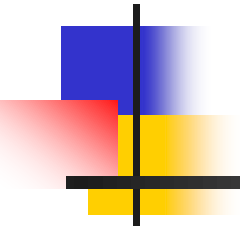
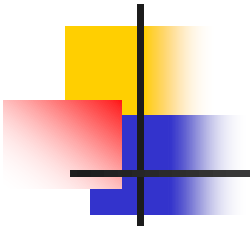
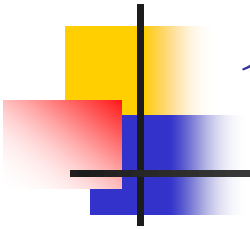


第5章 命名系统



内容提要

- 名称、标识符和地址
- 无层次命名
- 结构化命名
- 基于属性的命名

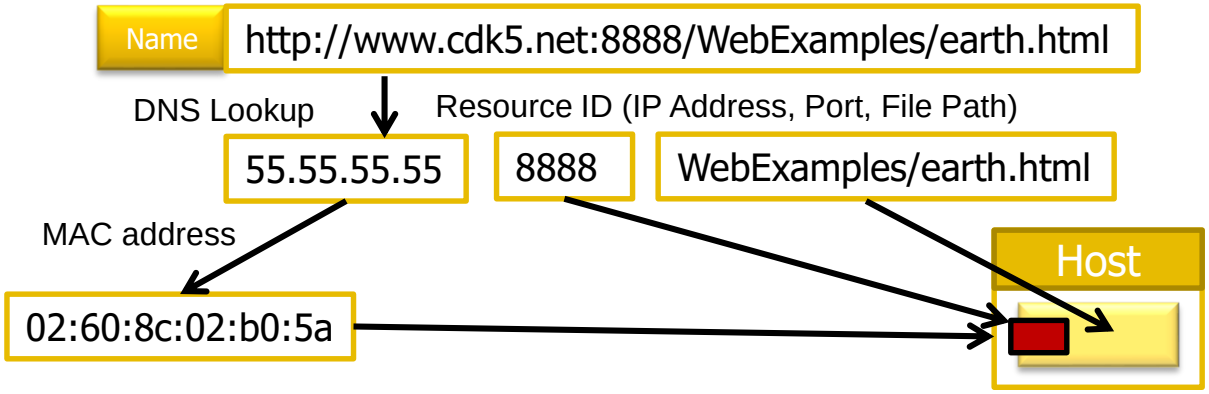


命名系统

- 位置无关（location independent）
- 用户友好（human-friendly name）
- 高效搜索（effectively search）

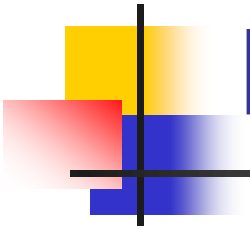
Naming

- Names are used to uniquely identify entities in Distributed Systems
  - Entities may be processes, remote objects, newsgroups, …
- Names are mapped to entities' locations using *name resolution*
- An example of name resolution



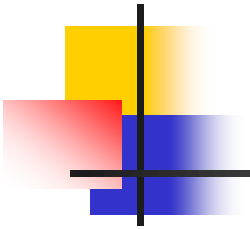
Names, Addresses and Identifiers

- An entity can be identified by three types of references
  1. Name
     - A name is a set of bits or characters that references an entity
     - Names can be human-friendly (or not)
  2. Address
     - Every entity resides on an access point, and access point has an address
     - Addresses may be location-dependent (or not)
     - e.g., IP Address + Port
  3. Identifier
     - Identifiers are names that *uniquely* identify entities
     - A *true identifier* is a name with the following properties:
       a. An identifier refers to at-most one entity
       b. Each entity is referred to by at-most one identifier
       c. An identifier always refers to the same entity (i.e. it is never reused)



Naming Systems

- A naming system is simply a middleware that assists in name resolution
- Naming systems are classified into three classes based on the type of names used:
  a. Flat naming 无层次
  b. Structured naming 结构化
  c. Attribute-based naming



