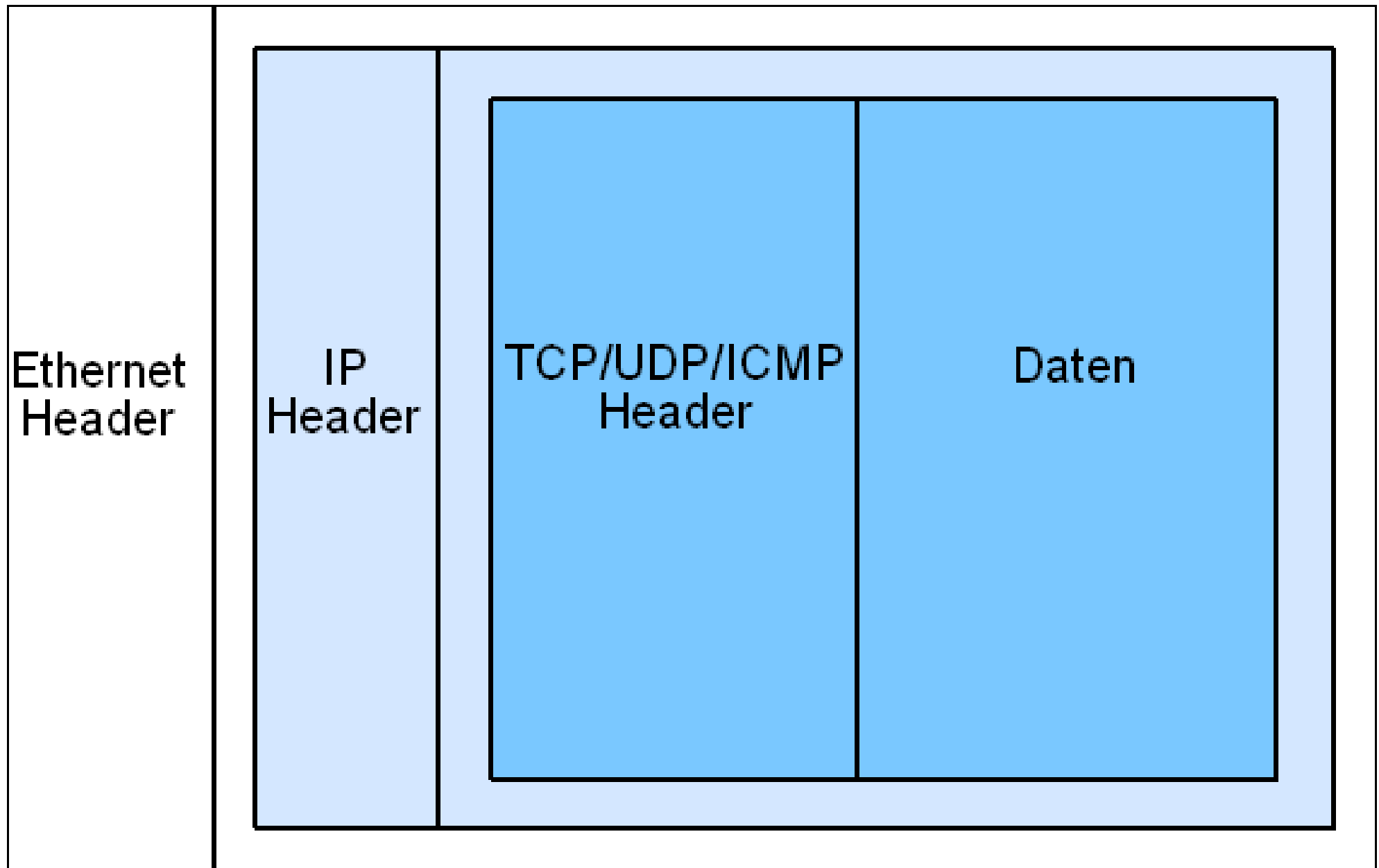


# TCP, UDP, ICMP

Die dritte Netzwerk-Schicht (Transport-Ebene) & ICMP

[lukas.feiler@lukasfeiler.com](mailto:lukas.feiler@lukasfeiler.com)

<http://teaching.lukasfeiler.com>



**Die Protokollsichten (TCP, UDP, ICMP setzen auf IP u. Ethernet auf)**

# TCP

Transmission Control Protocol

Anwendungsbereich: HTTP, FTP, SMTP, POP3, ...

