第5讲 命名系统

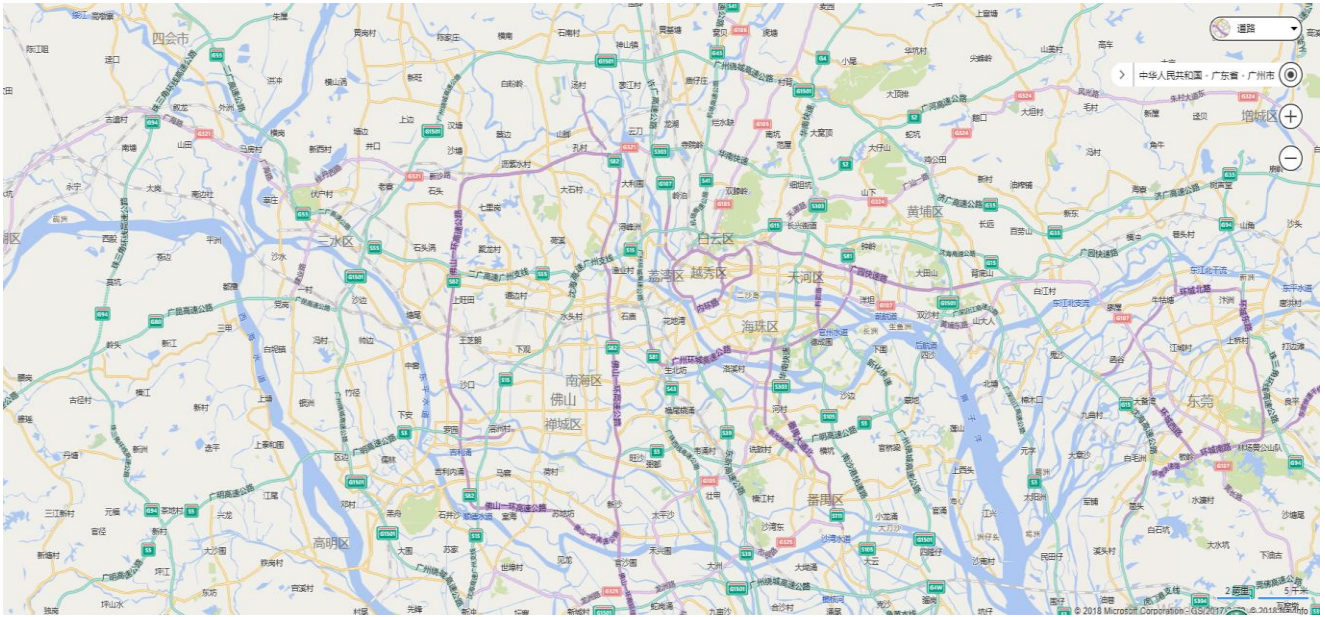
§5.1 基本概念

§5.2 无结构命名

§5.3 结构化命名

§5.4 基于属性的命名

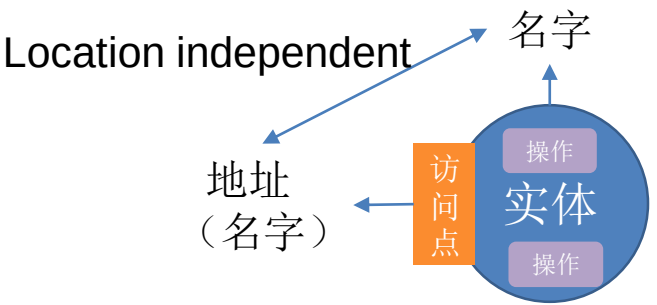
§5.1 基本概念



分布式系统 VS 地图

名称/名字

- 名称Name：
  - 用名字标识分布式系统中的实体对象
  - 由字符组成的字符串
- 实体Entity：
  - 可以是分布式系统中的任何事物，抽象的或者实际的
  - 如：主机、用户、消息等。
- 访问点Access point：
  - 访问实体的的入口或途径
  - 是一种特殊实体
- 地址Address：
  - 用于访问实体的地址
  - 其实是实体的访问点的名称



标识符（Identifiers）

- 用于唯一标识实体的名称
  - 一个标识符至多引用一个实体；
  - 每一个实体最多由一个标识符引用；
  - 一个标识符始终引用一个实体
    - 标识符不会重新使用
- 一个标识符可以是随机字符串，也可以是有特定内容的字符串
- Machine readable vs. Human friendly
  - E.g. hash based id, domain name

命名系统

- 实现名称到地址的绑定
  - 即name-to-address的对应表
- 命名：绑定名称和地址的过程
- 解析：标识符解析为地址的过程
- 分布式命名系统
  - 分布式在多台机器上的命名系统

§5.2 无结构命名

- Flat/Unstructured names
  - 命名只是单纯的标识符，可能是随机生成的串
  - 名字完全不反映位置/地址信息
  - 命名简单
- 关键问题：如何解析?
  - 简单方案：广播、转发指针
  - 宿主地址
  - 分布式哈希表
  - 分层定位

1. 简单命名解析—广播

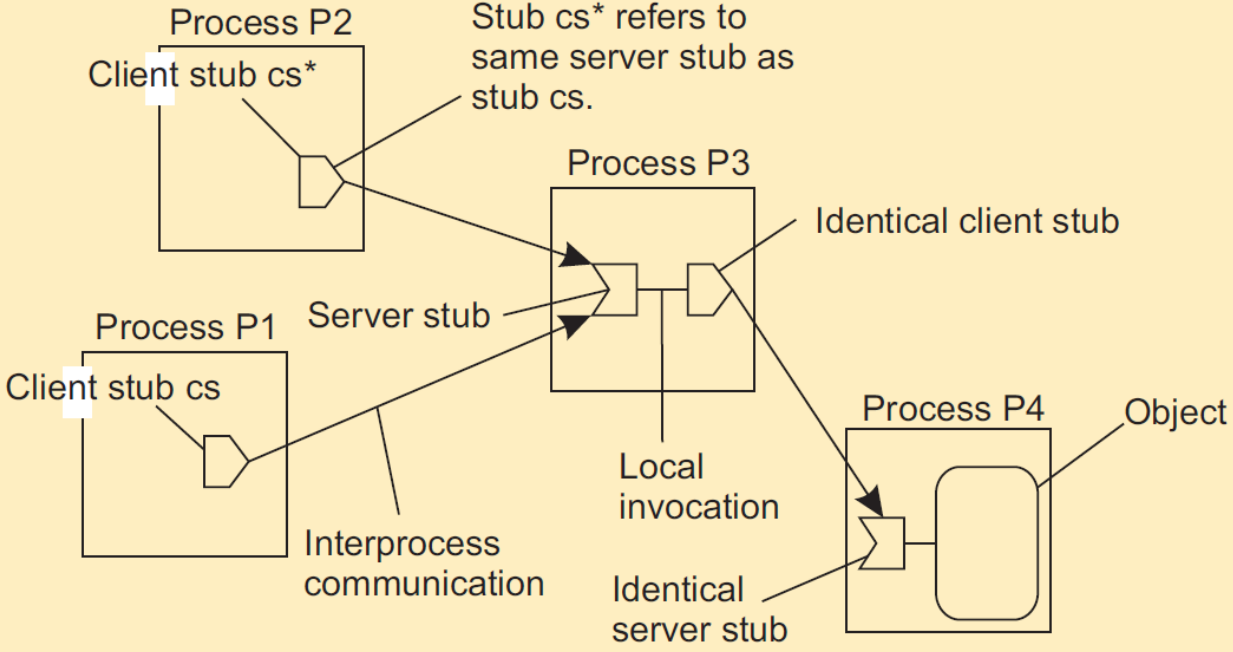
- 广播：广播ID，请求实体返回其当前地址
  - 难以逾越局域网
  - 效率低：浪费实体资源处理非目标消息
  - 采用多播：可以降低通讯开销
- 例子：地址解析协议（ARP）
  - 找出与 IP 地址相关的MAC地址
  - 广播查询“who has this IP address?”

