

Verteilte Systeme

16. Namensdienste

Namen sind mehr als Schall und Rauch

Eindeutige Bezeichnung von Dingen
(Bezeichner)

Anwendungsgebiete

- Namen in einem Programm
 Variablen, Prozeduren, Konstanten, ...
- Rechner
- Dienste
- Dateien

Funktionen eines Namens

- Erklären
 printer3, sturm@uni-trier.de, /de/uni-trier/fb4/cs/sturm/printer
- Identifizieren
- Lokalisieren
 Auffinden und Zugriff auf Ding über den Namen



3 Funktionen \Rightarrow 3 Namenstypen

Symbolische Namen

- Erklärungsfunktion (für Menschen)
- Lesbare Zeichenkette, Umfangreich
- Aufwendiges Parsen durch den Rechner
- Beispiel: /de/uni-trier/fb4/cs/sturm/printer

Identifikator

- Identifikationsfunktion
- Für den Menschen meist "unverständlich"
- Meist konstante Größe
- Beispiel: 136.199.54.210, Port 80

Adresse

- Lokalisierungsfunktion
- Direktes Ansprechen über Adresse möglich
- Beispiel: 08.00.20.88.94.1e

Pure und Impure

Pure Names

- Enthalten keine Ortsinformation
- Nur auf Gleichheit testbar
- Beispiele?
- Vor- und Nachteile?

Impure Name

- Enthält Ortsinformation
- Orts- oder Adreßanteil wird meist bei der Lokalisierung verwendet
- Gültigkeit der Ortsinformation?
- Beispiele?
- Vor- und Nachteile?

Vor- und Nachteile

Pure Names

Vorteile

- Häufig Erklärungsfunktion
- Unabhängig vom Objektort
Erleichtert Migration
- Zusätzliche Indirektionsstufe
Delegation möglich

Nachteile

- Direkte Lokalisierung des Objekts nicht möglich
- Zusätzliche Indirektionsstufe
Mehrkosten

Impure Names

Vorteile

- Direkte Lokalisierung des Objekts bedingt möglich
- Schnelle Lokalisierung

Nachteile

- Adreßinformation erschwert
Objektmigration
- Delegation problematisch
- Häufig wenig verständlich

Verteilte Systeme, Sommersemester 1999

Folie 16.5

Pure Adresse !?

Beispiele

- IP-Adresse
- Ethernet-Adresse

Lokalisierungsverfahren

- Einstufig
Broadcast
Gesuchte Station meldet sich
- Mehrstufig
Routing
 - Gesuchtes Objekt bekannt
 - Weitergabe an übergeordnete Instanz

Caching der "Weginformation"

Bindung Name - Adresse



Verteilte Systeme, Sommersemester 1999

Folie 16.6

Unterschied zwischen Name und Adresse

Im Prinzip keiner

Abhängig vom gewählten Abstraktionsgrad

Name

- Direktes Ansprechen des gewünschten Objekts nicht möglich
- Bindung an eine Adresse
- 2 sichtbare Schritte
 1. Ermittlung der Adresse über einen Namensdienst
 2. Kontaktieren des gewünschten Objekts

Adresse

- Direktes Ansprechen des gewünschten Objekts tatsächlich (z.B. Speicheradresse) oder scheinbar (z.B. IP-Adresse) möglich
- Unterste sichtbare Stufe
 - Kontaktieren des gewünschten Objekts

Zeitpunkt der Namensbindung

Statische Bindung

- Beim Übersetzen, Binden (vor dem Ausführen)
- Beispiele
 - Variablen- oder Prozedurname wird an Adresse gebunden
 - Feste Portnummern für NFS, Portmapper, ...

Halbdynamisches Binden

- Zum Startzeitpunkt
- Beispiele
 - Dynamische Bibliotheken (DLLs, Shard Libs, ...)
 - Eingeben der Serveradresse beim Client

Dynamisches Binden

- Spätestens unmittelbar vor dem Zugriff auf das Objekt
- Typisch für verteilte Systeme
- Namensdienst

Namensräume

Eindeutigkeit relativ zu einem Namensraum

Trivillösung

- Flacher Namensraum (1)
- Beispiele?