- verbindungsorientiert
  - expliziter Verbindungsaufbau -und Abbau
  - kein Datentransfer ohne Verbindung
- zuverlässig
  - stellt sicher, dass alle Daten tatsächlich ankommen
  - checksum für Header & Daten (beschädigte Pakete werden erneut gesendet); IP kennt nur checksum f. Header
  - Pakete in falscher Reihenfolge werden sortiert
  - doppelte Pakete werden verworfen
- ununterbrochener Datenstrom
  - höhere Protokollschichten erhalten ununterbrochenen Datenstrom
  - Operationen zur Fehlerkorrektur bleiben höheren Protokollschichten verborgen

# TCP: Ports

dienen dazu den richtigen Dienst auf dem Server zu finden

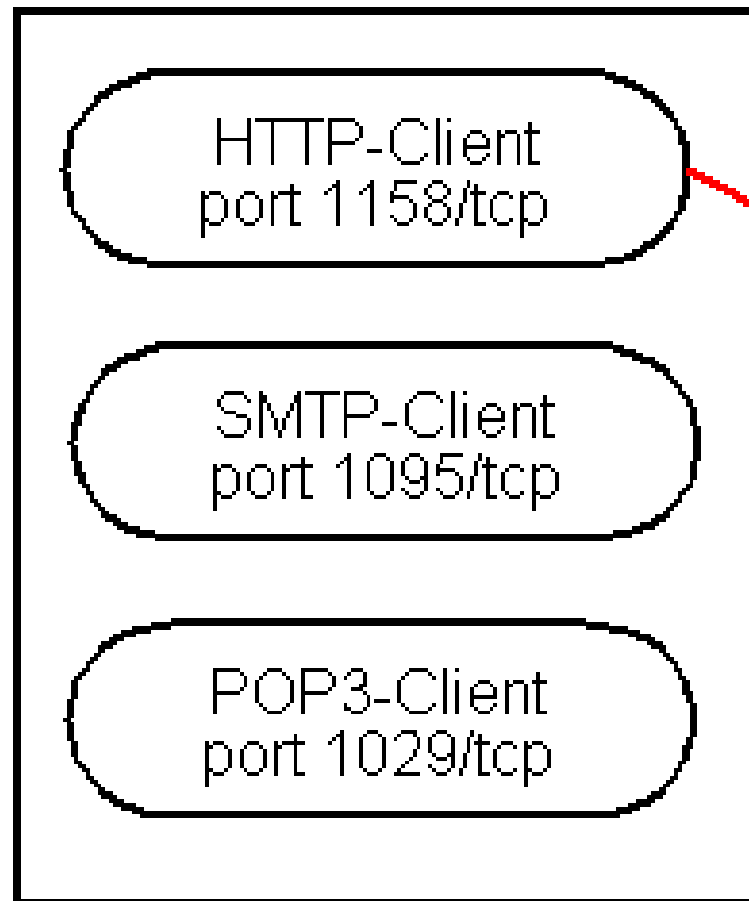
von 1 bis 65 535 ( $2^{16}$ )

von 1 bis 1023 sind "privilegierte ports"