1.简单命名解析—转发指针

- 适用于“移动性”实体
- 当实体移动时，在当前位置留下到下一个位置的指针
- 引用过程简单：只需要沿着指针链搜索
  - 当实体的当前地址找到后，更新客户端的引用
- 扩展性问题（地理可扩展性）：
  - 较长的传播链很脆弱，容易断开
  - 实体的定位开销比较大

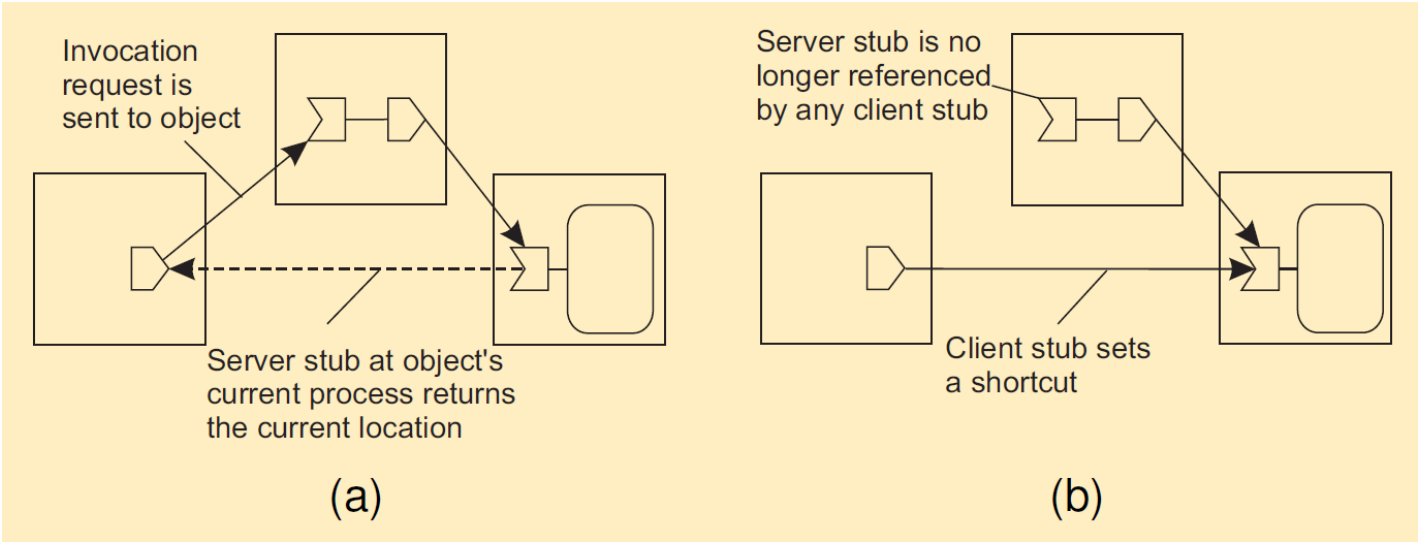
SSP (Stub Scion Pairs)

- RPC转发（client stub, server stub）



SSP (Stub Scion Pairs)

