Distributed Systems

Name and Directory Services

Prof. Dr. Oliver Hahm

Frankfurt University of Applied Sciences
Faculty 2: Computer Science and Engineering
oliver.hahm@fb2.fra-uas.de
https://teaching.dahahm.de

21.05.2024

Introduction

- **Numerical identifiers** and **addresses** (e.g., IP addresses) are difficult to memorize for human users
- Names provide an *abstraction* of the concrete services and objects

→ But how can we get from a name to its address?

- Remember: In the previous chapter we saw how a binder deals with **naming** and **locating** for an RPC system
- More general we require:
  - Name services
  - Directory services
- Name resolution does exist in other IT contexts (e.g., variable name → memory address)

Agenda

- Names
- Name Services
- Directory Services
- Location Services

Agenda

- Names
- Name Services
- Directory Services
- Location Services

What is a name?
Which properties could a name have?

Names

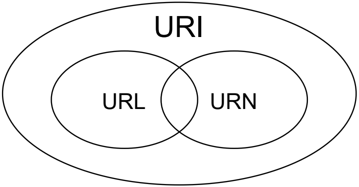
- Names
  - Names are used to identify objects (e.g., a resource or service)
  - A name is a *sequence of bits or characters*
  - Binding: the process which binds a name to an object
- Name properties
  - unique: a name identifies exactly one object unambiguously
  - pure: a name is only a bit pattern and does not convey any other information
  - impure: a name implies additional information about the specified object

Examples

- unique
  - "‘Joe Smith"’ is not unique
    → A name in combination with birthday and location of birth is typically unique
  - UUIDs (Universally Unique Identifiers) are unique
    - 128 bit number
    - Specified in RFC 4122 and ITU-T Rec. X.667 | ISO/IEC 9834-8:2005
    - Various versions exist
    - Can be generated, for instance, by the tool `uuidgen`
    - In Microsoft ecosystem also called Global Unique Identifier (GUID)
    - Example: `123e4567-e89b-12d3-a456-426614174000`
- pure
  - UUIDs as names of DCOM objects or classes are pure
- impure
  - DNS names often imply additional information
    - e.g., `mail.fra-uas.de`

URI, URL, and URN

- A Uniform Resource Identifier (URI) identifies a specific resource, e.g., `ISBN 978-1543057386`
- A Uniform Resource Locator (URL) additionally specifies how the resource can be accessed, e.g., `https://www.libra-buchhandlung.de/shop/item/9780131217867`
- A Uniform Resource Name (URN) identifies a specific resource in a persistent and location-independent way, e.g., `urn:isbn:978-1543057386`

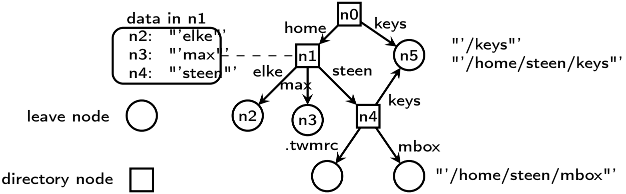


Name Spaces

- Names have their meaning only in a certain context
- Names are structures in name spaces
- Name spaces define the syntax rules for the names
- **Examples:** Name spaces in C++, DNS, ISBN . . .

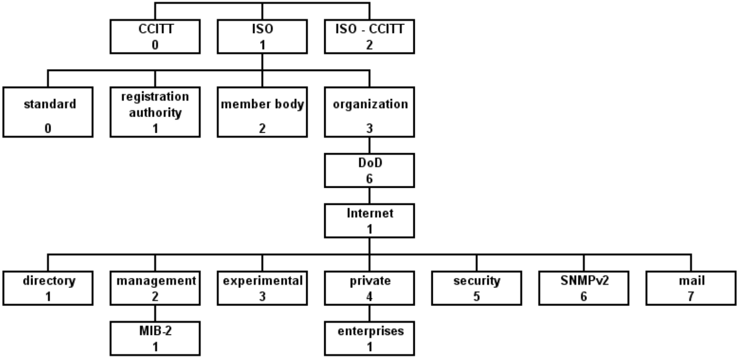
Name Space Structures

- Flat name spaces (less common today, e.g., Unix UIDs)
- Hierarchical name spaces are typically organized as **directed graphs** with labels
- In these name spaces the context is given by the prefix
  - Directory nodes and leaves
  - Absolute and relative paths
  - Global and local names
- **Example:** Unix file system name spaces