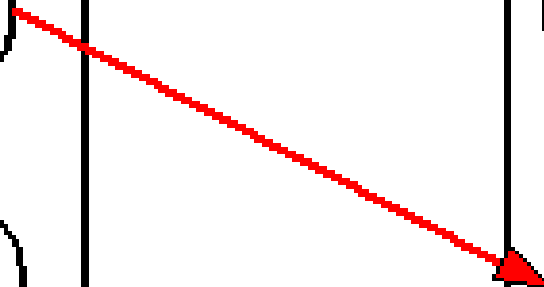
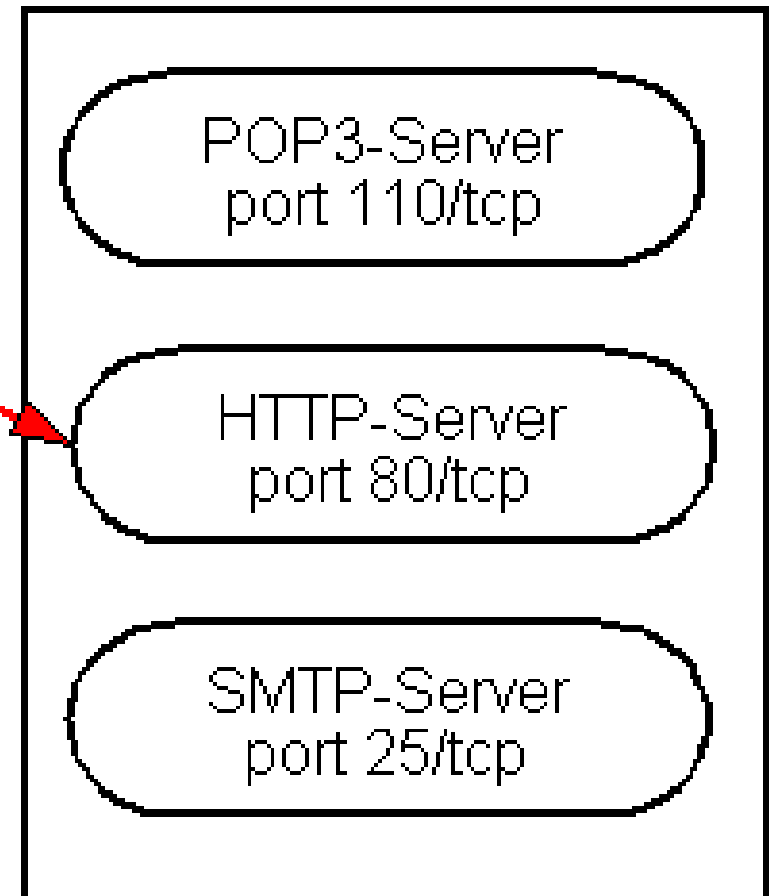
„Socket“: Kombination aus IP & Port

Schreibweise: `www.example.com:80`

# Client



# Server



Eine Connection zu einem HTTP-Server

- TCPView

<http://www.sysinternals.com/Utilities/TcpView.html>

GUI & nur für Windows

- netstat

CLI; für Windows & Unix (auch Mac OS X)