内容提要

- 名称、标识符和地址
- 无层次命名
- 结构化命名
- 基于属性的命名

5.2无层次命名

- 问题提出

  当只给出非结构化的名称（比如说一个标识符），我们如何定位它的访问点？

  - 简单方法：广播和多播，转发指针
  - 基于宿主位置的方法
  - 分布式散列表
  - 分层方法

5.2.1简单方法

- 广播和多播

  广播一个包含实体的标识符，要求拥有该实体的机器返回它当前的地址。

  - 不能超出局域网
  - 要求所有的进程监听定位请求

- 转发指针

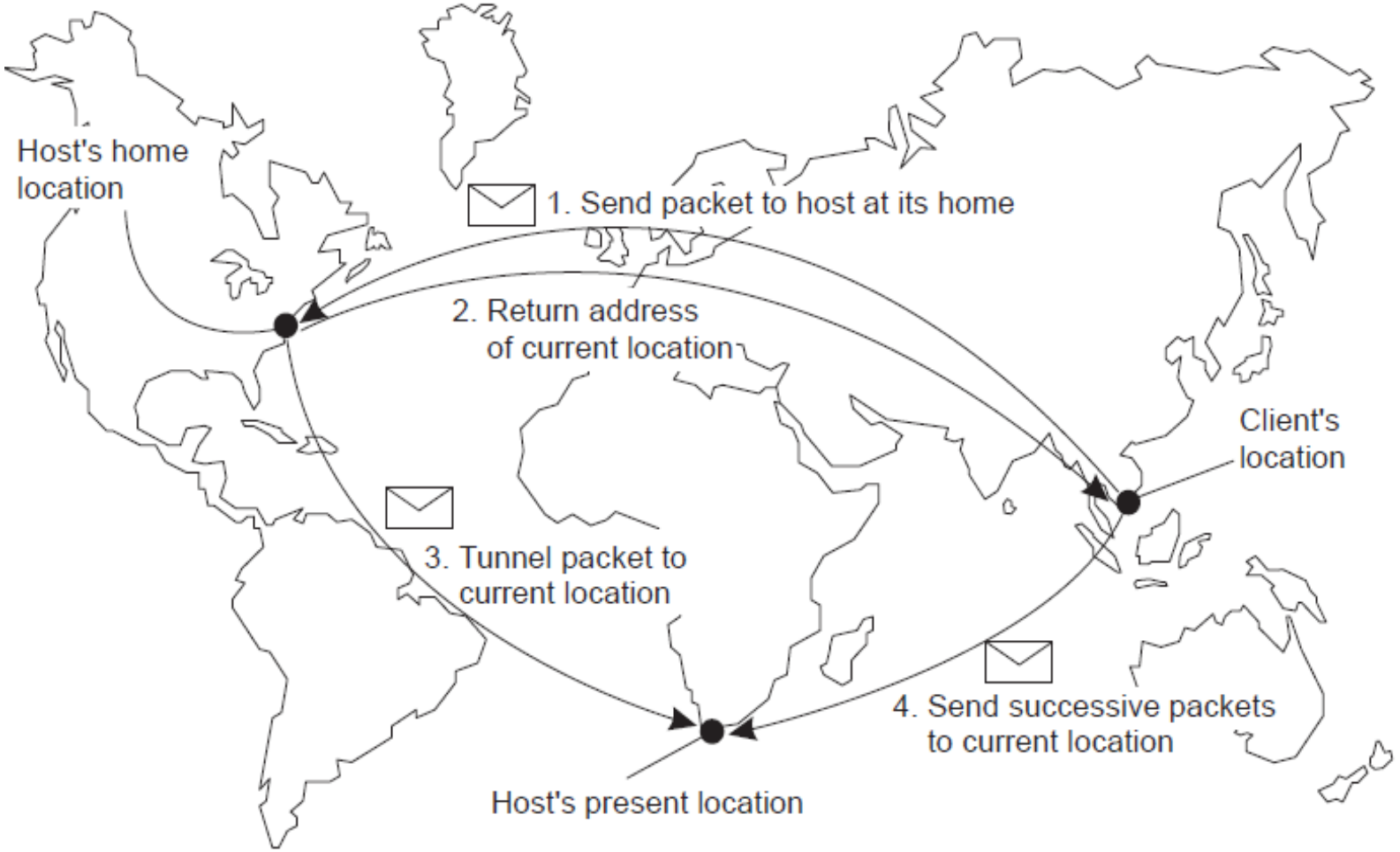
  当实体移动时，它留下一个指针指向下一个位置。

5.2.2基于宿主位置的方法

利用宿主追踪实体的位置

1. 注册宿主位置
2. 注册实体所在的远程主机
3. 宿主与实体所在的远程主机保持联系
4. 客户首先与宿主建立连接，然后与远程主机连接

5.2.2基于宿主位置的方法



5.2.2基于宿主位置的方法

- 基于宿主位置的方法的问题
  - 宿主地址必须在实体的生存期内有效
  - 宿主地址是固定的
  - 可扩展性差
- 问题

  我们如何解决永久转移的问题？

5.2.3分布式散列表（DHT）

Chord

将节点组织成逻辑环

- 各个节点被赋予一个随机的m位标识符。
- 每个实体被赋予一个特定的m位键值。
- 含有键值K的实体位于含有最小标识符ID>=K的节点之内（称之为K的后继者）。

