

Verteilte Systeme

Namensdienste und
Internet Standardanwendungen

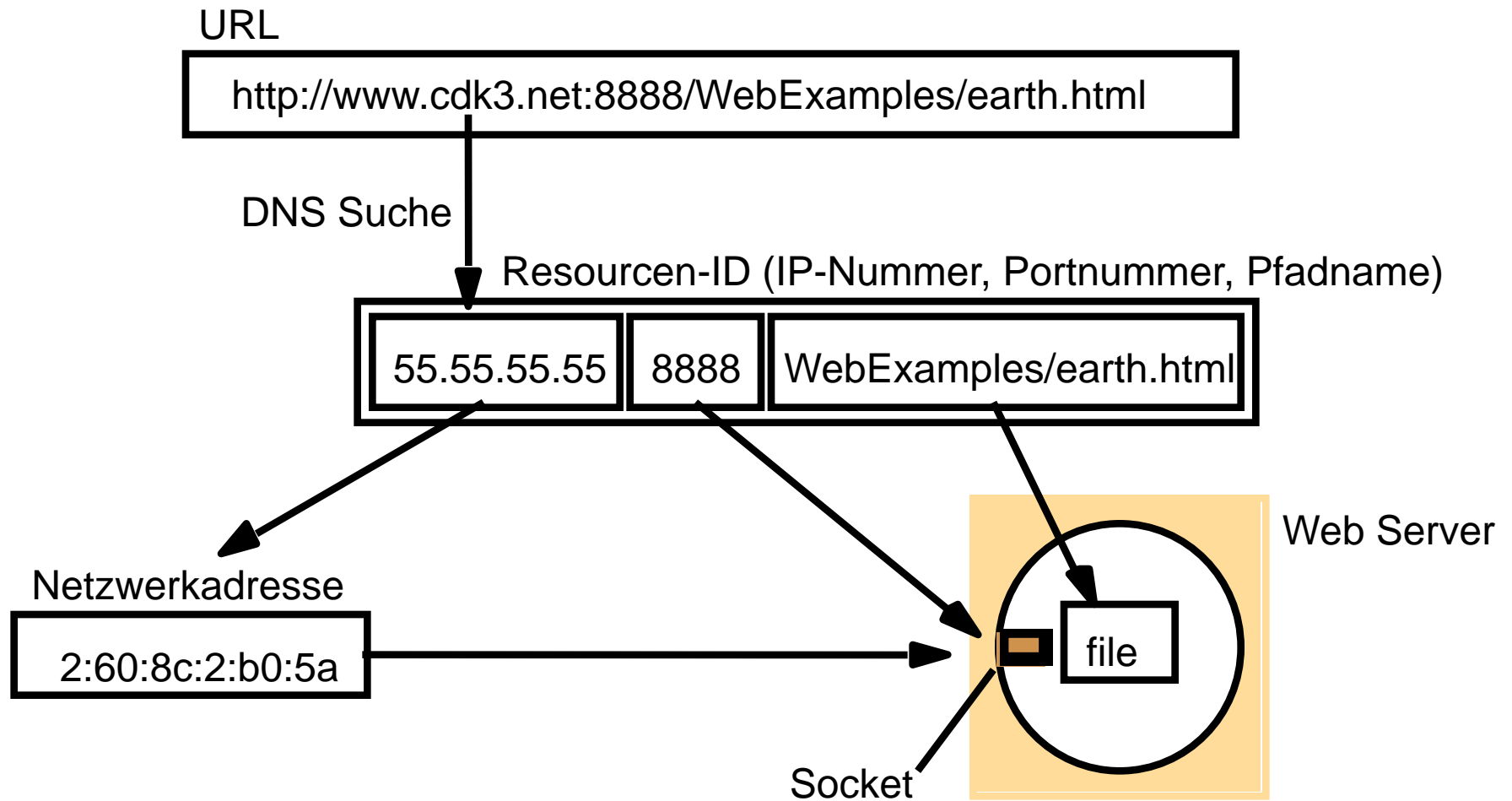
Namen in verteilten Systemen

- ◆ Namen dienen der **(eindeutigen) Bezeichnung** von Objekten (Bezeichner)
 - Variablen, Prozeduren, Datentypen, Konstanten, ...
 - Rechner, Dienste, Dateien, Geräte
- ◆ Zweck von Namen:
 - **Erklären:** laser88, klauck@informatik.haw-hamburg.de, /etc/passwd/
 - **Identifizieren:** laser88 statt "der mit „laser88“ bezeichnete Drucker" oder 10030 als RPC-Programmnummer statt "die mit „HelloWorld“ bezeichnete Prozedur"
 - **Lokalisieren:** Zugriff auf Objekt über den Namen: 131.246.19.42:1633 als Adresse der entfernten Funktion „HelloWorld“
- ◆ Für die **Praxis wichtig:**
 - Kenntnisse über Namensauflösung
 - Kenntnisse über Lokalisierung, speziell mobiler Objekte

Namen und Adressen

- ◆ Jedes Objekt hat eine Adresse:
 - Speicheradresse
 - Ethernet-Adresse
 - Internet-Adresse
 - Internet-Adresse, Protokoll-Port
- ◆ Adressen sind "physische" Namen (Namen unterster Stufe)
 - **Direkte Lokalisierung** eines Objektes
 - in einem **Kontext** (Adreßraum, Namensraum) **eindeutig**
- ◆ Entkopplung von Namen und Adressen unterstützt **Ortstransparenz**
- ◆ **Zuordnung** "Name ---> Adresse" notwendig (Bindung)

Beispiel: URL-Abbildung



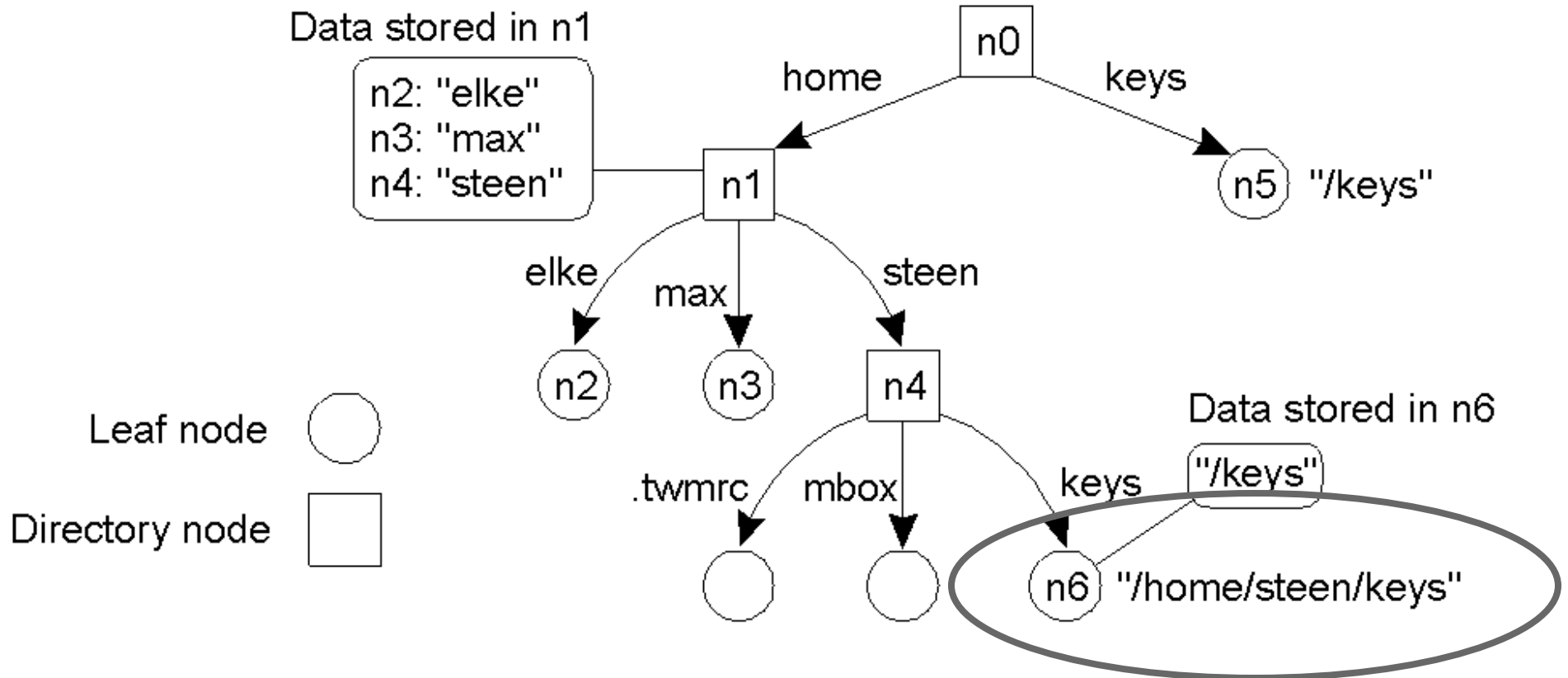
Identifikation von Ressourcen

- ◆ **URI:** Universal Resource Identifier
 - Ein String, der eine Ressource im Netz identifiziert, ohne auf die Zugriffsart einzugehen
 - Wird zur Zeit heftig diskutiert
- ◆ **URN:** Uniform Resource Name
 - dauerhafter Name (Problem der hängenden links bei URL)
 - URN-Suchdienst verbindet Namen mit URL
 - haben **URC:** Uniform Resource Characteristics; Beschreibung/Attribute der Eigenschaften einer Web-Resource (ähnlich X.500 bzw. Gelbe Seiten)
- ◆ **URL:** Uniform Resource Locator
 - URLs sind spezielle URIs
 - Eine URL identifiziert eindeutig ein Dokument im WWW, auf das z.B. mittels HTTP zugegriffen wird
 - URLs haben eine feste Syntax, die das Zugriffsprotokoll und den Ort im Netz identifizieren (DNS-Computername + Pfadname auf dieser Maschine)
 - Definiert zuerst in RFC 1738, erweitert in RFC1808, RFC2368, RFC2396
 - Kompromiss zwischen Adresse und Name

Namen und Binden

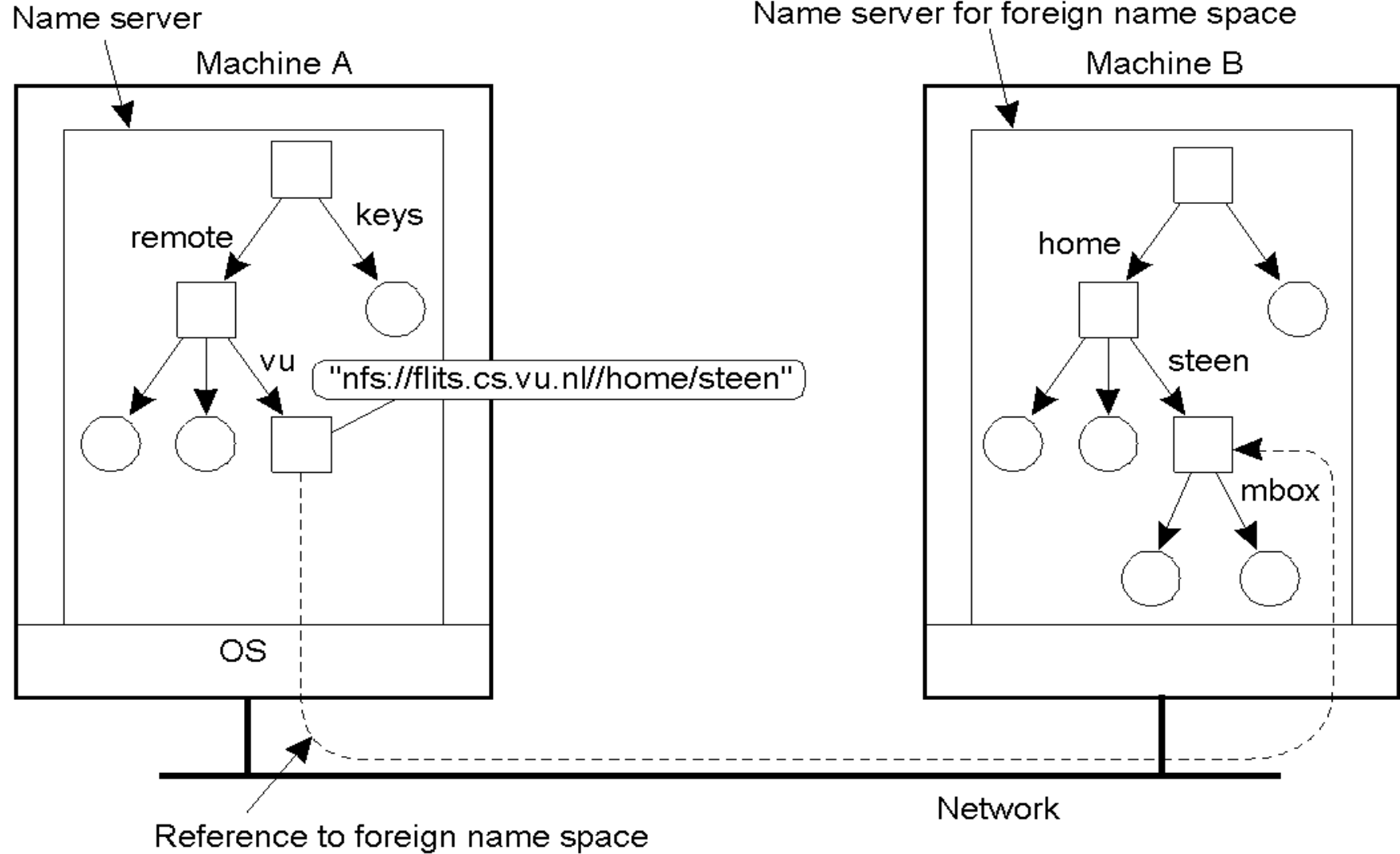
- ◆ Zuordnung Name ---> Adresse
- ◆ Bindezeitpunkt:
 - beim Übersetzen (**statisches Binden**)
z.B. bei Programmiersprachen
 - beim Starten ("**halb**"-**dynamisches Binden**)
z.B. moderne Binder (SunOS), nach dem Start in der Regel nicht änderbar
 - beim Zugriff (**dynamisches Binden**)
in verteilten Systemen angebracht:
 - ◆ Neue Dienste
 - ◆ Verlagerung existierender Dienste

