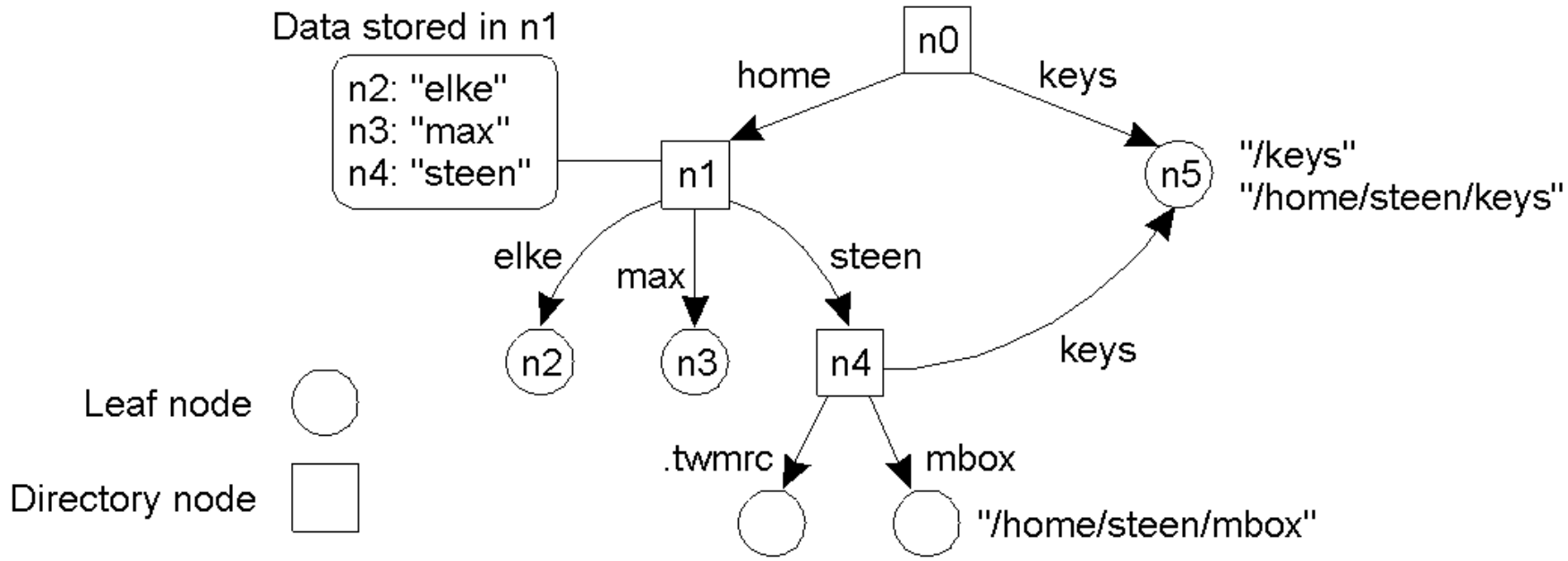
Location-Independent names

- a name that is independent from its addresses
- A true identifier is a name that has the following properties:
  1. An identifier refers to at most one entity
  2. each entity is referred to by at most one identifier
  3. an identifier always refers to the same entity (i.e., it is never re-used).

Human-friendly names

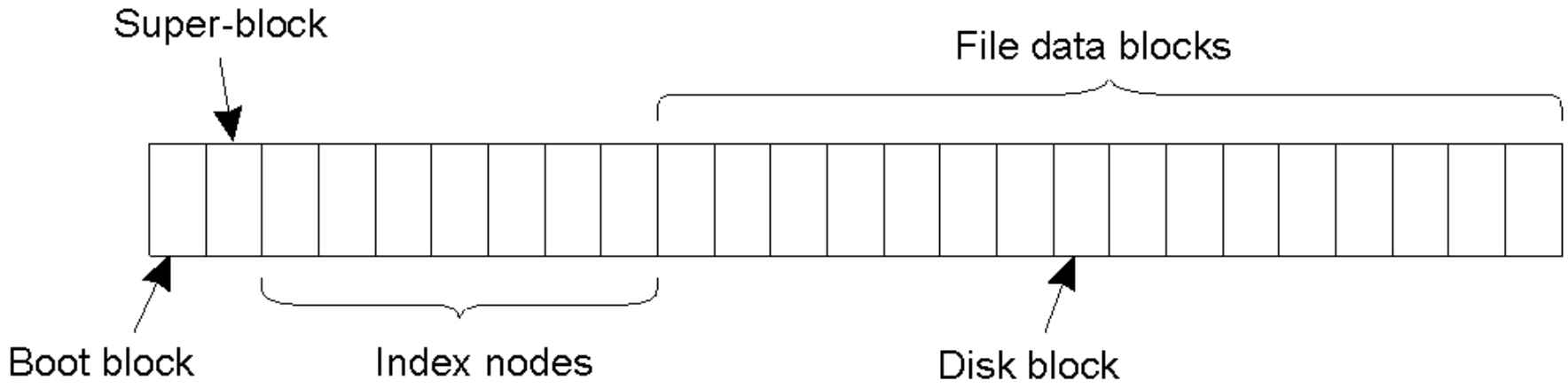
- names tailored to be used by humans -> generally represented as a character string