DHTs: Finger表

原理

- 每个节点P维护一个最多M个实体的指状表FTp[]。

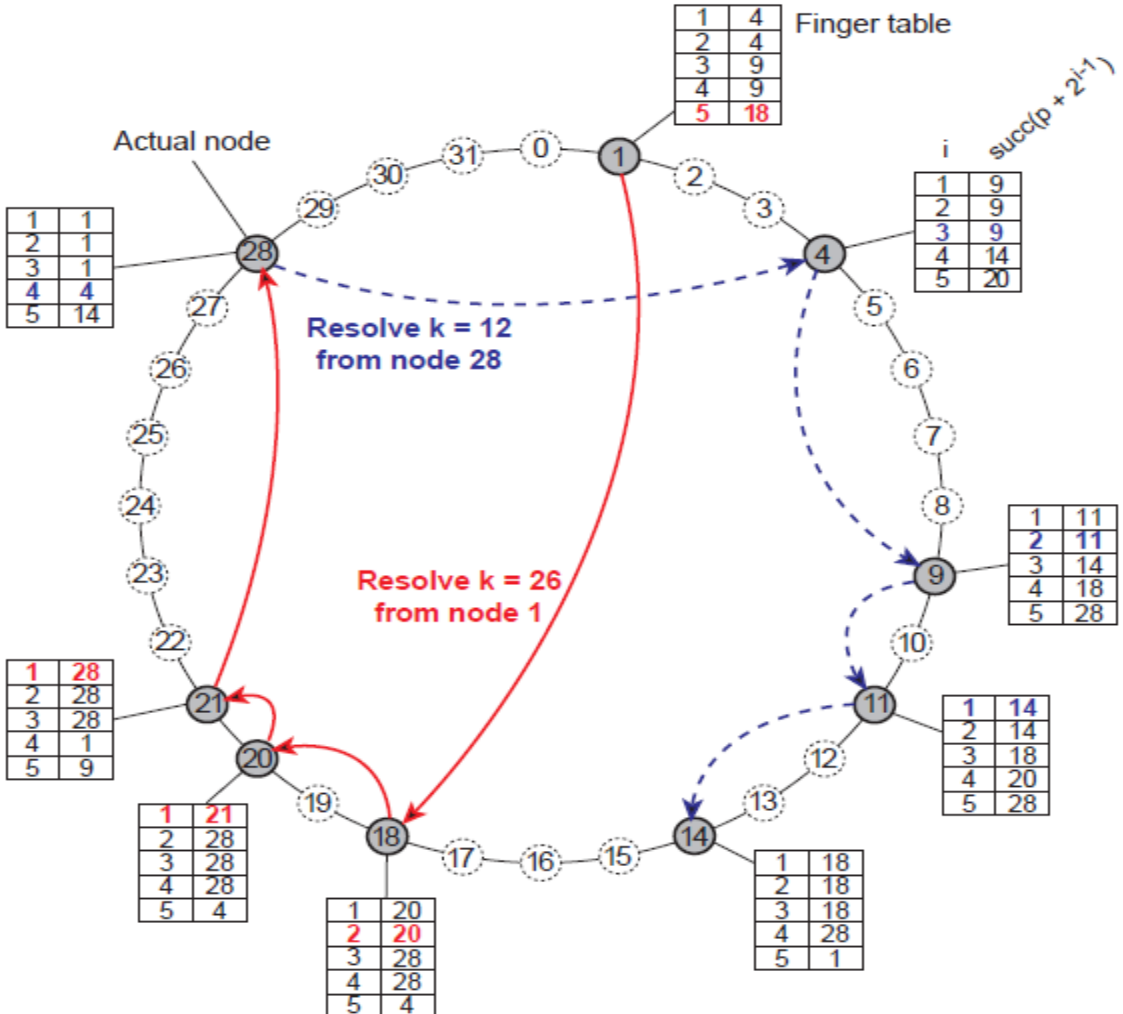
$$FTp[i] = succ(p+2^{i-1})$$

- 要查找键k，节点P立即把该请求转发给在P的指状表中索引为J的节点Q。

$$q = FTp[j] \;\; k < FTp[j+1]$$

- 如果$p < k < FTp[1]$，请求也要转发给FTp[1]。

DHTs: Finger表



利用网络邻近

- 潜在问题

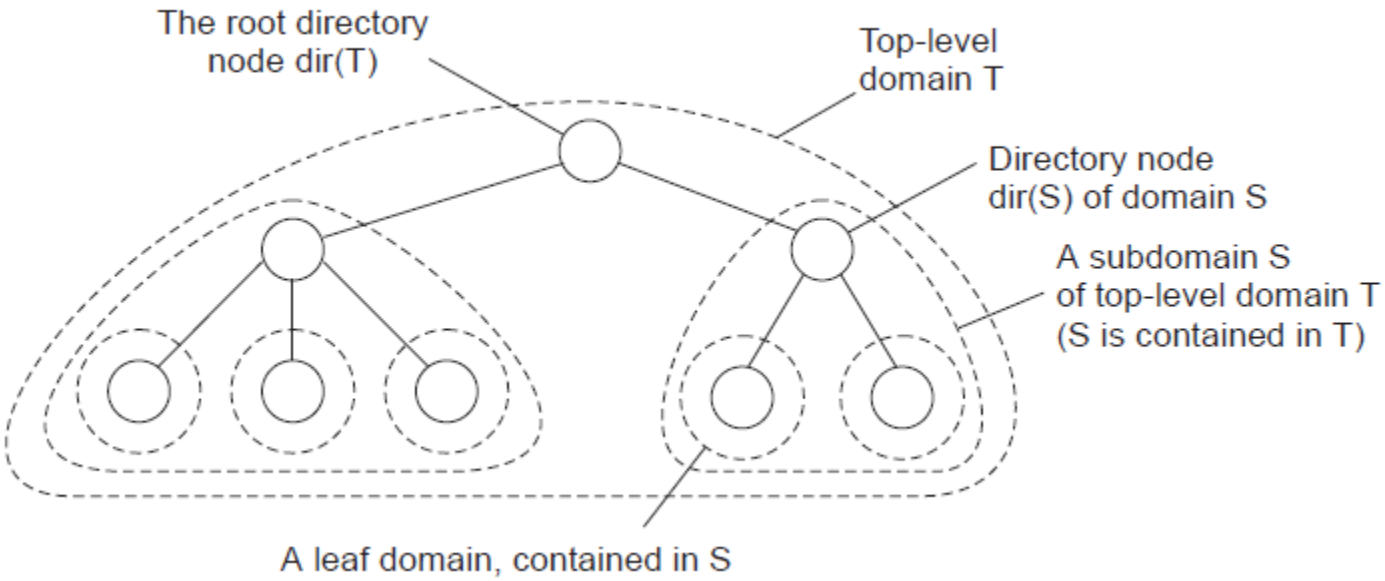
请求跨因特网进行路由时不稳定:结点k和后继结点(k+1)可能相距很远。

- 基于拓扑的结点标识符赋值：在标识符赋值时，两个邻近结点所赋给的标识符也是靠近的。很难实现。
- 邻近路由：每个结点维护多个后继者，转发给最近者。
  - 例：FTq[i]指向 $[p+2^{i-1};p+2^{i}-1]$区间内的第一个结点。 结点P也能跟踪该区间的其他结点。
- 邻近邻结点选择：选择最近的结点作为邻结点。

5.2.4分层方法（HLS）

基本思想

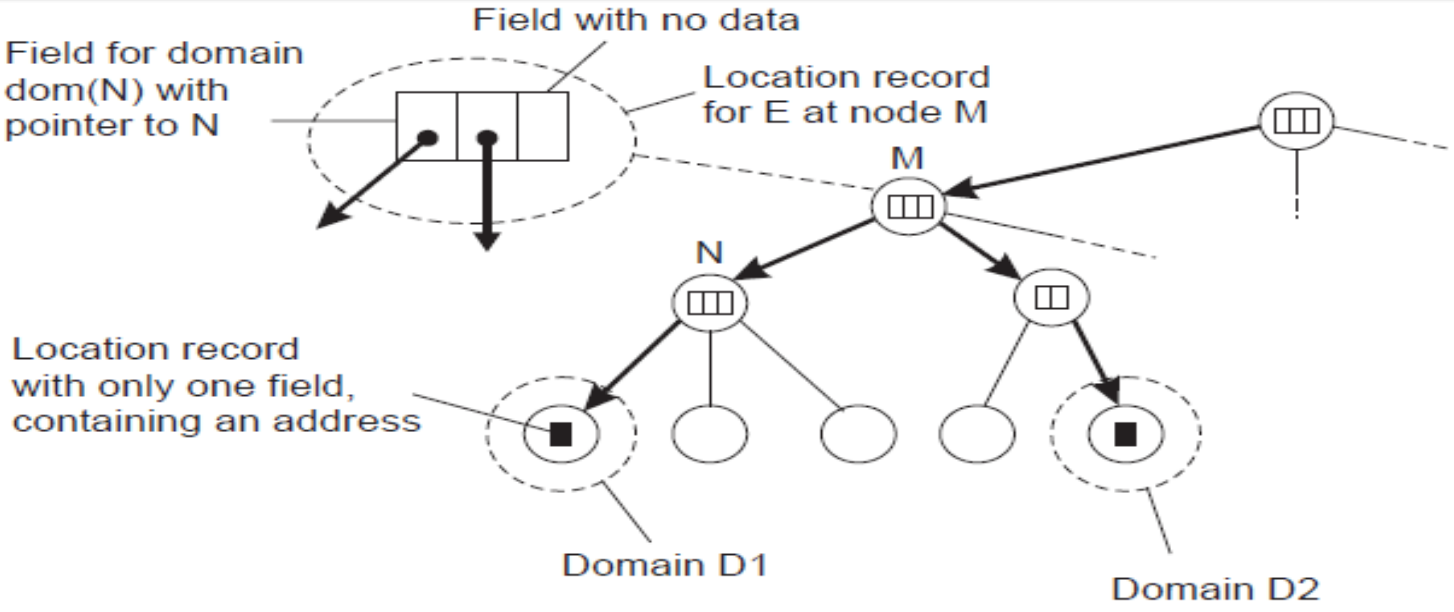
将底层网络划分为一组域，构建一个大规模的搜索树。每个域都拥有关联的目录节点DIR。



HLS：目录树组织

不变性

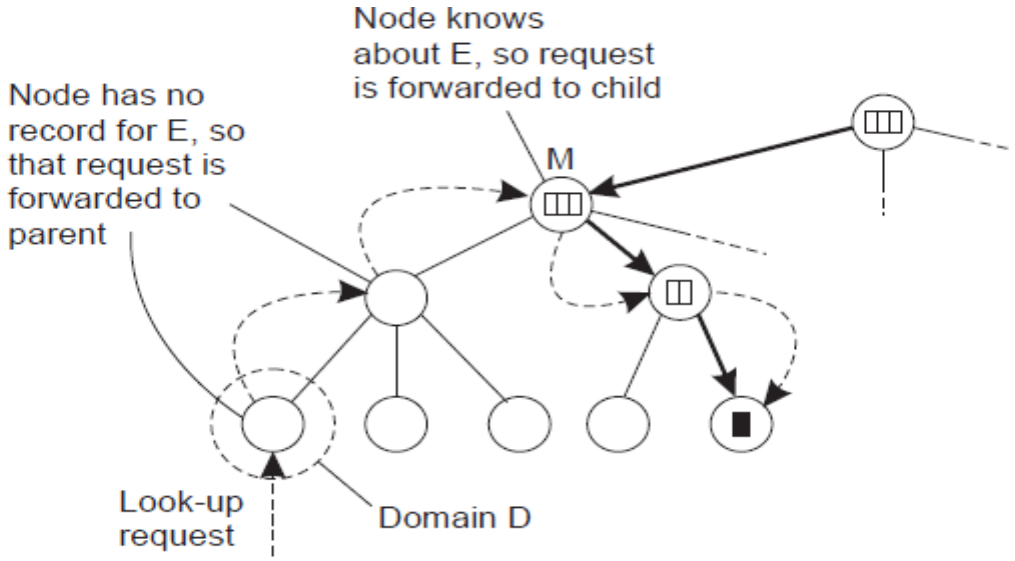
- 实体E的地址存储在叶域或者中间节点。
- 中间节点含有一个指向更底层子域目录节点的指针。
- 根节点掌握所有实体的信息。