Mounting



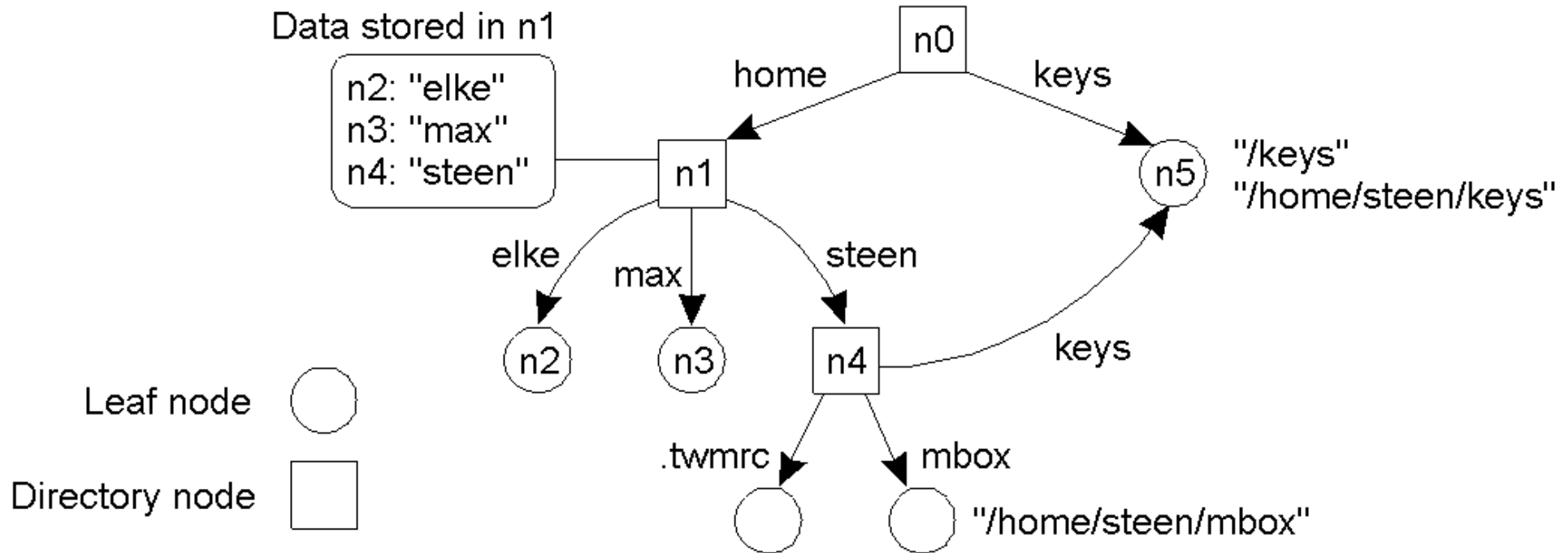
Der Begriff einer symbolischen Verbindung erklärt am Beispiel eines Namensgraphen

Mounting



Mounting eines entfernten Namensraum durch ein spezielles Protokoll

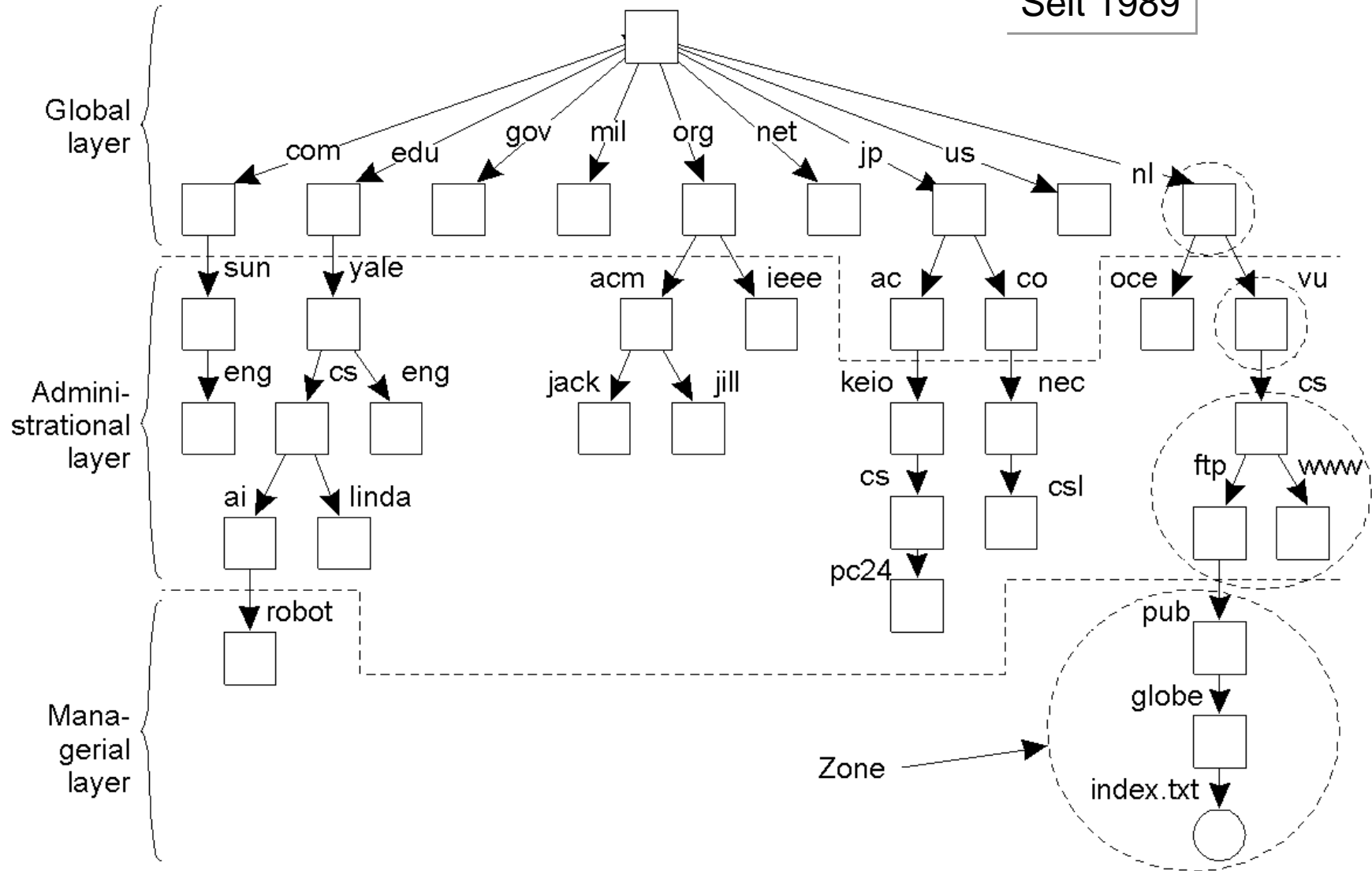
Namen und Namensräume



- ◆ Namen nur relativ zu einem Namensraum **eindeutig**
- ◆ Existiert nur ein Namensraum --> **flacher Namensraum**
- ◆ Namensräume haben selbst wieder Namen --> **hierarchischer Namensraum**

Logische Schichten

Seit 1989



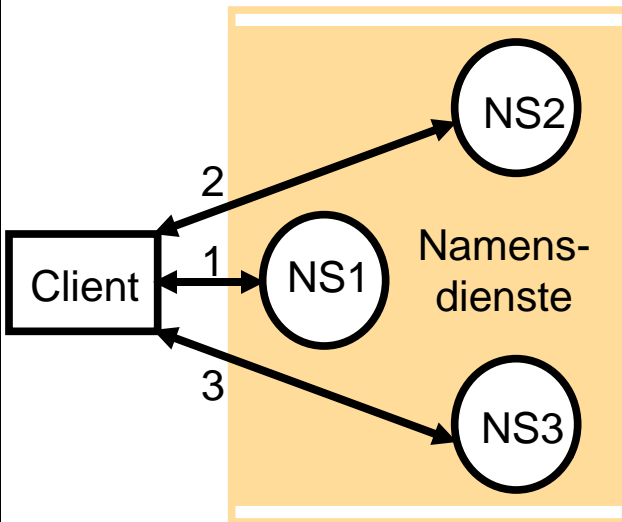
Anforderung an Namensräume

- ◆ Einfache, aber **bedeutungsvolle Namen** erlauben
- ◆ Potentiell eine **unendliche Anzahl** von Namen zulassen
- ◆ **Strukturiert**
 - Um ähnliche Sub-Namen ohne Kollisionen zu erlauben
 - Um verwandte Namen zu gruppieren
- ◆ **Re-Strukturierung** von Namensbäumen erlauben: Für einige Arten der Änderung sollten alte Programme weiterhin funktionieren
- ◆ **Lange Lebensdauer** des Dienstes (Investitionssicherung)
- ◆ **Hohe Verfügbarkeit**
- ◆ Management des **Vertrauens**
- ◆ **Tolerierung von Misstrauen** (in einem großen System sind nicht alle Komponenten gleich vertrauenswürdig)

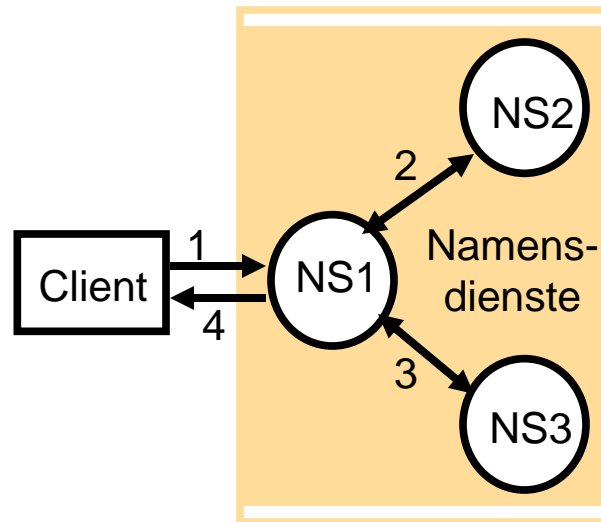
Namensauflösung

- ◆ Namensauflösung: Prozess des Findens der Bindung eines Namens
- ◆ Strategien für die Navigation:

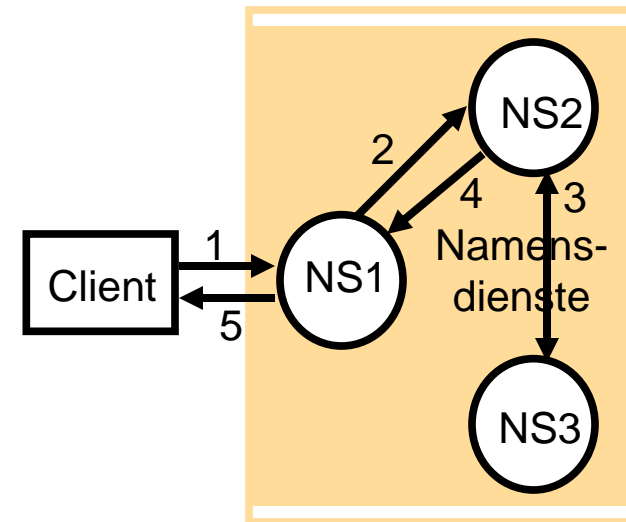
Iterativ



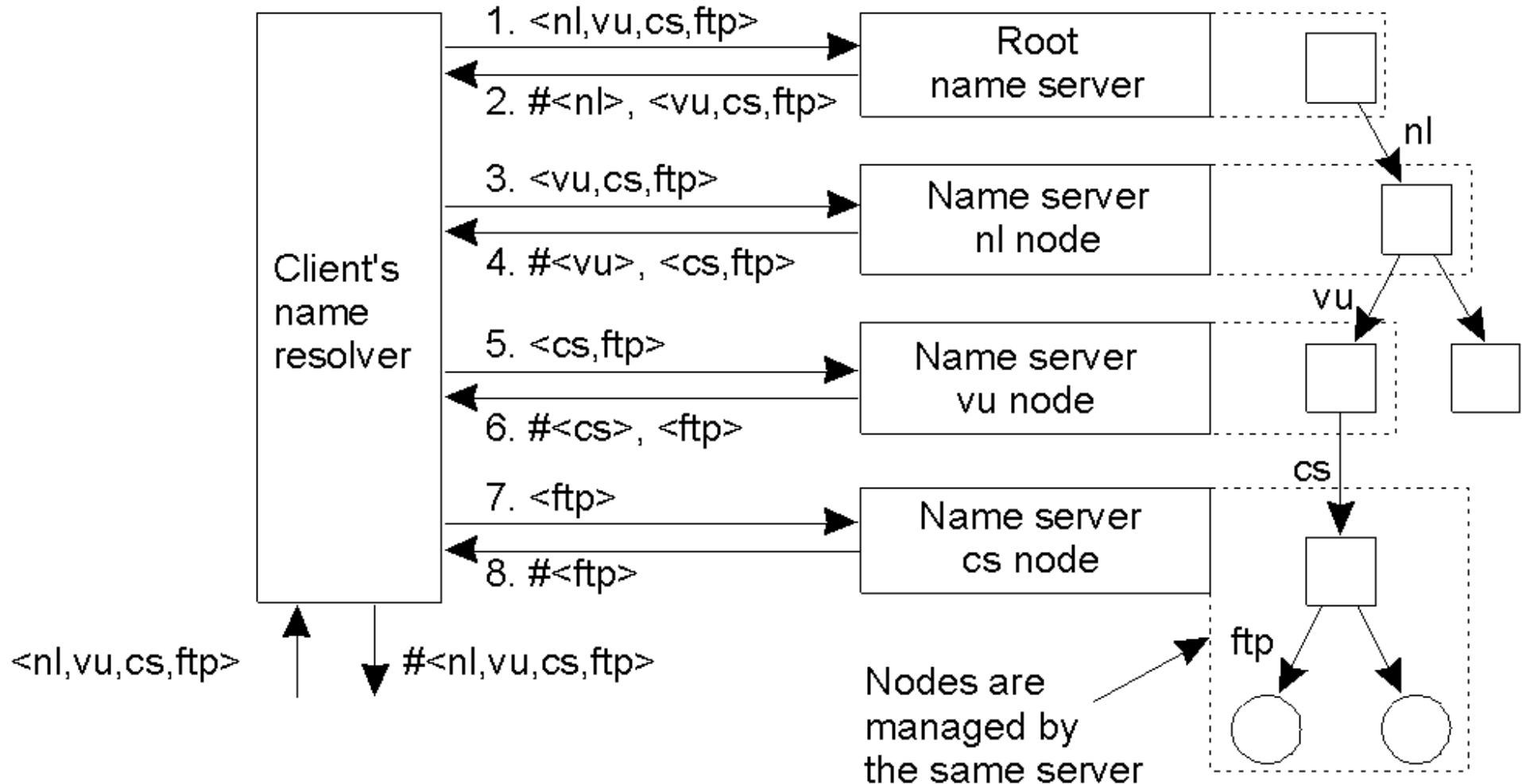
Iterativ
Server gesteuert



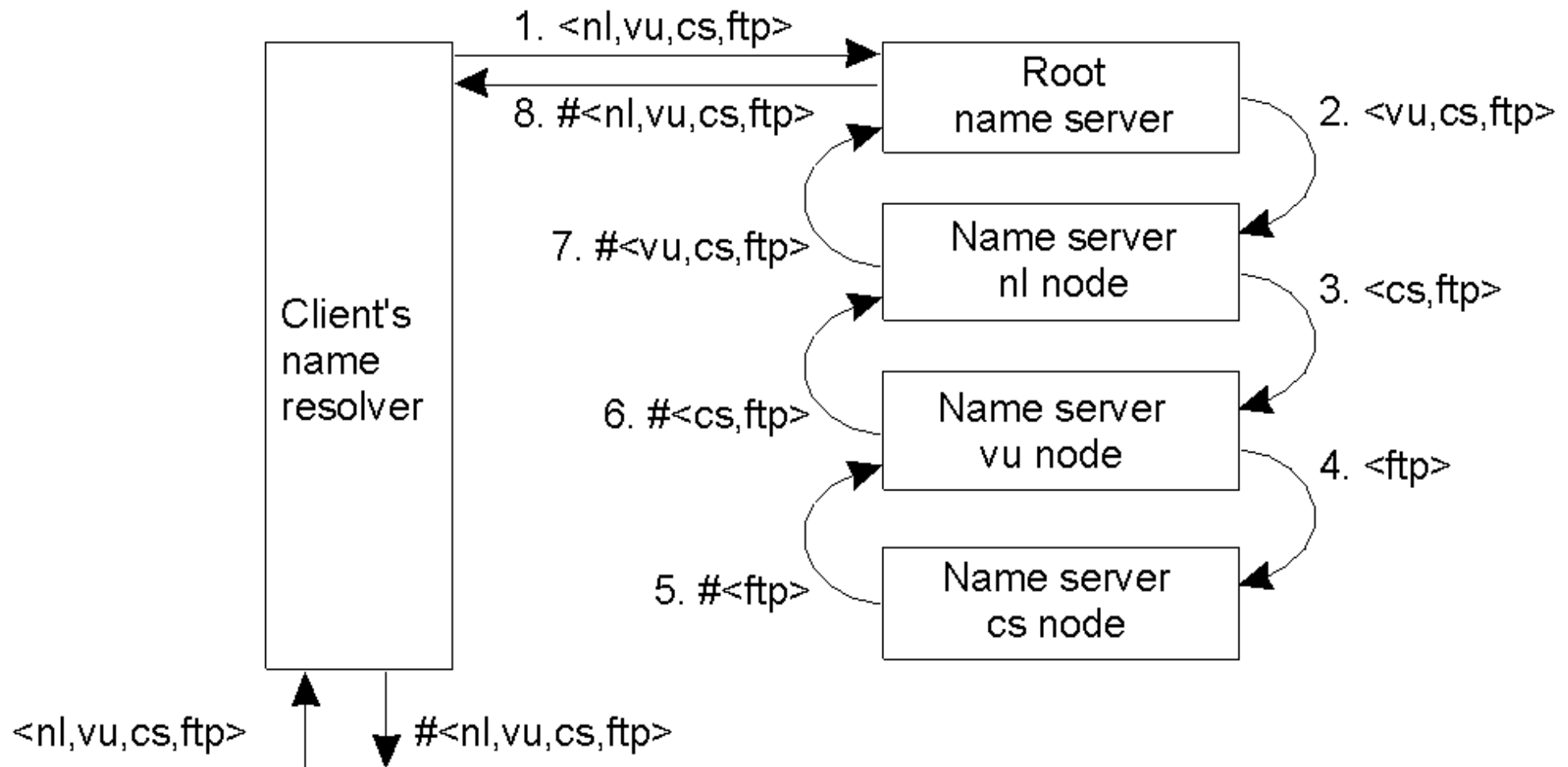
Rekursiv
Server gesteuert



Beispiel: Iterative Namensauflösung



Beispiel: Rekursive Namensauflösung



Beispiel: Namensdienste

- ◆ Das Telefonbuch
- ◆ Die „Gelben Seiten“
- ◆ Internet: Domain Name System (DNS)
- ◆ Jini lookup service
- ◆ X.500 Directory Service (CCITT/ISO) Standard verwendet in OSF/DCE, als „Verbesserung“:
LDAP (Lightweight Directory Access Protocol)
- ◆ CORBA NameService
- ◆ JNDI (Java Naming and Directory Interface)
- ◆ NIS = Network Information Service
- ◆ UDDI (Universal Description, Discovery, and Integration)
- ◆ ...

Verzeichnisdienste

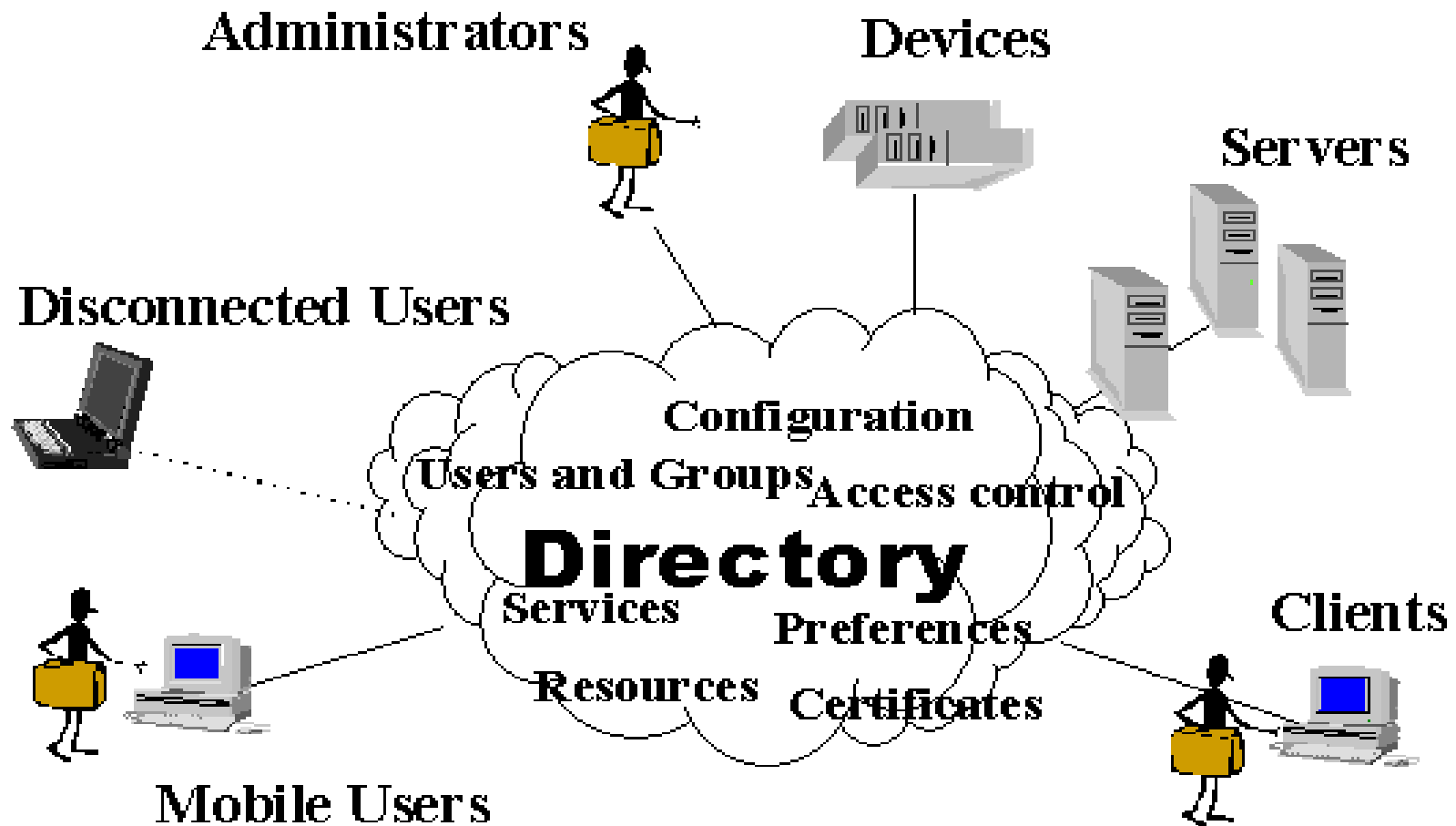
Rechnernetze benötigen in der Praxis weitere, anwendungsspezifische Informationen:

- Login-Daten
- Informationen über netzweite Dienste
- Informationen über Software und Geräte ...

Hierfür sind Verzeichnisdienste gut geeignet:

- Flexibel hierarchisch strukturierbar
- Eigene Zugriffs- und Verteilungsfunktionen
- Beispiele: NIS, NDS, ADS, LDAP

Vision



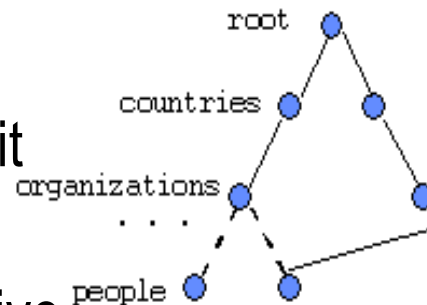
LDAP

Ein LDAP-Baum gliedert sich in Knoten

Jeder Knoten bildet ein Objekt mit zugehörigen Attributen

Jeder Knoten besitzt einen Relative Distinguished Name als Informationsschlüssel

Bsp:

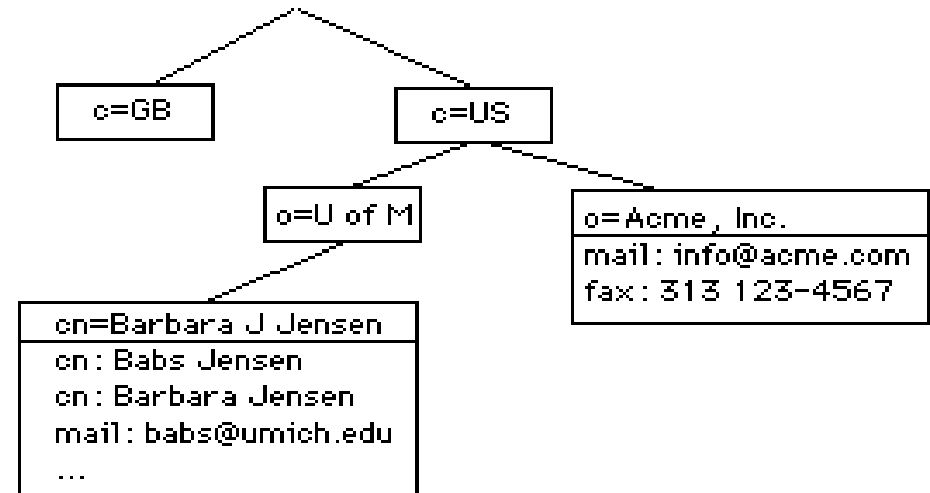


```
cn: Babs Jensen
sn: Jensen
mail: babs@ace.com
jpegPhoto: AdQ13...
```

cn= Babs Jensen

LDAP Struktur

- ◆ Beinhaltet Informationen über Objekte
- ◆ Objekte haben ein oder mehrere Attribute
- ◆ Attribute können mehrere Werte annehmen
- ◆ Ein Attribut bildet den RDN (cn=Barbara J Jensen)
- Zusammen mit den darrüberliegenden Knoten des Objekts wird der Distinguished Name gebildet:



dn: cn= Barbara J Jensen, o=U of M, c=US

LDAP-Operationen

LDAP Zugriffe bilden drei Gruppen:

Lesen - Schreiben - Verbinden

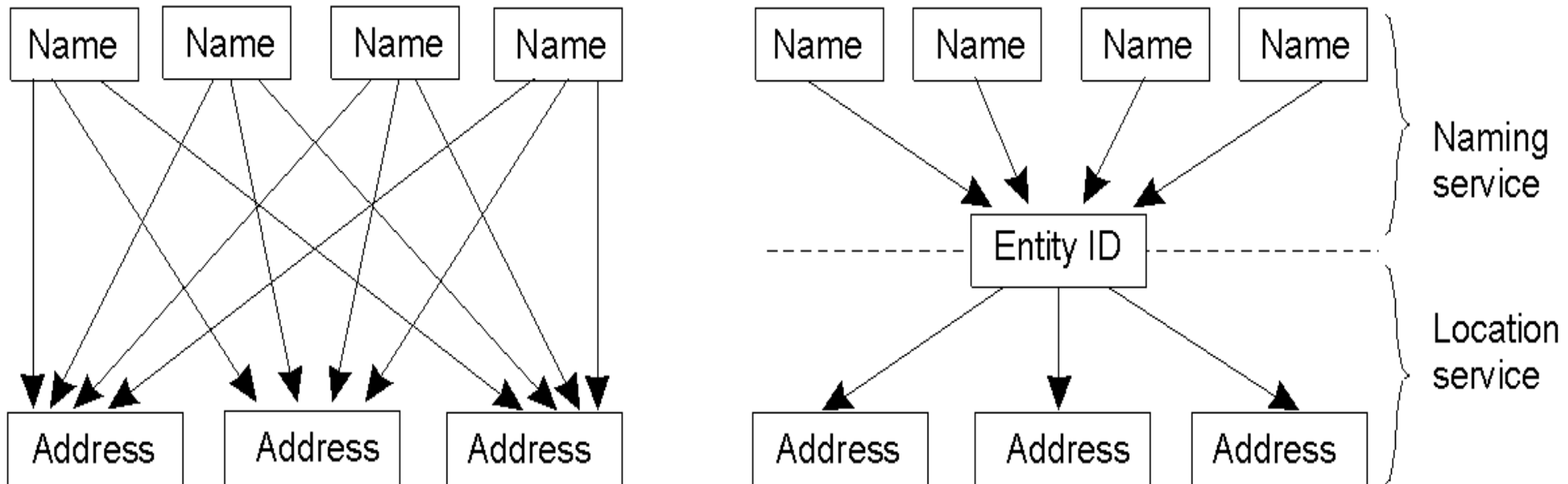
- Bind
- Search
- Compare
- Add
- Delete
- Modify
- ModifyRDN
- Abandon

Bsp `ldapdelete` : `-h ldap.uom.edu -D "cn=LDAP Admin, o=U of M, c=US" -w "secret" "cn= Barbara J Jensen, o=U of M, c=US"`

LDAP-Technologien

- LDAP ist ein offener Standard (RFC 3377/4510)
- LDAP hat breite Betriebssystemunterstützung
- LDAP - Server beherrschen Replikation und Delegation (Referral)
- LDAPv3 unterstützt SSL, SASL und Kerberos-Authentisierung
- Die meisten LDAP-Daten sind Textstrings (einfache Kodierung bei Netzübertragung), Binärdaten können verarbeitet werden
- JAVA Zugriff: JNDI

Lokalisierung Mobiler Objekte



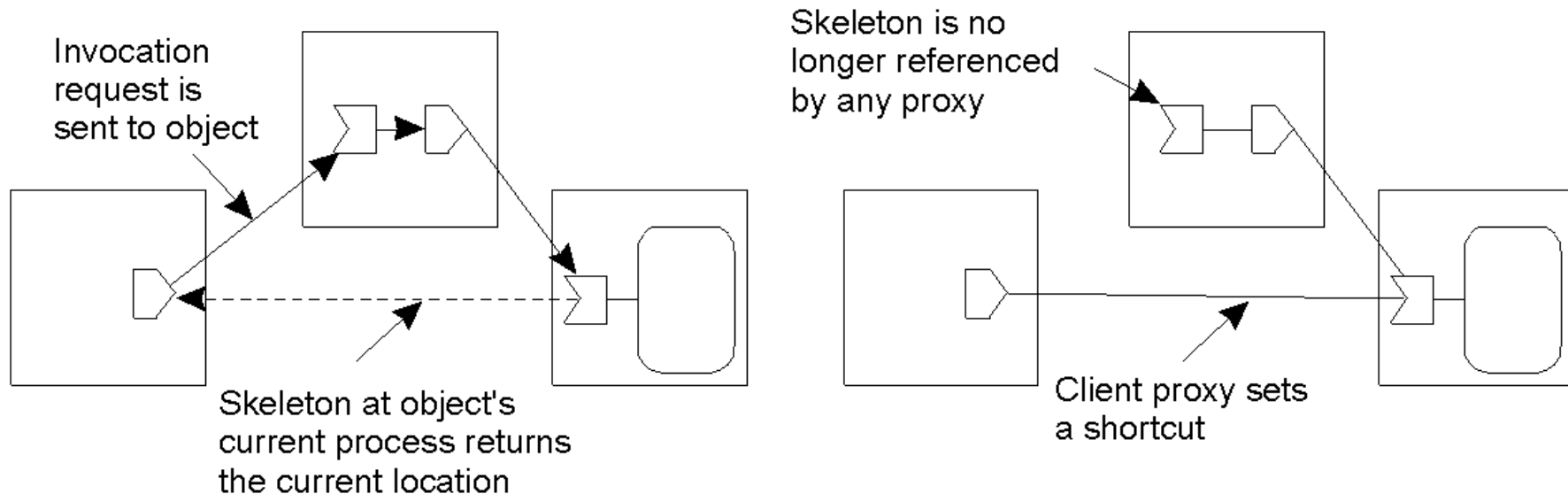
- ◆ **Einebenen-Zuordnung:** Direkte Abbildung von Namen auf Adressen. Änderung bei Namens- oder Adressänderungen.
- ◆ **Zweiebene-Zuordnung:** Einführung eines eindeutigen Identifiers, der statisch immer dem selben Objekt zugeordnet ist.

Location Service: Broad-/Multicasting

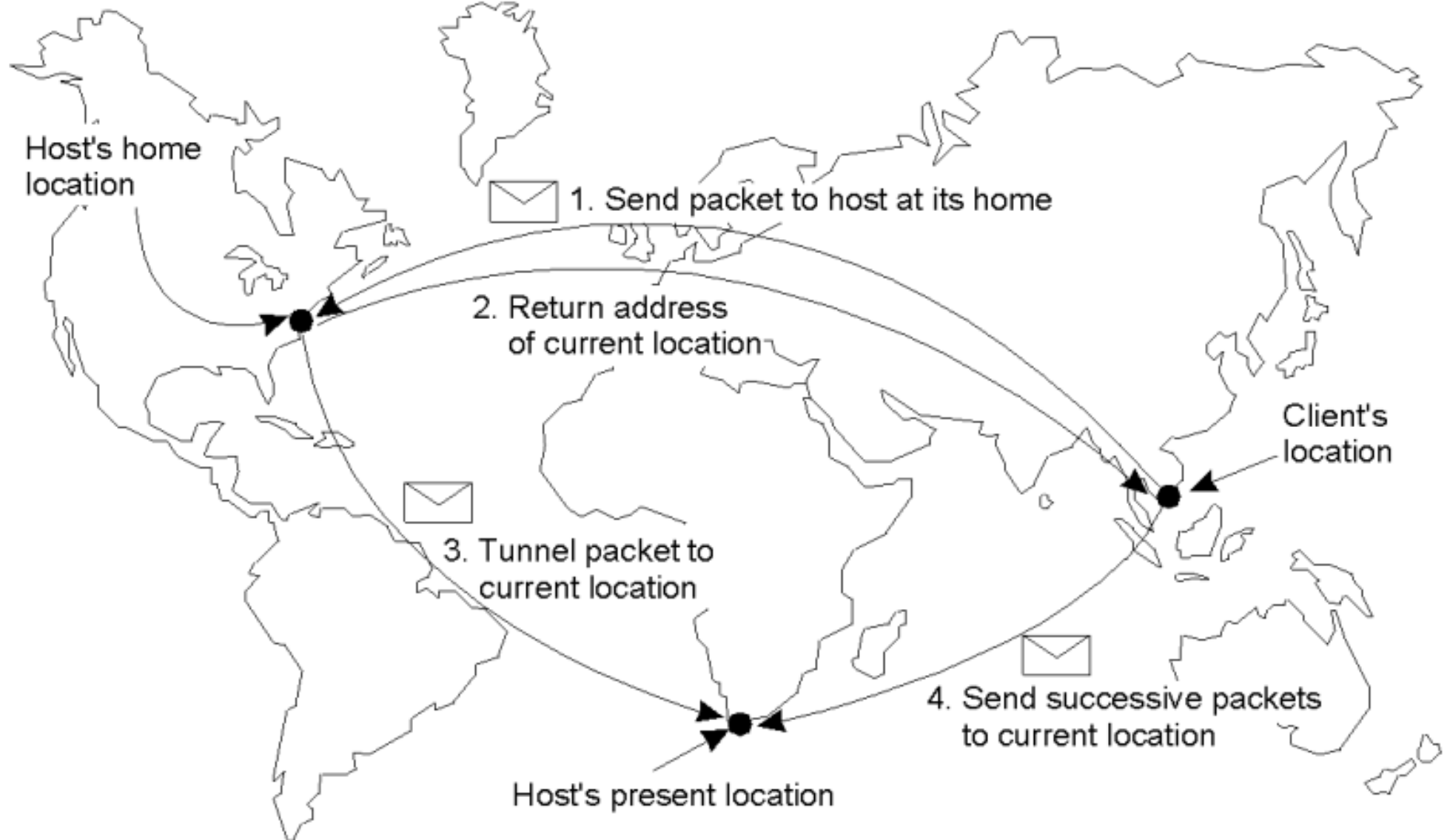
- ◆ Nur im LAN mit „überschaubarer Größe“ anzuwenden.
- ◆ Ablauf:
 - 1 Nachricht mit Identifiziere eines Objekts wird per Broadcast im LAN versendet.
 - 2 Rechner, der Zugangspunkt für das Objekt bereitstellen kann, sendet Antwort mit Adresse des Zugangspunktes
- ◆ Ineffizient bei „größeren“ LAN's. Einschränkung durch Multicastgruppen möglich.

Location Service: Forwarding Pointers

- ◆ Nur im LAN mit „überschaubarer Größe“ anzuwenden.
- ◆ Ablauf: Bei Migration von Rechner A zu Rechner B wird auf Rechner A eine Referenz auf den neuen Ort B hinterlassen..
- ◆ Ineffizient bei „häufiger“ Migration: Es entstehen zu lange Ketten. Einschränkung durch „Kettenreduktion“ möglich.



Location Service: Home Location



Home Location: Mobile IP

- ◆ Jeder mobile Host nutzt eine **feste IP-Adresse**
- ◆ Ganze Kommunikation geht an „**Home Agent**“ (an der Home Location, d.h. gleiches Netzwerk)
- ◆ Bei **Bewegung** des mobile Host
 - Anforderung einer (temporären) **care-of-Adresse** durch mobilen Host
 - Registrierung der care-of-Adresse beim Home Agent
 - care-of-Adresse nur für Kommunikation
- ◆ Feste IP-Adresse ist **eindeutiger, statisch zugeordneter Identifizier**

Home Location: Nachteile

- ◆ **gesteigerte Kommunikationslatenz** (permanente Nachfrage beim Home Agent)
 - Lösung: Zweischicht-Schema (aus dem Mobilfunk)
 - ◆ zuerst eine lokale Registrierung
 - ◆ dann (bei Mißerfolg) Home-Position kontaktieren

- ◆ die Verwendung einer **festen** Home-Position (= IP-Adresse) (Erreichbarkeit ? Entfernung zur tatsächlichen Position ?)
 - Lösung: Verwendung eines Namensdienstes
 - ◆ Caching der Position möglich
 - ◆ Bei Mißerfolg lookup beim Namensdienst

IP Mobility: Challenges & Terms

Objective:

Application persistence while roaming between IP subnets / providers

Preserve upper layer (L 4+) communication when changing IP subnets

Key Aspects:

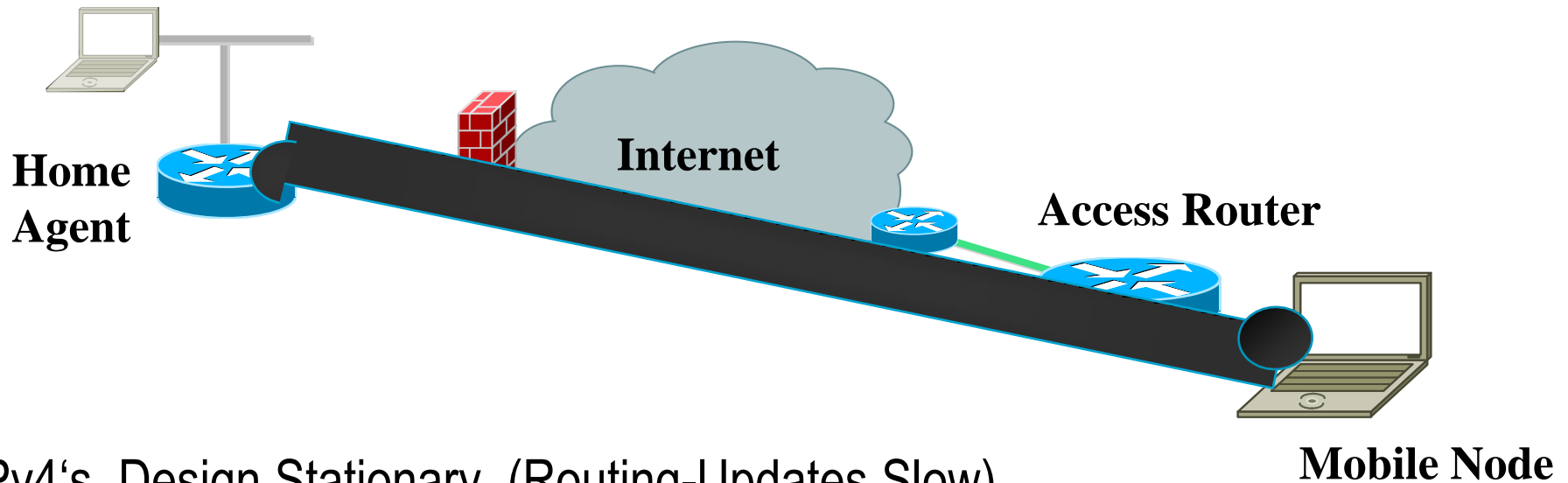
- **Mobile Node** (MN) globally addressable: fixed **Home Address** (HoA)
- **Home Agent** (HA) to permanently represent MN at home network
- Mobile Node locally addressable: changing **Care of Address** (CoA)
- Sustain partner sessions: update **Correspondent Nodes** (CN)
- Enable efficient communication (route optimisation)