Name space



- name space, leaf nodes, directory nodes, paths: absolute path name - relative path name, trees, DAGs
- Global name: is always interpreted with respect to the same directory node.
- Local name: relative name whose directory in which it is contained is implicitly known.

Name Spaces



The general organization of the UNIX file system implementation on a logical disk of contiguous disk blocks.

Name Resolution

- Given a path name, it should be possible to look up any information stored in the node referred to by that name.
  - N: <label-1, label-2, … , label-n>
- In UNIX, a node identifier is implemented as the index number of an inode.

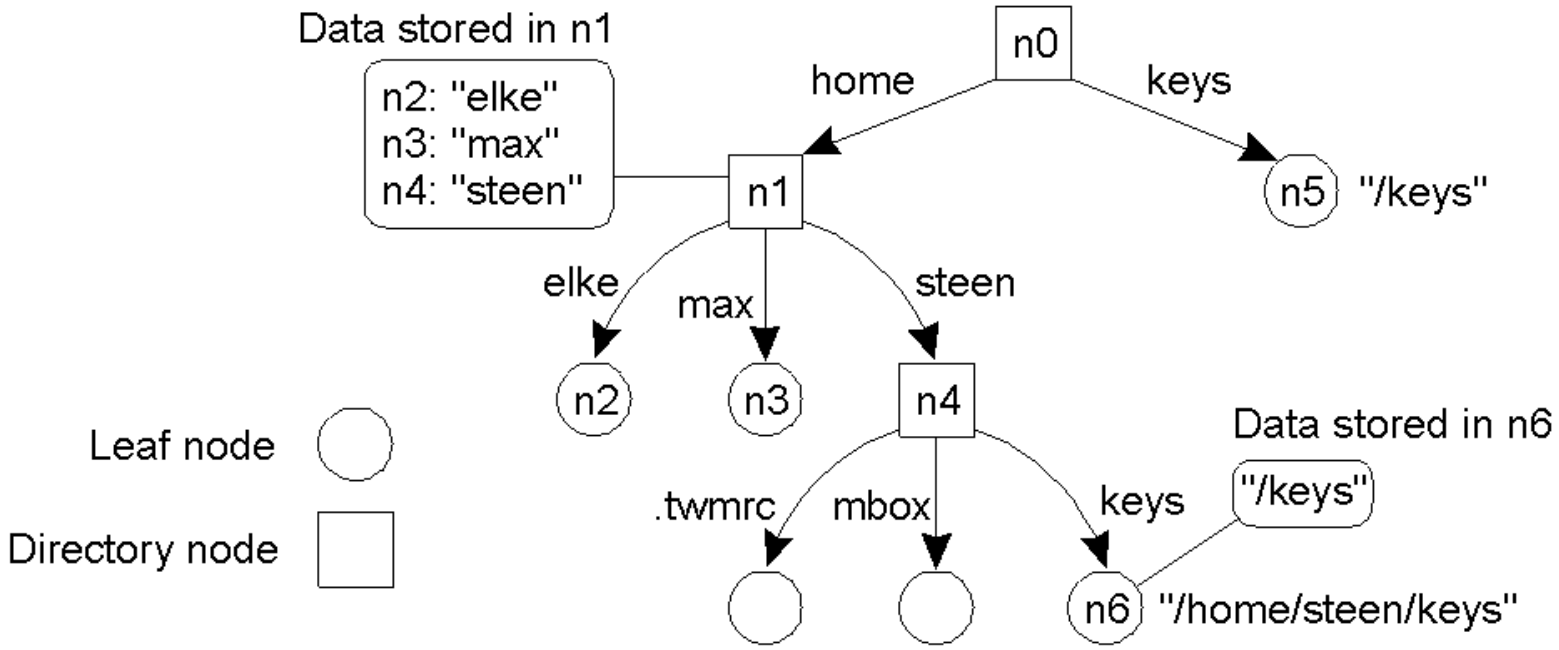
Closure Mechanism

- Knowing how and where to start name resolution is generally referred to as a closure mechanism.