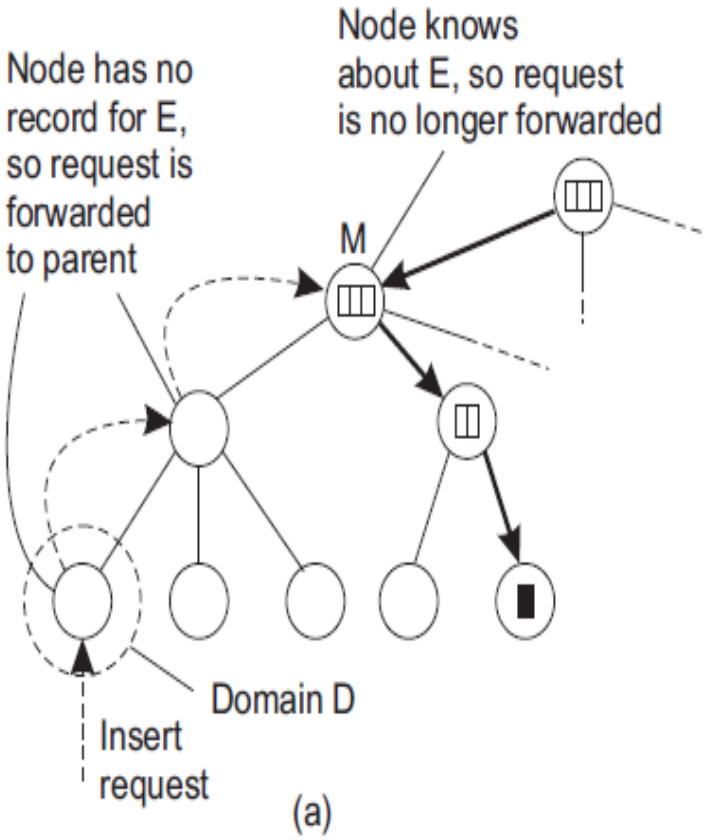
HLS：查找操作

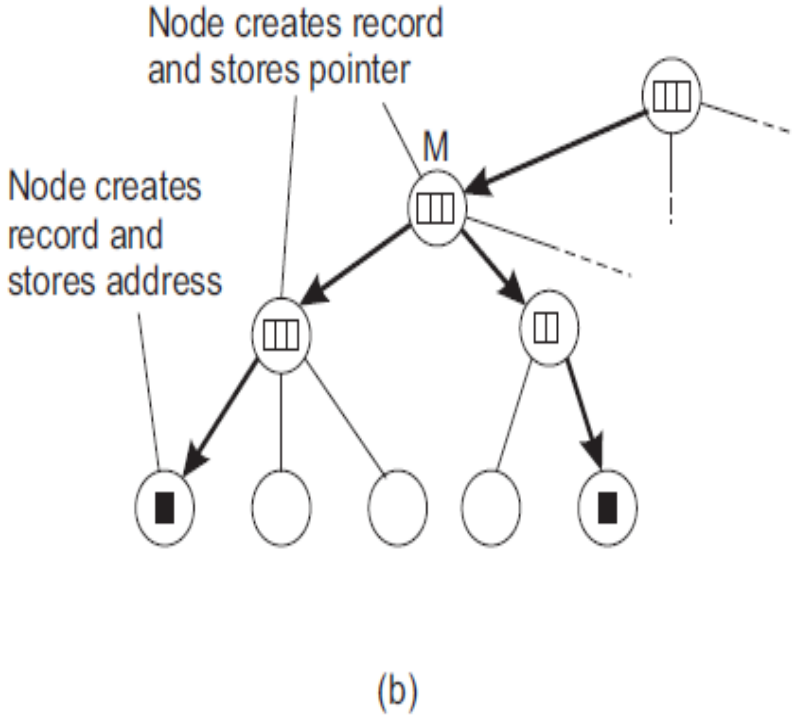
基本原理

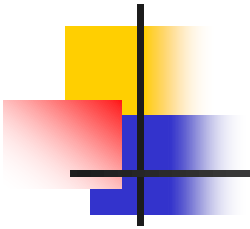
- 首先在客户所在叶域的目录节点查找实体E。
- 节点里含有实体E的位置记录=>跟随向下查找指针，
- 如果没有E的位置记录，继续下一步向父节点所在域查找实体E的位置信息，以此类推。



HLS：插入操作


Node has no record for E, so request is forwarded to parent
Node knows about E, so request is no longer forwarded
M
Domain D
Insert request
(a)


Node creates record and stores pointer
M
Node creates record and stores address
(b)



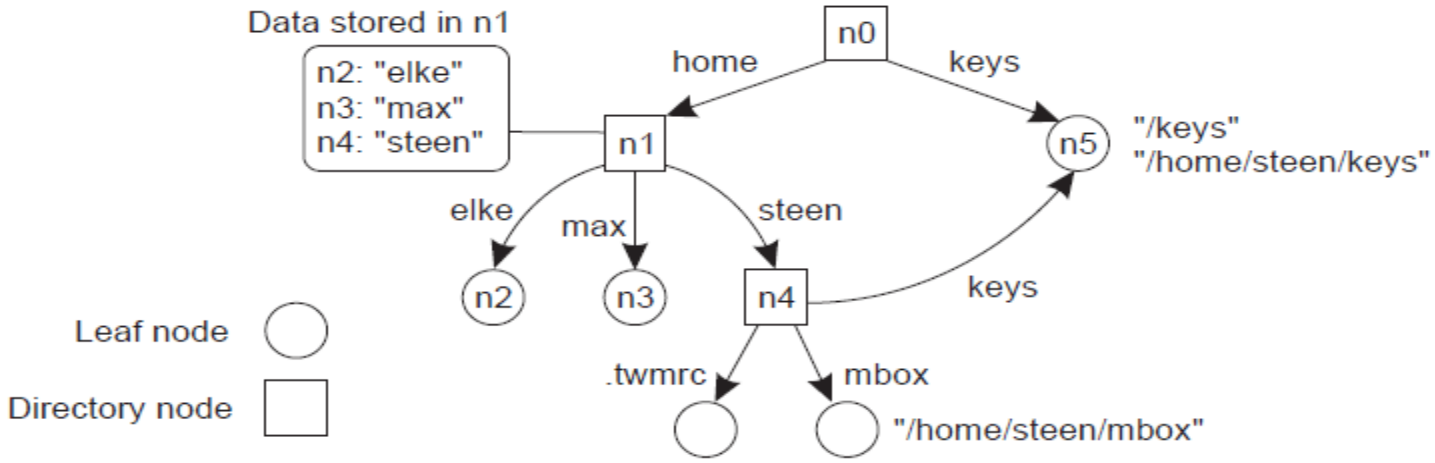
内容提要

- 名称、标识符和地址
- 无层次命名
- 结构化命名
- 基于属性的命名

5.3 结构化命名

- 命名空间本质

名称空间是一个有向图，其中叶节点代表一个实体。目录节点是指向其他叶节点的实体。



- 注意

目录节点用于存储一个表，其中每条分支边用一个对来表示。

5.3.1名称空间

- 结点通过名称存储各种属性，描述实体的各方面信息。
  - 实体类型
  - 实体标识符
  - 实体的位置信息
  - 别名

目录结点除了存储目录表外还能存储其他属性。

5.3.2名称解析

- 问题

  解析名称需要目录结点的信息，如何找到该结点。
- 终止机制

  终止机制如何处理从名称空间中选择初始结点。
  - www.cs.vu.nl:从域名服务器开始
  - /home/steen.mbox:从命名图的根结点的目录表开始。
  - 0031204447784：通过拨号
  - 130.37.24.8：路由到VU的web服务器
- 提问：为什么终止机制必须是隐式的？