Status Quo

- Namensräume haben ebenfalls Namen
- Hierarchischer Namensraum
- Impliziert meist auch Architektur eines verteilten Namensdiensts

Adventure **Action**

PC-Spiele

Länder

Deutschland
Trier
Frankfurt
München
Hintertupfingen
Koblenz
Hamburg

USA
New York
Frankfurt
Boston
Los Angeles

England **Spanien**

Verteilte Systeme, Sommersemester 1999 Folie 16.9

Architektur Aspekte

Zentraler Namensdienst

- Schlechte Skalierbarkeit
- Leistungengpaß
- Ausfallsicherheit

Skalierbarkeit

- Verteilter Namensdienst
- Hierarchische Struktur

Leistungengpaß

- Datenreplikation

Ausfallsicherheit

- Datenreplikation

Datenkonsistenz

- "Fast" Read-Only-Daten
- Inkonsistenz unkritisch

Nochmal Auskunft bei "falsch verbunden"

Verteilte Systeme, Sommersemester 1999 Folie 16.10

Funktionen eines Namensdiensts

Server verwaltet ein oder mehrere Namensräume

Clientseitiges API

- **Insert (Name)**
- **Delete (Name)**
Nicht immer direkt zugreifbar
- **Bind (Name1 ,Name2)**
- **Unbind (Name1 ,Name2)**
- **Query (Name1) returns Name2**

Serverseitiges API

- **Resolve (Name)**
Anfrage bei einem übergeordneten Server

Schutz- und Sicherheitsaspekte

- Wer darf welche Informationen abfragen?
- Wer darf welche Informationen hinzufügen/ändern?

Erweiterter Funktionsumfang

Directory Services

Speicherung von "beliebigen" Attributen zu einem Namen

Beispiele

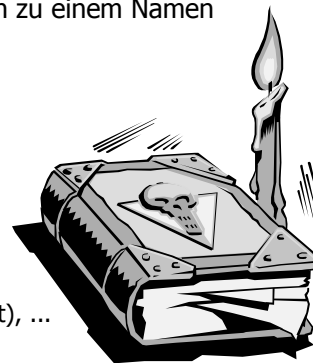
- Email-Directory
Name, Adresse, Fähigkeiten, ...
- Personaleinträge
Name, Adresse, Email, Gehalt, ...
- ...

Wachsende Verbreitung

- X.500, LDAP, Active Directory (Microsoft), ...

Erweiterte Queries

- Pattern Matching
- Filter



Replikation

```

graph TD
    Konzern --> Niederlassung
    Niederlassung --> Abteilung
    Abteilung --> Mitarbeiter
    
```

Leistungssteigerung und Ausfallsicherheit

Änderungen bei Namensräumen seltener als bei Namen

- Namensräume sind Ordnungsprinzip

Lesende Anfragen häufiger als Änderungen

Temporäre Inkonsistenzen tolerierbar

- Effiziente Replikat-"Konsistenz"
- Caching akzeptabel

Obere Ebenen besonders gut replizierbar

- Hohe Anfragehäufigkeit
- Geringe Änderungshäufigkeit

Verteilte Systeme, Sommersemester 1999 Folie 16.13

Replikatkonsistenz

Änderungszeitpunkt

- Periodische Aktualisierung
Inkonsistenzfenster tolerieren
Wie würde man vorgehen?
- Jede Änderung sofort "pushen"
Kleines Inkonsistenzfenster
Hohe Server- und Netzbelastung
Abhängigkeit von der Ebene der geänderten Daten?

Ausgangspunkt

- Primärserver
Alle Änderungen auf einem zentralen Server
Serverausfall, Leistungsengpaß
Beispiel: NIS
- Symmetrische Lösung
Verteilter Abgleich

Verteilte Systeme, Sommersemester 1999 Folie 16.14

16.1 Symbolischer Namensdienst Beispiel: LDAP

Übersicht LDAP

Basiert auf X.500

- Bekannter ISO-Standard eines Directory Service
- Volles ISO-OSI-Protokoll (Aufwand)
- ASN.1-Notation bei verwendeten Datenstrukturen
- Zugangsprotokoll: Directory Access Protocol (DAP)

Lightweight DAP = LDAP (Aktuelle Version 2)

- Direkte TCP-Adaption
- Stringrepräsentation (Heterogene Systeme)
- Protokoll für Anfrage an LDAP-Server
- LDAP \subset DAP: Anfrage an X.500-Server ebenfalls realisierbar

Standards

- RFC 1777: LDAP-Standard
- RFC 1778: Stringrepräsentation für Attribute
- RFC 1779: Stringrepräsentation für Namen
- RFC 1558: Syntax für Suchfilter

Einträge im Directory Service

Einträge setzen sich aus Attributen zusammen

Jedes Attribut besteht aus

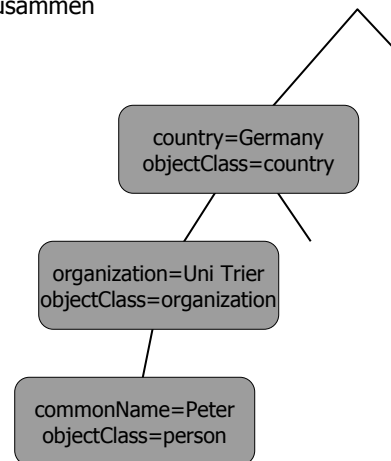
- Typ
- Ein oder mehreren Werten

Attributwerte sind Strings

Jeder Eintrag besitzt spezielles

Attribut **objectClass**

- Legt Attributtypen fest (Namen der Attribute)
- Welche Attribute benötigen zwingend Werte?
- Welche Attribute sind optional?