Approaches:

Mobile IPv4: **IP Mobility Support for IPv4 (RFC 3344)**

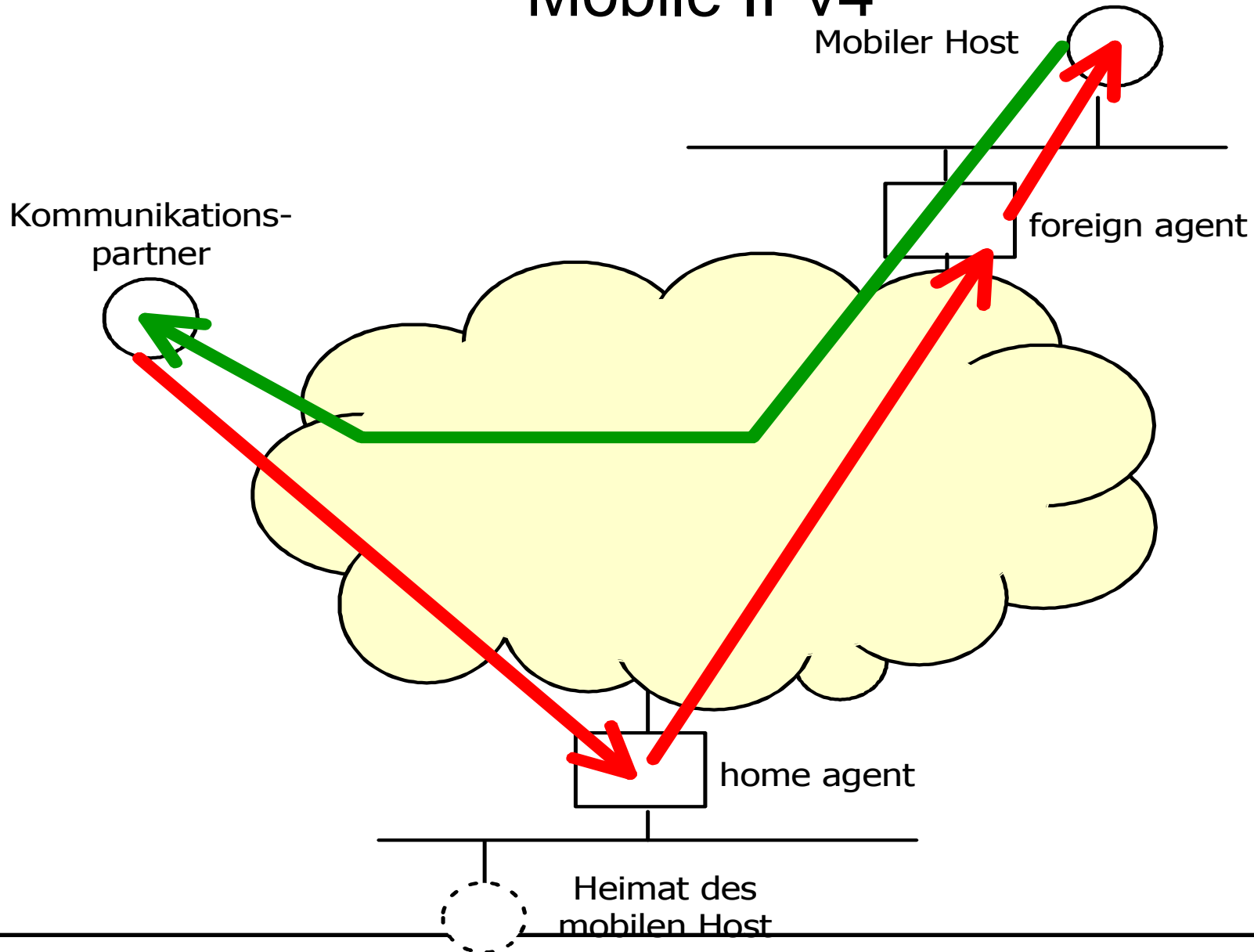
Mobile IPv6: **Mobility Support in IPv6 (RFC 3775/3776)**

Mobile IP



- ◆ IPv4's Design Stationary (Routing-Updates Slow)
- ◆ Implementation of Mobile Services: Tunneling via Home Agent
- ◆ IPv6 Potential:
 - Several Addresses (2 for Mobile Node, many for Mobile Networks)
 - Flexible Architecture - no dedicated Access-Services (Agents, DHCP)

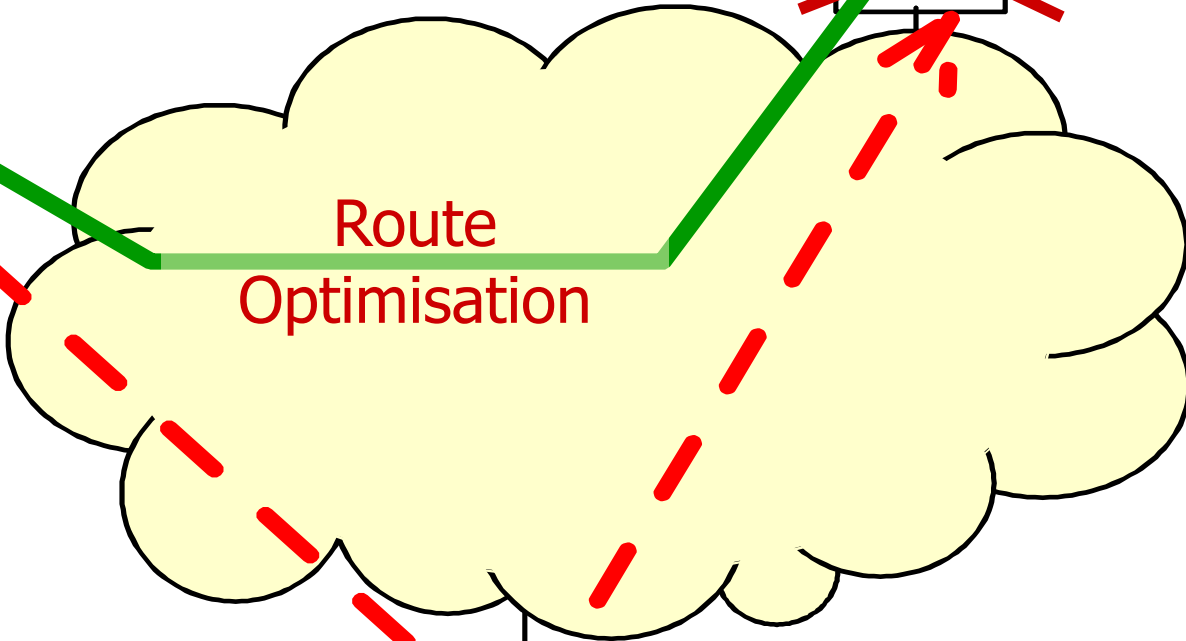
Mobile IPv4



Mobile IPv6

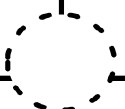
Kommunikations-
partner

Mobiler Host



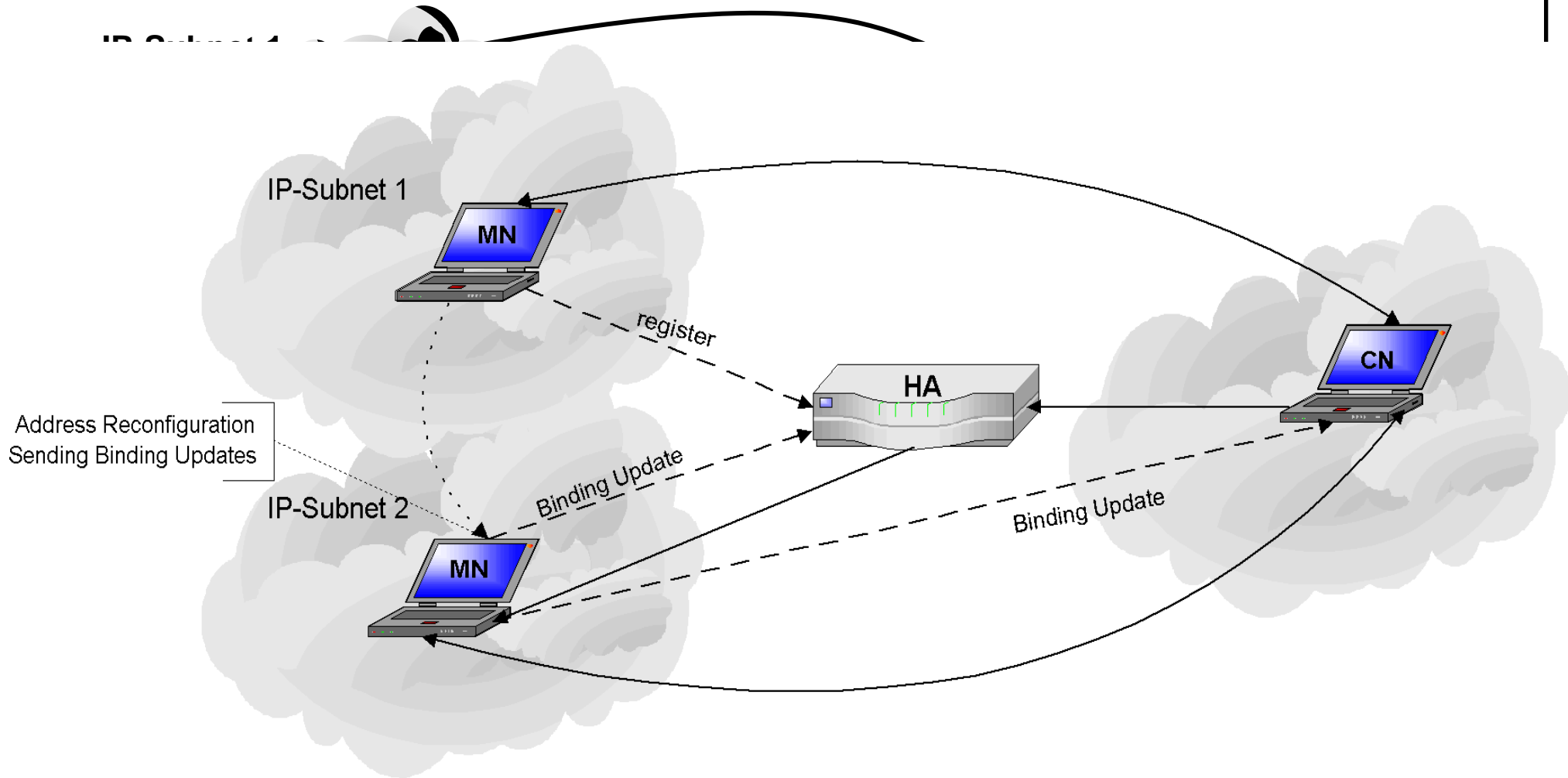
Route
Optimisation

home agent



Heimat des
mobilen Host

Mobile IPv6



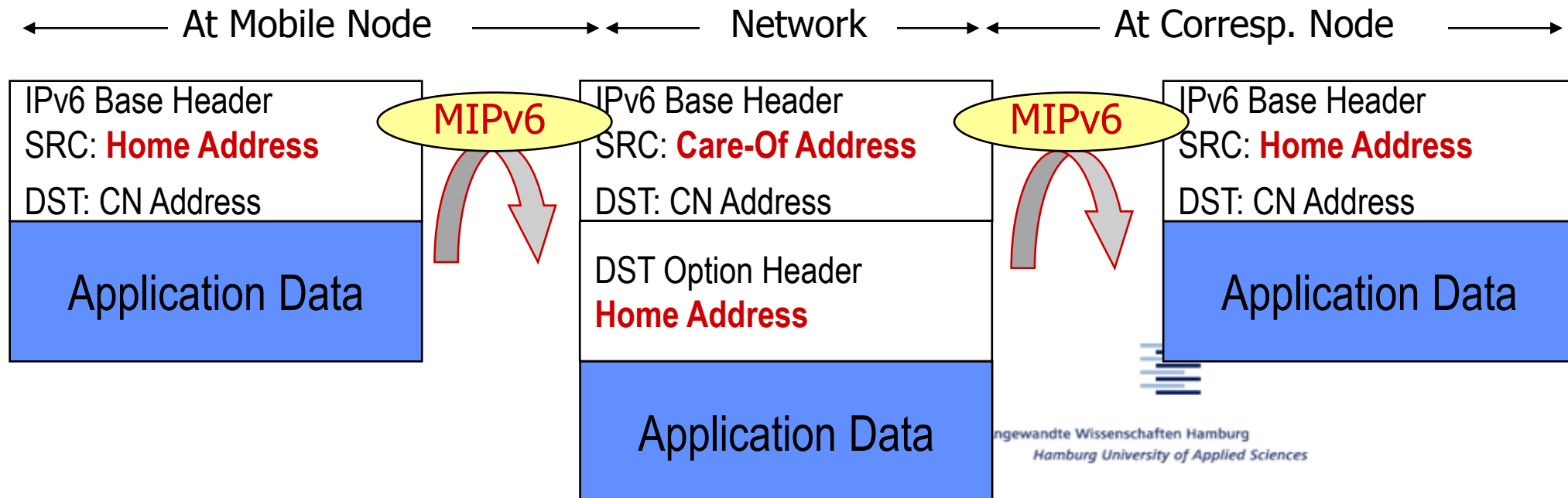
Basic Mobile IPv6

MIPv6 transparently operates address changes on IP layer by:

- ◆ MN's stateless configuration of Care of Address in a foreign network and Binding Updates (BUs) with Home Agent (HA) and Correspondent (CNs).
- ◆ MN continues to use its original Home Address in a Destination Option Header, thereby hiding different routes to the socket layer.
- ◆ CNs continues to use Home Address of the MN, placing current CoA in a Routing Header as Source Route.
- ◆ MN, CN & HA keep Binding Cache Tables.
- ◆ Home-Agent needed as Address Dispatcher.