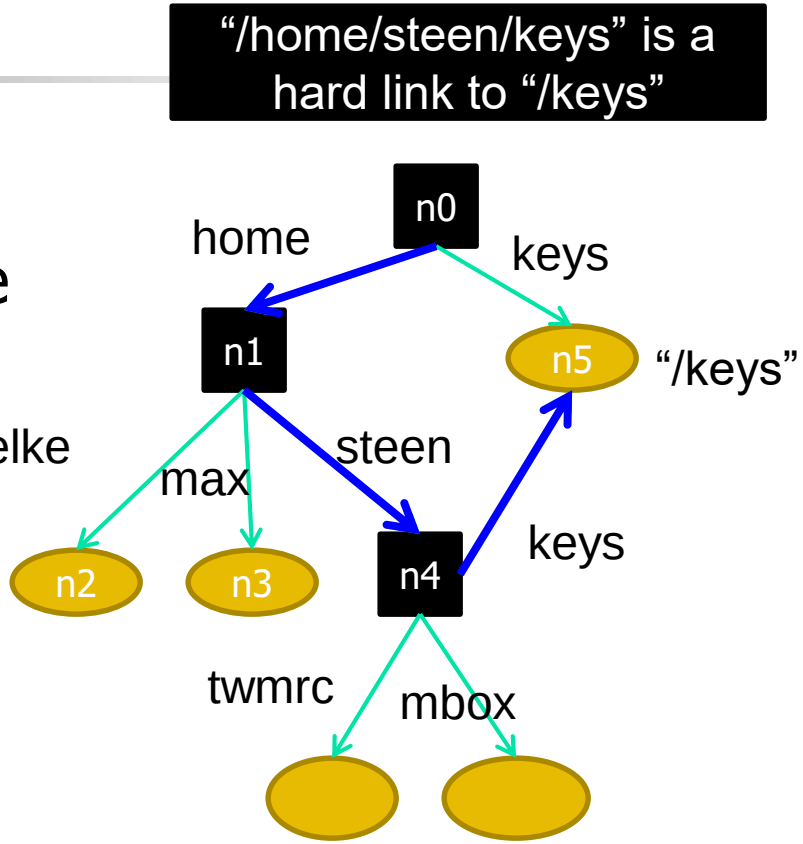
名称链接

- 硬链接 符号链接

  路径名：命名图中每个路径可以通过路径中的边的标签序列来指向，这样的序列称为路径名。

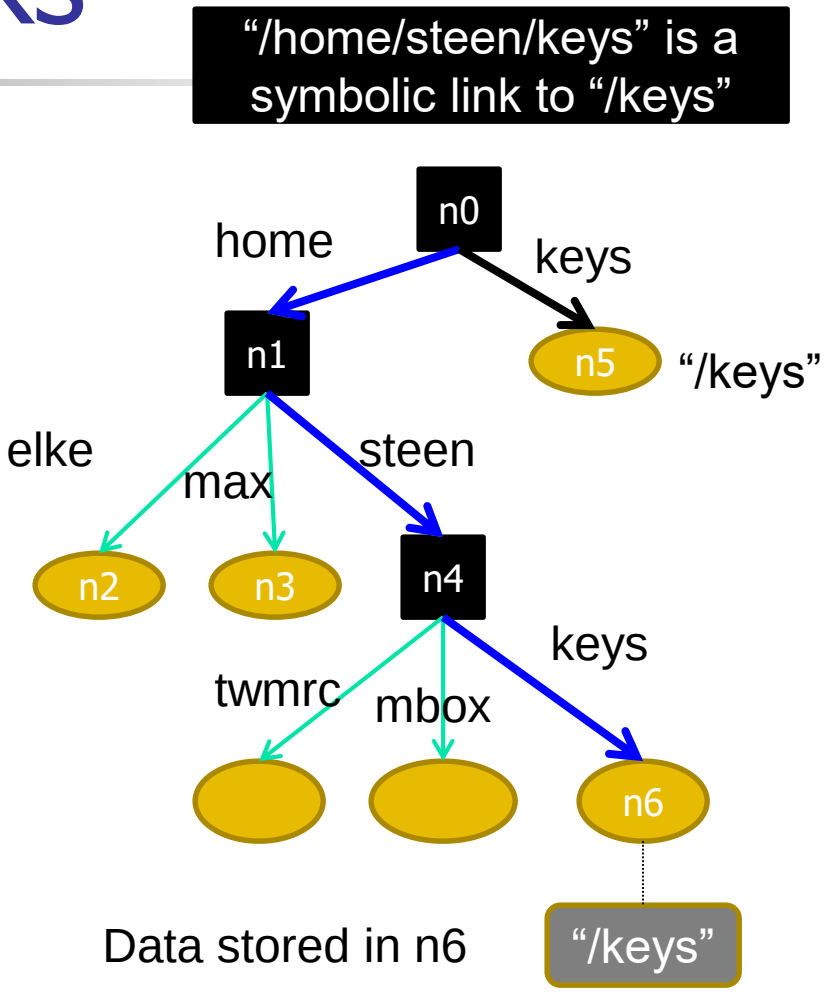
1. Hard Links

- There is a directed link from the hard link to the actual node
- Name Resolution
  - Similar to the general name resolution
- Constraint:
  - There should be no cycles in the graph

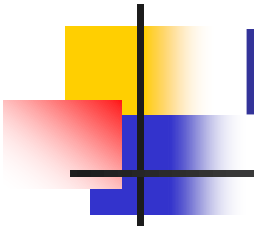


2. Symbolic Links

- Symbolic link stores the name of the original node as *data*
- Name Resolution for a symbolic link SL
  - First resolve SL's name
  - Read the content of SL
  - Name resolution continues with content of SL
- Constraint:
  - No cyclic references should be present