Example: MIB-2 Name Space

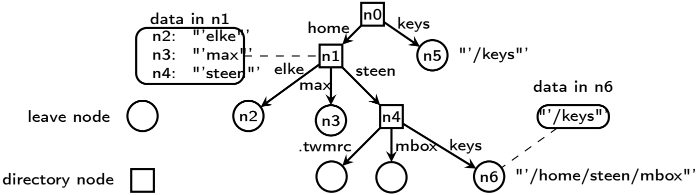
- **MIB: Management Information Base**



https://upload.wikimedia.org/wikipedia/commons/1/1c/SNMP.MIB-Tree.PNG

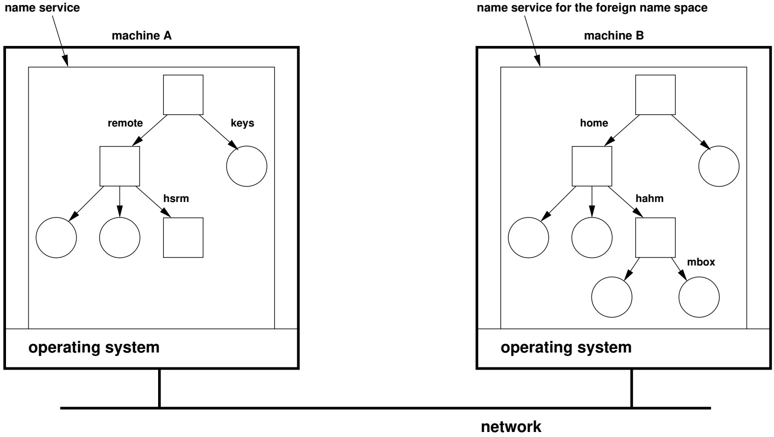
Links in a Name Space

- The name of an object is another name
- Forwarding or mapping of a name to another name
- **Example:** Unix soft link



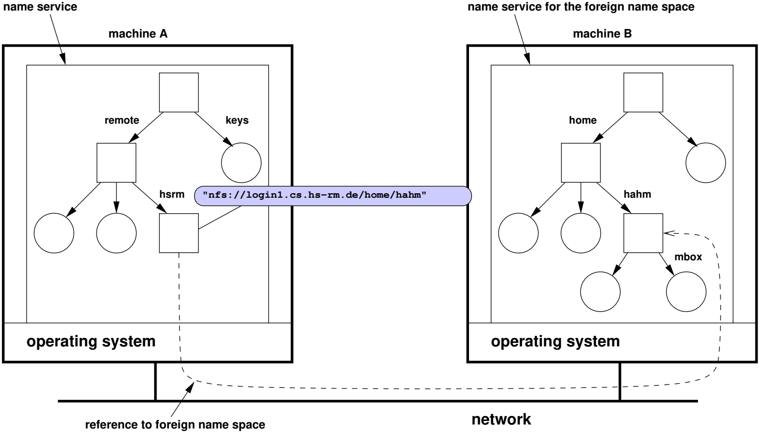
Links into Another Name Space

- Mounting



Links into Another Name Space

- Mounting


name service
machine A
remote
keys
hsrm
"nfs://login1.cs.hs-rm.de/home/hahm"
operating system
name service for the foreign name space
machine B
home
hahm
mbox
operating system
reference to foreign name space
network

