

# **Client/Server-Systeme**

**Prof. Dr.-Ing. Wilhelm G. Spruth**

**WS 2005/2006**

**Teil 7**

**Namens- und Verzeichnisdienste  
Middleware**

# Namensgebung: Definitionen

## Alphabet

Menge von Zeichen,  
Beispiel {A,B,C}

## Name (symbolische Adresse, „symbolischer Name“)

Folge von Zeichen, meist in der Länge begrenzt  
Beispiele: AB, AAC  
beschreibt ein Objekt, welches gebraucht wird  
Objekte sind physikalische (CPU, Drucker) oder  
logische (Datei, Programm) Ressourcen.

## Namensraum

Menge aller möglichen Namen. Beispiel:  
{AA,AB,AC, BA, BB, BC, CA, CB, CC}  
für Namenslänge 2 und Alphabet {A,B,C}

## Naming Kontext

Namensraum, innerhalb dessen ein Objekt Name eindeutig ist.  
Namen können relativ zu ihrem Kontext definiert werden.

## Adresse

identifiziert den Ort, wo sich das Objekt befindet

## Binding

Zuordnung eines Namens zu einem Objekt  
statische und dynamische Bindung

## Objekt Identifier (OID), Universal Identifier (UID)

systemweiter Name, der ein Objekt eindeutig identifiziert.

# Verwendung symbolischer Adressen

Das IP Protokoll (Schicht 3) des Internet verwendet 32 Bit Integer Werte als Adressen.

Der Benutzer verwendet meistens symbolische Namen. Symbolische Namen haben nichts mit IP Adressen zu tun, werden aber durch definierte Mechanismen auf sie abgebildet.

Die Übersetzung von symbolischen Namen (sog. Host Namen), z.B.

jedi.informatik.uni-leipzig.de

in 32 Bit lange IP Adressen, z.B.

Hex	8b	12	4	23
binär	1000 1011	0001 0010	0000 0100	0010 0011
oder in Dezimalnotation	139.18.4.35			

findet in der Schicht 6 statt.

Die Zuordnung (Binding) symbolischer Name (host name) zu IP Adresse befindet sich in der lokalen Host Tabelle ( /etc/hosts )..

In der Anwendungsschicht ist für IP Adressen auch die Verwendung der Dezimalnotation möglich, z.B.

http://139.18.4.35

an Stelle von

http://jedi.informatik.uni-leipzig.de

## Client Process Using Stream Sockets

```
/* Error checking has been omitted to allow for easy
 * reading. However, error checks should always be
 * implemented. */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <netinet/in.h>
#define PORTNUM 1503
#define MESSAGE "Socket Test"

main()

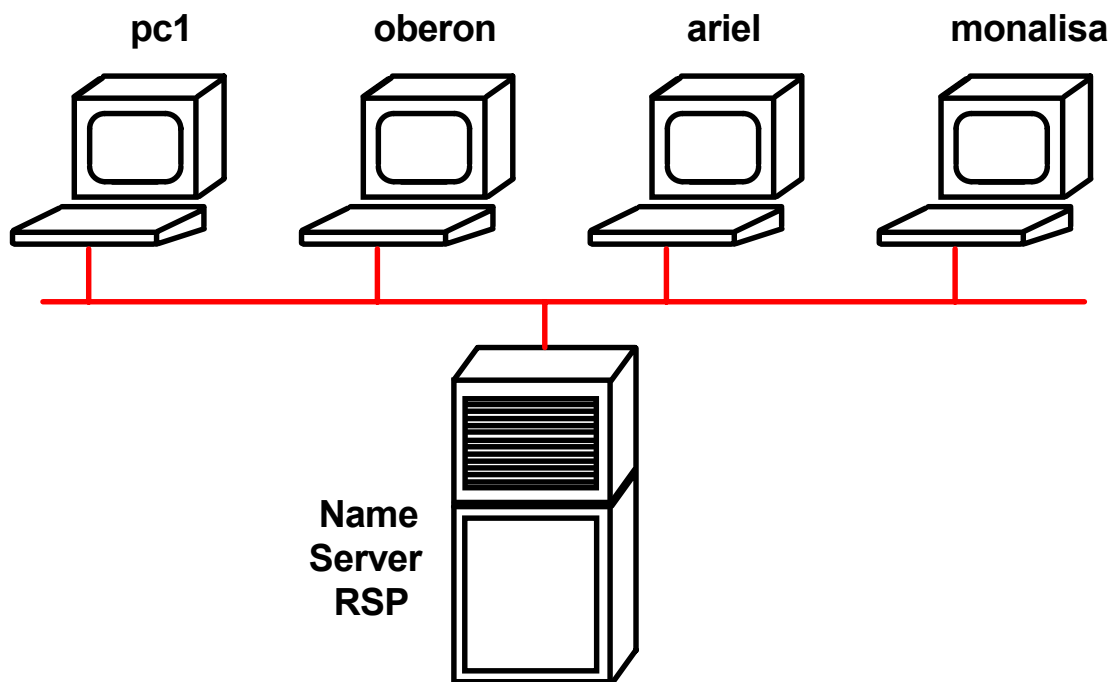
int sd;                /* client socket descriptor */
struct sockaddr_in
    servername;        /* server socket name */
struct hostent *host;  /* server host struct */

/* create client socket */
sd = socket(AF_INET, SOCK_STREAM, 0);

/* Obtain network address of server.
 * "myhost" is used as server name.
 */
host = gethostbyname( "myhost" );

/* Initialize server socket address */
bcopy( host->h_addr,
        (char*) &servername.sin_addr, host->h_length);
servername.sin_family = AF_INET;
servername.sin_port = PORTNUM;
/* Attempt to connect to server */
connect (sd, (struct sockaddr *)&servername,
         sizeof(servername));

/* write data to socket; close socket */
write ( sd, MESSAGE, sizeof( MESSAGE ));
close(sd);
}
```



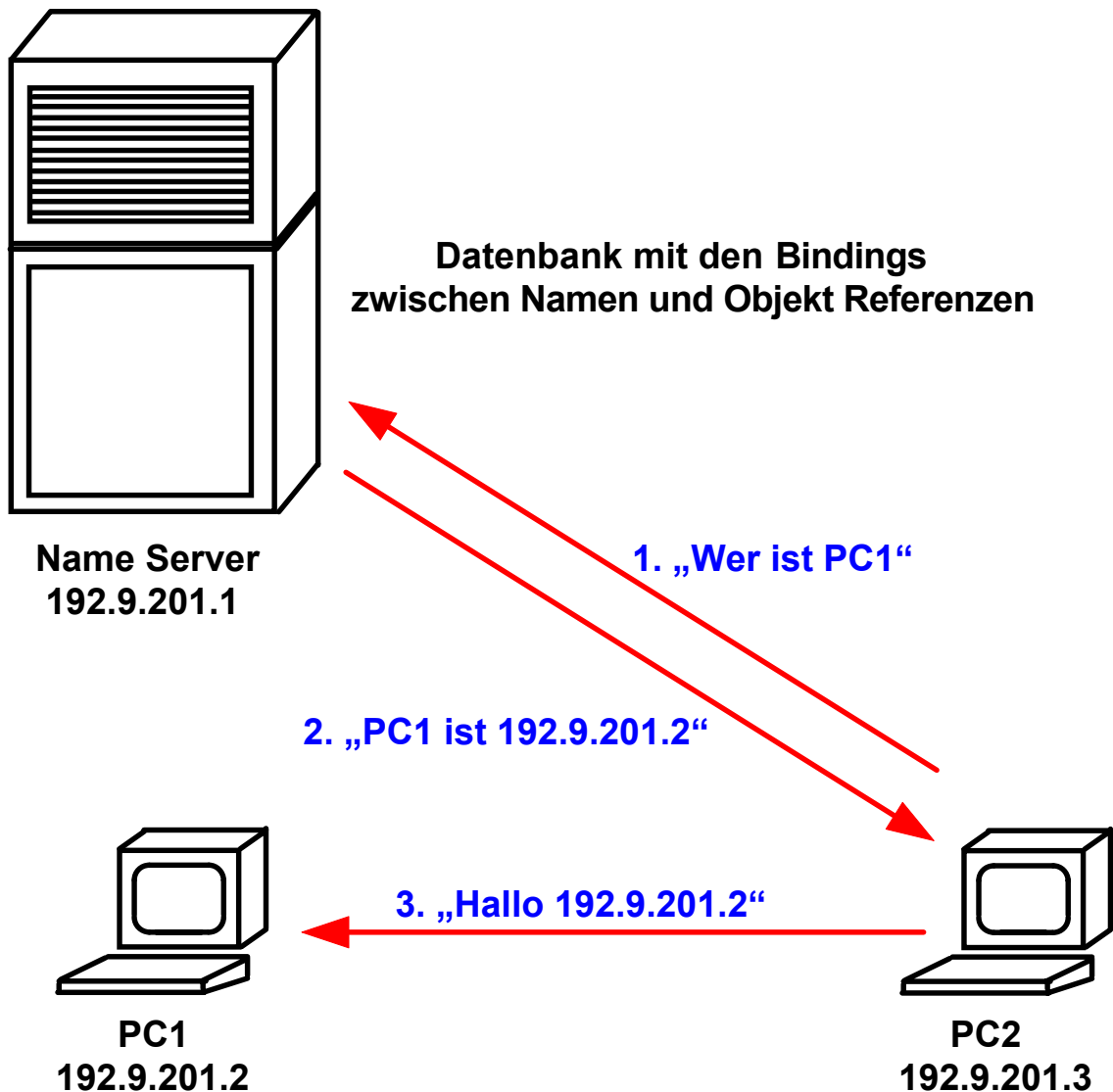
```
# hosttable SERVER RSP
192.9.201.1 rsp # SERVER
192.9.201.2 PC1
192.9.201.4 oberon
192.9.201.5 ariel
192.9.201.9 monalisa
```

## Probleme mit lokalen Hosttabellen

In einem kleinen Netz speichert die lokale Hosttabelle eines Klienten die Adressen aller angeschlossenen Rechner (Klienten und Server).

In einem großen Netz mit vielen Rechnern ist es schwierig, alle Hosttabellen gleichzeitig auf dem richtigen Stand zu halten (Administrationskosten).

**Lösung:** Einer der Rechner übernimmt die Rolle des „Name Servers“.



## Adressauflösung durch Name Server

Der Namensdienst (Name Service) bildet die für einen menschlichen Benutzer verständlichen Bezeichnungen auf Objekt ID's ab (OID's, auch als „Objekt Referenzen“ bezeichnet, u.a. CORBA Begriff). Jedes Objekt hat eine eindeutige Objekt ID.