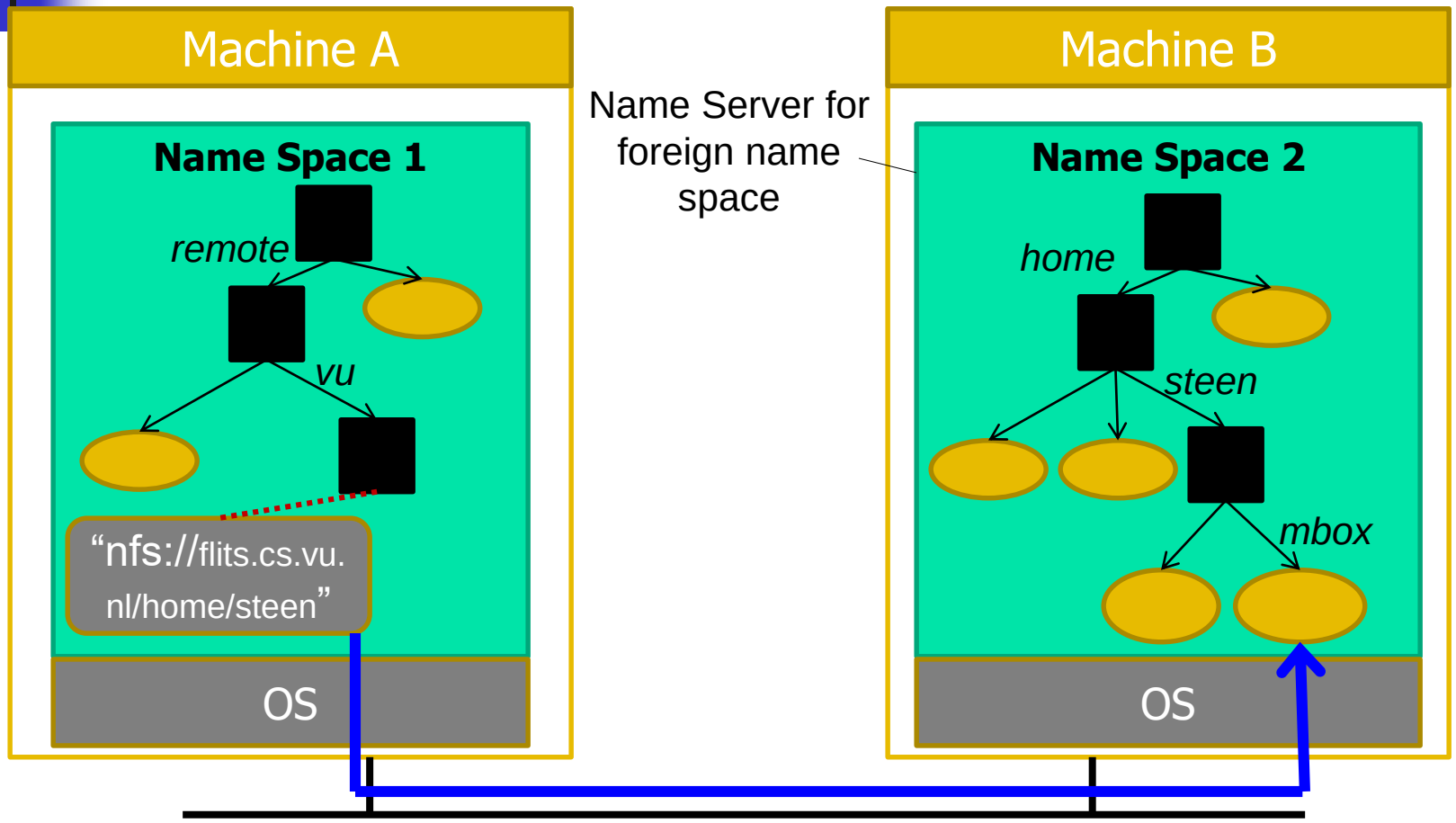


Mounting of Name Spaces



- Two or more name spaces can be merged transparently by a technique known as mounting
- In mounting, a directory node in one name space will store the identifier of the directory node of another name space
- Network File System (NFS) is an example where different name spaces are mounted
  - NFS enables transparent access to remote files

Example of Mounting Name Spaces in NFS



Name resolution for "/remote/vu/home/steen/mbox" in a distributed file system

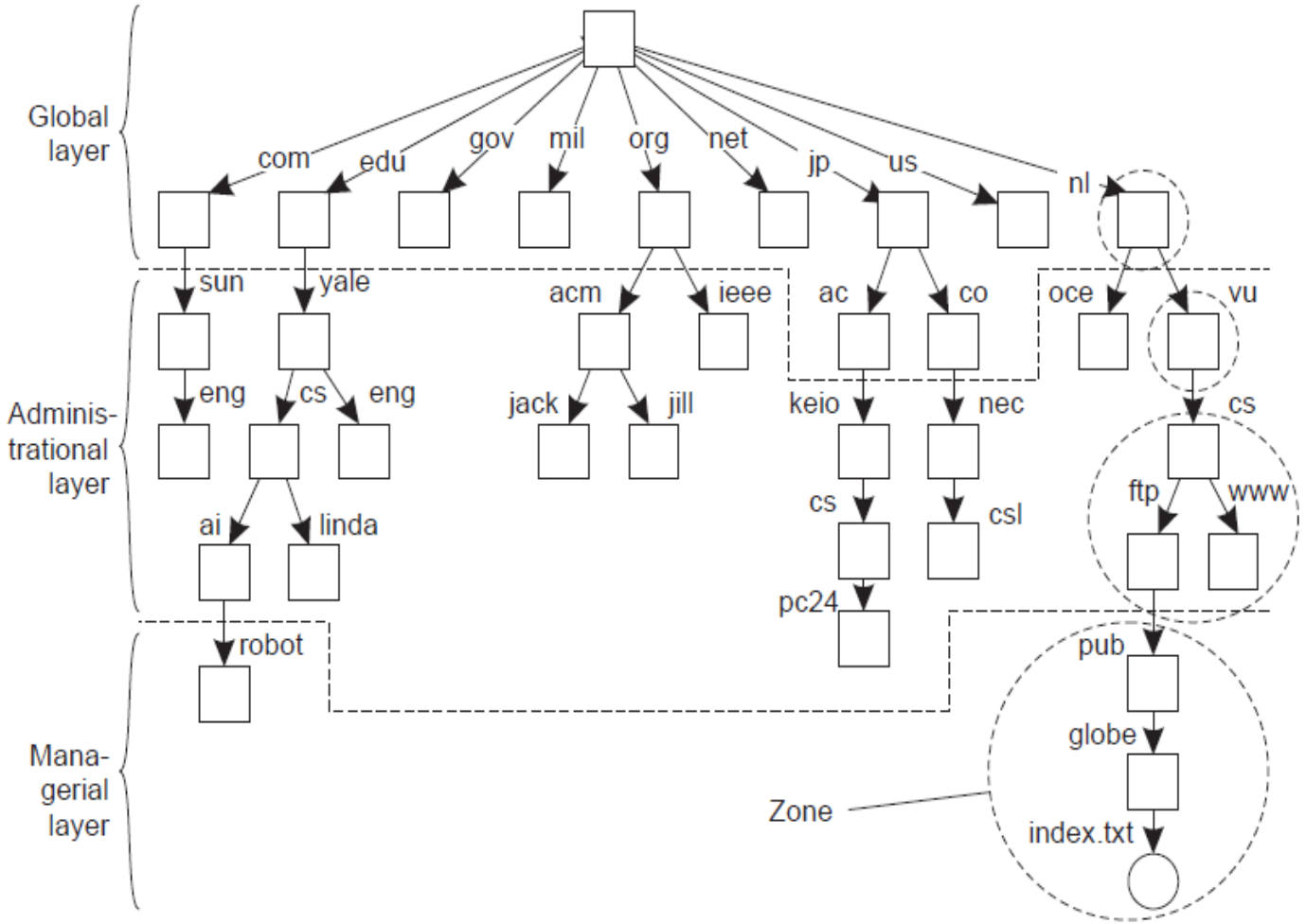
5.3.3名称空间的实现

- 基础问题

  通过将命名图结点分布存储实现分布式名称解析。

- 命名空间分层
  - 全局层：由最高级别的结点组成。
  - 行政层：由那些在单个组织内一起被管理的目录结点组成。
  - 管理层：由那些经常改变的结点组成。

5.3.3名称空间的实现


Global layer
Adminis-trational layer
Mana-gerial layer
com
edu
gov
mil
org
net
jp
us
nl
sun
yale
acm
ieee
ac
co
oce
vu
eng
cs
eng
jack
jill
keio
nec
cs
ftp
www
ai
linda
cs
csl
pc24
robot
pub
globe
index.txt
Zone