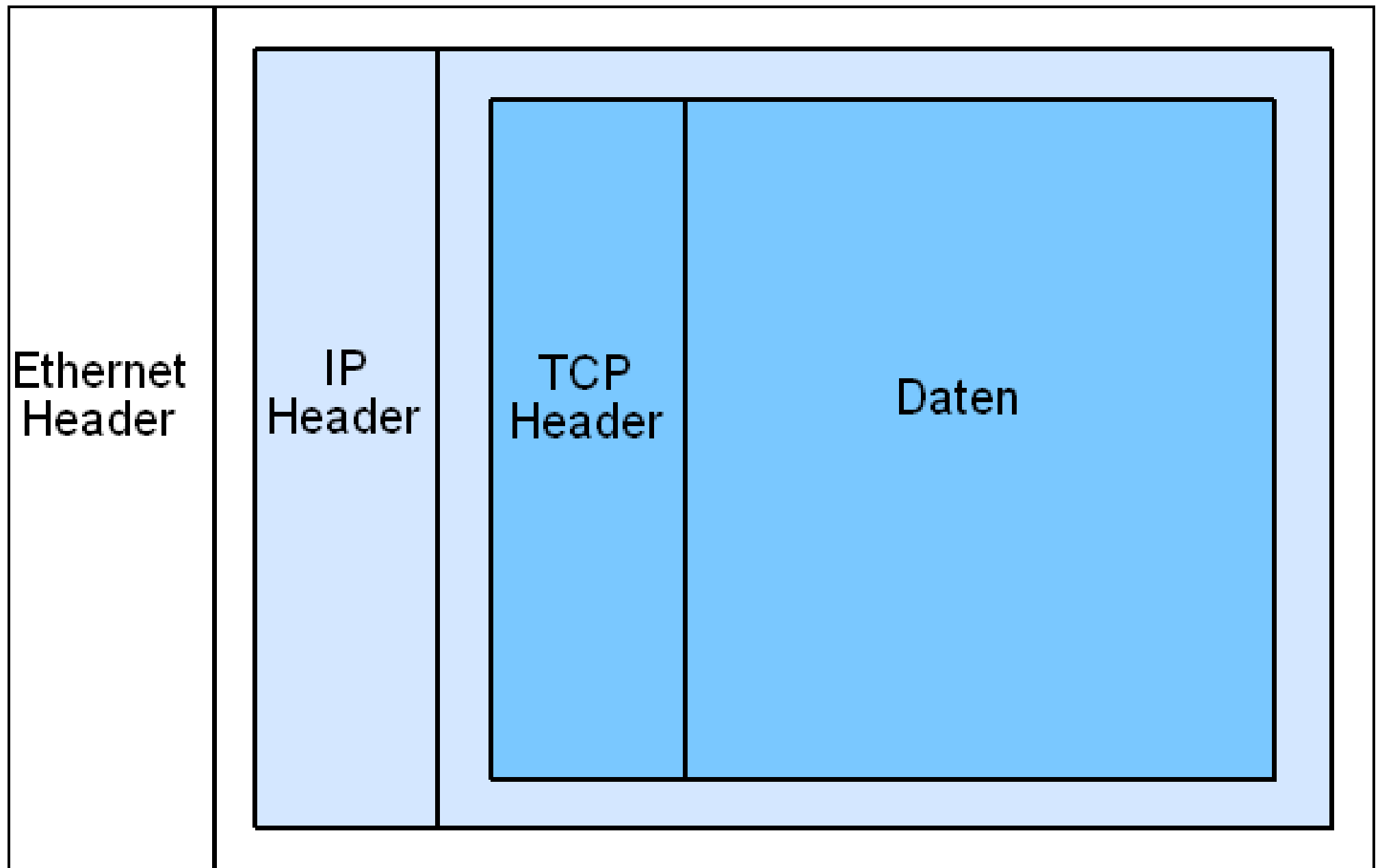
*netstat -n*

**Anzeigen offener Verbindungen und verwendeter Ports**



# TCP-Pakete

Auch auf TCP-Ebene werden Daten nur als Pakete verschickt.



**Ein TCP-Paket (eingepackt in ein IP-Paket)**

source port number:  
TCP-Quellport

destination port number:  