Linking and Mounting

- Alias: another name for the same entity
- Hard links: allow multiple absolute path names to refer to the same node in a anaming graph.
- Symbolic links: represent an entity by a leaf node but store an absolute path name in this node.
- Mounted file system: let a directory store the identifier of a directory node from a different name space.
- Mount point: the directory node storing the identifier.
- Mounting point: the directory node in the foreign name space.

Linking and Mounting

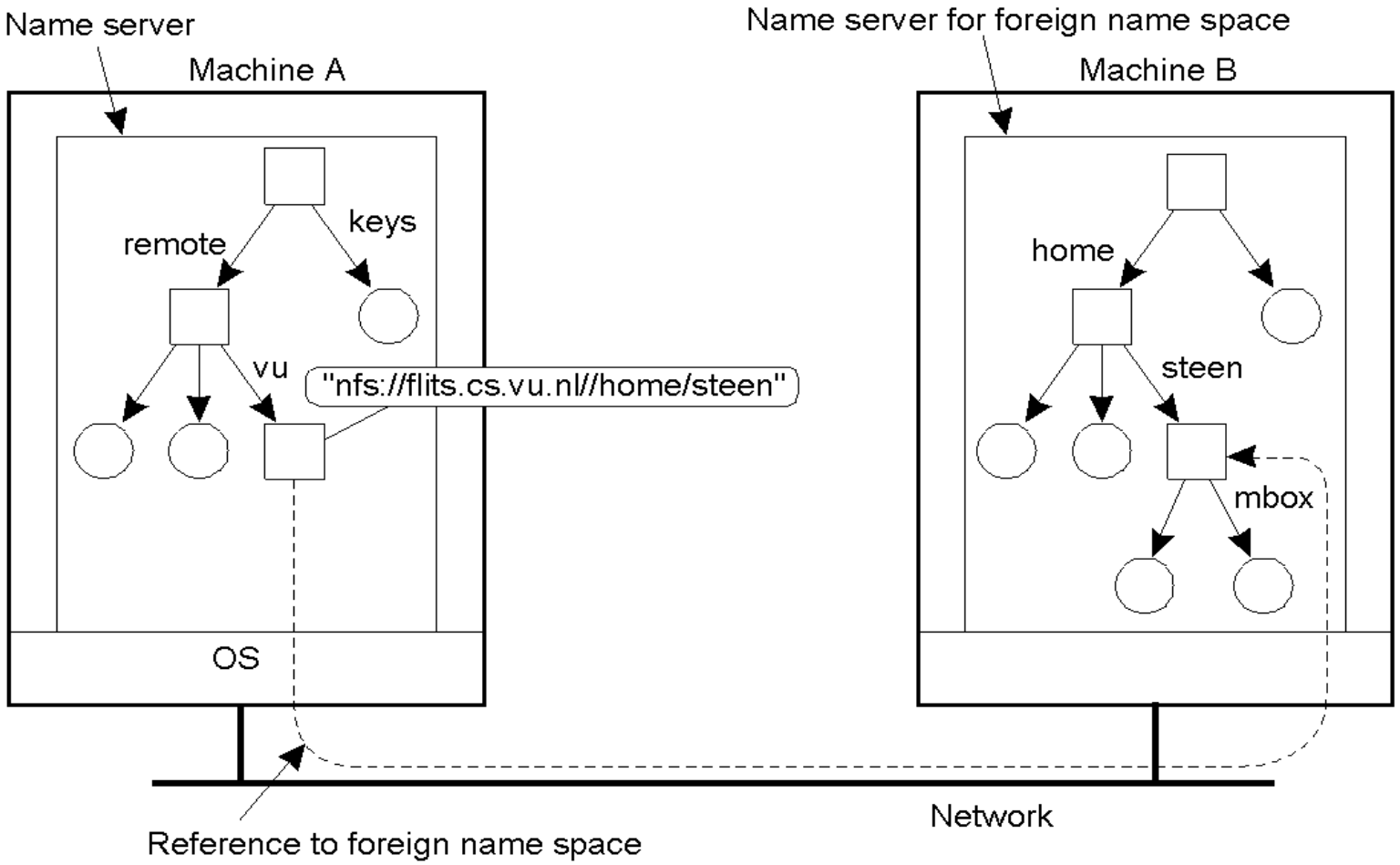


The concept of a symbolic link explained in a naming graph

Linking and Mounting

- To mount a foreign name space in a distributed system requires at least the following information:
  - the name of an access protocol
  - the name of the server
  - the name of the mounting point in the foreign name space
- Example
  - nfs://flits.cs.vu.nl//home/steen

Linking and Mounting



Mounting remote name spaces through a specific process protocol.