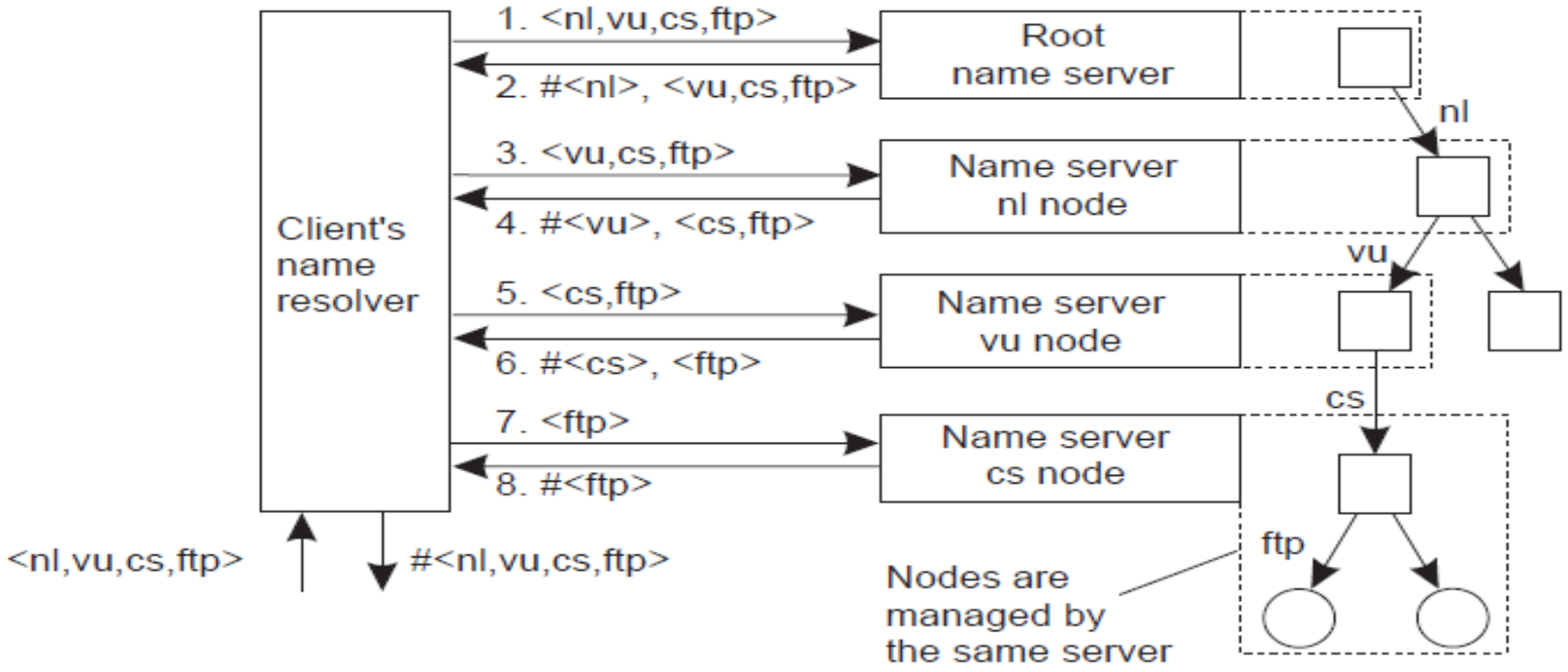
5.3.3名称空间的实现

Item	Global	Administrational	Managerial
1	Worldwide	Organization	Department
2	Few	Many	Vast numbers
3	Seconds	Milliseconds	Immediate
4	Lazy	Immediate	Immediate
5	Many	None or few	None
6	Yes	Yes	Sometimes

1: Geographical scale	4: Update propagation
2: # Nodes	5: # Replicas
3: Responsiveness	6: Client-side caching?

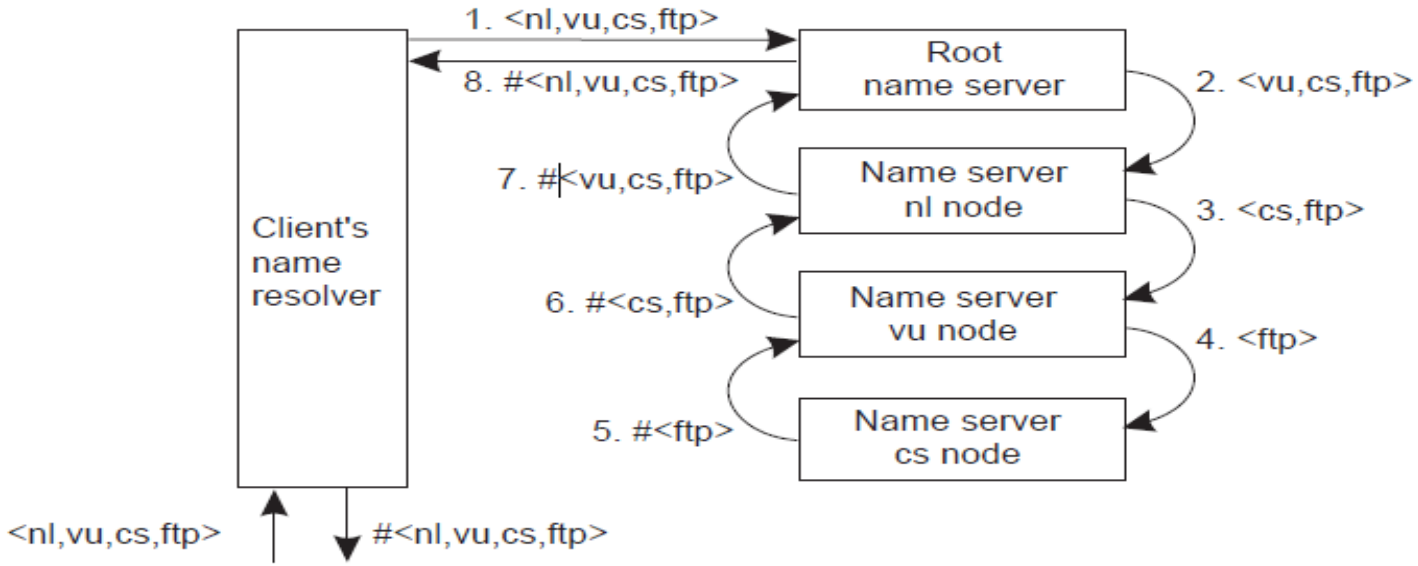
迭代名称解析

- 解析程序将完整名称(dir,[name1,...namek])发给server0。
- Server0将(dir,name)解析为dir1,返回sever1(存储dir1)的地址。
- Client发送解析(dir,[name1,...namek])请求给sever1。



递归名称解析

- 解析程序将完整名称(dir,[name1,...namek])发给server0。
- Server0将(dir,name)解析为dir1,发送解析(dir,[name1,...namek])请求给sever1返回sever1(存储dir1)的地址。
- Server0等待server1的响应，然后发送给client.