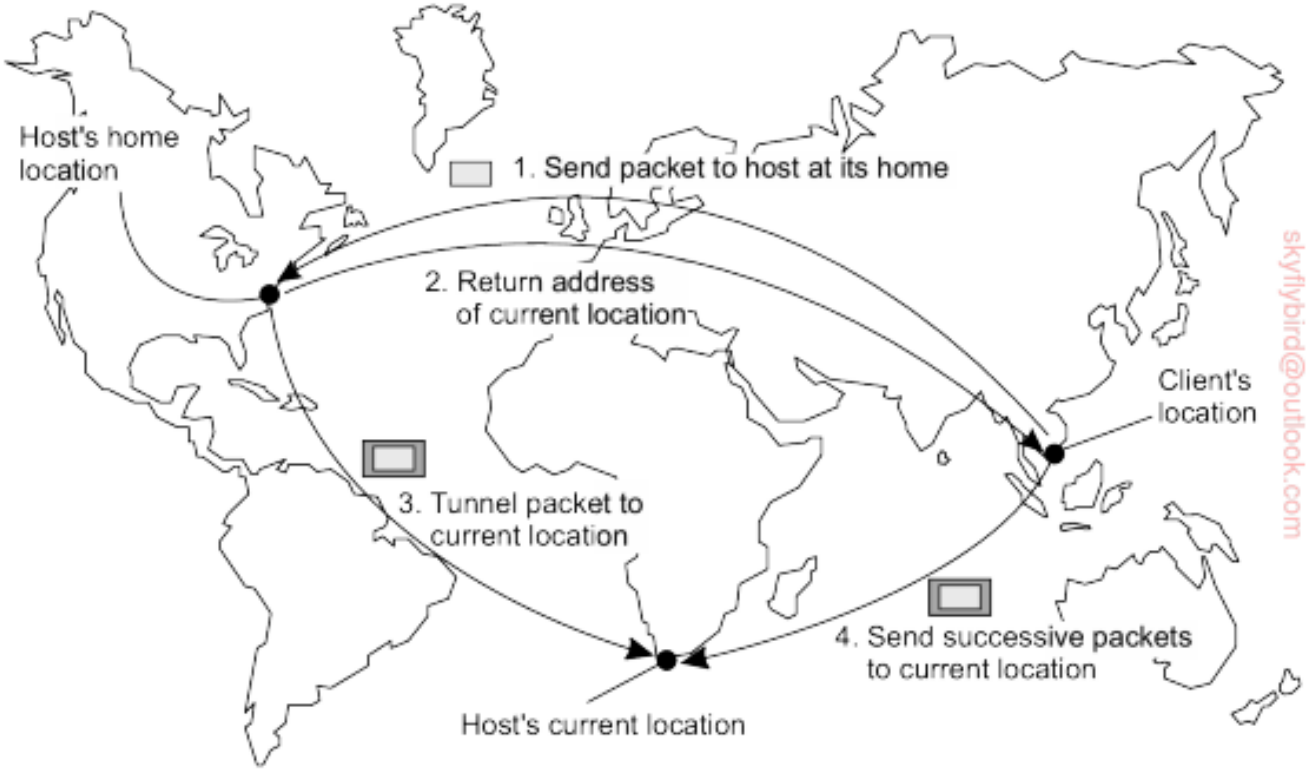
- 指针更新



2. 基于宿主地址的解析

- 宿主机记录实体的当前地址（外部地址）
- 宿主地址注册在命名服务上
- 客户端需要先联系宿主机，然后与外部地址联系
- 本质：单层转发指针
- 主要问题：
  - 宿主机需要伴随实体的整个生命周期；
  - 宿主机的地址是固定的 => 如果实体对象永久迁移会带来问题；
  - 较差的地域可扩展性 （实体可能就在客户端旁边）；

移动IP机制

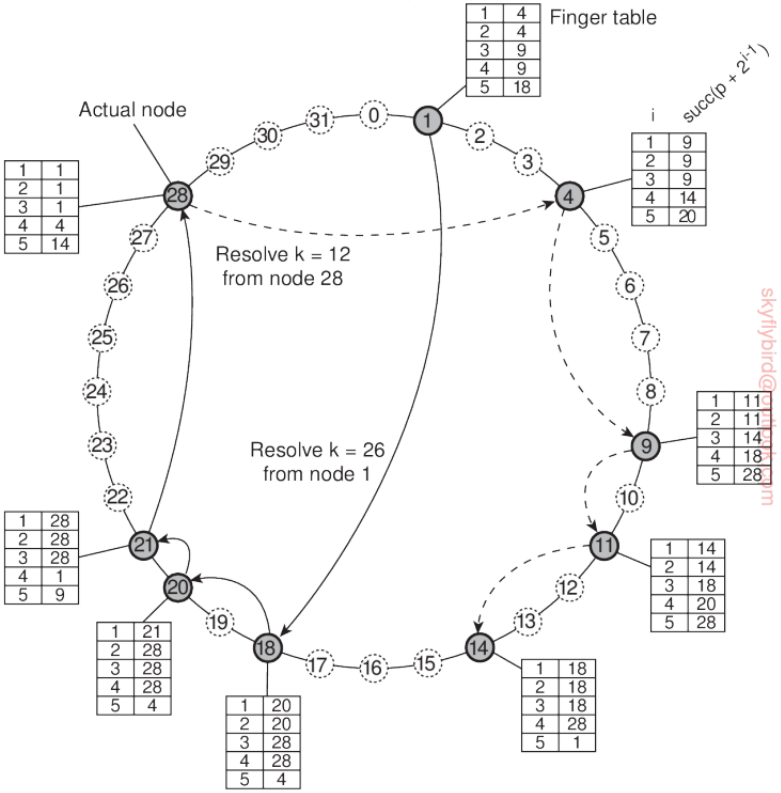


3. 分布式哈希表

- Distributed hash tables
- 基本机制：基于哈希值的环行结构
  - 每一个节点被赋予一个由m位（哈希值）构成的标识符;
  - 每一个实体被赋予一个唯一的m位（哈希值）的健值;
  - 健值为 k 的实体存储在满足 id >= k 的最小标识符节点上，成为 k 的后继者, succ(k)。
- 搜索（即解析）
  - 每一个节点记录它的邻居，并且进行线性的搜索。

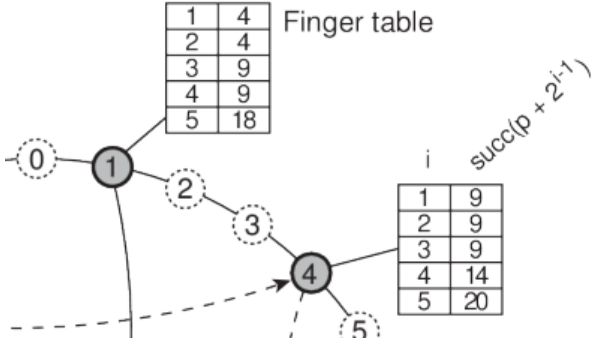
Chord

- Chord结构：通过finger table提高搜索效率



Chord

- 指状表实为搜索捷径
- 根据哈希值范围确定
  - 关键：不会错过目标实体



- Each node p maintains a finger table $FT_p[]$ with at most m entries:

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

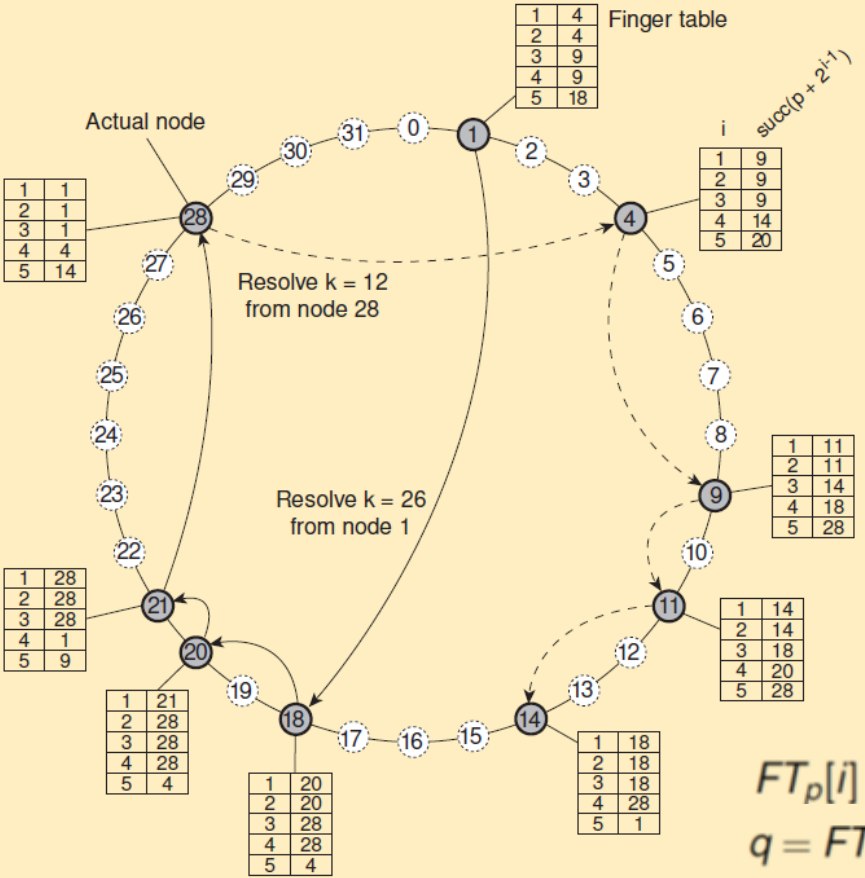
Note: the i-th entry points to the first node succeeding p by at least 2^{i-1}.