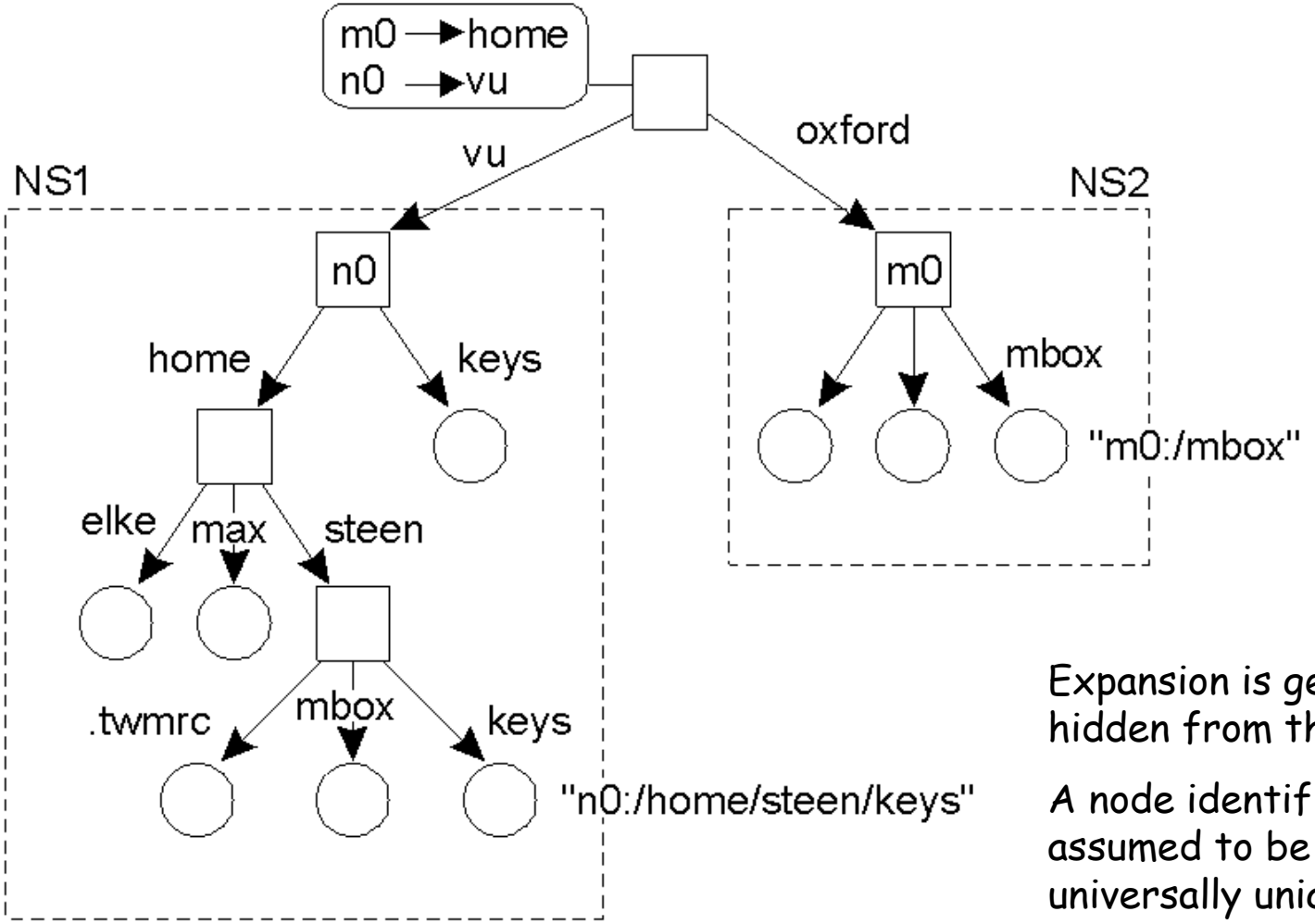
The user is spared the details of the actual access to the remote server.