递归名称解析服务器缓存

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>
nl	<vu,cs,ftp>	#<vu>	<cs,ftp>	#<cs> #<cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>
root	<nl,vu,cs,ftp>	#<nl>	<vu,cs,ftp>	#<vu> #<vu,cs> #<vu,cs,ftp>	#<nl> #<nl,vu> #<nl,vu,cs> #<nl,vu,cs,ftp>

可扩展性

- 规模的可扩展性

  必须保证服务器在每个时间单元能处理大量的请求，特别对于高层的服务器。

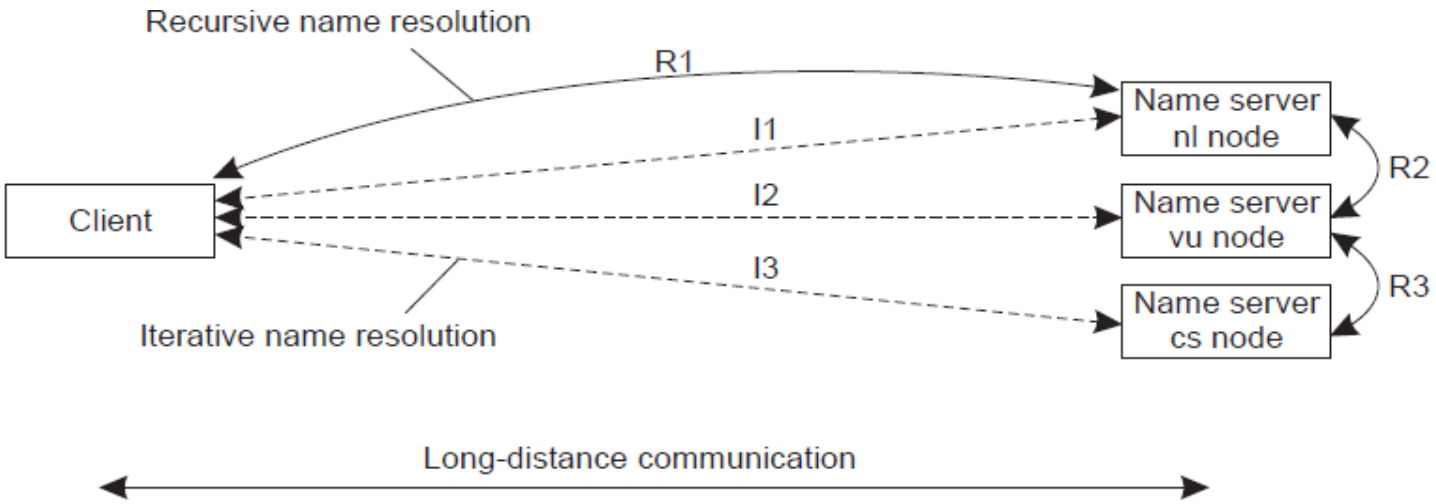
- 解决方案

  假设结点的信息稳定。就可以通过将结点映射到多个服务器来增加副本数量，以至于可以从最近的服务器进行名字解析。

可扩展性

- 地理跨度

  必须保证名字解析进程能**跨越**很大的**地理空间**。



- 问题

  通过将结点映射到服务器能定位到任何地方，我们引入隐含的位置从属。

示例：分布式域名系统

- 基本思想

  将**DNS全名**哈希成一个**键值k**，然后在基于DHT的系统中查询k值。缺点：不能在子域中查询所有的结点。

- 结点信息

SOA	Zone	Holds info on the represented zone
A	Host	IP addr. of host this node represents
MX	Domain	Mail server to handle mail for this node
SRV	Domain	Server handling a specific service
NS	Zone	Name server for the represented zone
CNAME	Node	Symbolic link
PTR	Host	Canonical name of a host
HINFO	Host	Info on this host
TXT	Any kind	Any info considered useful

DNS on Pastry

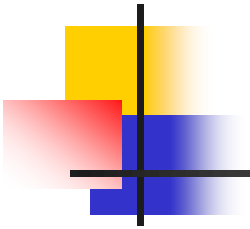
- Pastry

带有前缀的键值构成的基于DHT的系统。考虑一个4位键值的系统。一个键值ID为3210追踪以下的结点。

n_k	prefix of ID(n_k)	n_k	prefix of ID(n_k)
n_0	0	n_1	1
n_2	2	n_{30}	30
n_{31}	31	n_{33}	33
n_{320}	320	n_{322}	322
n_{323}	323		

- 注意

结点3210负责处理前缀为321的键。如果获得一个键值为3012的请求，它将转发给结点N30。对于DNS：负责键值k的结点用一个哈希值存储DNS记录。



内容提要

- 名称、标识符和地址
- 无层次命名
- 结构化命名
- 基于属性的命名

5.4 基于属性的命名

- 概论

  在分布式系统中，描述实体最常用的方法是利用属性。

- 问题

  查找操作很耗时，因为它要求**匹配**请求的**属性值**而不是实体的值。

- 解决方案

  结合传统的结构化命名，将目录服务作为数据库

5.4.1 目录服务

- 是指基于属性的命名系统，而支持结构化命名的系统通常称为命名系统
- 属性设计因人而异
- 引入资源描述框架（resource description framework RDF）

基于属性的命名系统基本上要求对**所有资源描述**进行穷举搜索。

5.4.2 Light-weight Directory Access Protocol (LDAP)

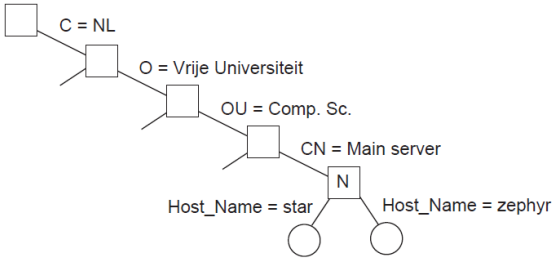
- LDAP Directory Service consists of a number of records called "directory entries"
  - Each record is made of (attribute, value) pair
  - **LDAP Standard** specifies five attributes for each record
- Directory Information Base (DIB) is a collection of all directory entries
  - Each record in a DIB is unique
  - Each record is represented by a distinguished name

    e.g., /C=NL/O=Vrije Universiteit/OU=Comp.

Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Comp. Sc.
CommonName	Main server
Host_Name	star
Host_Address	192.31.231.42

分层实现：LDAP

- All the records in the DIB can be organized into a hierarchical tree called *Directory Information Tree (DIT)*

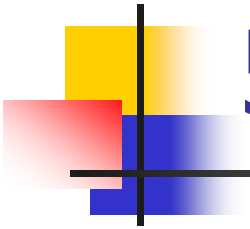


Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Comp. Sc.
CommonName	Main server
Host_Name	star
Host_Address	192.31.231.42

Attribute	Value
Country	NL
Locality	Amsterdam
Organization	Vrije Universiteit
OrganizationalUnit	Comp. Sc.
CommonName	Main server
Host_Name	zephyr
Host_Address	137.37.20.10

- LDAP provides advanced search mechanisms based on attributes by traversing the DIT
- Example syntax for searching all Main_Servers in Vrije Universiteit:

```
search("&(C = NL) (O = Vrije Universiteit) (OU = *) (CN = Main server)")
```



5.4.3 非集中式实现

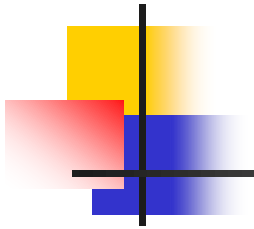
- 映射到分布式散列表
- 语义覆盖网络

映射到分布式散列表

- 将资源描述给出的（属性，值）对转换为DHT的一个键值
- 键值中的某些位用于表示属性，某些位用于表示值。

语义覆盖网络（semantic overlay network）

语义相近的邻结点及其链接构成语义覆盖网。



END