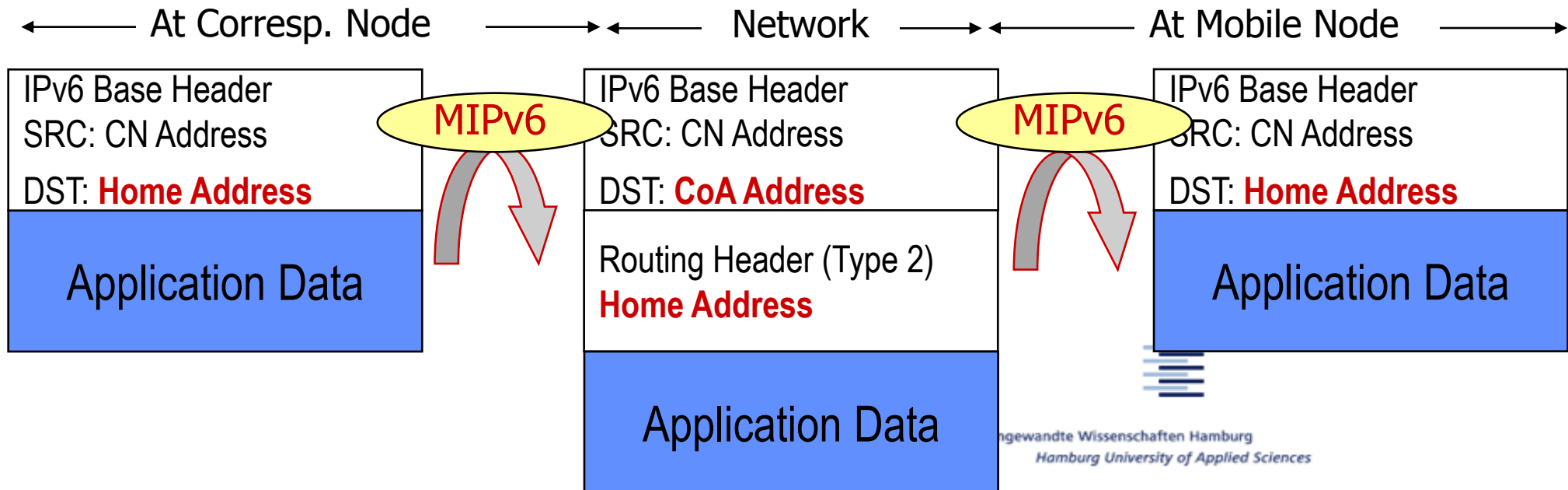
MIPv6 Transparent Communication MN → CN

- o Application persistence requires continuous use of HoA
- o Infrastructure requires use of topologically correct source address: CoA
- o MIPv6 stack moves HoA to Destination Option Header



MIPv6 Transparent Communication CN → MN

- o Application persistence requires continuous use of HoA
- o Route optimisation operates with CoA
- o MIPv6 extracts CoA from Binding Cache and initiates source routing to HoA via CoA



Security: The Core of the Problem?

For Authentication

A Mobile Node must proof ownership of HoA

But: Certification Infrastructure (PKI) is out of scope

Idea in IPv6:

Cryptographically Generated Addresses (Aura, Castellucia, Montenegro & Petander – RFC 3972):

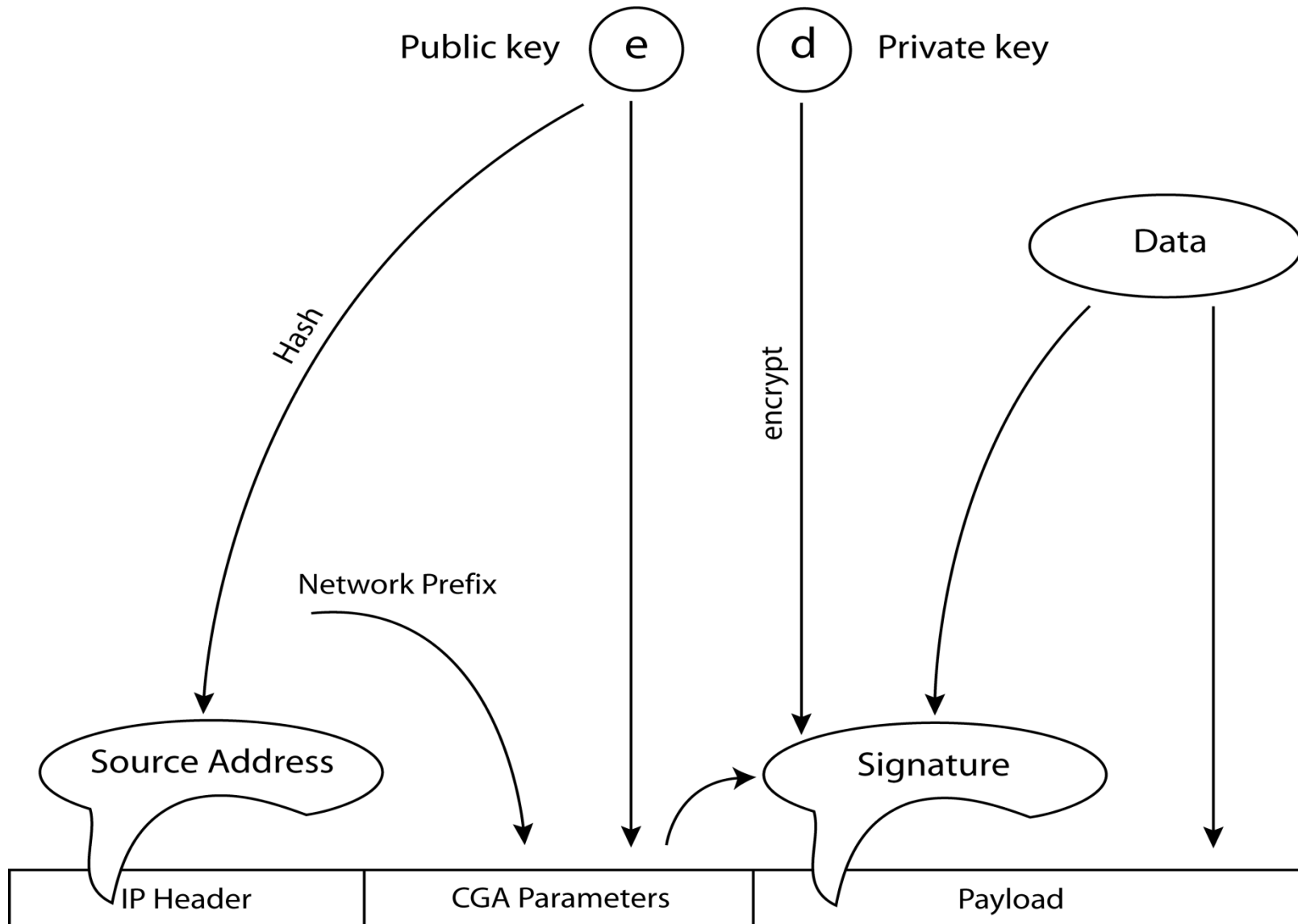
- o Generate public/private key pair: e, d
- o Generate host-ID from public key: 64 sha1(e)
- ➔ Packets now can authenticate their address (and content) self-consistently!

Cryptographically Generated Addresses (IPv6)

Problem: In IP infrastructure protocols the sender of a message frequently has to prove its 'ownership of address' to a receiver, it never met before. Authentication between unknown partners normally requires a public key infrastructure.

- Cryptographically Generated Addresses (CGAs) are source addresses formed from the public key.
- This mechanism allows the authentication of sender's address and the signing of data without a PKI.

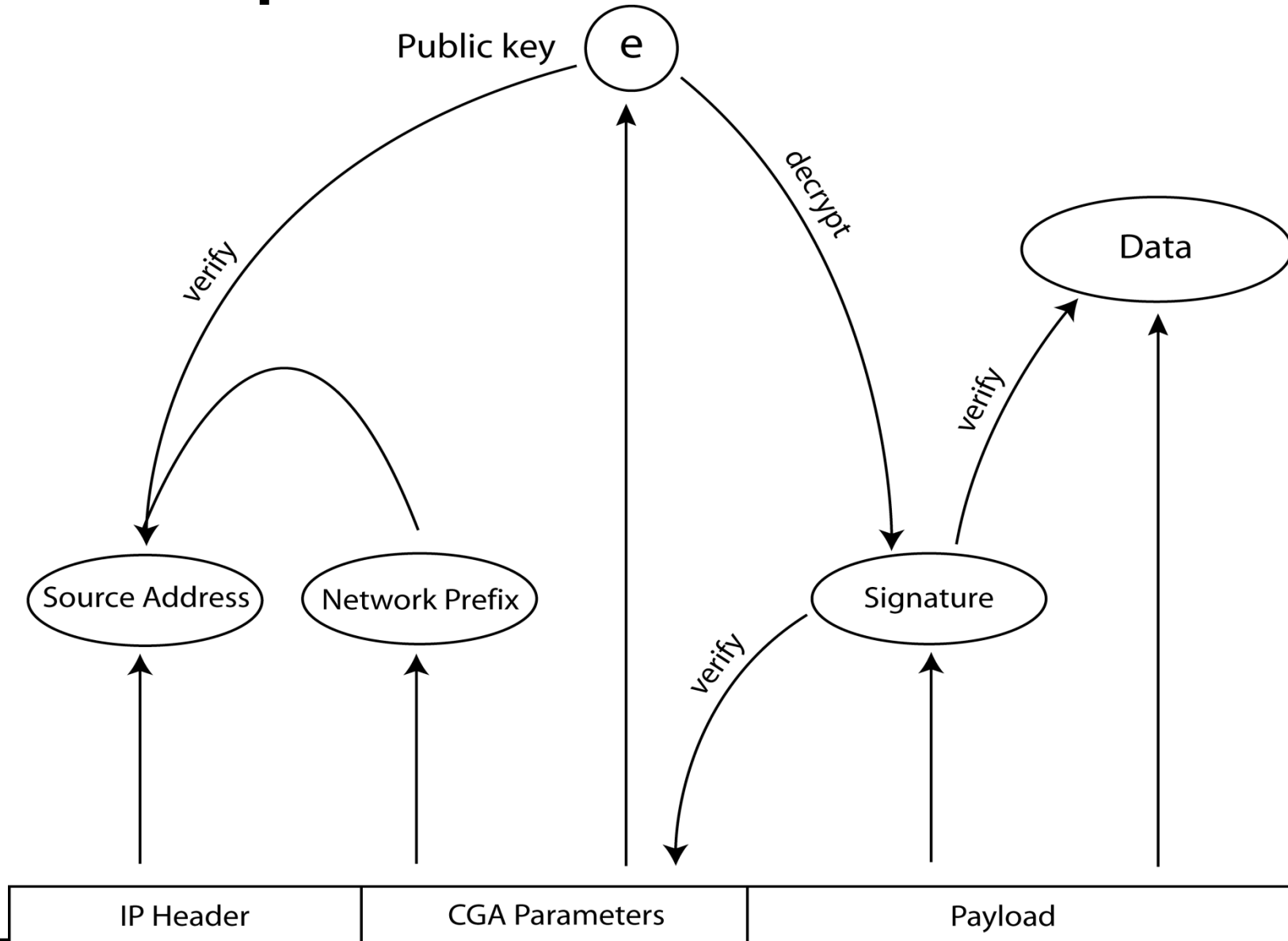
CGA Encapsulation



CGA: Encapsulation Steps

1. Sender forms public/private key pair e and d , calculates (node-) source address as a 64 bit hash from e .
2. Sender computes signature of network prefix, public key e , data ... encrypted with its private key d .
3. Sender includes (unencrypted) network prefix, e and the signature in a CGA parameter header within the packet.
4. Sender adds data and sends packet.

CGA Decapsulation



CGA: Decapsulation Steps

1. Receiver extracts network prefix and the public key e from the CGA parameter header.
2. It decrypts the signature with the public key e and verifies the CGA parameters + Data.
3. Receiver re-calculates and verifies sender's source address as a 64 bit hash from e .
4. The receiver can now be sure, that the received packet has been originally sent by the owner of the claimed IP address.