- To look up a key k, node p forwards the request to node with index j satisfying

$$q = FT_p[j] \leq k < FT_p[j+1]$$

- If $p < k < FT_p[1]$, the request is also forwarded to $FT_p[1]$

Chord



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

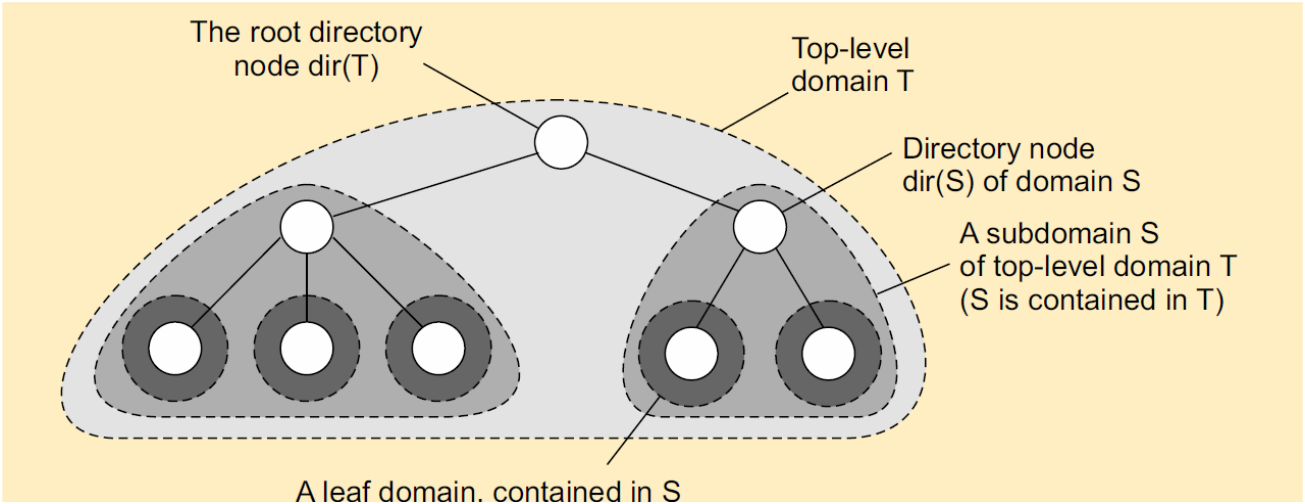
$$q = FT_p[j] \le k < FT_p[j+1]$$

其他DHT

- 其他DHT结构
  - CAN、Pastry
- DHT优点：简单、扩展性好、容错好
- 负载均衡问题：可以通过虚节点机制解决
- 物理拓扑不匹配：融合网络物理信息，如：
  - 基于拓扑的节点标识符赋值： 物理邻近节点赋值近。
  - 邻近路由优化：每个节点有多个后继者，查询时选择最近的一个。
  - 邻节点选择优化：优化路由表，使得选择最近的节点作为邻结点。

4. 分层定位

- 创建一个大规模的搜索树
- 底层的网络被划分成多个分层的域
- 每一个域由一个目录节点表示



4.分层定位

- 树型结构
  - 实体 E 的地址存储在叶节点或者相应的中间节点上
  - 中间结点包含了指向其孩子节点的指针
    - 当且仅当根植于孩子节点的子树存储有实体的地址
    - 如果有实体有副本则会有两个独立指针
  - 根节点的指针也是指向其子节点
- 一个实体可以有多个地址（副本）

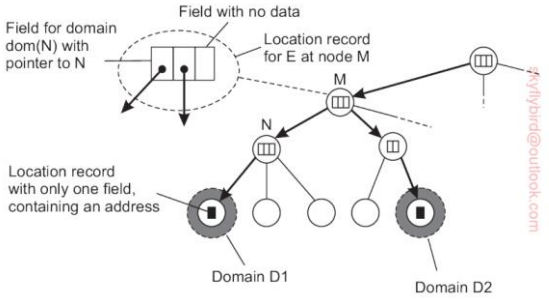
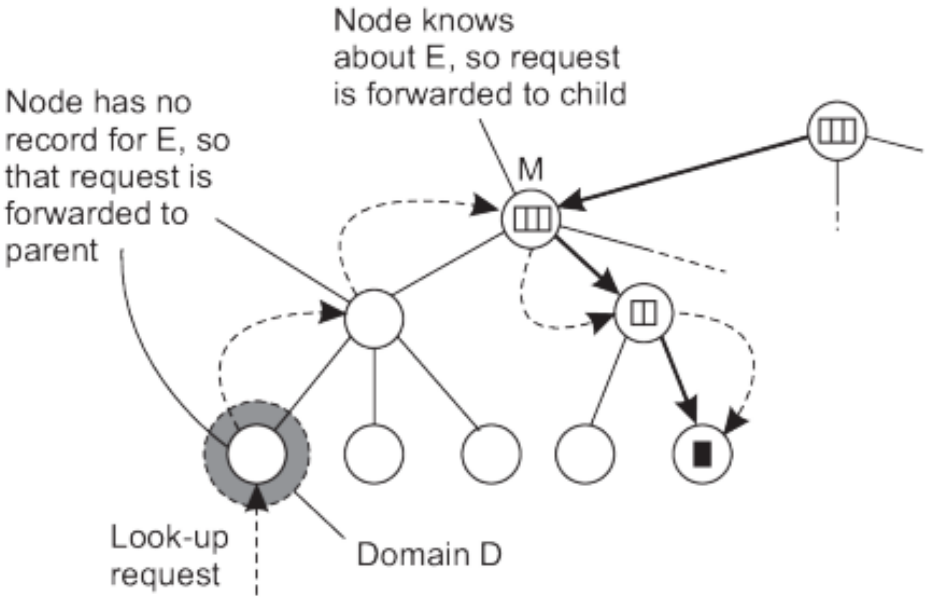


Figure 5.7: An example of storing information of an entity having two addresses in different leaf domains.