Linking and Mounting

Another approach to mounting (Global Name Service, GNS)

- Add a new root node and make the existing root nodes its children.
- names in GNS always (implicitly) include the identifier of the node from where resolution should normally start.
- When adding a new root node, the node stores a table mapping the identifier of the root node to the name under which that root is known in the new name space.

Linking and Mounting



Expansion is generally hidden from the user!

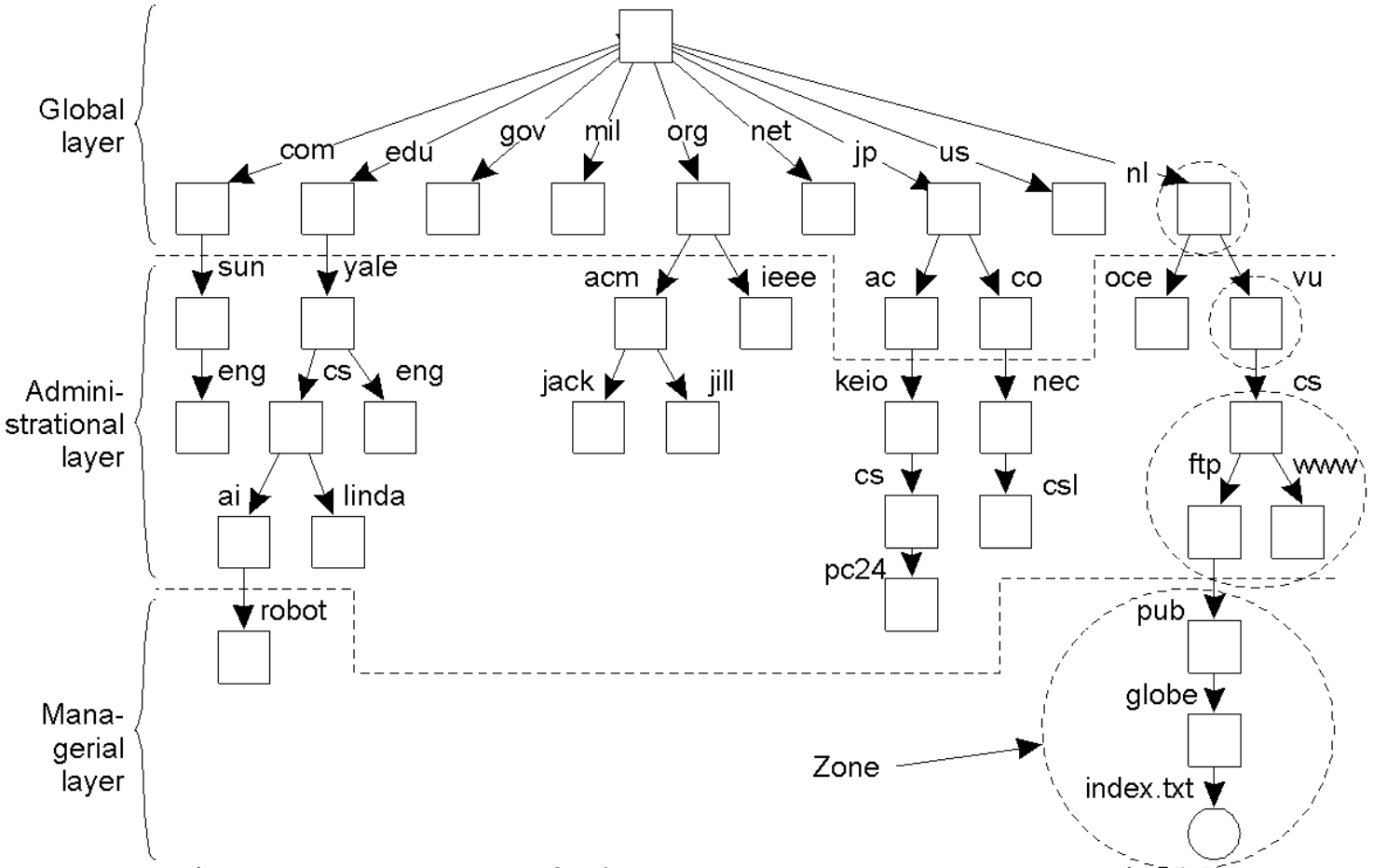
A node identifier is assumed to be universally unique.