\*

Als „name binding“ bezeichnen wir die Zuordnung  
Name - Objekt\_Referenz

Der Namensdienst unterhält eine Datenbank der Bindings zwischen Namen und Objekt Referenzen.

# Domain Name System - DNS

In der Anfangszeit des Internet hielt nic.ddn.mil (Defense Data Network Information Center) eine zentrale Datei vor, die auf alle Hosts des Internet verteilt wurde.

Die Verwaltung einer einzigen Host Tabelle wurde mit der wachsenden Größe des Internets unmöglich

Der „Domain Name Service“ (DNS) ist die Lösung für das Internet. Festgehalten in RFC 1035.

Aufteilung des Internets in (weiterhin aufteilbare) Einheiten (Domänen).

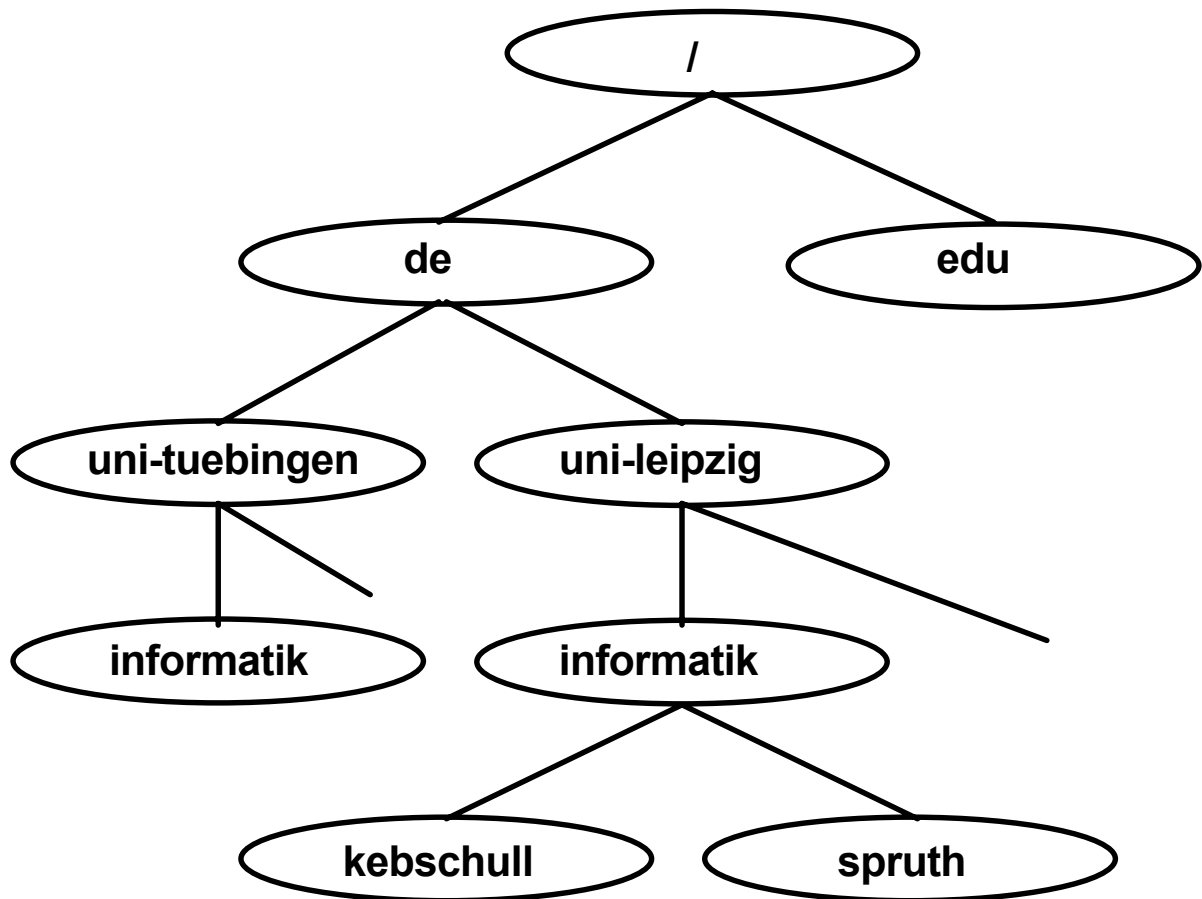
Grundlage ist eine

- hierarchisch gegliederte
- verteilte
- replizierte

Datenbank

Verwendet das UDP Protokoll (für lange Nachrichten TCP)

# Hierarchisch strukturierter Namensraum

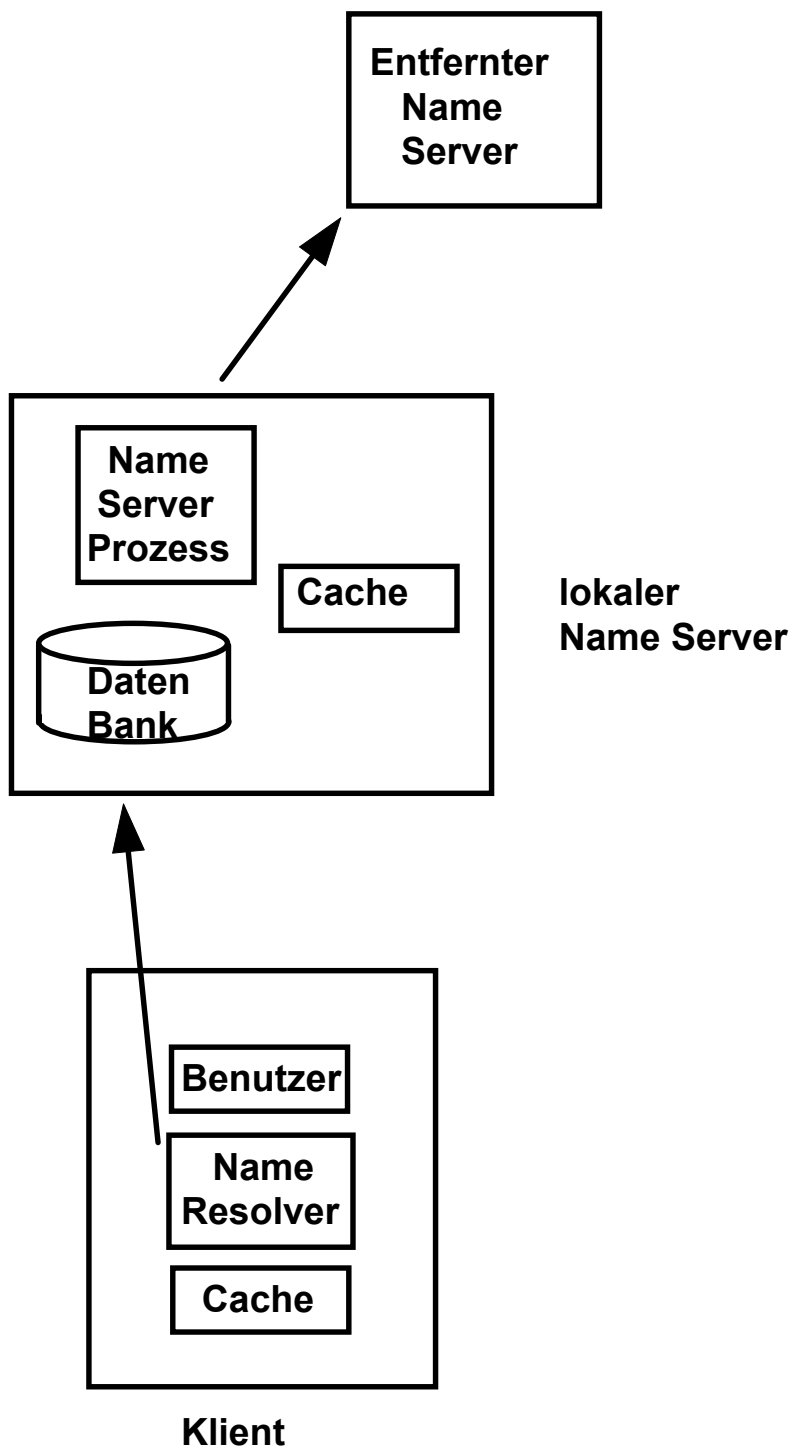


**Hierarchie kann nachbilden:**

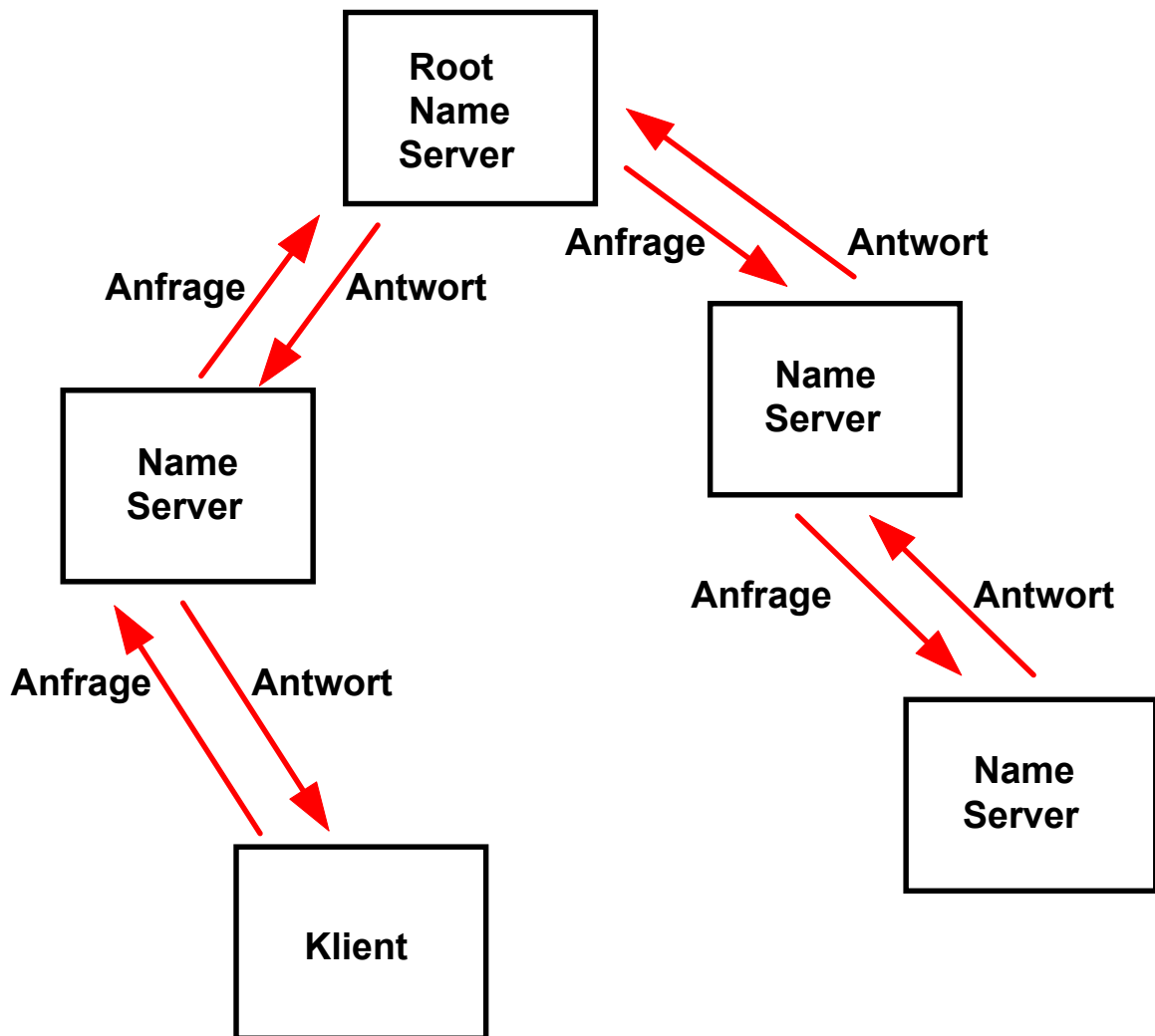
- Organisatorische Struktur**
- Systemtopologie**
- geographische Verteilung**