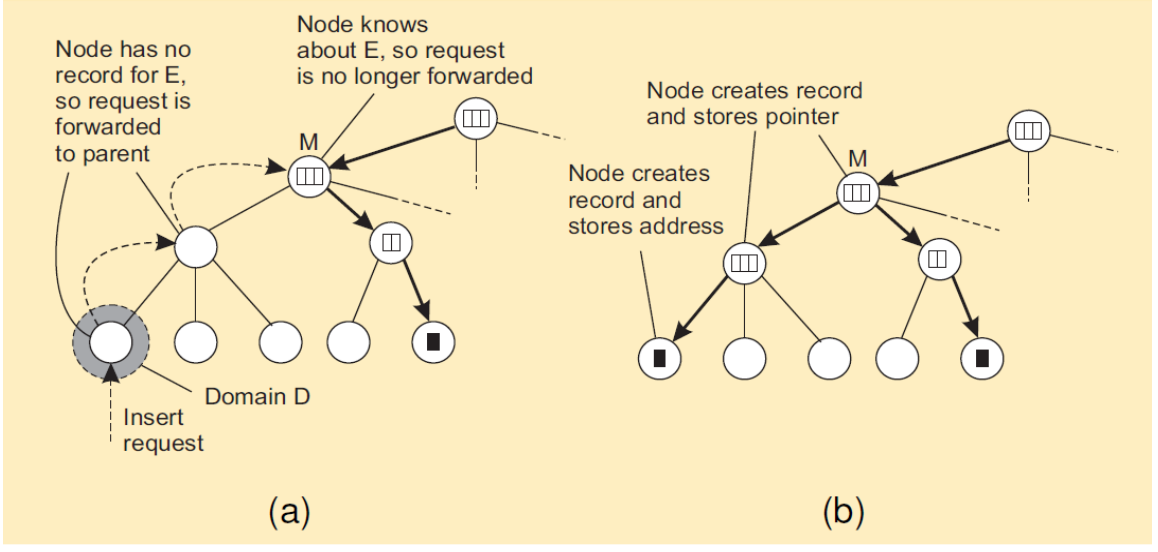
4.分层定位

- 搜索查找过程
  - 从叶子节点开始往上搜索，直到有节点知道目标实体
  - 然后沿此节点向下搜索到目标。



4.分层定位

- 插入新节点
  - 插入请求被转发到知道实体 E 地址的第一个节点
  - 建立一条从第一个知道实体 E 地址的节点到实体 E 的指针链

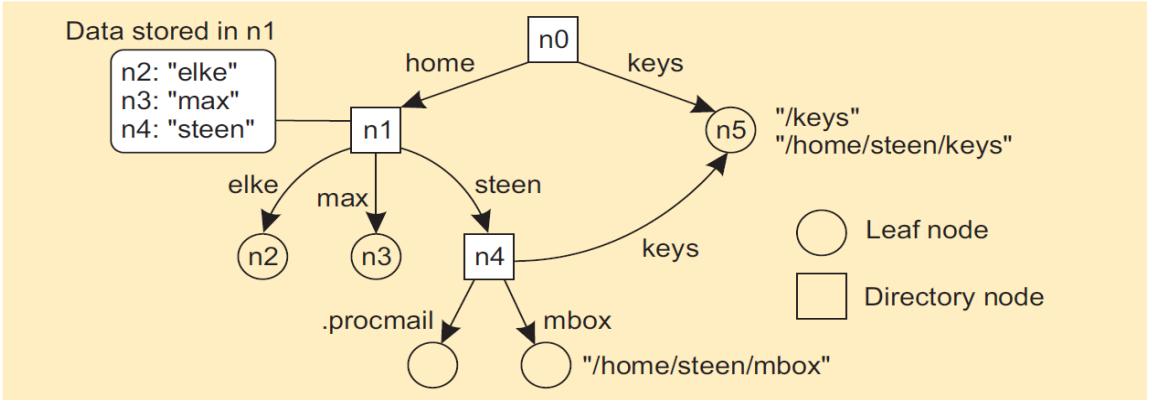


§5.3 结构化命名

- 根据一定结构或者规则对实体进行命名
- Human-readable names
- 如：
  - 基于目录结构的文件命名
  - 基于层次的主机名

1.命名空间

- 命名空间：命名图（naming graph）
  - 一张叶节点表示实体的图
  - 叶节点还可以存储实体的属性、状态等信息
  - 目录节点具有一定数量的边，用于引用其他节点
- 例：一张通用的具有一个根节点的命名图



2.名称解析

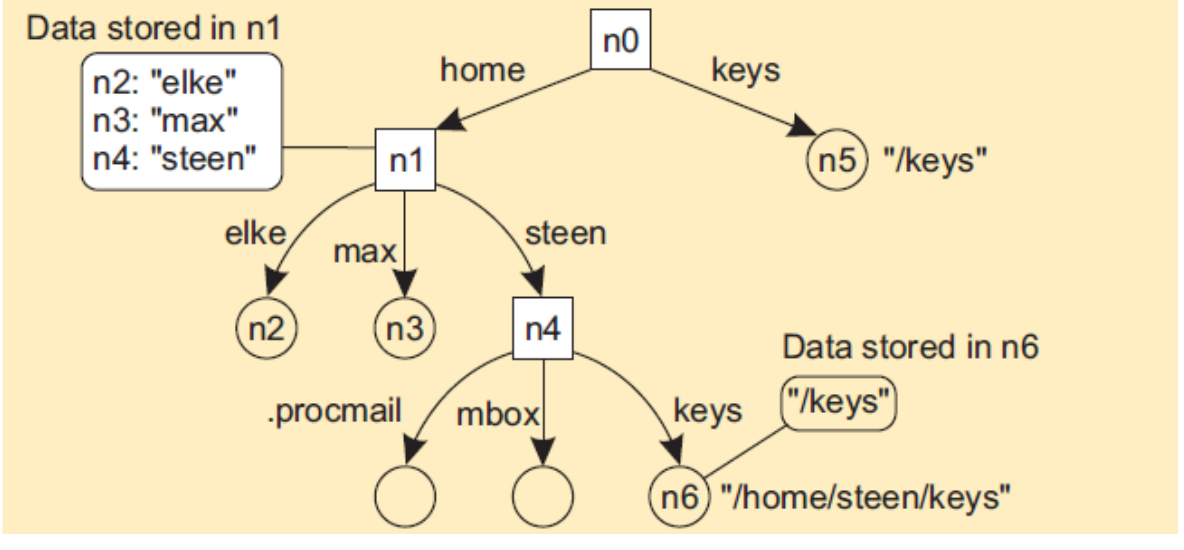
- 名称解析：
  - 给定一个路径名，查找出由该名称所指向的节点的地址
- 为了解析一个名字我们需要一个目录节点
- 问题：该如何找到初始节点?
- 答案：闭包（Closure）机制
  - 知道如何启动以及在何处启动名称解析的机制
- 几个例子
  - www.distributed-systems.net, 从DNS服务器开始；
  - /home/marten/mbox: 从本地的NFS服务器开始；
  - 13587569903：拨打电话号码；
  - 222.200.145.180: 把消息路由到特定的IP地址；

名称链接

- 硬链接（Hard link)
  - 我们所描述的路径名即用于在命名图中按照特定路径搜索节点的名字就是“硬链接”；
- 软链接（Soft link)
  - 允许一个节点 N 包含另外一个节点名字
  - 用叶节点表示实体，而不是存储实体的地址和位置，该节点存储绝对路径名。