Organisation of Huge Name spaces

- To manage huge name spaces effectively, these are typically divided into three layers:
    - **Global Layer**
        - High-level nodes (entry points)
    - **Administrative Layer**
        - Name spaces within an organization
    - **Managerial Layer**
        - Name spaces for names that frequently change
- Properties:

	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

Example: DNS (Domain Name Service)

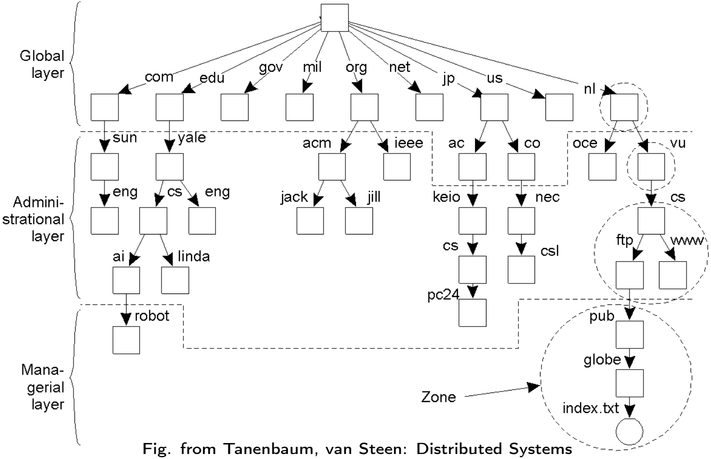


Fig. from Tanenbaum, van Steen: Distributed Systems

Addresses

- Addresses are attributes of names which can be used to interact with or access objects
    - Examples for addresses
        - Street, house number, city
        - Phone number
        - IP address or (IP address, port number)
        - Memory address
- Advantage for the use of names over addresses
    - Location independent (preferable)
    - Easier to memorize
    - Abstracts from many (protocol) details of an address

Agenda

- Names
- Name Services
- Directory Services
- Location Services

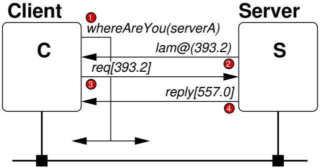
Name Services

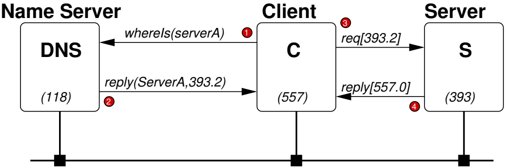
- Name resolution:
  - Process to find the address property for a given name of an object
- Name service:
  - Provides name resolution for requesting clients
  - For RPC systems it is also called binder
  - Typical operations:
    - Register/Bind
    - Deregister/Unbind
    - Resolve/Lookup

How could we implement name resolution?

Types of Name Resolution

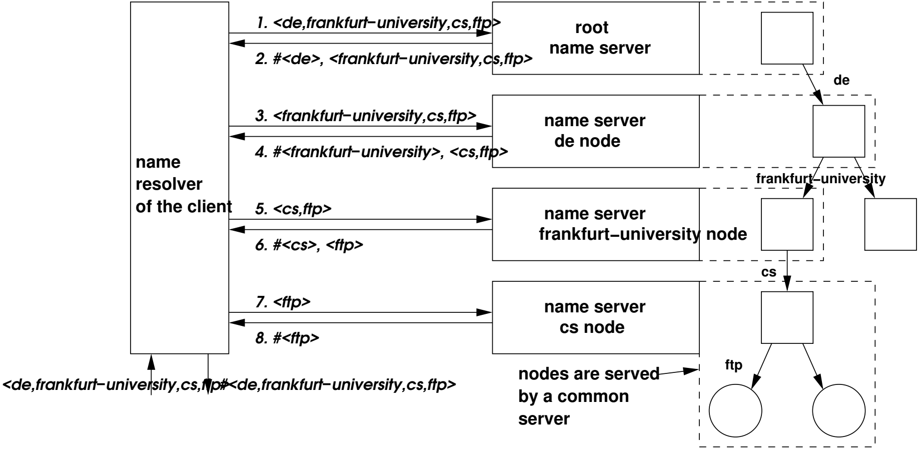
- Search via broadcast
  - Request is sent to everyone and only the node which can resolve the name responds
  - **Drawback:** Does not scale
  - **Example:** ARP → Resolution of IP addresses to MAC addresses
- Via name server
  - Requests are sent to a dedicated server which maintains a mapping
  - **Drawback:** Requires a well-known address
  - **Example:** DNS