Verteilte Systeme, Sommersemester 1999

Folie 16.17

Distinguished Names

Relative Distinguished Name (RDN)

- Relativ zum gemeinsamen Parent eindeutiger Name
- Besteht aus den Werten ein oder mehrerer Attribute
- Höchstens ein Wert pro Attribut

Distinguished Name (DN)

- Eindeutige und absolute Objektbezeichnung
- Alle RDNs von der Wurzel zum Objekt

X.500-Namen basieren auf ASN.1

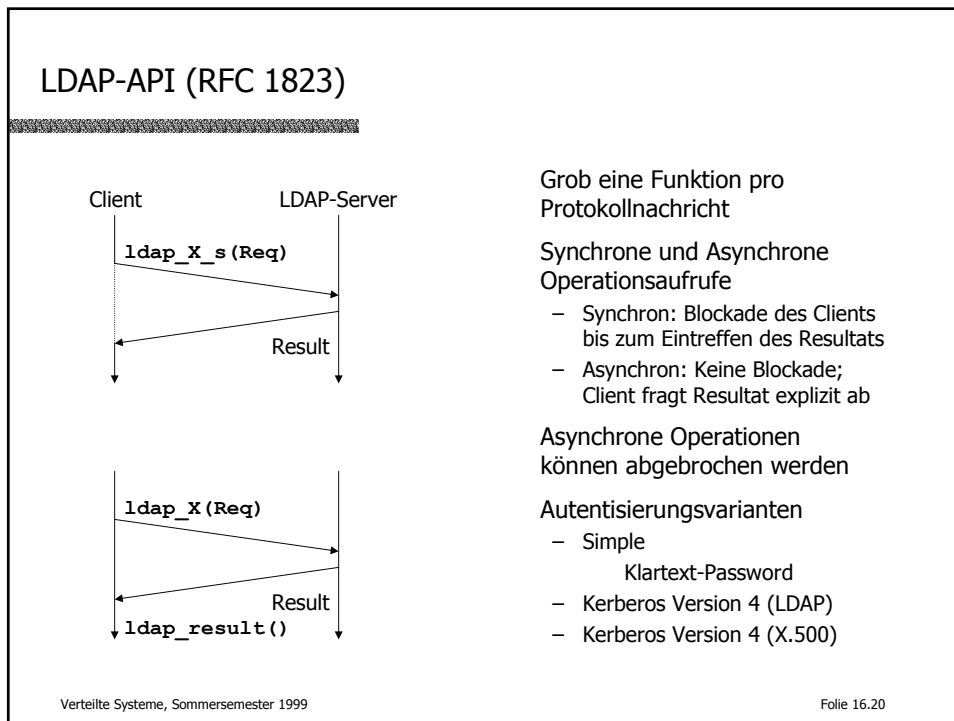
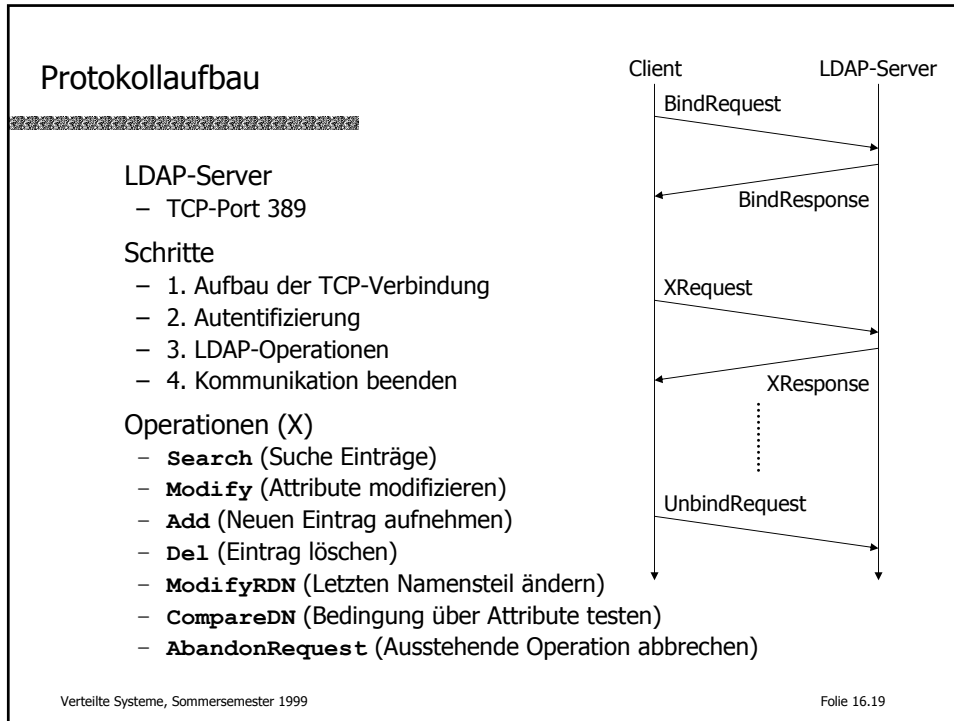
LDAP-Namen