**Ein relativer Name identifiziert ein Objekt eindeutig im Kontext seines Vaterobjektes. Beispiel: informatik/kebschull**

**Ein absoluter Name identifiziert ein Objekt global eindeutig. Beispiel: /de/uni-leipzig/informatik/kebschull**

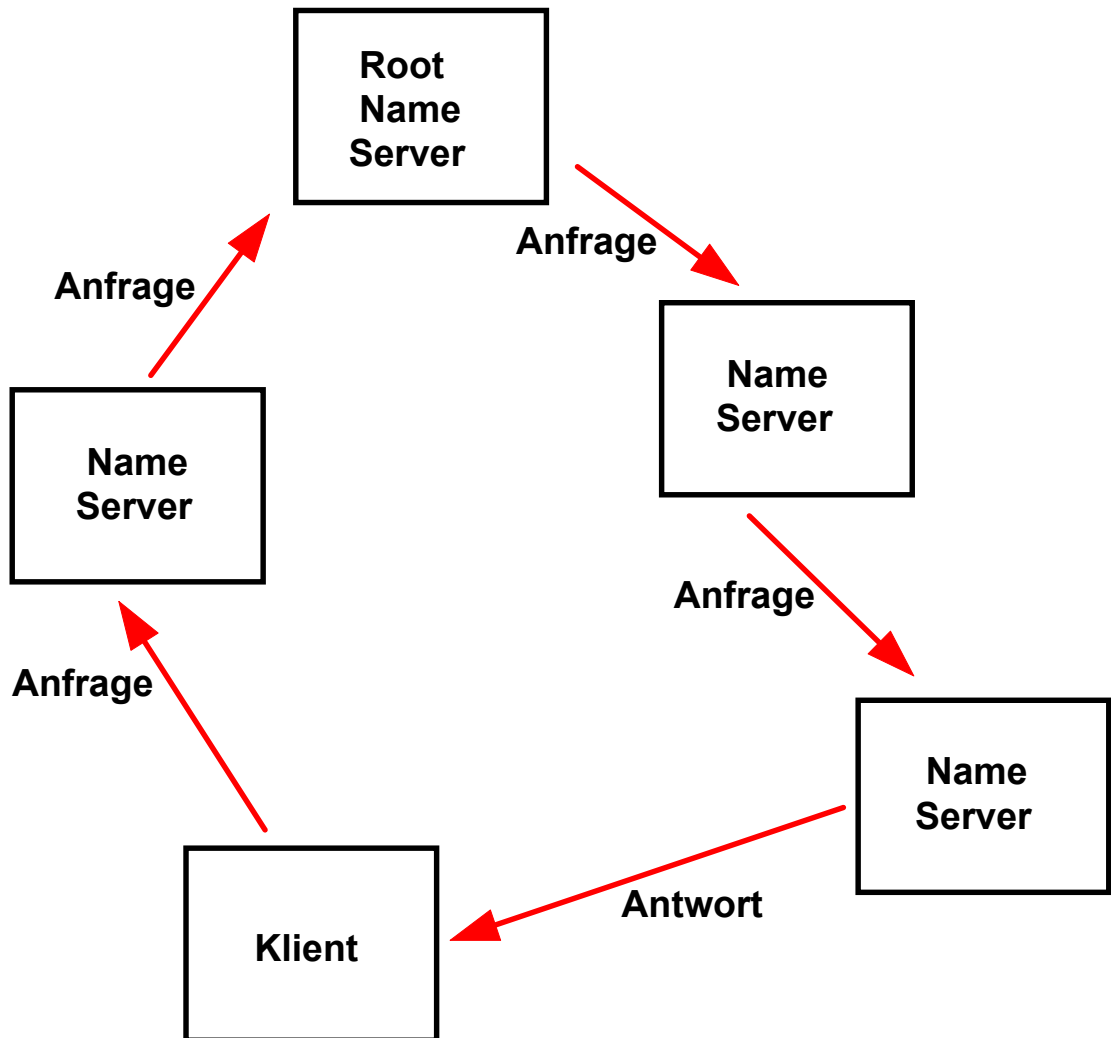


## Rekursive Namensauflösung



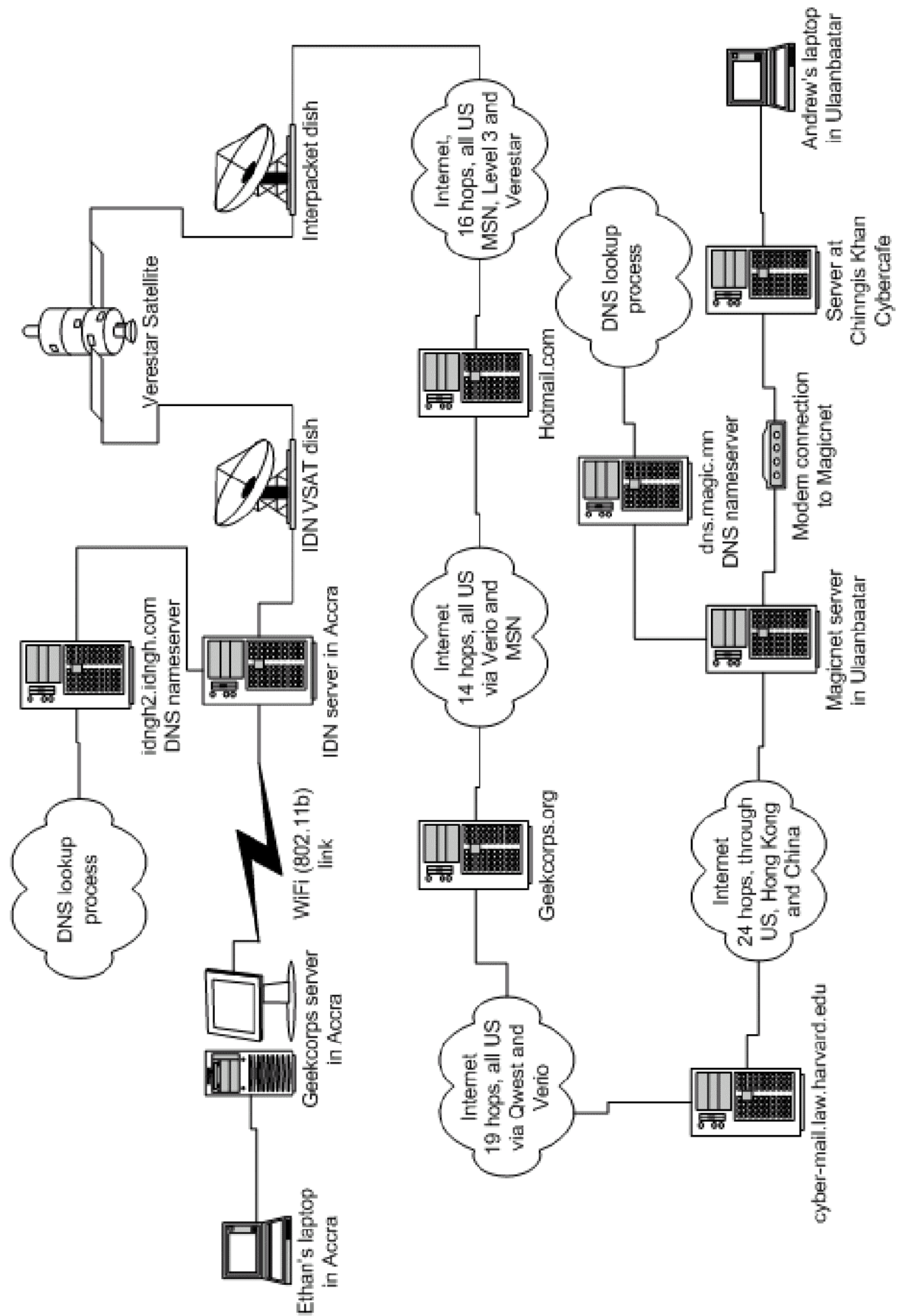
## Rekursive Namensauflösung im DNS

**Anfrage wird von Server zu Server weitergegeben**  
**Antwort wird auf dem selben Weg zurückgegeben**

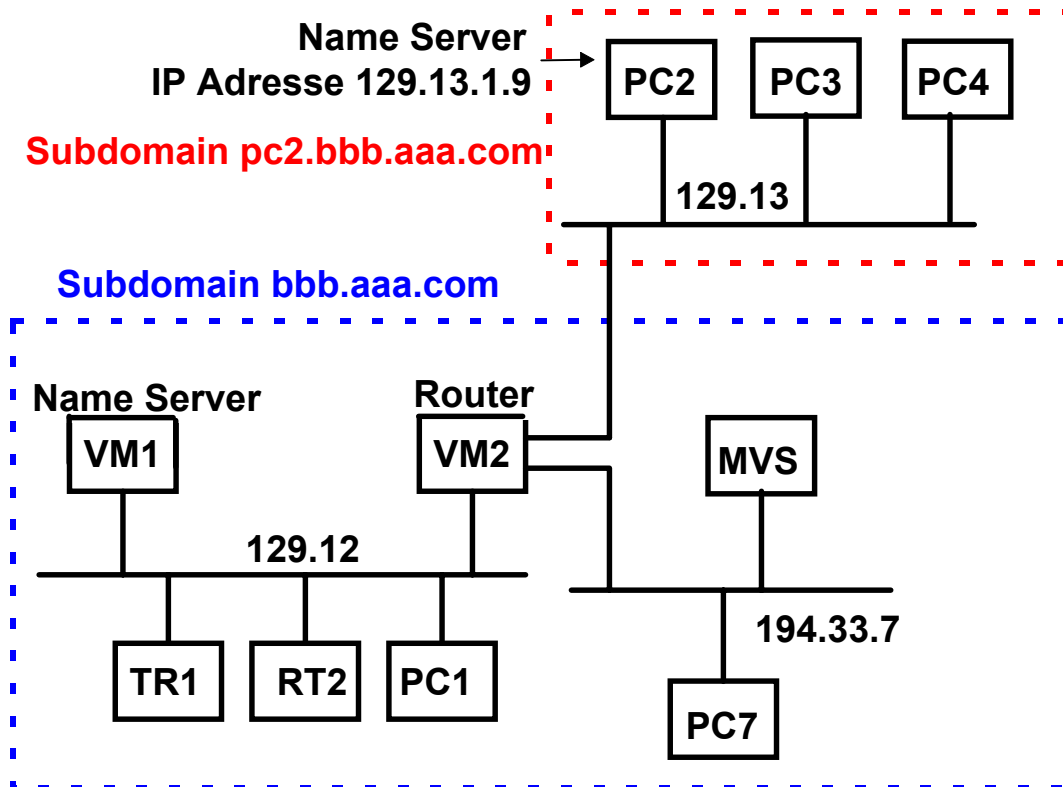


## Rekursive Namensauflösung im X.500

**Anfrage wird von Server zu Server weitergegeben**  
**Antwort wird auf direktem Weg zurückgegeben**



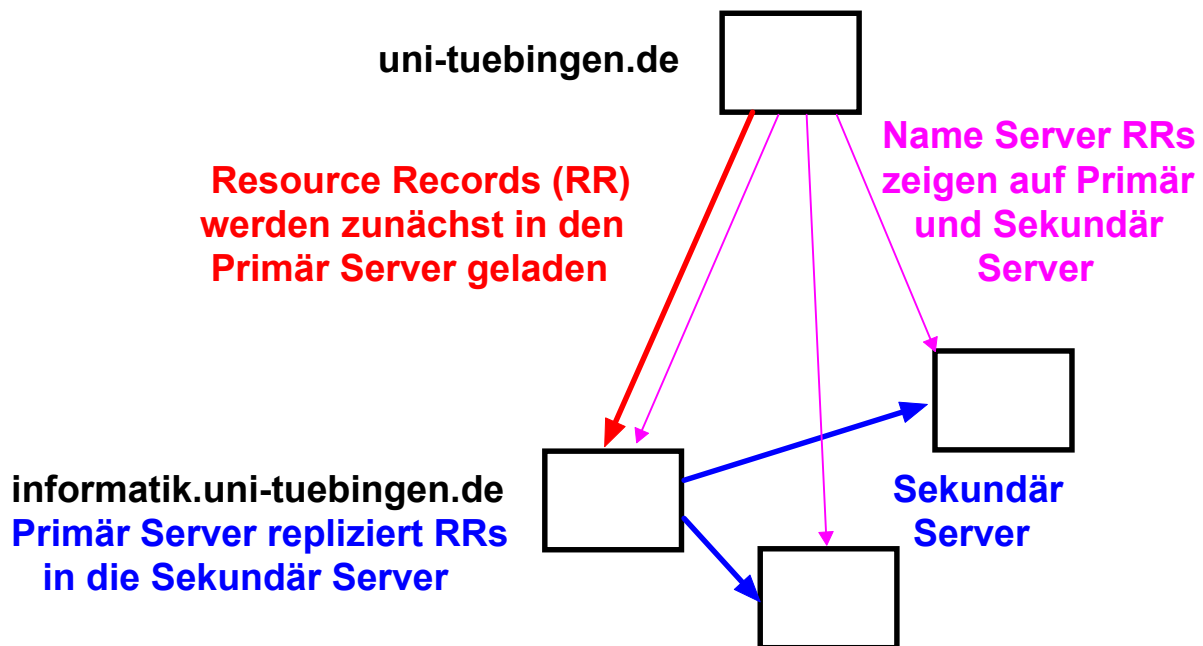
## Subdomänen und Zonen



DNS ist eine logische Hierarchie, unabhängig von der physikalischen Konfiguration. Die Netze 129.12 und 194.33.7 bilden die Subdomain `bbb.aaa.com`. Der Rechner VM1 dient als Name Server. Das Netz 129.13 bildet eine eigene Subdomain `ccc.bbb.aaa.com`, die hierarchisch untergeordnet ist. Der Rechner PC2 mit der IP Adresse 129.13.1.9 dient als Name Server.

Einträge im Name Server VM1 besagen, dass PC2 der Name Server für die Subdomain `ccc.bbb.aaa.com` ist, und über die IP Adresse 129.13.1.9 erreicht werden kann.

# Replikation der Namensserver



Update der Sekundär Server in unregelmäßigen Zeitabständen. Alle Server speichern Anfragen aus jüngster Zeit in ihrem Cache. Integrität der Information ist nicht gewährleistet. „Time-to-Life“ Angabe für maximale Lebensdauer im Cache.

Transient Inconsistency der Daten.

Subdomains sind Teil der Domain Hierarchie und durch einen Domain Namen identifiziert.

Der DNS Namensraum ist in nonoverlapping *Zonen* aufgeteilt. Jede Zone enthält einen Teil des DNS-Baums (eine oder mehrere Subdomains) und hat einen Primär-Namensserver sowie möglicherweise mehrere Sekundärserver. Jeder Namensserver hat einen konfigurierbaren Zeitgeber.