名称链接

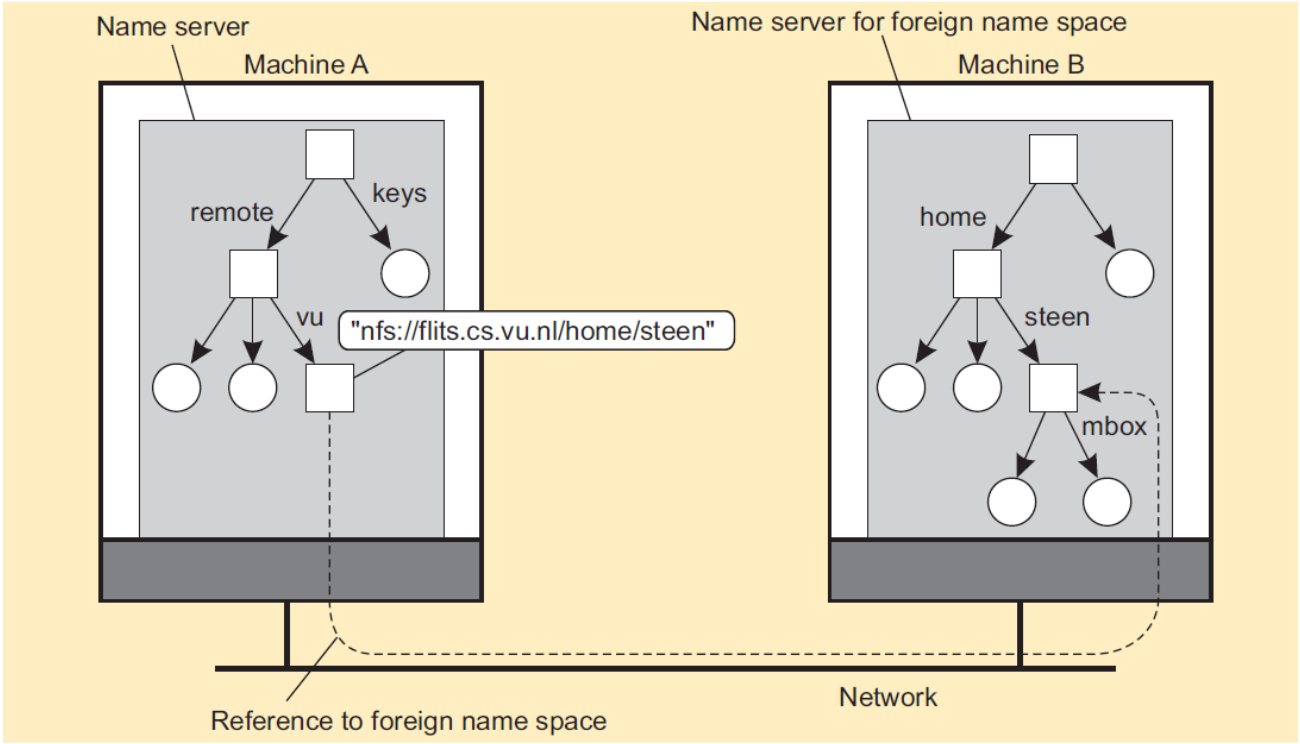
- 软链接解析
  - 首先解析 N 的名字;
  - 读取 N 的内容返回名字;
  - 利用返回的名字进行名字解析;



挂载—命名空间关联/合并

- 挂载
  - 将当前命名空间节点与另一空间的节点标识符相关联
- 外部命名空间
  - 需要访问的命名空间;
- 挂接点
  - 在当前命名空间中用于存储节点标识符的目录节点;
- 挂载点
  - 外部名称空间中的目录节点称为挂载点;
  - 挂载点是命名空间的“根”;

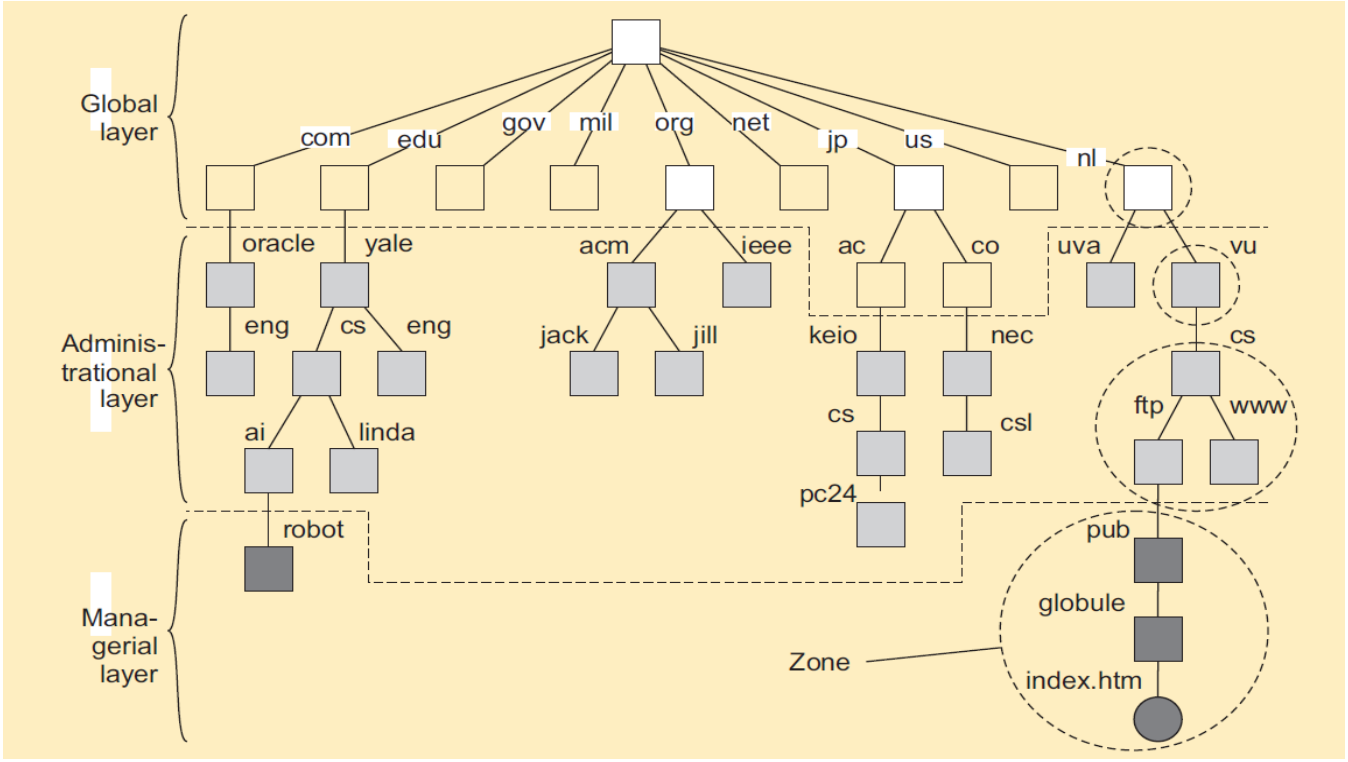
NFS挂载


Name server
Machine A
Name server for foreign name space
Machine B
remote
keys
vu
"nfs://flits.cs.vu.nl/home/steen"
home
steen
mbox
Network
Reference to foreign name space