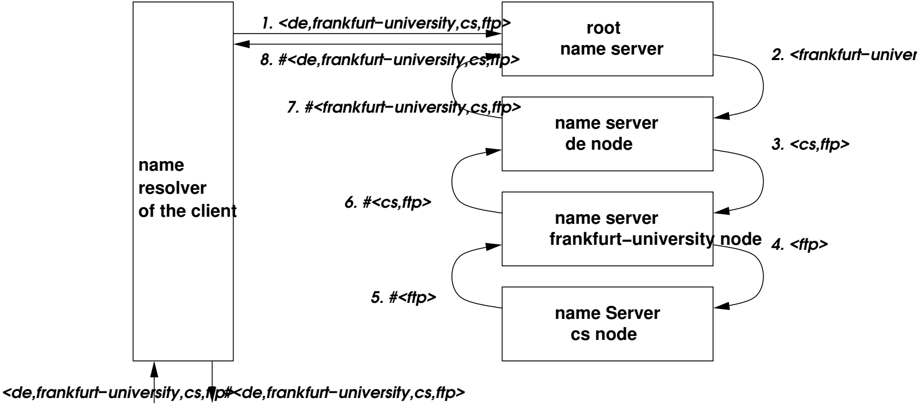
Iterative Name Resolution

- Starting at the client
- Caching only on client


name resolver of the client
1. <de,frankfurt–university,cs,ftp>
2. #<de>, <frankfurt–university,cs,ftp>
root name server
3. <frankfurt–university,cs,ftp>
4. #<frankfurt–university>, <cs,ftp>
name server de node
5. <cs,ftp>
6. #<cs>, <ftp>
name server frankfurt–university node
7. <ftp>
8. #<ftp>
name server cs node
<de,frankfurt–university,cs,ftp>
#<de,frankfurt–university,cs,ftp>
nodes are served by a common server
de
frankfurt–university
cs
ftp

Recursive Name Resolution

- Caching on server is possible
- Less traffic
- More load for the root servers


name resolver of the client
root name server
name server de node
name server frankfurt–university node
name Server cs node
1. <de,frankfurt–university,cs,ftp>
2. <frankfurt–univer
3. <cs,ftp>
4. <ftp>
5. #<ftp>
6. #<cs,ftp>
7. #<frankfurt–university,cs,ftp>
8. #<de,frankfurt–university,cs,ftp>
<de,frankfurt–university,cs,ftp>
#<de,frankfurt–university,cs,ftp>

Common Name Services

- DNS (Internet Domain Name Service)

  → Lecture Computer Networks
- Java RMI Registry
- CORBA INS (Interoperable Naming Service)
  - URLs as names for CORBA objects

Agenda

- Names
- Name Services
- Directory Services
- Location Services

Directory Services

- Difference to name service:
  - Extension by attributes
  - Analogy: *yellow pages* vs. phone book
  - Entries in a directory service are mainly searched by their properties – not by their name

Directory Services

- Difference to name service:
  - Extension by attributes
  - Analogy: *yellow pages* vs. phone book
  - Entries in a directory service are mainly searched by their properties – not by their name
- Standards
  - *X.500* (ITU-T)
    - Complex, used ISO/OSI stack and Directory Access Protocol (DAP)
  - *LDAP* (Lightweight Directory Access Protocol)
    - Implements only a part of the X.500 standard
    - Builds up on top of TCP/IP
    - LDAP = Lightweight version of DAP
    - Typically LDAP does not only refer to the access protocol but to the directory server (LDAP server) itself