Binding Update

Enhanced Route Optimization for Mobile IPv6 (RFC 4866)

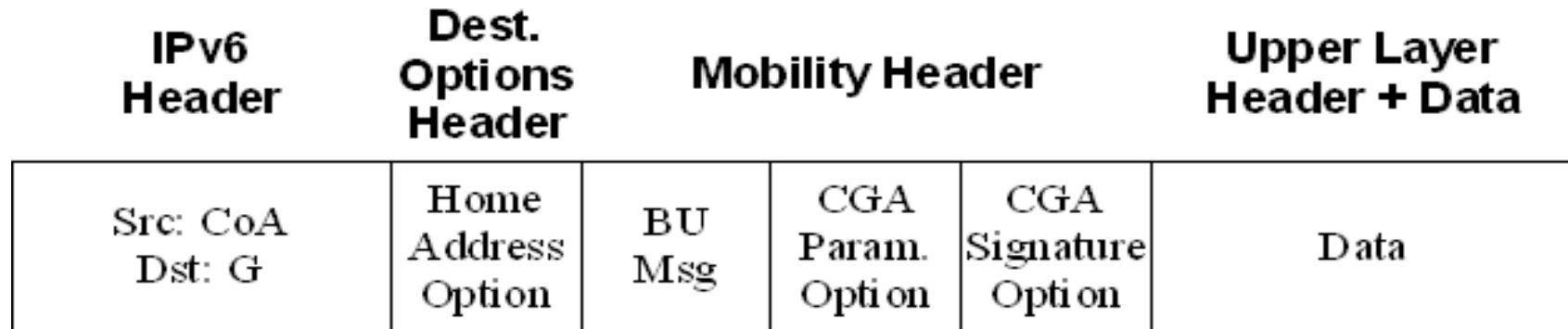
IPv6 Header	Dest. Options Header	Mobility Header			Upper Layer Header + Data
Src: CoA Dst: G	Home Address Option	BU Msg	CGA Param. Option	CGA Signature Option	Data



Base header is Home Address unaware.

Binding Update

Enhanced Route Optimization for Mobile IPv6 (RFC 4866)



The destination receives the Home Address in the Destination Options Header.

Binding Update

Enhanced Route Optimization for Mobile IPv6 (RFC 4866)

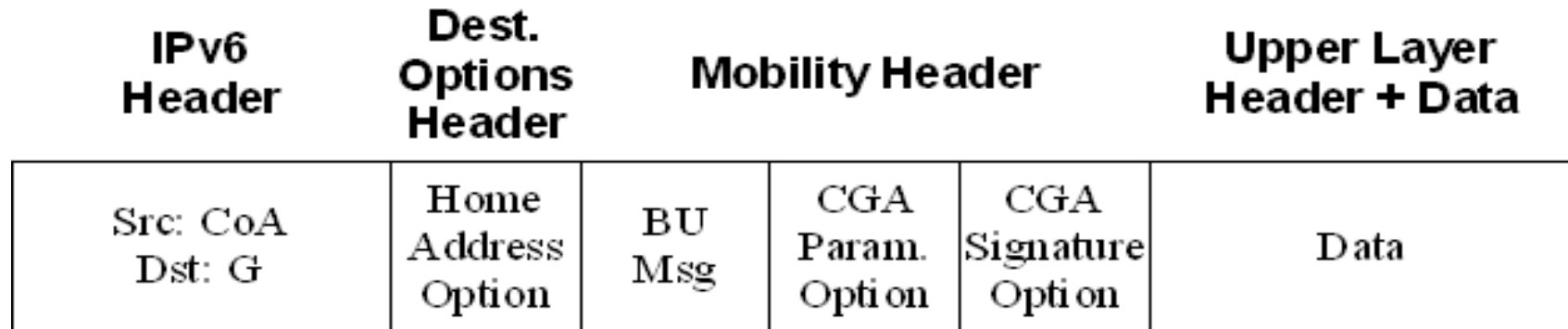
IPv6 Header	Dest. Options Header	Mobility Header			Upper Layer Header + Data
Src: CoA Dst: G	Home Address Option	BU Msg	CGA Param. Option	CGA Signature Option	Data



The update itself is stored in the Mobility Header.

Binding Update

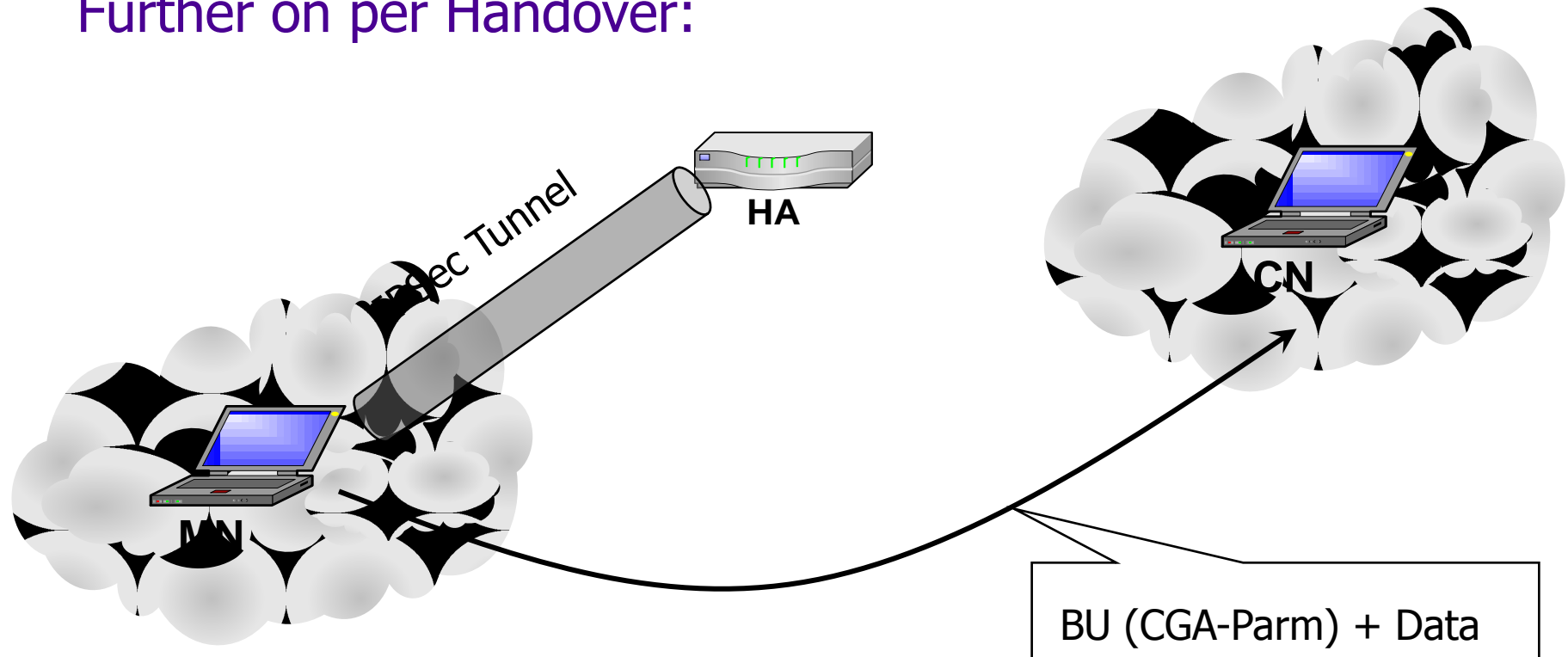
Enhanced Route Optimization for Mobile IPv6 (RFC 4866)



CGA options verify the HA and sign the packet

CGA-Authenticated BU (RFC 4866)

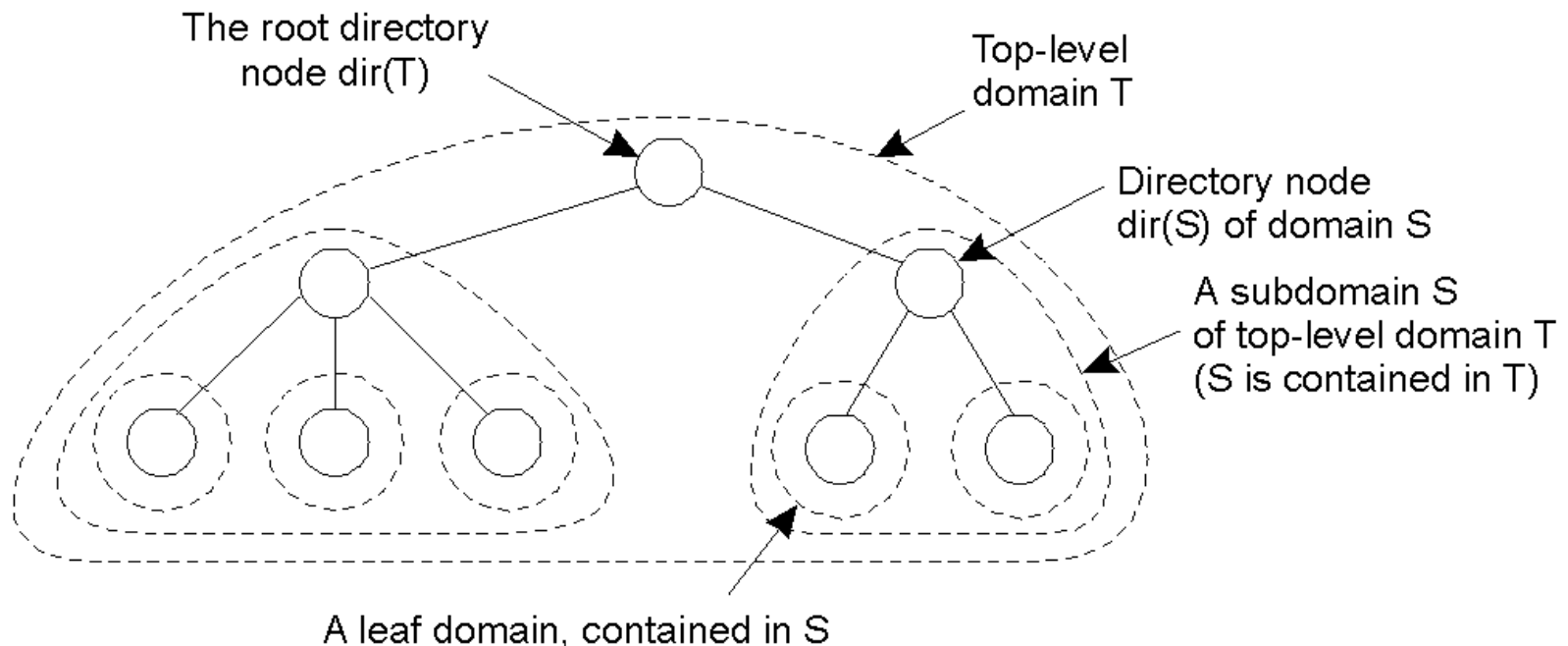
Initial HoA-Reachability Test
Further on per Handover:



Further Reading: *Mobility in IPv6: Standards and Upcoming Trends*, Uptimes (GUUG), 2007

Location Service: hierarchische Ansätze

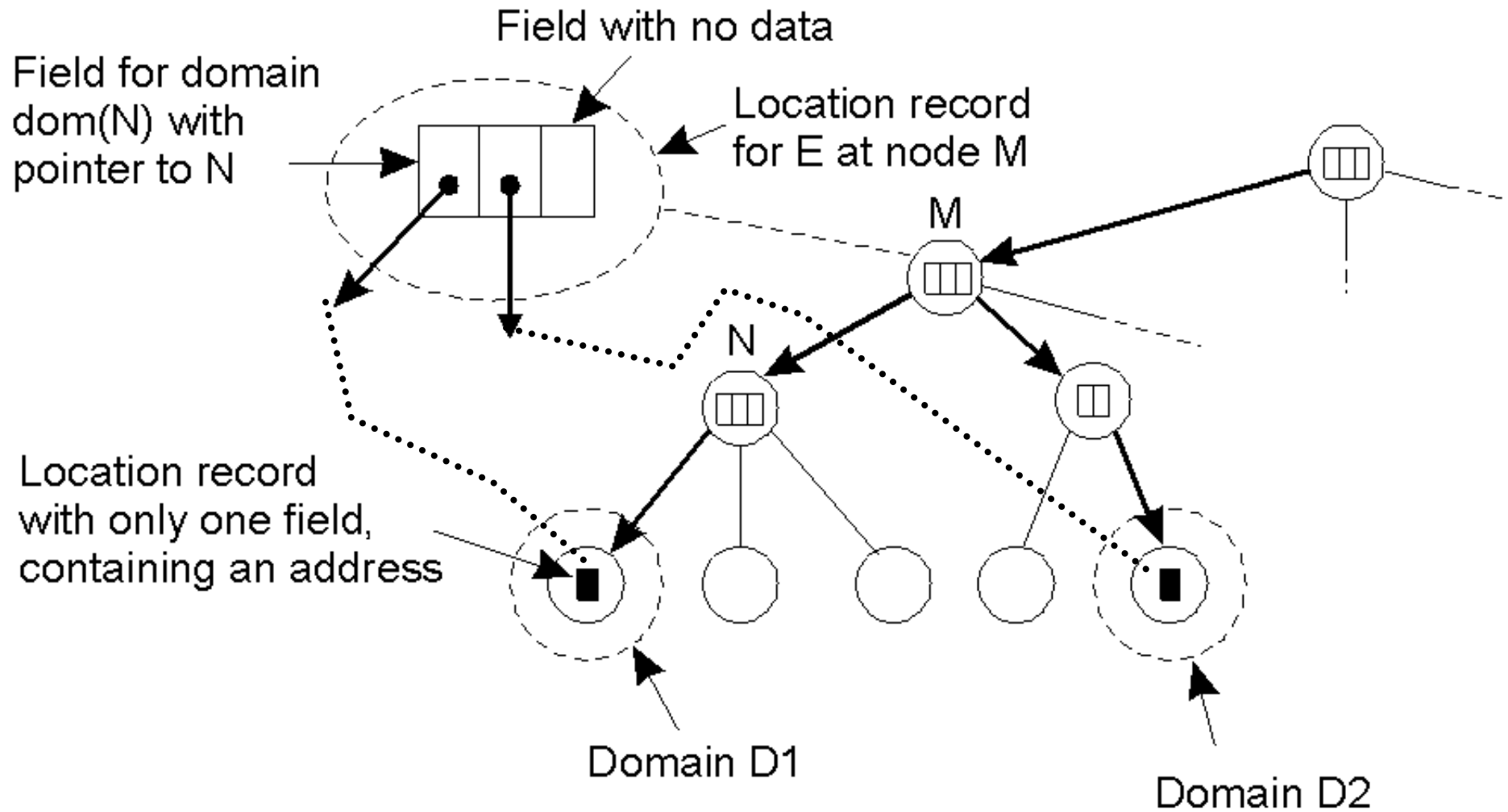
- ◆ Erweiterung der zweistufigen Home-Location zu mehreren Stufen
- ◆ ist ein (allgemeines) **hierarchisches Suchschema**
- ◆ **Ausgangspunkt:**
 - Netzwerk in mehrere Domänen hierarchisch unterteilt (ähnlich DNS)
 - Jeder Domäne ist Verzeichnisknoten $\text{dir}(D)$ zugeordnet