TCP-Zielpport

TCP flags:

SYN: Synchronization; Verbindungsaufbau

ACK: Acknowledgement; bei jedem außer dem ersten

FIN: leitet Verbindungsende ein

RST: Reset; Verbindungsabbruch

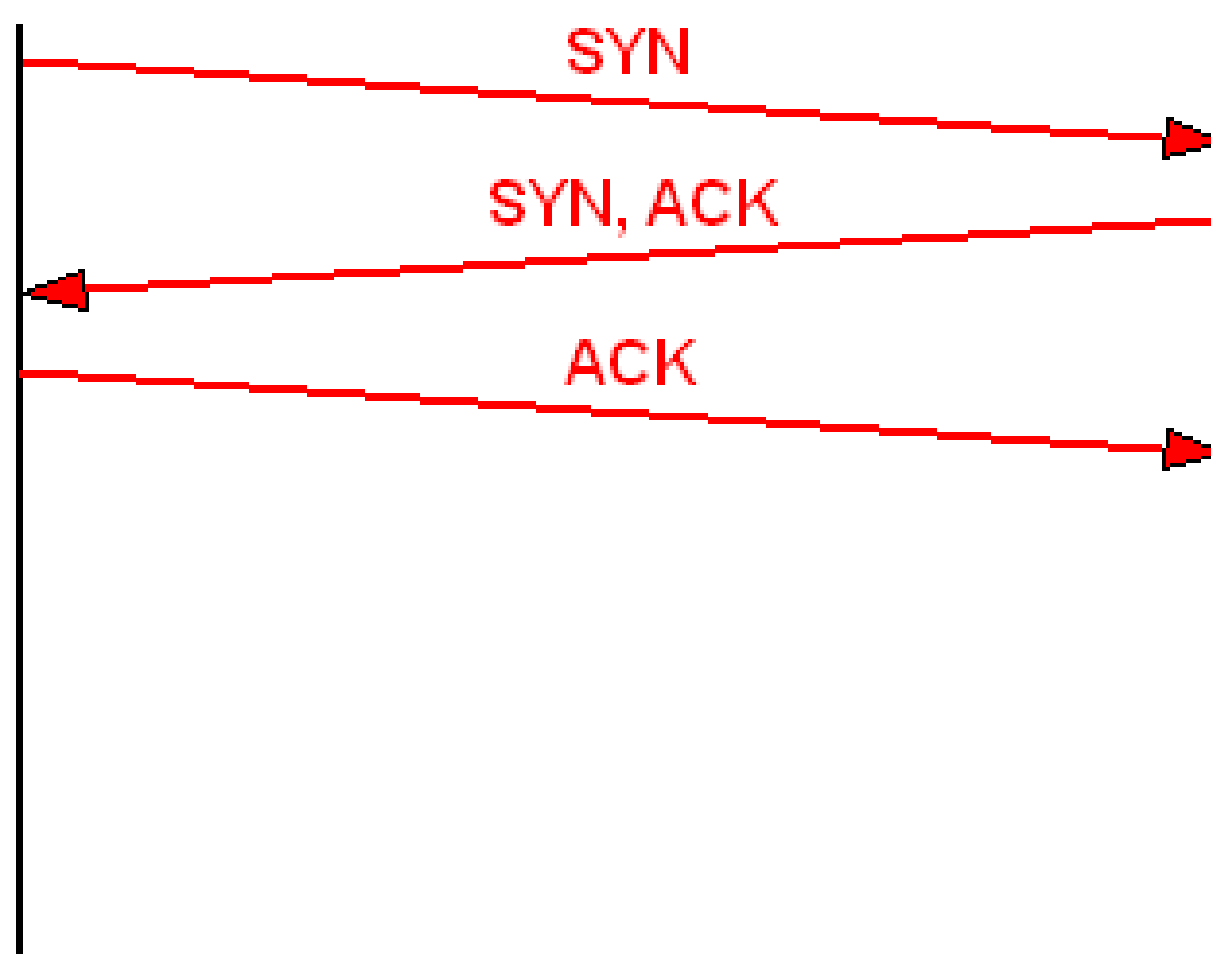
TCP checksum: Checksumme über Header und -Daten  
Daten