# Autorisierung von Namen

**Die Administration der einzelnen Subdomänen (oder Zonen) erfolgt dezentral und unabhängig voneinander durch die jeweiligen Administratoren.**

## **Beispiel**

**tech01.informatik.uni-leipzig.de**

**Die Benutzung des Feldes "tech01" wurde vom Netzwerk-Administrator des Informatik-Institutes autorisiert.**

**Die Benutzung des Feldes "Informatik" wurde vom Netzwerk-Administrator des Universitätsnetzes autorisiert.**

**Die Benutzung des Feldes "uni-leipzig" wurde vom übergeordneten Netzwerk-Administrator (DENIC) autorisiert.**

**Die Benutzung des Feldes "de" wurde vom Internet Network Information Center (ICANN) autorisiert.**

# DNS - Datenbank

Die DNS-Datenbank besteht aus *Resource Records*. (RR). Jeder Resource Record enthält folgende Felder: (RFC 2929)

- Subdomain Name
- Type
- Class (wenig benutzt)
- Time to live
- Datenlänge
- Daten

Es existieren 20 unterschiedliche Typen. Sie wichtigsten sind:

Address Record (A)	Zusammenhang zwischen Host Name und IP-Adresse
Name Server Record (NS)	Adresse eines Name Servers, der für eine bestimmte Domain zuständig ist
Mail Exchange Record (MX)	Über MX Records können Hosts definiert werden, die für andere Hosts Mail entgegennehmen (z.B. weil letztere nicht ständig mit dem Netz verbunden ist. UUCP benötigt MX Records)

Jede Anwendung, die eine Anfrage an einen Namens-Server stellt, muß den Typ angeben

z.B. E-mail Anwendung spezifiziert den Typ *mail-exchanger*

login Anwendung spezifiziert den Typ *IP-Adresse*

# Domain Name System (2)

**Alle Einträge besitzen ein "Typ" Feld.**

**Jede Anwendung, die eine Anfrage an einen Namens-Server stellt, muß den Typ angeben**

**z.B. E-mail Anwendung spezifiziert den Typ**

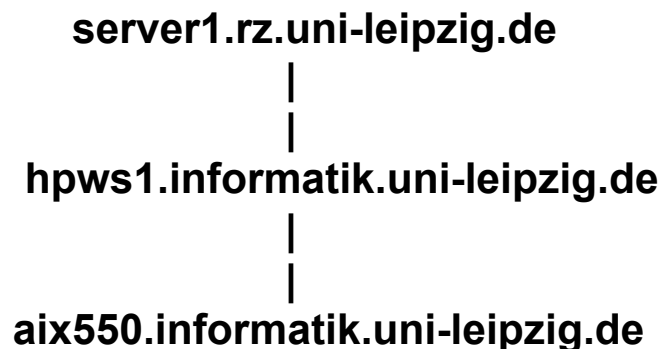
**mail-exchanger**

**login Anwendung spezifiziert den Typ**

**IP-Adresse**

# Adressenauflösung electronic mail

Beispiel: `spruth@informatik.uni-leipzig.de`



Alle Internet Nachrichten werden in Leipzig von der Maschine `server1.rz.uni-leipzig.de` empfangen.