Organization of the DEC Global Name Service

Name Space Distribution

- Name-spaces for a large-scale, worldwide distributed system are usually organized hierarchically.
- The name space is partitioned into logical layers.
  - global layer: root and directory nodes close to the root
  - administrational layer: directory nodes that are managed within a single organization
  - managerial layer: nodes that may typically change regularly
- The name space is also divided into non-overlapping parts, called zones in DNS.
  - A zone is a part of the name space that is implemented by a separate name server.

Name Space Distribution



An example partitioning of the DNS name space, including Internet-accessible files, into three layers.

Name Space Distribution – Availability and Performance

- **Global Layer**
  - High availability is extremely critical
  - The results of lookup operations can be effectively cached.
  - Throughput is more important than worst-case response time per lookup request.
  - Techniques employed: (1) replication of servers, (2) client-side caching
- **Administrational Layer**
  - Availability is important for clients in the organization as the name server and less important for external clients.
  - Lookups should respond within a few milliseconds.
  - Updates shold be processed quicker than those of the global layer.
  - Techniques: (1) use high-performance machines to run name servers, (2) client-side caching combined with replication.
- **Managerial Layer**
  - Availability requirements are less demanding.
  - Performance is crucial.

Name Space Distribution

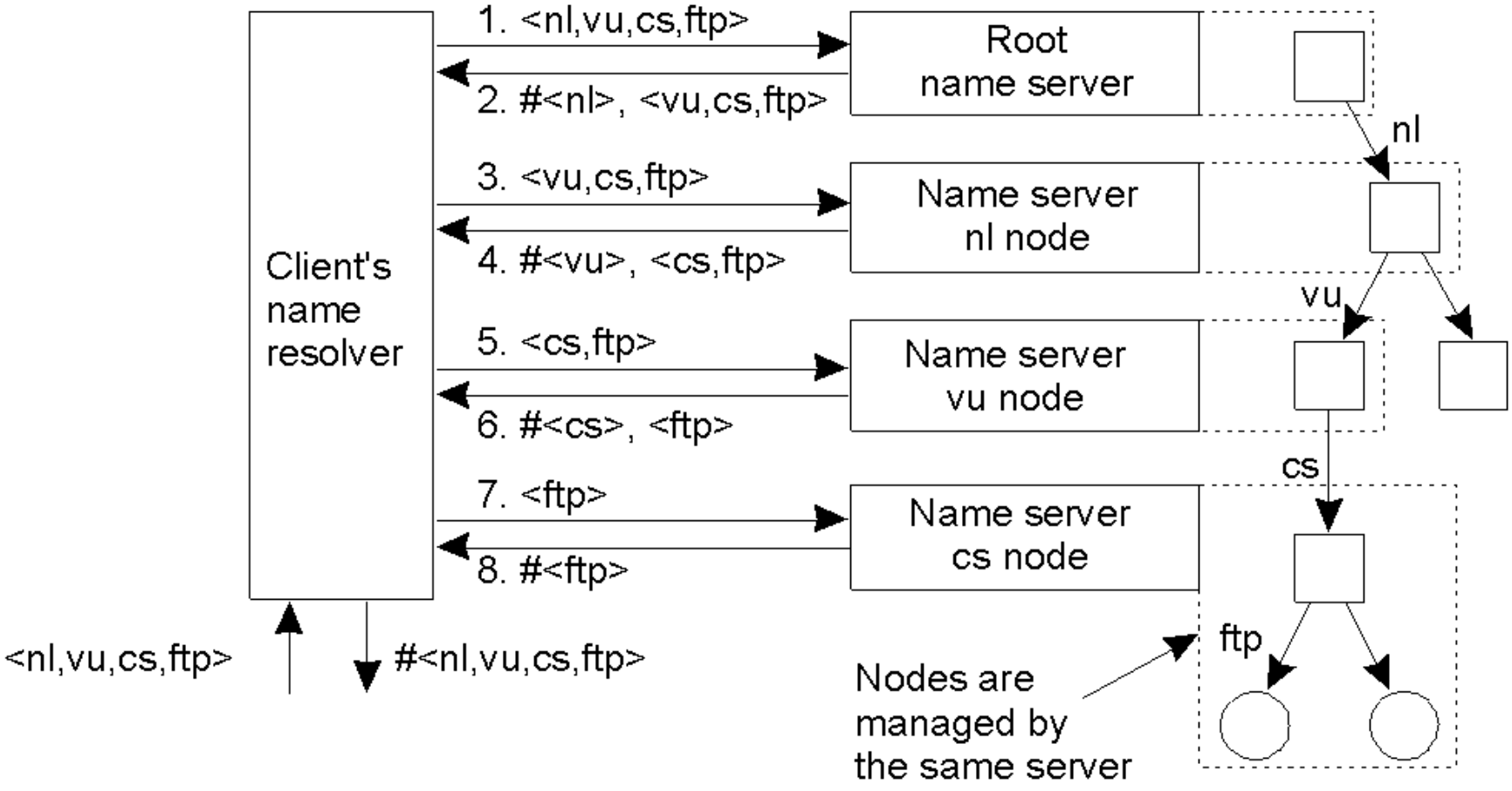
Item	Global	Administrational	Managerial
Geographical scale of network	Worldwide	Organization	Department
Total number of nodes	Few	Many	Vast numbers
Responsiveness to lookups	Seconds	Milliseconds	Immediate
Update propagation	Lazy	Immediate	Immediate
Number of replicas	Many	None or few	None
Is client-side caching applied?	Yes	Yes	Sometimes