z.B. HTTP-Header & Daten

**Ein TCP-Paket kennt (unter anderem) folgende Felder**

# Client

# Server



Aufbau einer TCP-Verbindung mittels „Three Way Handshake“

# UDP

User Datagram Protocol

Anwendungsbereich: Streaming, zeitkritische  
Anwendungen, bei sehr kleinen Datenmengen

- Datagramm-orientiert (nicht verbindungsorientiert)  
kein Verbindungsaufbau bzw. -Abbau
- unzuverlässig  
stellt NICHT sicher, dass alle Daten tatsächlich ankommen  
beschädigte Pakete werden nicht noch einmal gesendet  
Pakete in falscher Reihenfolge werden NICHT sortiert  
doppelte Pakete werden NICHT verworfen
- Kein ununterbrochener Datenstrom

**Eigenschaften von UDP (ein „Send and Pray“ Protocol)**

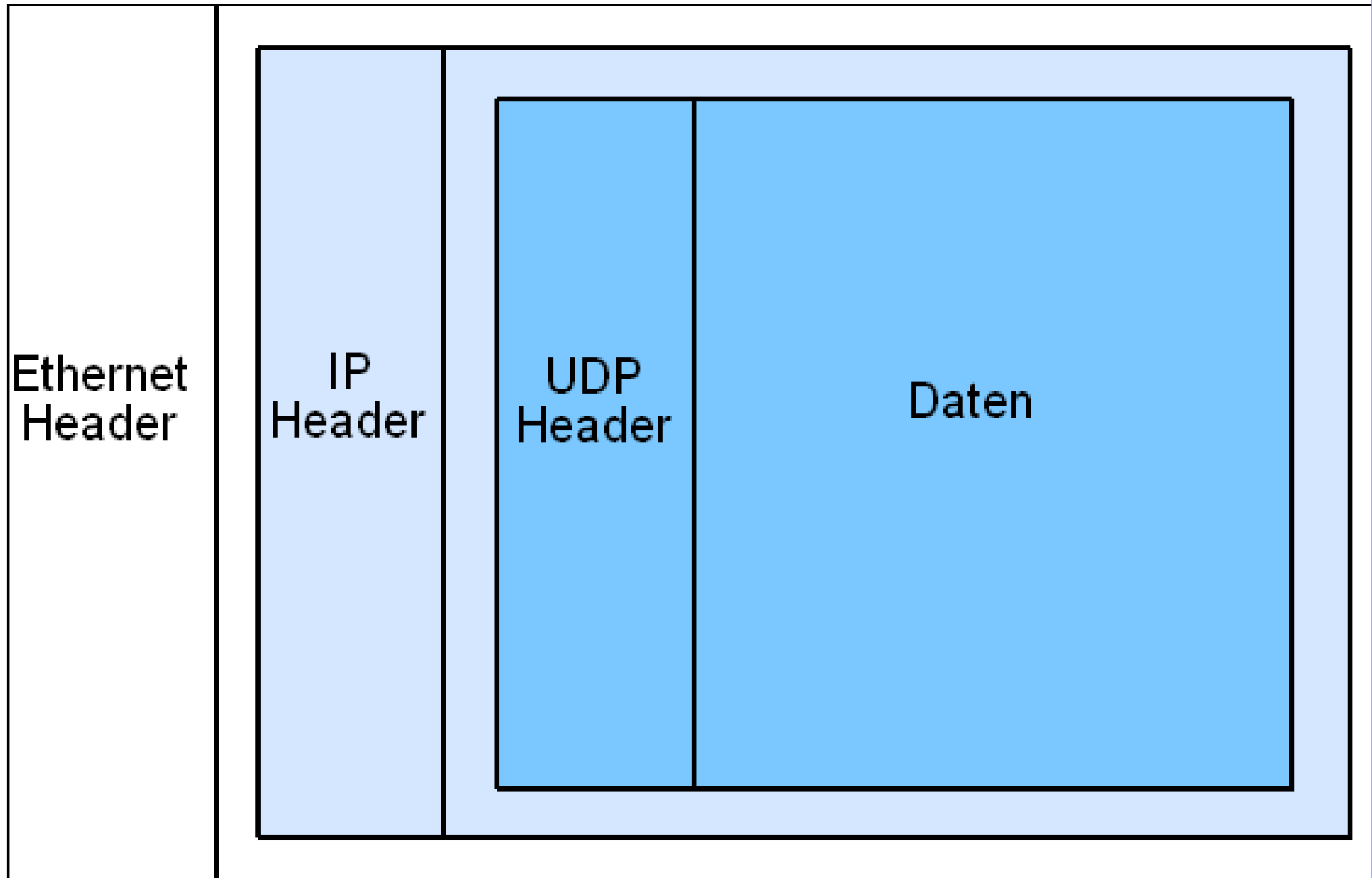
# UDP: Ports

wie TCP-Ports  
kollidieren nicht mit TCP-Ports

# UDP-Pakete

Auf UDP-Ebene werden Daten nur als Pakete verschickt.