- Stringrepräsentation
- Beispiel

"co=Germany, O=Uni Trier, cn=Peter"

Verteilte Systeme, Sommersemester 1999

Folie 16.18



Anfrage

```
int ldap_search_s (
    LDAP *ld,
    char *base,
    int scope,
    char *filter,
    char *attrs[],
    bool attrsonly,
    LDAPMessage *result
)
```

Verbindung zum LDAP-Server

- Handle `ld`

Scope (LDAP_SCOPE_...)

- **BASE**
- **ONELEVEL**
- **SUBTREE**

Suchfilter `filter`

Welche Attribute und eventuell Attributwerte sollen zurückgegeben werden (`attrs`)

Nur Attribute oder auch Werte (`attrsonly`)?

Variante mit Timeout

- `ldap_search_st()`

Angabe von Filtern

```
<filter> ::= '(' <filtercomp> ')'  
<filtercomp> ::= <and> | <or> | <not> | <item>  
<and> ::= '&' <filterlist>  
<or> ::= '|' <filterlist>  
<not> ::= '!' <filter>  
<filterlist> ::= <filter> | <filter> <filterlist>  
<item> ::= <simple> | <present> | <substring>  
<simple> ::= <attr> <filtertype> <value>  
<filtertype> ::= <equal> | <approx> | <greater> | <less>  
<equal> ::= '='  
<approx> ::= '~='  
<greater> ::= '>='  
<less> ::= '<='  
<present> ::= <attr> '='  
<substring> ::= <attr> '=' <initial> <any> <final>  
<initial> ::= NULL | <value>  
<any> ::= '*' <starval>  
<starval> ::= NULL | <value> '*' <starval>  
<final> ::= NULL | <value>
```

16.2 Lokalisierungsdienst Beispiel: DNS

Übersicht DNS

Domain Name Service

Hierarchischer Namensraum: balvenie.uni-trier.de

Transformation

- Domain Name → IP-Adresse (Lookup)
- IP-Adresse → Domain Name (Reverse Lookup)

UNIX-API

- `gethostbyname ()`
- `gethostbyaddr ()`

IETF-Standard

- RFC 1034 (Konzepte)
- RFC 1035 (Implementierungsdetails)

Gängige Implementierung

- Berkeley Internet Name Domain Server (BIND)

Organisation der DNS-Server

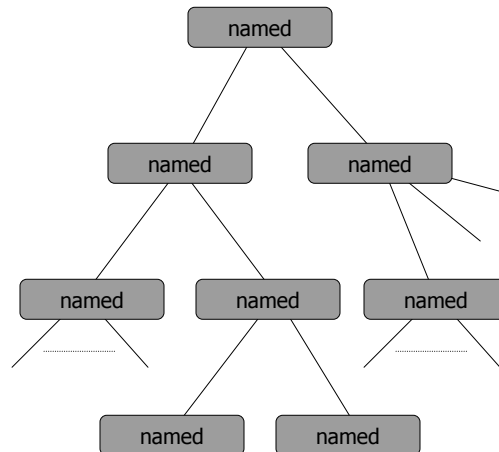
Hierarchische Struktur

Ausfallsicherheit

- Primary Name Server
- Secondaries

Administrative Einheiten

- Zone



Verteilte Systeme, Sommersemester 1999

Folie 16.25

Einträge

Ca. 20 verschiedene Eintragsformen

- Viele nicht mehr gültig

Gängige „Record“-Einträge

- A: IP-Adresse
- NS: NS-Server
- CNAME: Kanonischer Name
- PTR: Pointer-Record
- HINFO: Host-Information
- MX: Mail Exchange Record

Pointer-Query

- Gegeben IP-Adresse; Gesucht: Kanonischer Name
- `in-addr.arpa` Domain

Verteilte Systeme, Sommersemester 1999

Folie 16.26