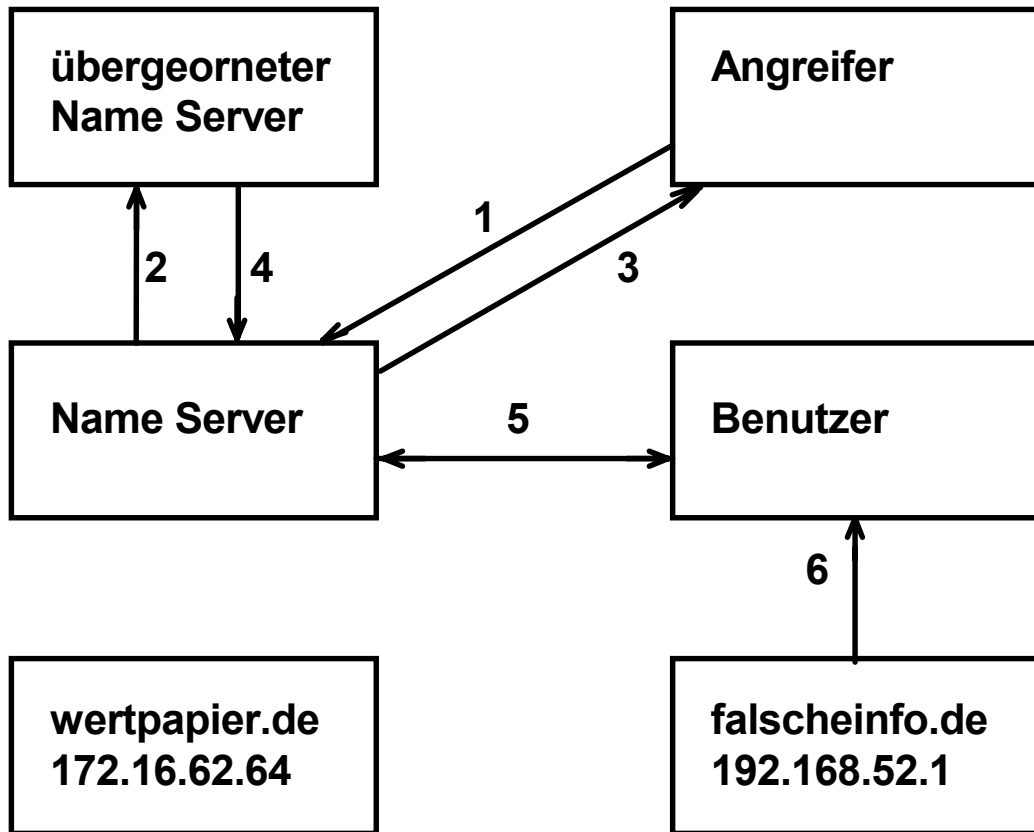
Alle Nachrichten die für die Informatik bestimmt sind, werden mit Hilfe einer Adresstabelle an `hpws1.informatik.uni-leipzig.de` weitergeleitet.

Der Server auf dieser Maschine untersucht die Nachricht und stellt z.B. fest, daß es sich um eine electronic mail handelt. Darauf schaut der mail-Server in seinem Adressenverzeichnis nach, und stellt fest, daß der Benutzer `SPRUTH` seine mail-box auf der Maschine `aix550.informatik.uni-leipzig.de` hat. Er leitet die mail an die `aix550` weiter.

Der dortige Mail-Server wiederum stellt die mail-Nachricht in die mailbox für den Benutzer `SPRUTH` ein.

Bei ftp würde die Adresse `spruth@informatik.uni-leipzig.de` nicht ausreichen, da der ftp-Server auf der `hpws1.informatik.uni-leipzig.de` über keine entsprechende Adresstabelle verfügt. Für den ftp transfer ist die adresse `spruth@aix550.informatik.uni-leipzig.de` erforderlich.

# DNS Spoofing



1. Der Angreifer erfragt die IP Adresse von wertpapier.de
2. Der Name Server leitet die Anfrage an seinen übergeordneten Name Server weiter.
3. Der Angreifer erstellt eine falsche Antwort mit der IP Adresse 192.168.52.1. Der Name Server stellt diese Antwort in seinen Namens-Cache
4. Der übergeordnete Name Server schickt die richtige Antwort mit der IP Adresse 172.16.62.64. Diese wird als scheinbares Duplikat verworfen
5. Der Benutzer erfragt vom Name Server die IP Adresse von wertpapier.de und erhält die falsche Antwort 192.168.52.1
6. Der Benutzer erhält von dem trojanischen Pferd 192.168.52.1 irreführende Information

# Verzeichnisdienst (Directory Service)

Ähnlichkeit mit Namensdienst, z.B. DNS. Kann als Namensdienst eingesetzt werden.

Erweiterte Funktionalität. Ein Verzeichnisdienst ist eine Datenbank, aber andersartige Eigenschaften als z.B. eine relationale Datenbank: Replikation ähnlich DNS, keine Gewährleistung der Datenkonsistenz, ungeeignet z.B. für Transaktionsverarbeitung.

Verzeichnisdienste organisieren Informationen in einer Baum-Struktur als eine Hierarchie von *Objekten*. Innerhalb des Baum zwei Typen von Einträgen: *Organizational Units* (OUs) und *Objekte*. OUs sind Container die andere OUs oder Objekte enthalten. Hierarchie kann auf unterschiedliche Server aufgeteilt sein.

Objekte werden zu Klassen zusammengefasst (z.B. Personen, Server, X.509 Zertifikate, Drucker, oder Kommunikationseinheiten). Eine OU kann unterschiedliche Klassen von Objekten enthalten.

Objekte eines Verzeichnisdienstes sind keine echten Objekte: Nur Daten, keine Methoden, keine Vererbung.

Beispiele für Verzeichnisdienst Implementierungen sind:

- OSI X.500
- LDAP (z.B. Novell NDS)
- Microsoft Active Directory Services (ADS)
- Java, Corba bauen häufig auf LDAP auf

# Aufgabe eines Verzeichnisdienstes

**Beispiel:**

**Sie wollen Mail an Ihren Freund Fritz Müller Schicken.**

**Fritz Müller hat eine neue E-Mail Adresse, Sie wissen aber nicht welche.**

**Ein Verzeichnisdienst enthält eine Auflistung aller Personen. Der Eintrag für Fritz Müller enthält auch seine (neue) E-Mail Adresse.**

**Die Information des Verzeichnisdienstes ist in Klassen gegliedert.**

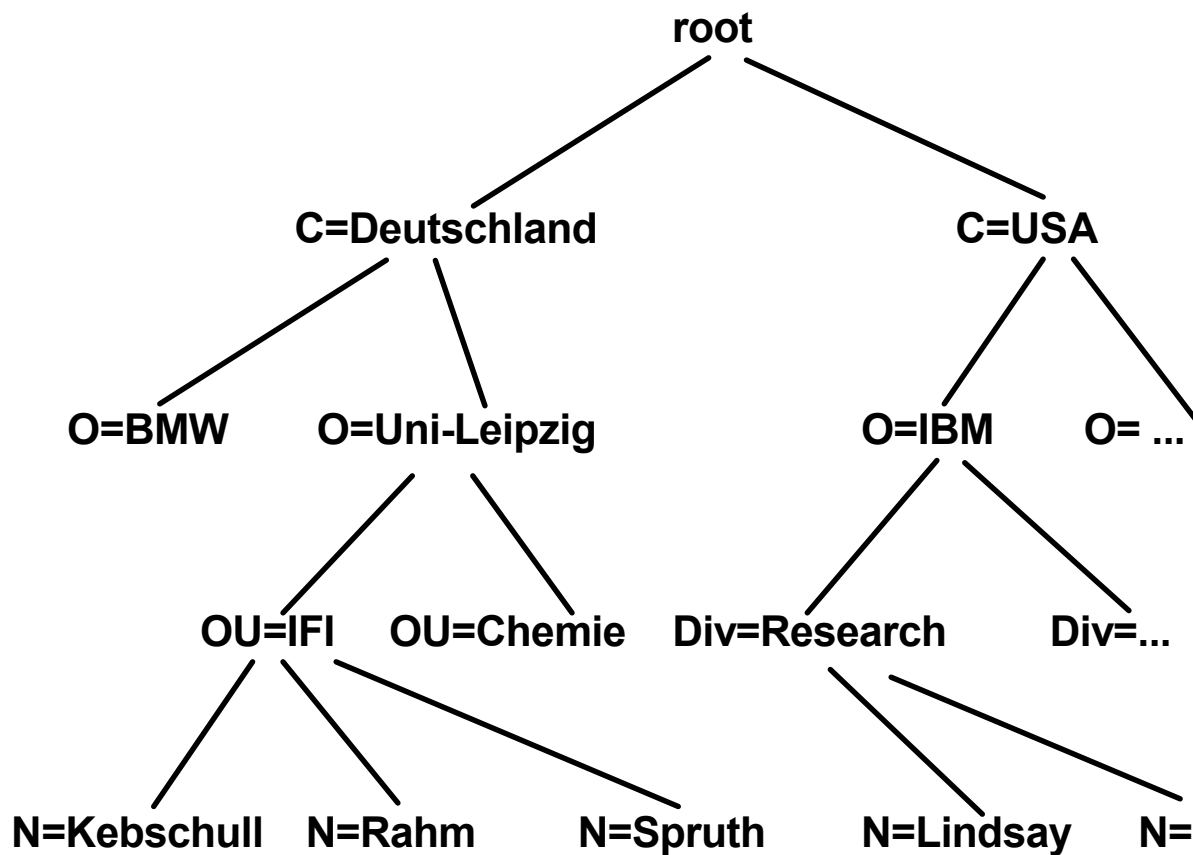
**Neben der „Klasse“ Personen (Fritz Müller ist eine Instanz dieser Klasse) kann der Verzeichnisdienst auch weitere Klassen enthalten.**

**Beispielsweise kann ein Prozess die Netzwerkadresse eines anderen Prozesses erfragen. Ein Klient kann den öffentlichen Schlüssel eines Servers abfragen.**

# X.500 Directory Service

OSI-Standard, der heute auch mit anderen Netzwerkprotokollen als OSI verwendet werden kann, z.B. TCP/IP.