**Ein UDP-Paket (eingepackt in ein IP-Paket)**

source port number:  
UDP-Quellport

destination port number:  
UDP-Zielpport

Checksum

Checksumme über Header & Daten

Bei Nichtübereinstimmung: Paket wird verworfen

Daten

z.B. RTSP (Real Time Streaming Protocol) Daten &  
Header

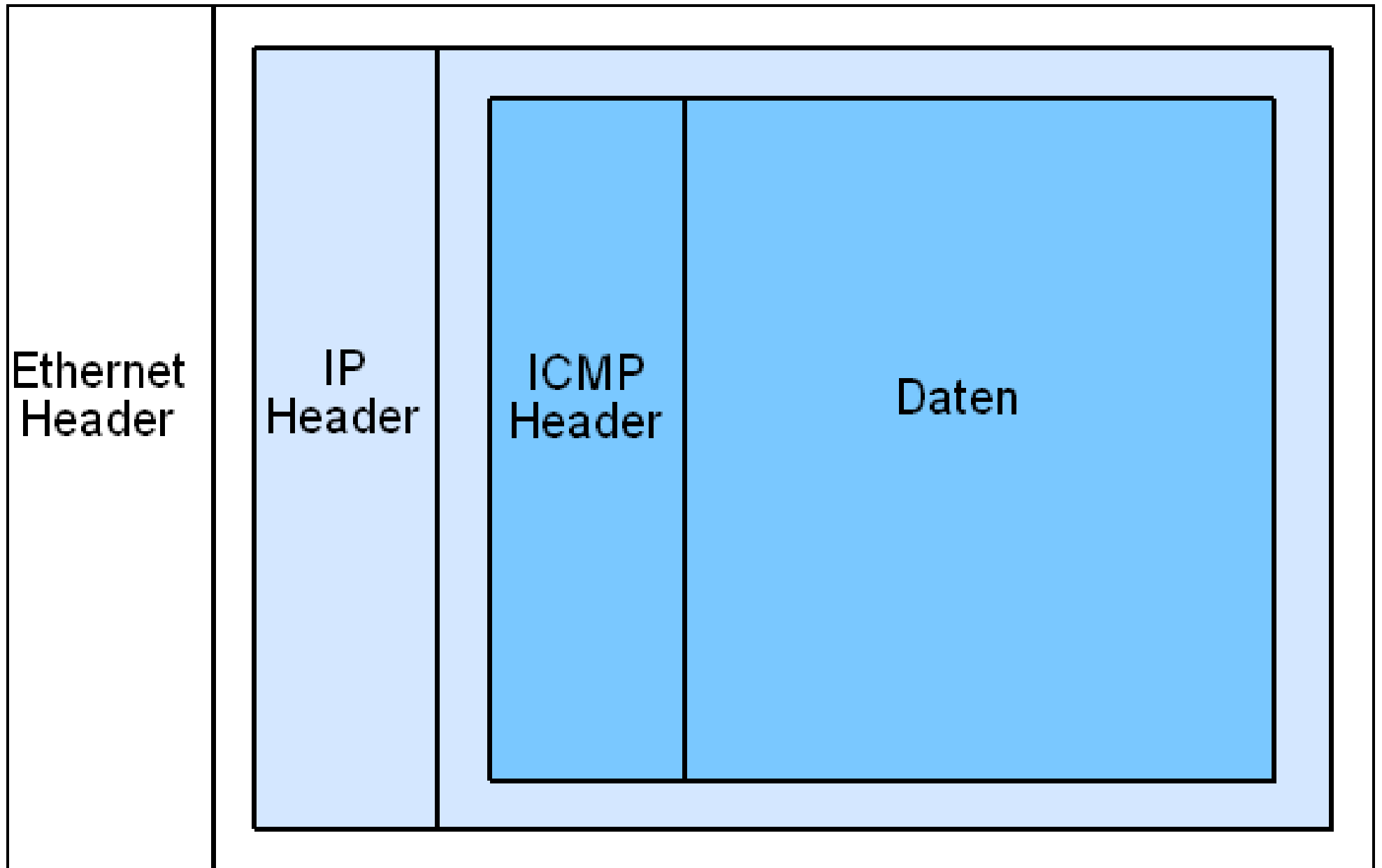
**Ein UDP-Paket kennt (unter anderem) folgende Felder**

# ICMP

Internet Control Message Protocol

Benachrichtigung über Fehler im Netzwerk

- verbindungslos
- kennt keine Ports



**Ein ICMP-Paket (eingepackt in ein IP-Paket)**

icmp type: Art der Nachricht

echo request: Aufforderung zum Senden eines echo reply

echo reply: Antwort auf echo request

destination unreachable: Netzwerk, Server ausgefallen

source quench: Zielrechner droht überladen zu werden

time exceeded: TTL-Zähler eines IP-Pakets hat 0 erreicht

parameter problem: allgemeine Fehlermeldung

Checksum:

Checksumme über gesamte ICMP Nachricht

**Ein ICMP-Paket kennt (unter anderem) folgende Felder**

# Das command line tool „ping“

auf allen Betriebssystemen vorhanden

Aufruf mit zu pingender IP-Adresse

Sendet ein ICMP-Paket vom Typ „echo request“

Der „gepingte“ Rechner antwortet mit „echo reply“

```
ping 192.168.1.10
```

od.

```
ping www.example.com
```

# DNS,SMTP,POP3,HTTP,HTTPS

Die vierte Netzwerk-Schicht (Applikations-Ebene)



# DNS – Domain Name System

Computer „denken“ am liebsten in Zahlen – Menschen nicht!