3.命名空间实现

- 命名空间分布化
  - 将命名图分布在多个节点上实现命名管理和解析；
- 经典的三层命名空间
  - 全局层：由最高级别的节点构成，即由根节点以及其他逻辑上靠近根节点的目录节点组成。
    - 特点：稳定，目录表很少改变，可以代表组织或者组织群。
  - 行政层：由那些在单个组织内一起被管理的目录节点组成。
    - 特点：代表属于同一组织或行政单位的实体组；相对稳定，但是比全局层的目录变化频繁；
  - 管理层：由经常改变的节点组成。
    - 如代表本地主机的节点及本地文件系统等，由终端用户维护。

DNS命名空间


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

命名服务器特点

- 全局层服务器
  - 变化不频繁，缓存作用大
  - 可用性要求高、响应速度要求不高、吞吐量要求高
  - 一般可以通过复制服务器实现
- 行政层服务器
  - 可用性要求最高（影响整个机构的用户）
  - 性能要求与全局层类似（缓存仍然高效）
    - 但是数据更新反应时间要短
  - 性能相对好的服务器可以实现
- 管理层服务器
  - 可用性要求不高（影响范围小）
  - 性能要求高：响应要快（数据更新多，缓存效果差）

命名服务器特点

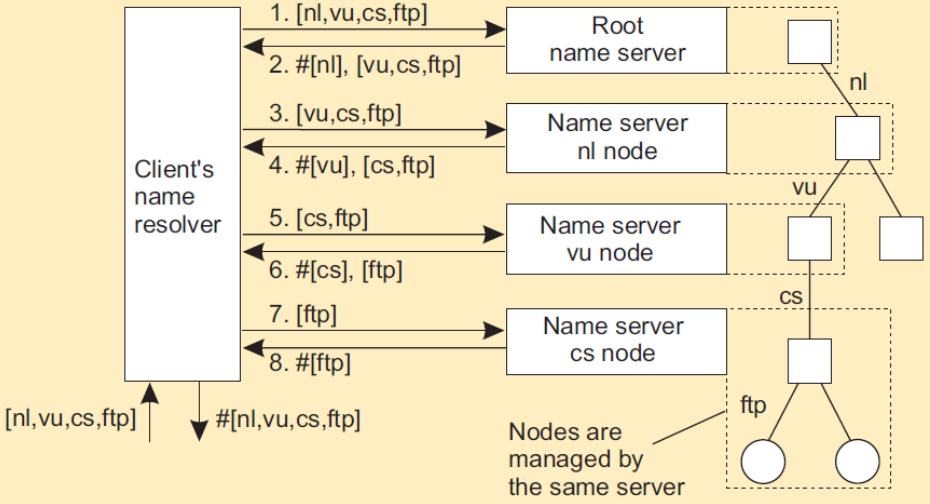
Item	Global	Administrational	Managerial
1	Worldwide	Organization	Department
2	Few	Many	Vast numbers
3	Seconds	Milliseconds	Immediate
4	Lazy	Immediate	Immediate
5	Many	None or few	None
6	Yes	Yes	Sometimes

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

名称解析实现

- 迭代式命名解析

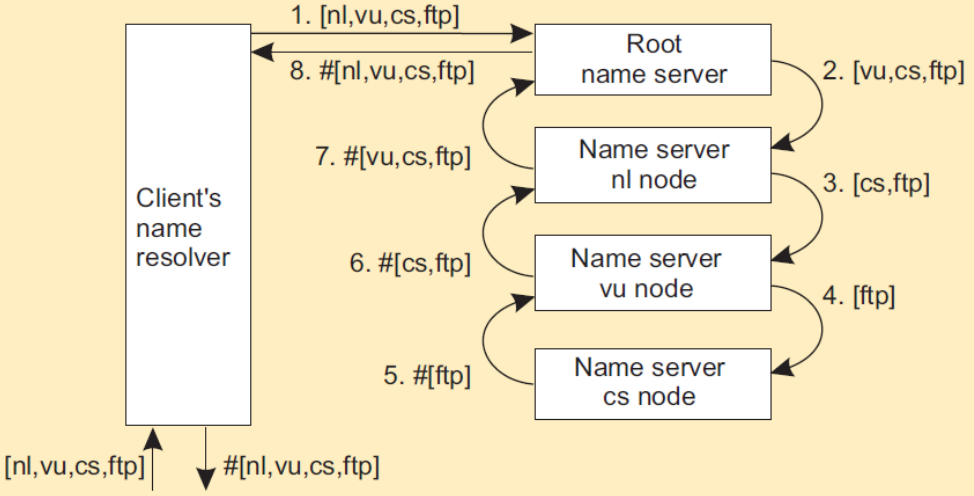
1. $resolve(dir, [name_1, ..., name_K])$ sent to $Server_0$ responsible for dir
2. $Server_0$ resolves $resolve(dir, name_1) \rightarrow dir_1$, returning the identification (address) of $Server_1$, which stores dir_1.
3. Client sends $resolve(dir_1, [name_2, ..., name_K])$ to $Server_1$, etc.



名称解析实现

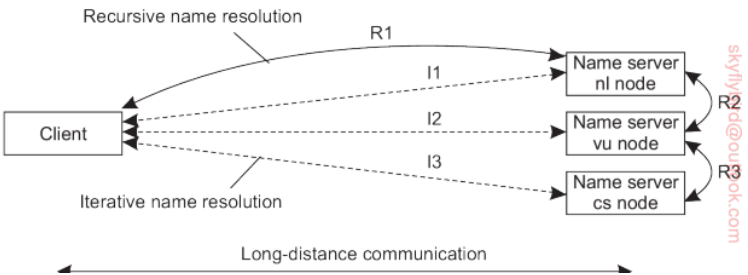
- 递归式命名解析

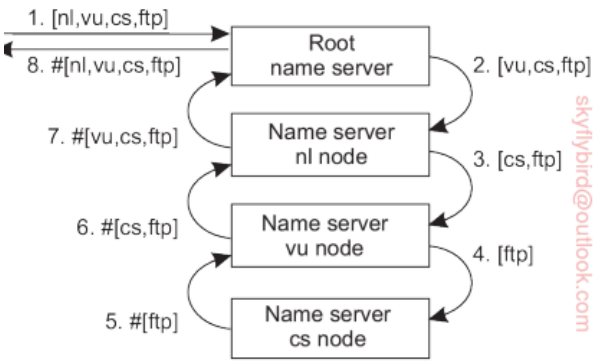
1. *resolve*(*dir*, [$name_1$, ..., $name_K$]) sent to $Server_0$ responsible for *dir*
2. $Server_0$ resolves *resolve*(*dir*, $name_1$) → dir_1, and sends *resolve*(dir_1, [$name_2$, ..., $name_K$]) to $Server_1$, which stores dir_1.
3. $Server_0$ waits for result from $Server_1$, and returns it to client.