LDAP (Protocol)

- Current version 3 is specified in RFC 4511
- Support by many operating systems
- LDAP server support replication and delegation (referral)
- LDAPv3 supports TLS, SASL, and Kerberos authentication
- Most LDAP data are strings (simple encoding for network transmission), but binary data can be processed as well

LDAP (Directory)

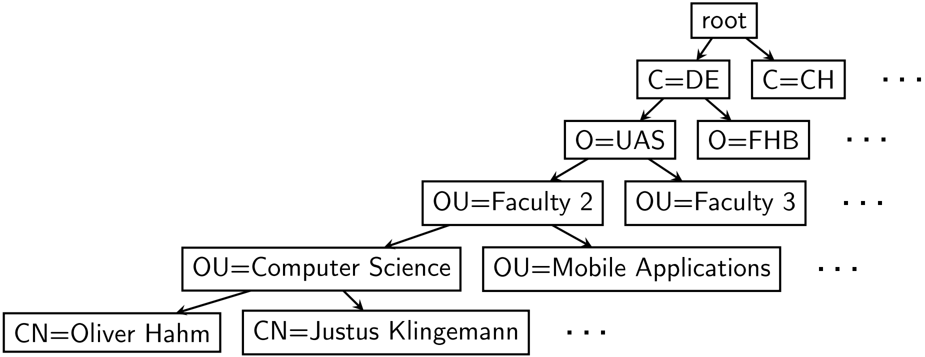
- Hierarchical name space: Directory Information Tree (DIT)
- Entries (the nodes of the tree) can be any LDAP objects
- LDAP objects consist of a set of `<attribute, value>` pairs
- Classes define object types with particular attribute and value sets
- Each object belongs to at least one class
- Schemata for predefined classes (e.g., person, organization) exist
- Inheritance is possible
- Application specific extensions are possible

Example

Attribute	Abbreviation	Value
Country	C	DE
Locality	L	Frankfurt
Organization	O	UAS
OrganizationalUnit	OU	Faculty 2
OrganizationalUnit	OU	Computer Science
CommonName	CN	Oliver Hahm