## Beispiel



## Beispiel für eine X.500 Adresse

C=DE.O=Uni-Leipzig.OU=Informatik.N=Spruth

als Alternative zu `spruth@informatik.uni-leipzig.de`

## **Beispiel für eine X.500 Adresse**

**C=DE.O=Uni-Leipzig.OU=Informatik.N=Spruth**

**als Alternative zu [spruth@informatik.uni-leipzig.de](mailto:spruth@informatik.uni-leipzig.de)**

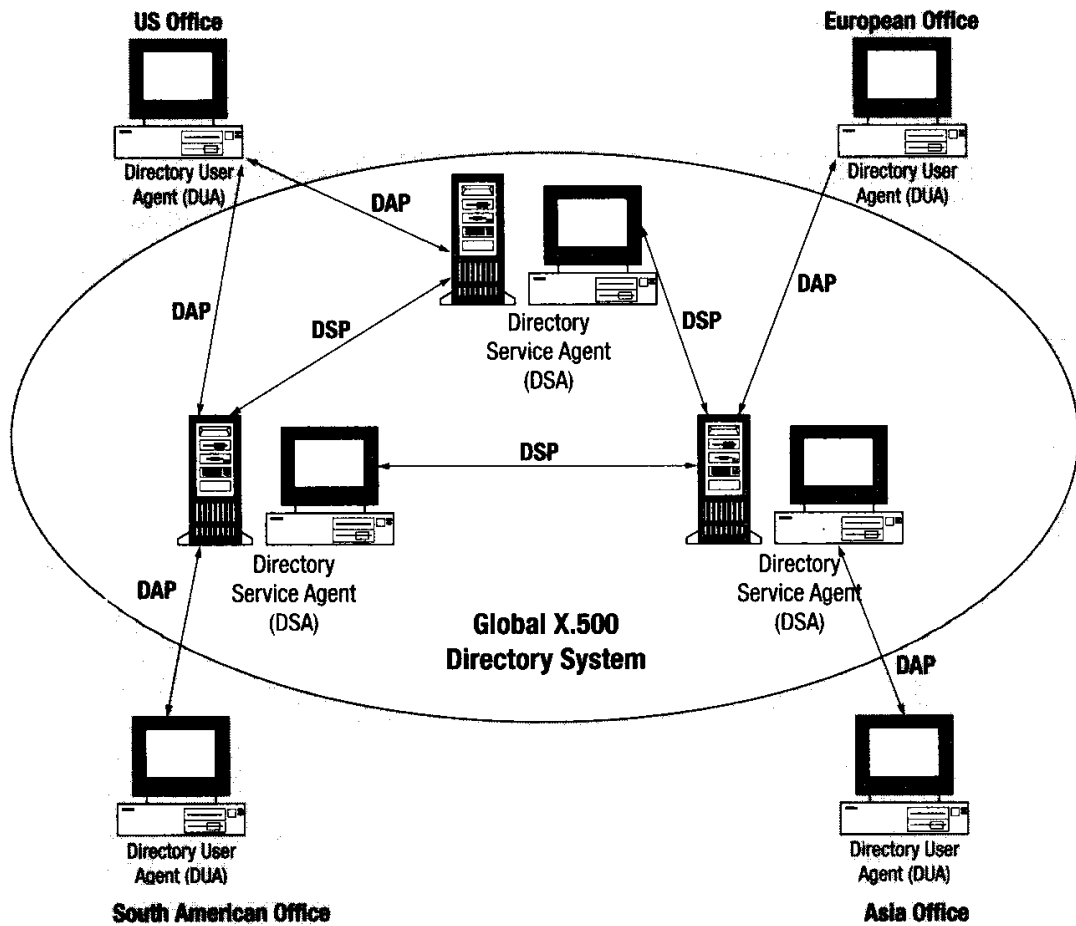
## **X.500 Begriffe**

**Directory User Agenten (DUA) sind X.500 Klienten.**

**Directory System Agenten (DSA) sind Server, welche gemeinsam die Directory Information Base (DIB) halten.**

**Die Directory Information Base (DIB) ist die Summe aller gespeicherten Einträge (ein Eintrag pro Objekt). Die DIB ist in der Form eines Baumes, des Directory Information Tree, implementiert. Die Absicht ist, daß weltweit eine einzige DIB existiert, wobei Teile der DIB auf einer Vielzahl unabhängiger Server gespeichert sind.**

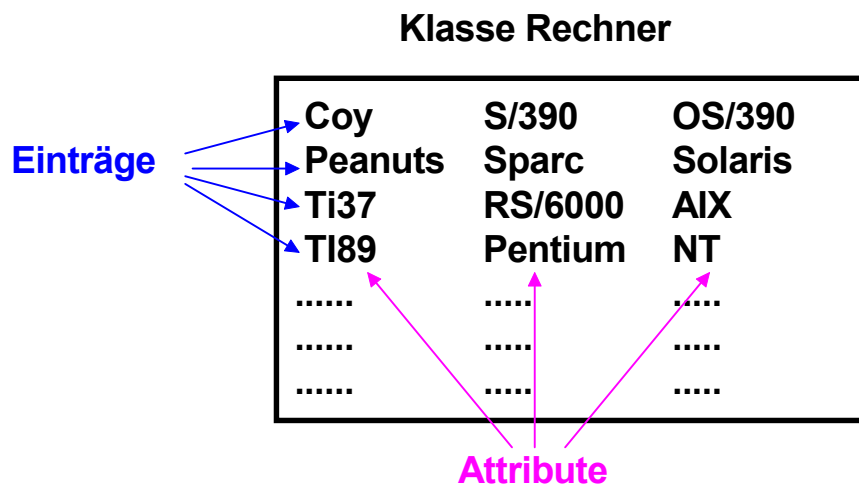
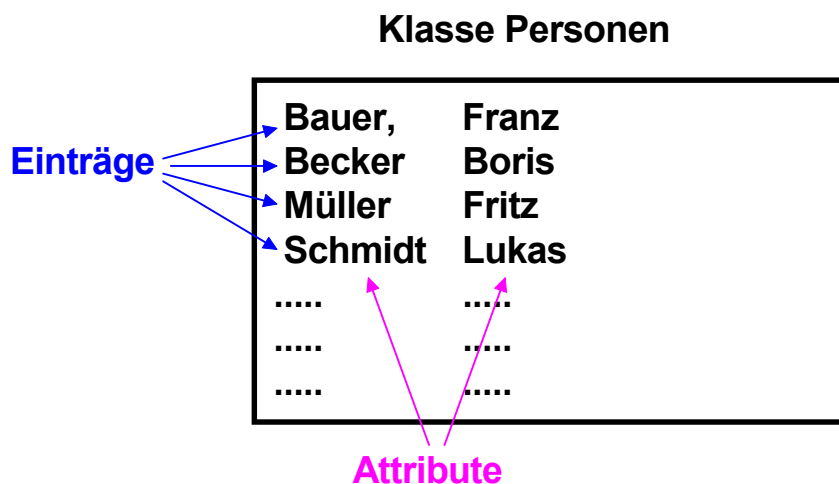
**Anwendungen können auf die DIB mit Hilfe einer von mehreren API's zugreifen. Es existieren die X/open Directory Service Interface (XDS), das X.500 Directory Access Protokoll (DAP) und das Internet Lightweight Directory Access Protocol (LDAP). Die Benutzung braucht extensive Vorkenntnisse.**



## Beispiel für ein globales X.500 Verzeichnissystem

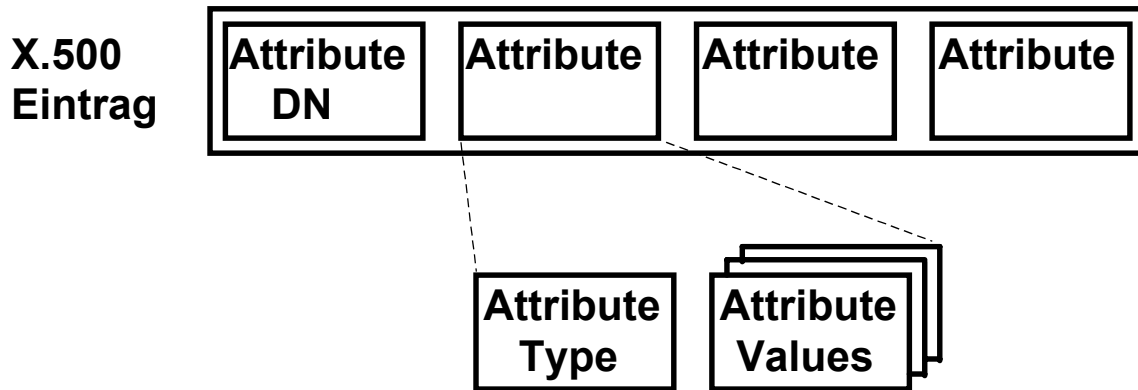
Die Directory Information Base enthält Einträge zu Objekten; je 1 Eintrag pro Objekt.

Eine Objektklasse definiert einen Satz von erforderlichen und von optionalen Attributen eines Eintrags. Anwendungen können sich auf den minimalen Inhalt eines Eintrags verlassen.



Besteht ein DIB Eintrag aus mehreren Attributen, dann ist eines der Attribute ein „Distinguished Attribute“. Dieses bestimmt den Namen des Eintrags.

# Struktur eines X.500 DIB Eintrages



Ein Eintrag besteht aus mehreren Attributen; eines der Attribute ist der DN (Distinguished Name) des Objektes. „C=DE.O=Uni-Leipzig.OU=Informatik.N=Spruth“ ist ein Beispiel für einen DN.

Jedes Attribute besteht aus einem Attribute Type und einem oder mehreren Attribute Werten (Values). Attribute mit mehreren Werten werden als „Multivalued Attributes“ bezeichnet. Einer der Attribute Values ist der „Distinguished Value“.

Der Attribute Typ beschreibt die Charakteristik des Attributes. Zu jedem Attribute Type gehören:

- Type Name

- Type Definition, einschließlich Type Description

- Syntax Definition (in ASN.1)

Attribute werden von manchen Herstellern als „Properties“ bezeichnet.