Implementation of Name Resolution

- Assumptions:
  - name servers are not replicated
  - no client-side caches are used.
- Each client has access to a local name resolver, which is responsible for ensuring that the name resolution process is carried out.
- Example:
  - Resolve the name:

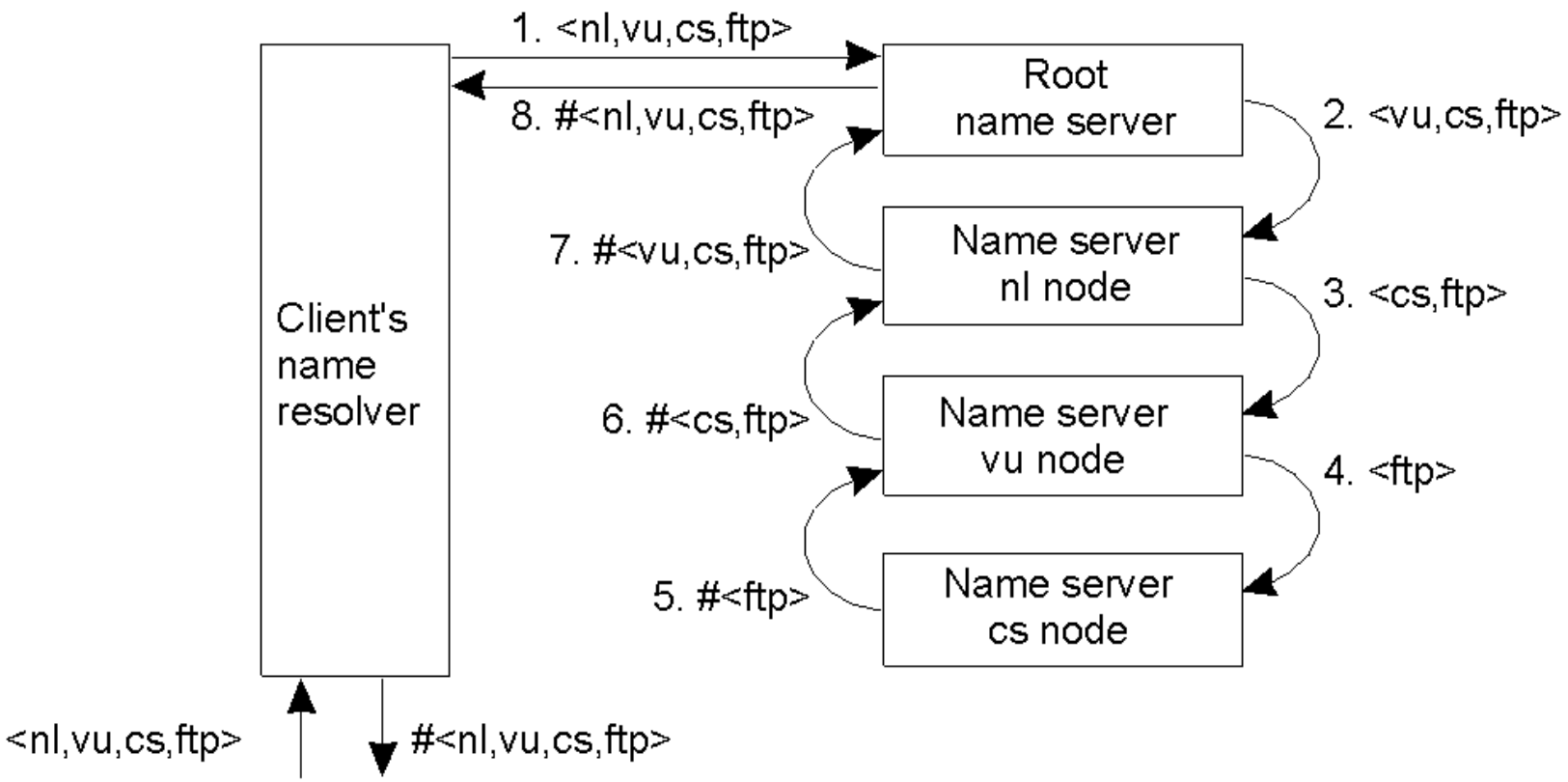
    root:<nl, vu, cs, ftp, pub, globe, index.txt>