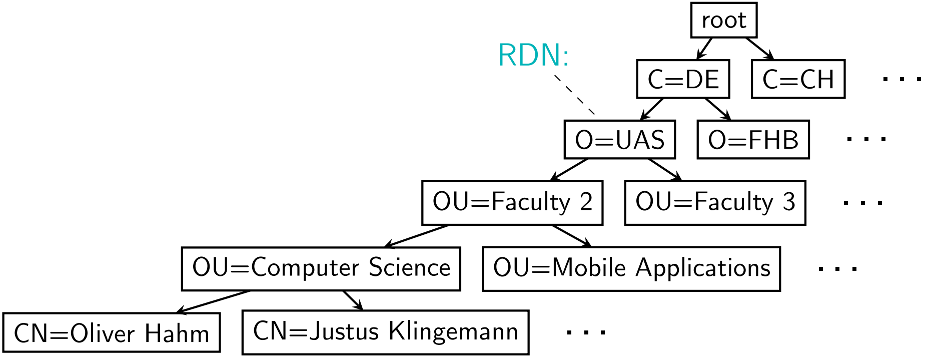
RDN and DN

- Starting point: DIT root → *Base Object*



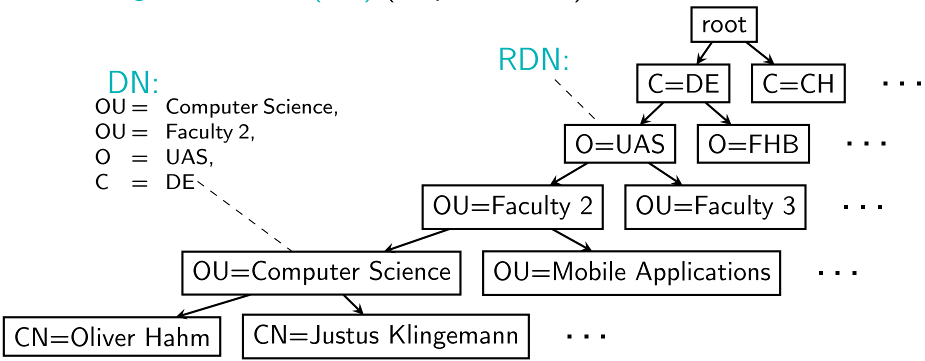
RDN and DN

- Starting point: DIT root → *Base Object*
- Each node has a unique name in its layer, called Relative Distinguished Name (RDN)



RDN and DN

- Starting point: DIT root → *Base Object*
- Each node has a unique name in its layer, called Relative Distinguished Name (RDN)
- Concatenation of RDNs from the node towards the root is called Distinguished Name (DN) (→ path names)



Operations

- Bind — Authentication
- Add — Adding an entry
- Delete — Remove an entry
- Search — Search an entry
- Compare — Compare LDAP objects
- Modify — Modify a LDAP object
- ModifyRDN — Move or rename an object
- Abandon — Cancel a running operation
- Unbind — Logout of a client

Requests

- Usage as name service
  - Find an object by its given Distinguished Name
  - e.g., `read(/C=DE/O=UAS/OU=Computer Science/CN=Oliver Hahm)`, for access to all attributes of the object
- Search objects with certain attributes
  - Request can return a list of results
- Requests may be complex:
  - Wildcards, regular expressions, e.g., &(C=DE)(CN=*Hahm)

Replication

- Parts of the name space are typically replicated on multiple servers
  - In order to improve fault tolerance and performance
  - Especially central parts
  - Replication may take hours
  - Primary-Replica [1] configuration
    - Modifications happen only at the primary
    - Propagation to replicas

[1] In older literature called *Master-Slave* configuration

Replication

- Parts of the name space are typically replicated on multiple servers
  - In order to improve fault tolerance and performance
  - Especially central parts
  - Replication may take hours
  - Primary-Replica [1] configuration
    - Modifications happen only at the primary
    - Propagation to replicas
- **Problem:** Duration for updates upon modifications
  - Updates are not immediately visible globally
  - Can only be tolerated, if . . .
    - Big read/write relation
    - Reading of outdated entries is uncritical

[1] In older literature called *Master-Slave* configuration

Applications

- User/identity management
  - Scheme: inetOrgPerson (RFC 2798)
- Address books of mail systems
  - e.g., Thunderbird interface to LDAP
- Company organization
  - Information according to organigrams
- Inventory system or infrastructure management

LDAP/X.500 Products

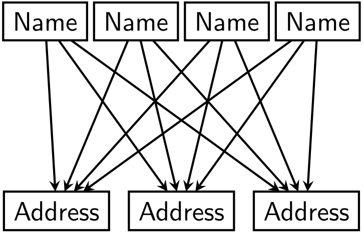
- OpenLDAP (Open Source)
- NetIQ eDirectory (formerly Novell eDirectory, before that Novell Directory Services NDS)
- MS Active Directory (with LDAP Interface)
- Atos DirX (formerly Siemens DirX)
- Oracle Directory Server (formerly Sun Directory Server)

Agenda

- Names
- Name Services
- Directory Services
- Location Services

Location Services

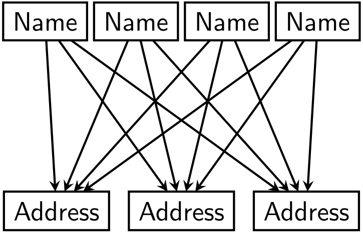
- **Problems** arise when objects may **change their** (phyiscal) **address quickly**
  - Each time the name server entries must be changed ⇒ problem with replication and caching