# Beispiel eines X.500 DIB Eintrags

## *info*

**Prof. Dr.-Ing. Wilhelm G. Spruth  
Institut fuer Informatik, Universitaet Leipzig**

## *commonName*

**Wilhelm.G.Spruth  
Wilhelm.Spruth  
Wilhelm Spruth  
W. G. Spruth**

## *surname*

**Spruth**

## *telephoneNumber*

**0341 97 32211  
0172-8051-485**

## *uid*

**spruth**

## *mail*

**spruth@informatik.uni-leipzig.de  
spruth@informatik.uni-tuebingen**

## *roomNumber*

**0239**

## *userClass*

**Professor**

# Probleme mit X.500

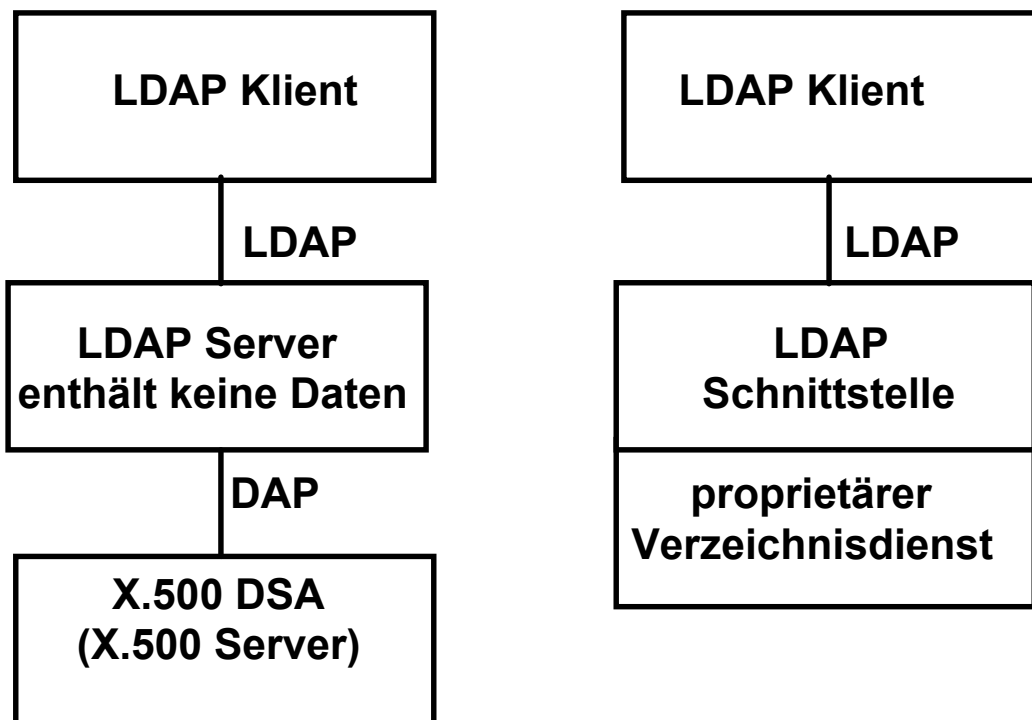
**Komplex zu benutzen**

**Viele nur selten verwendete Funktionen**

**Verwendet den OSI Protokoll Stack  
Umsetzung auf TCP/IP nach RFC 1066**

**Light Weight Directory Access Protocol (LDAP)  
RFC 1777, Yeong, Howes, Kille, 1995**

**einfaches Gateway für den Zugriff auf einen  
X.500 Server**



# **LDAP Bestandteile (Modelle)**

## **Informationsdarstellung**

**Namensgebung**

**sehr X.500 ähnlich**

**umgekehrte Reihenfolge, Komma als Delimiter**

**cn=Spruth, ou=Informatik, o=Uni-Leipzig, c=DE**

## **Organisation**

**X.500 ähnliche Baumstruktur**

## **Funktionalität**

**Abfragen**

**Aktualisierung**

**Authentifikation**

## **Sicherheit**

**Bind Operation (Client authentifiziert sich gegenüber dem Server)**

**Verbesserung mit LDAP Version 3**

# LDAP Operationen

**Bind**

**Search**

**Compare**

**Add**

**Delete**

**ModifyRDN**

**Alle LDAP Nachrichten und Antworten sind in ASN.1  
kodiert**

# Microsoft Active Directory Services

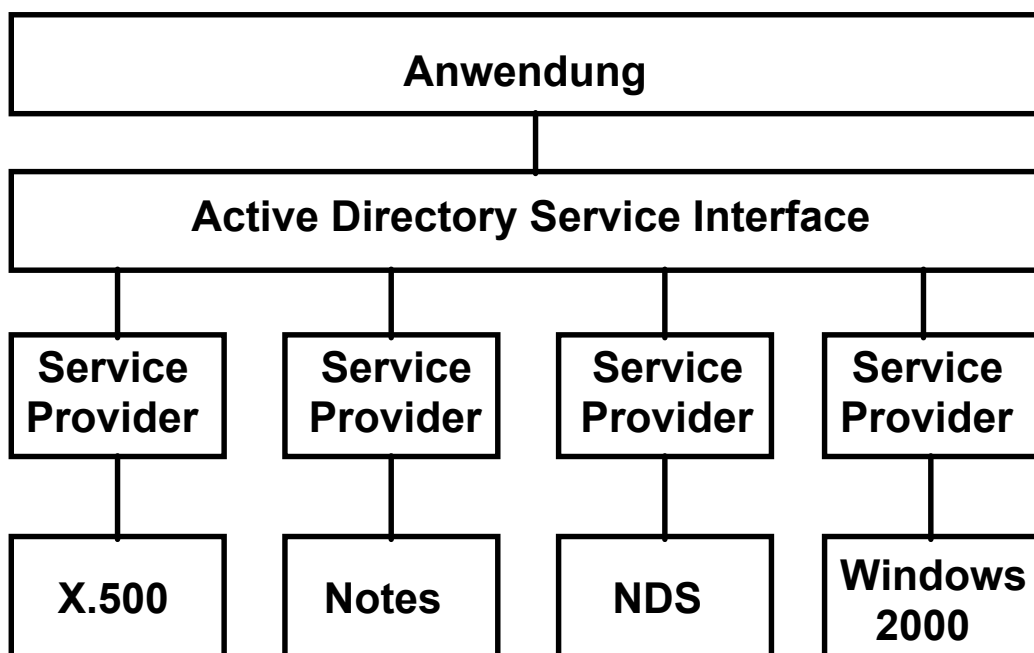
**Bestandteil von Windows 2000**

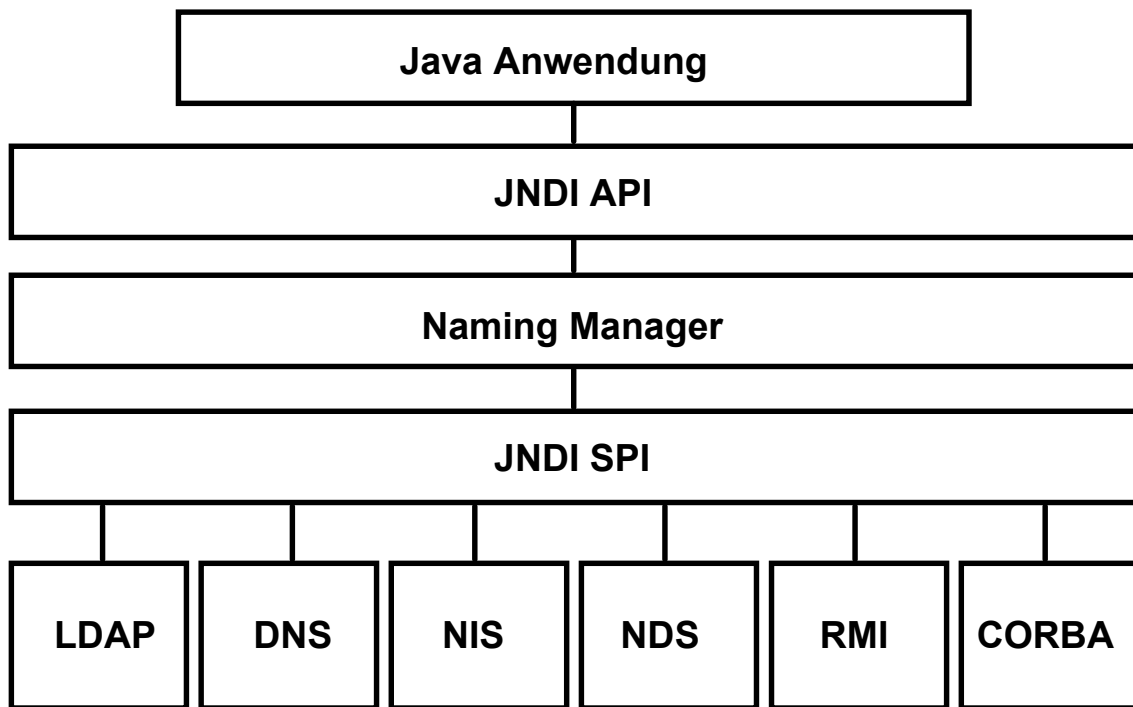
**Kombination von DNS und LDAP**

**Einheitliche ADSI Schnittstelle für Zugriffe auf Directory Services unterschiedlicher Hersteller**

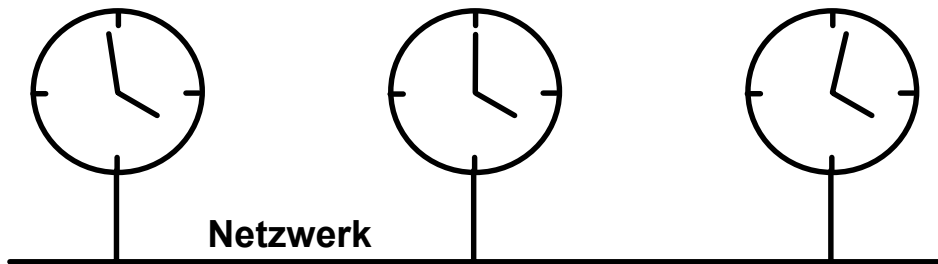
**Kombination mit DHCP**

**Kerberos Unterstützung**



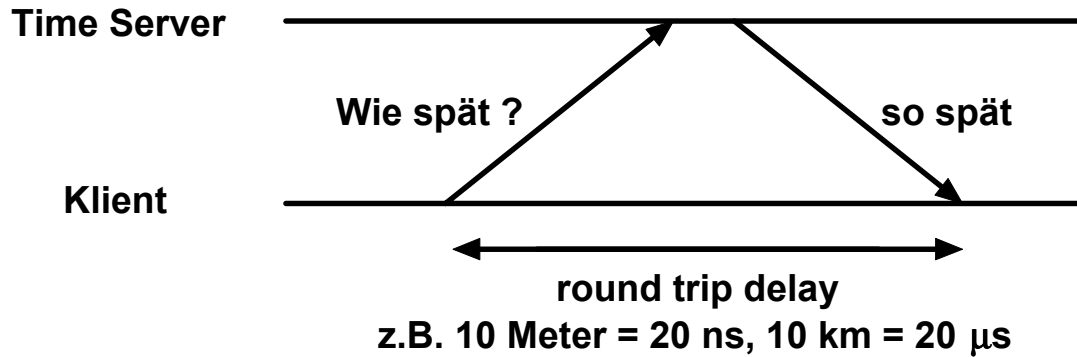


**Die JNDI Service Provider Interface ermöglicht die Integration von Naming und Directory Services für Java anwendungen (z.B. Servlets).**