Hierarchischer Ansatz

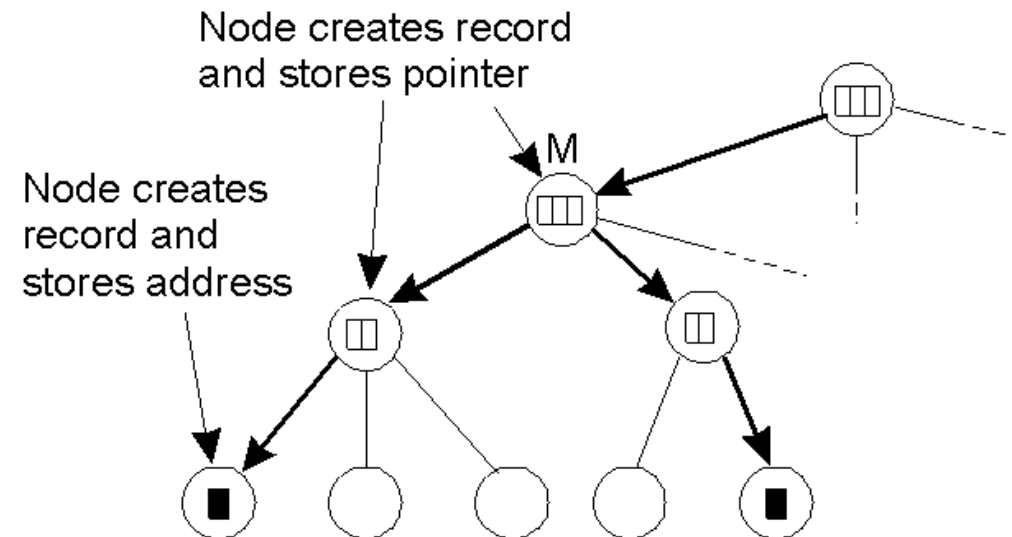
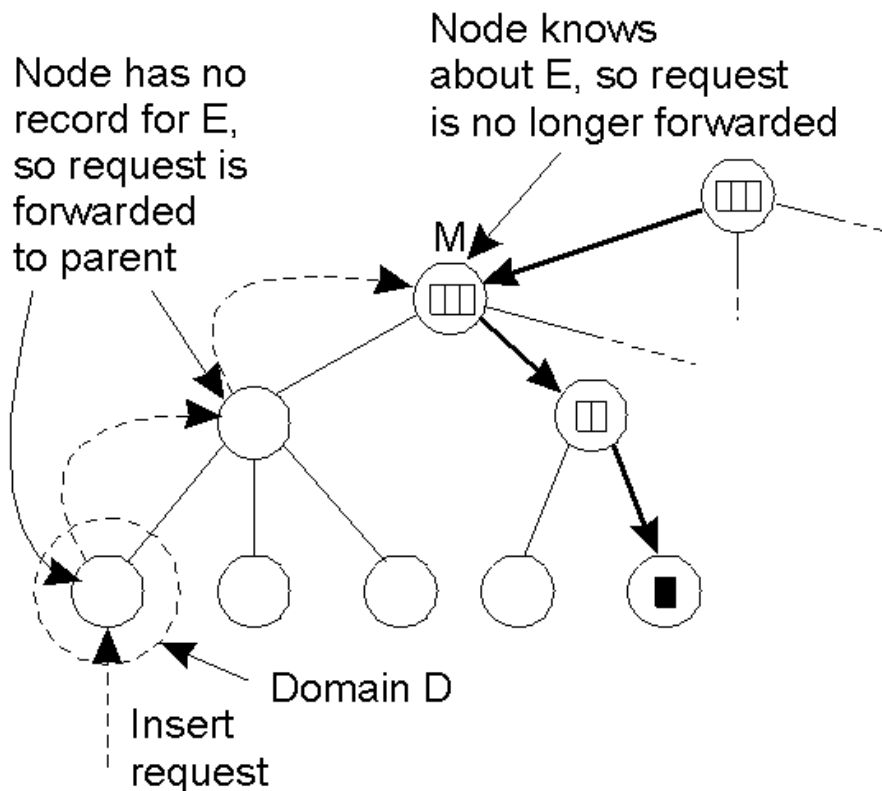
- ◆ Jedes Objekt E in Domäne D wird durch **Positionsdatensatz** in $\text{dir}(D)$ dargestellt.
 - Positionsdatensatz für Objekt O im Verzeichnisknoten N für Blattdomäne D enthält die **aktuelle Adresse** von O
 - Positionsdatensatz für Objekt O im Verzeichnisknoten N' für nächst höhere Domäne D' enthält nur **Zeiger auf N** .
 - Positionsdatensatz für Objekt O im Verzeichnisknoten N'' für nächst höhere Domäne D'' enthält nur **Zeiger auf N'** .
 - usw.
- ◆ **Wurzel-Knoten** besitzt Positionsdatensatz für **jedes Objekt**
- ◆ Bei **mehreren Adressen** (z.B. durch Replikate) existieren **mehrere Zeiger**

Hierarchischer Ansatz: Beispiel

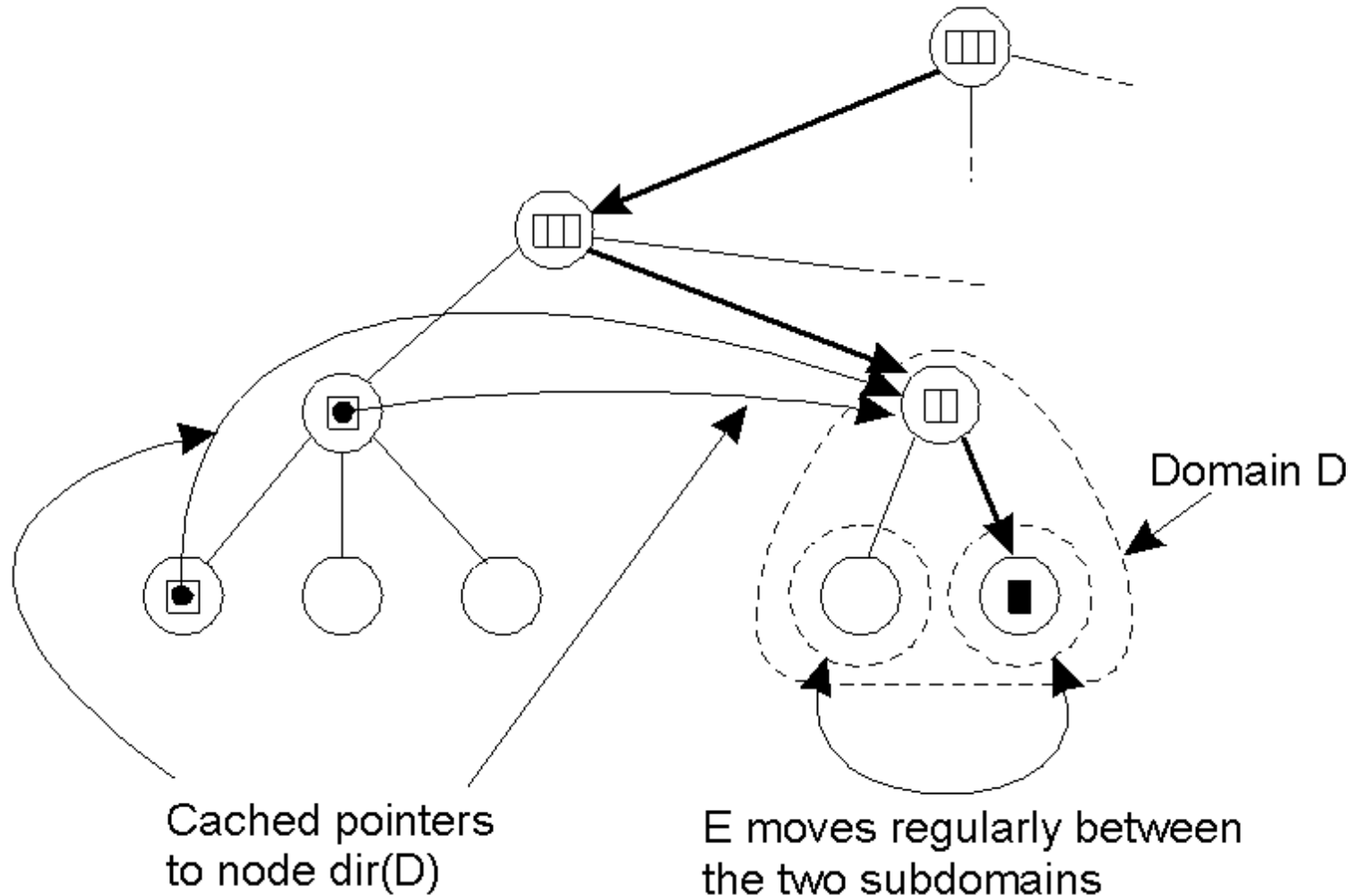


Hierarchischer Ansatz: Replikate

- ◆ Suchoperation nutzt die **Lokalität** aus
- ◆ Im **schlechtesten Fall** Suche bis Wurzel-Knoten
- ◆ Anforderung entlang **Abwärtspfad**es an Blattknoten weitergeben

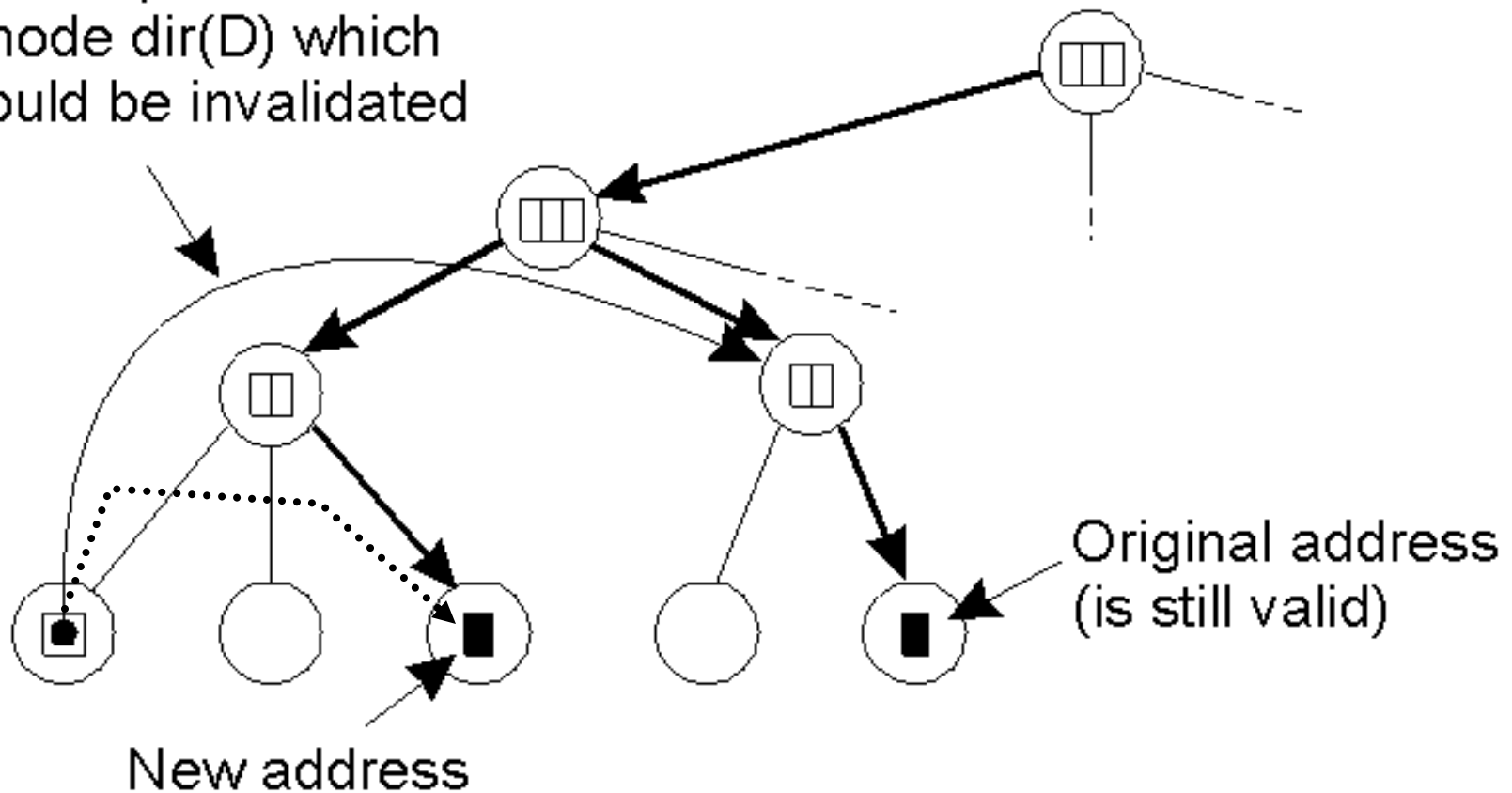


Hierarchischer Ansatz: Zeiger-Caches



Hierarchischer Ansatz: Zeiger-Caches

Cached pointer
to node $\text{dir}(D)$ which
should be invalidated



Peer-to-Peer Ansatz: Beispiel SIP