Implementation of Name Resolution



Iterative Name Resolution

Implementation of Name Resolution



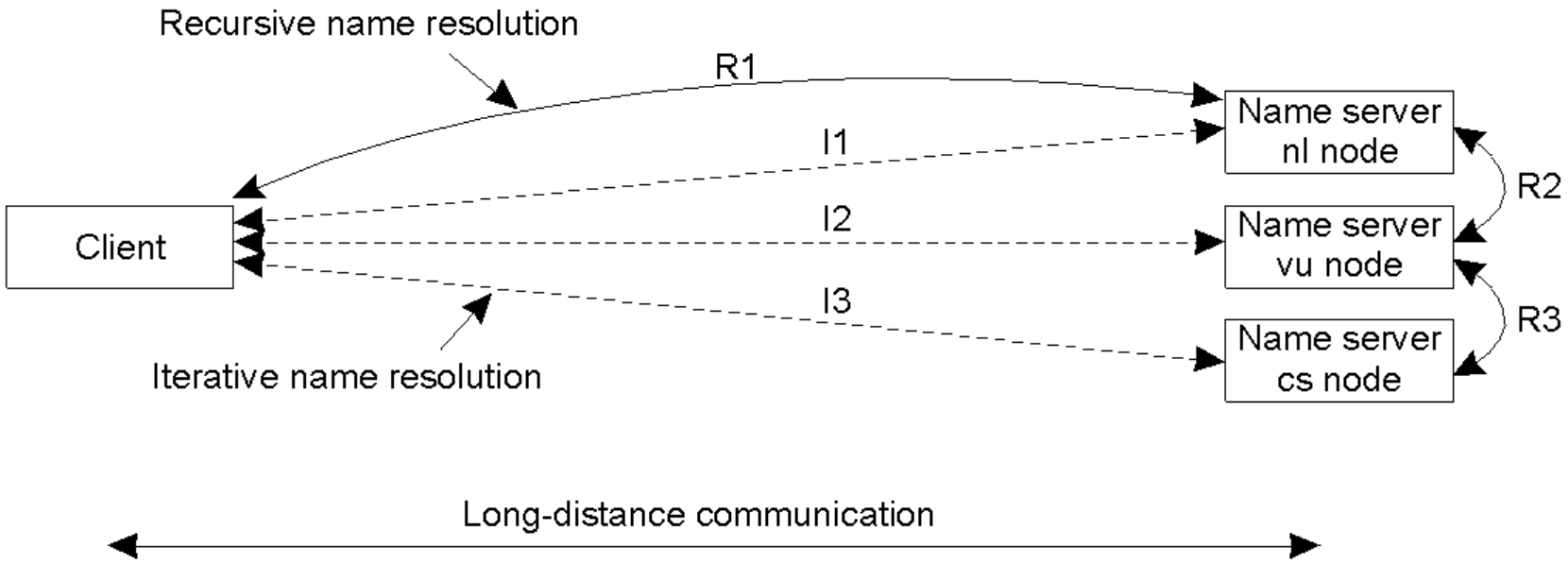
Recursive name resolution

Implementation of Name Resolution

Server for node	Should resolve	Looks up	Passes to child	Receives and caches	Returns to requester
cs	<ftp>	#<ftp>	--	--	#<ftp>
vu	<cs,ftp>	#<cs>	<ftp>	#<ftp>	#<cs> #<cs, ftp>
ni	<vu,cs,ftp>	#<vu>	<cs,ftp>	#<cs> #<cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>
root	<ni,vu,cs,ftp>	#<nl>	<vu,cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>	#<nl> #<nl,vu> #<nl,vu,cs> #<nl,vu,cs,ftp>

Recursive name resolution of <*nl, vu, cs, ftp*>. Name servers cache intermediate results for subsequent lookups.

Implementation of Name Resolution



The comparison between recursive and iterative name resolution with respect to communication costs.