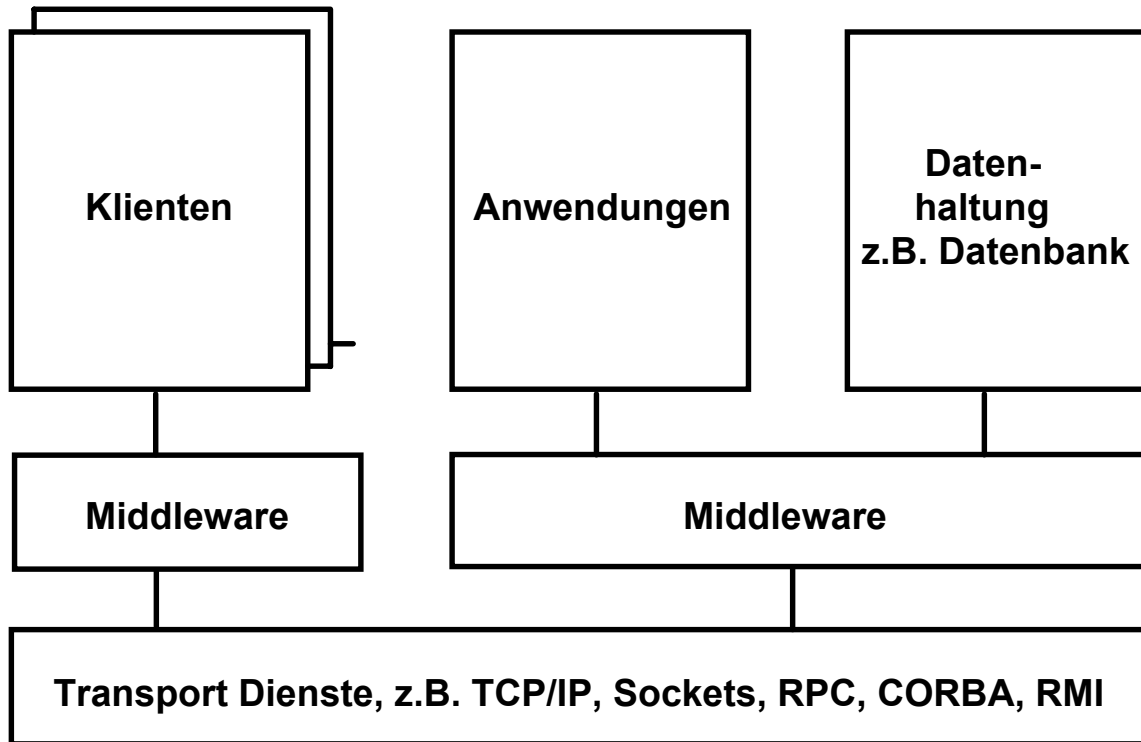
Ungenauigkeit des Workstation Kristalls  $\approx$  1 Sekunde/Tag

## Time Service



Lichtgeschwindigkeit    im Vakuum    = 300 000 km/s  
                                   auf Leitungen    = 200 000 km/s

Taktzeit einer 4GHz CPU = 250 ps



## Middleware

Middleware kombiniert Schicht 5 Funktionen wie Sockets und RPC mit Funktionen der Schicht 6 (Datenrepräsentation, Sicherheit, Verzeichnisdienste usw.).

Beispiele sind:

- DCE
- Transaktionsmonitore  
BEA Tuxedo, IBM CICS, IMS, MS Transaction Server,
- Web Application Server  
BEA WebLogic, IBM WebSphere, MS DotNet, SAP Netweaver
- Groupware  
IBM Lotus Notes, MS Exchange, Novell Groupwise
- Systems Management  
CA Unicenter, HP Openview, IBM Tivoli,

Die entsprechenden Middleware Funktionen müssen auf allen partizipierenden Klienten- und Server Rechnern installiert sein.

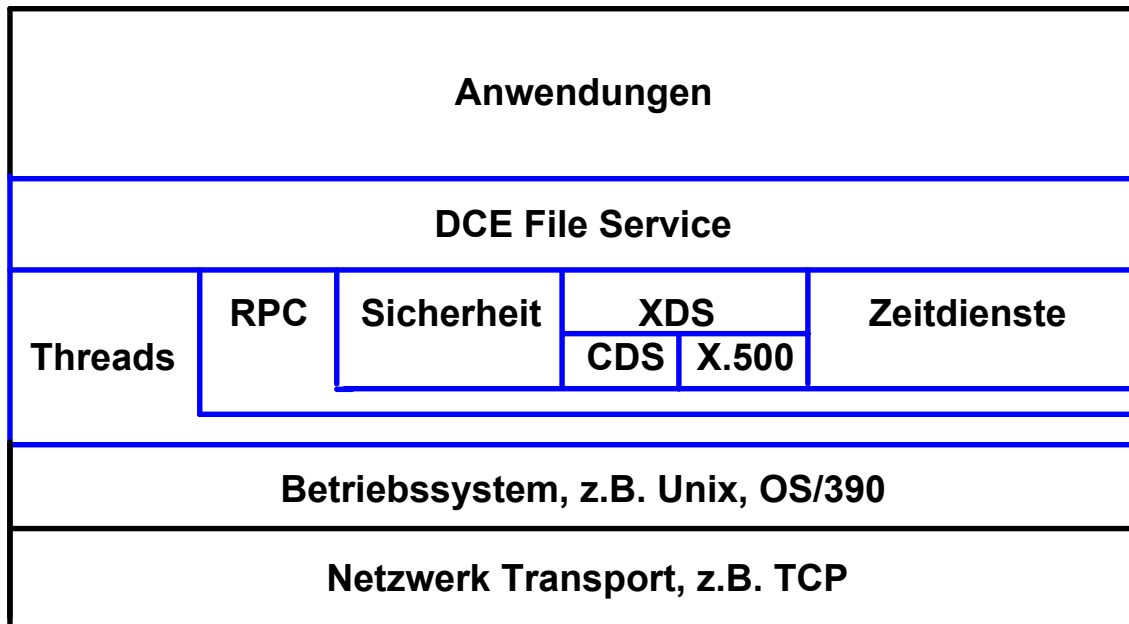
# DCE

## Distributed Computing Environment

Integrierter Satz an Werkzeugen, Einrichtungen und Diensten

Ermöglicht die Kooperation zwischen unterschiedlichen Rechnern

Schirmt den Benutzer von den technischen Details der Schichten 1 - 4 ab



Anwendungs-Programmierschnittstellen

# **DCE Remote Procedure Call**

**Kann konfiguriert werden, die Host Computer Sicherheits-einrichtungen zu benutzen**

**Unterstützt verbindungslose und verbindungsorientierte Protokolle**

**20 unterschiedliche Nachrichten Typen definiert, davon einige nur für den verbindungslosen oder den verbindungsorientierten Transport. Die Nachrichten werden als Protocol Data Units (PDU's) bezeichnet**

**Der Microsoft RPC ist eine auf DCOM basierende objektorientierte Erweiterung des DCE RPC (auch als „Objekt RPC“ bezeichnet. Die Syntax der übertragenen Pakete ist nahezu vollständig mit der ursprünglichen DCE Spezifikation kompatibel. Wird als Bestandteil von NT ausgeliefert.**

# DCE Unterstützung

**DEC OSF/1, VMS**

**HP-UX**

**IBM AIX, z/OS, OS/400**

**Linux**

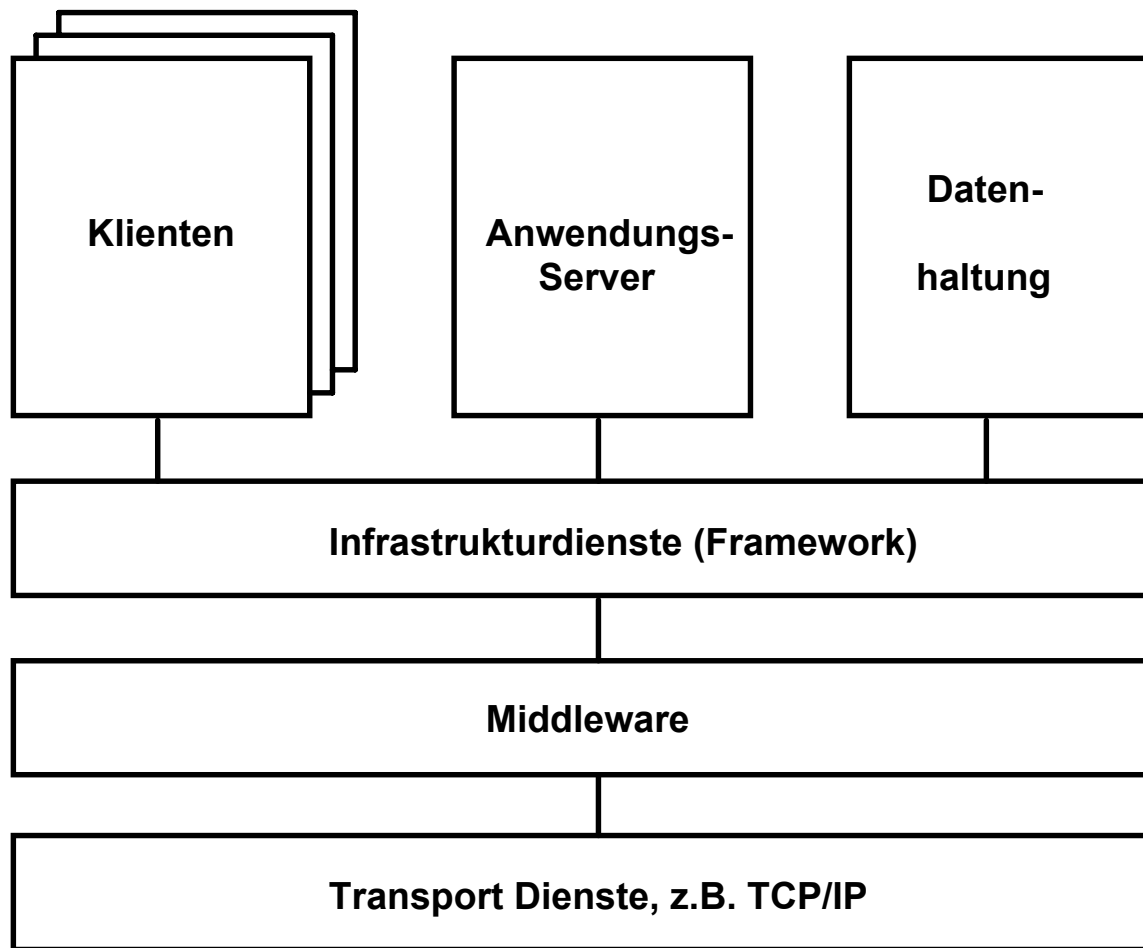
[www.entegrity.com](http://www.entegrity.com)

**Microsoft Windows XP**

**Siemens Sinix**

**Sun-Solaris**

**Tandem Guardian**



## Framework

- Policy Regimes (wer was macht, wann, wo und wie)
- Profile Manager
- Tasks
- Scheduling Funktion
- Sicherheits-Einrichtungen
- Rollback/Recovery
- Administration

# Framework

**Ein Framework ist ein Satz von Middlewarekomponenten, welche von ISVs (Independent Software Vendors) als Basis für eigenentwickelte Softwarekomponenten benutzt werden kann.**

**Zu den Komponenten eines Frameworks gehören**

**Repositories  
Schnittstellen  
Protokolle  
Topologiedienste  
Agent Technologien**