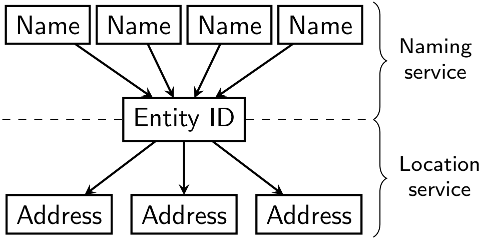


Location Services

- **Problems** arise when objects may **change their** (phyiscal) **address quickly**
  - Each time the name server entries must be changed ⇒ problem with replication and caching
- **Solution:**
  - Split into **naming** and **location service**
  - Mapping: Name → unique entity ID → location

  ⇒ Only one update required

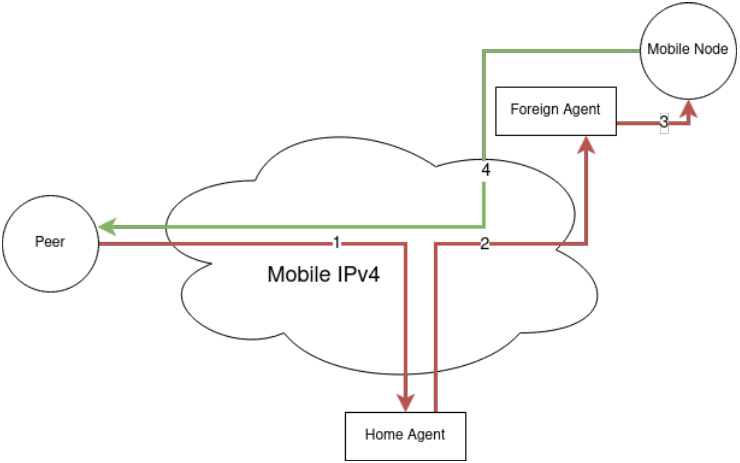




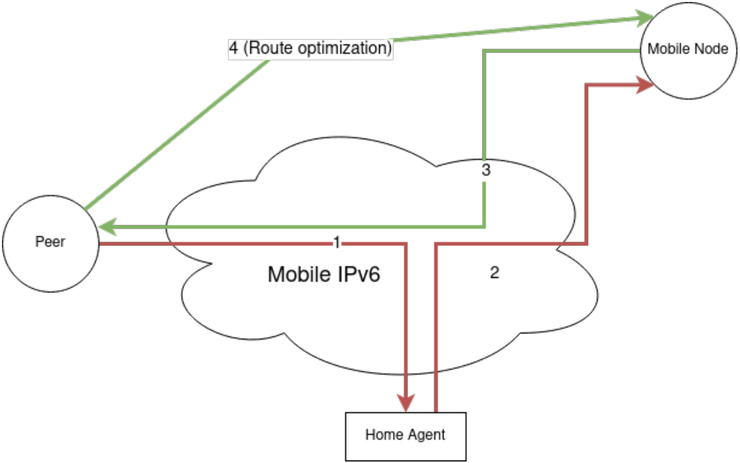
Identifier-Locator Split

- Each participant requires two designators (addresses)
  - An identifier (ID) specifies who it is
  - A locator (Loc) specifies where it is
- In static networks both designators are static:
  - ⇒ Can be reduced to one address
  - In the Internet the IP address represents ID and Loc
- In mobile networks the locator changes
  - ID and Loc diverge → ID-Loc split
  - Network and end systems have to handle this duality
- Specified in the Locator/ID Separation Protocol (LISP) (RFC 9300)

Mobile IPv4



Mobile IPv6


4 (Route optimization)
Mobile Node
3
Peer
1
Mobile IPv6
2
Home Agent

Important takeaway messages of this chapter

- Names can be unique and pure/impure
- Their name is only meaningful for a particular context
- Name services can be used for name resolution (name → address)
- Directory services extend this approach and allow for searching by further attributes