名称解析实现

- 递归式命名解析优点鲜明
  - 缓存机制更有效（服务器端缓存）
  - 通信开销低（服务器间距离近）





§5.3 基于属性的命名

- 基于属性的命名也称为目录服务
- 需求：
  - 在很多场景中，通过实体的属性查询实体很方便
  - 如：传统电话号码目录簿、门户网站站点列表
- 问题/难点：
  - 查找操作代价高：需要进行属性值匹配和比对

1.目录服务

- 属性构建（命名）
  - 定义属性、值
  - 基本需要手工设定
  - 规范化描述，如RDF（ resource description framework ）
  - <Person, name, Alice>:
    - a resource named Person whose name is Alice
- 名称存储
  - 一般用集中式节点
    - 方便穷尽搜索，保证搜索的完备性

分布式目录服务

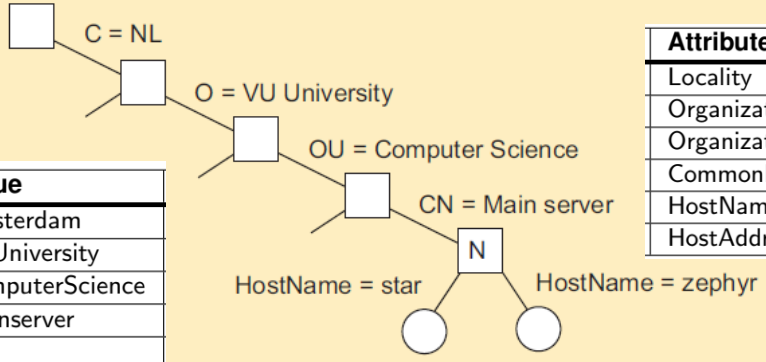
- 提高目录服务扩展性
- 可以有不同方案
  - 层次化结构，如LDAP
  - 去中心化结构

2. LDAP

Essence

- Directory Information Base: collection of all directory entries in an LDAP service.
- Each record is uniquely named as a sequence of naming attributes (called Relative Distinguished Name), so that it can be looked up.
- Directory Information Tree: the naming graph of an LDAP directory service; each node represents a directory entry.

Part of a directory information tree



Attribute	Value
Locality	Amsterdam
Organization	VUUniversity
OrganizationalUnit	ComputerScience
CommonName	Mainserver
HostName	star
HostAddress	192.31.231.42

Attribute	Value
Locality	Amsterdam
Organization	VUUniversity
OrganizationalUnit	ComputerScience
CommonName	Mainserver
HostName	zephyr
HostAddress	137.37.20.10

LDAP

- 与DNS非常类似
- DSA： **directory service agents**
  - 实际存储DIT的服务器，一般有多个
  - DIT分区存储：类似DNS
  - 与一般的命名服务器类似，但是提供搜索操作
- DUA：**directory user agents**
  - It exchanges information with a DSA to resolve names according to a standardized access protocol

LDAP

- 与DNS的主要差别
  - 支持更丰富的搜索查询服务
  - 基于属性查询
- 对DIB进行搜索
  - 基于属性条件
- 如：
  - 查询列出VU大学的所有main server

search(“(C=NL)(O=VU University)(OU=*)(CN=Main server)”)

LDAP

- *Forest* of LDAP
  - Allowing several trees to co-exist, while also being linked to each other.
  - 如：微软的Active Directory
  - 搜索变得复杂
    - 可以设置一个global的index服务先确定要搜索的DIB
- LDAP+DNS
  - Root of LDAP is known under a DNS name

3.去中心化目录服务

- Directory service in the P2P mode
- 一般的P2P只支持key/index查询
- 如何实现属性<attribute, value>查询?
  - <attribute, value>-to-key matching
- Two possible solutions
  - Distributed index server
  - Space-filling curves

分布式索引服务器

- 假设一共d个不同属性
- 每个属性配置一个server
  - 属性A的server维护（实体，属性值）列表
  - (E, val)表示实体E存储有A属性值为val的内容
- 例子：
  - 查询被发给三个server：Country、Organization、CommonName
  - Client基于三个服务器的结果选择交集
  - 然后Client发送进一步查询给相应实体

```
search("(Country=NL)(Organization=VU University)
       (OrganizationalUnit=*)(CommonName=Main server)")
```

空间填充曲线

- 基本思路：
  - 将N个属性构成的N维空间映射为1维线性空间
  - 然后可以用P2P网络，如Chord，直接处理
- 关键问题：
  - 值相近的(attribute, value) 尽量匹配给同一个server (locality)
- 方案：
  - 空间填充曲线

Hilbert Space-filling Curves

- 基本思路：
  - 切分空间->分配Index->连线填充
  - 可以递归重复（多order）
- 查询搜索：空间匹配->线段匹配

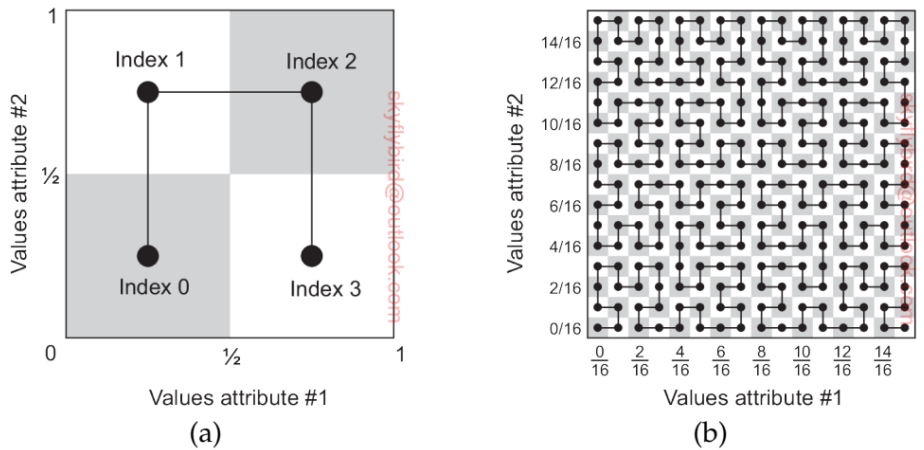


Figure 5.30: Reducing a two-dimensional space to a single dimension through a Hilbert space-filling curve of (a) order 1, and (b) order 4.

Summary

- 无层次命名：解析是关键
  - 广播/多播、转发指针、宿主位置、DHT、搜索树
- 结构化命名：基于路径来访问
  - 名称空间/命名图
  - 名称服务器：迭代vs.递归
- 基于属性命名：目录服务
  - <属性，值>
  - 集中式
  - 层次化：LDAP
  - 去中心化