DNS Verwendet UDP (aber auch TCP)

# Geschichte des DNS

Vom ARPAnet zum Internet

Heute: dezentrale Verwaltung durch unzählige Name-Server

www.sae.edu

www.sae.at

www.tuwien.ac.at

law.tuwien.ac.at

informatik.univie.ac.at

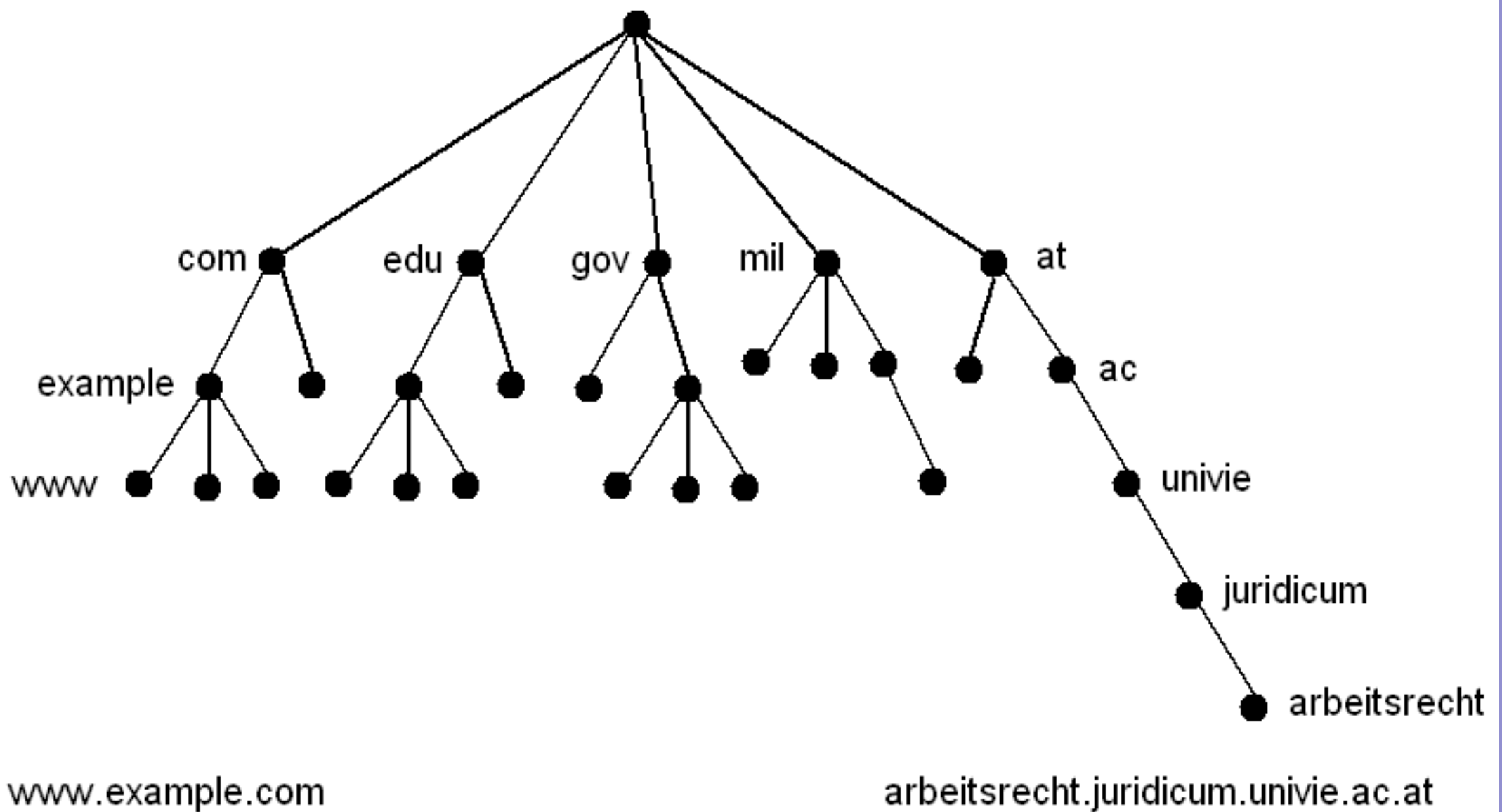
statistik.tuwien.ac.at

strafrecht.juridicum.univie.ac.at

vaio.sony.com

www.courtinfo.ca.gov

**Beispiele für einen Domain Name**



## Der Domain Name Space

Top Level Domain (TLD) bzw. first level domain:

z.B. „com“ in „example.com“; weitere TLD: edu, gov, mil, net, org, int, at, de, name, biz, info, ...

Second level domain:

z.B. „example“ in „example.com“

Domain

Ein subtree des Domain Name Space

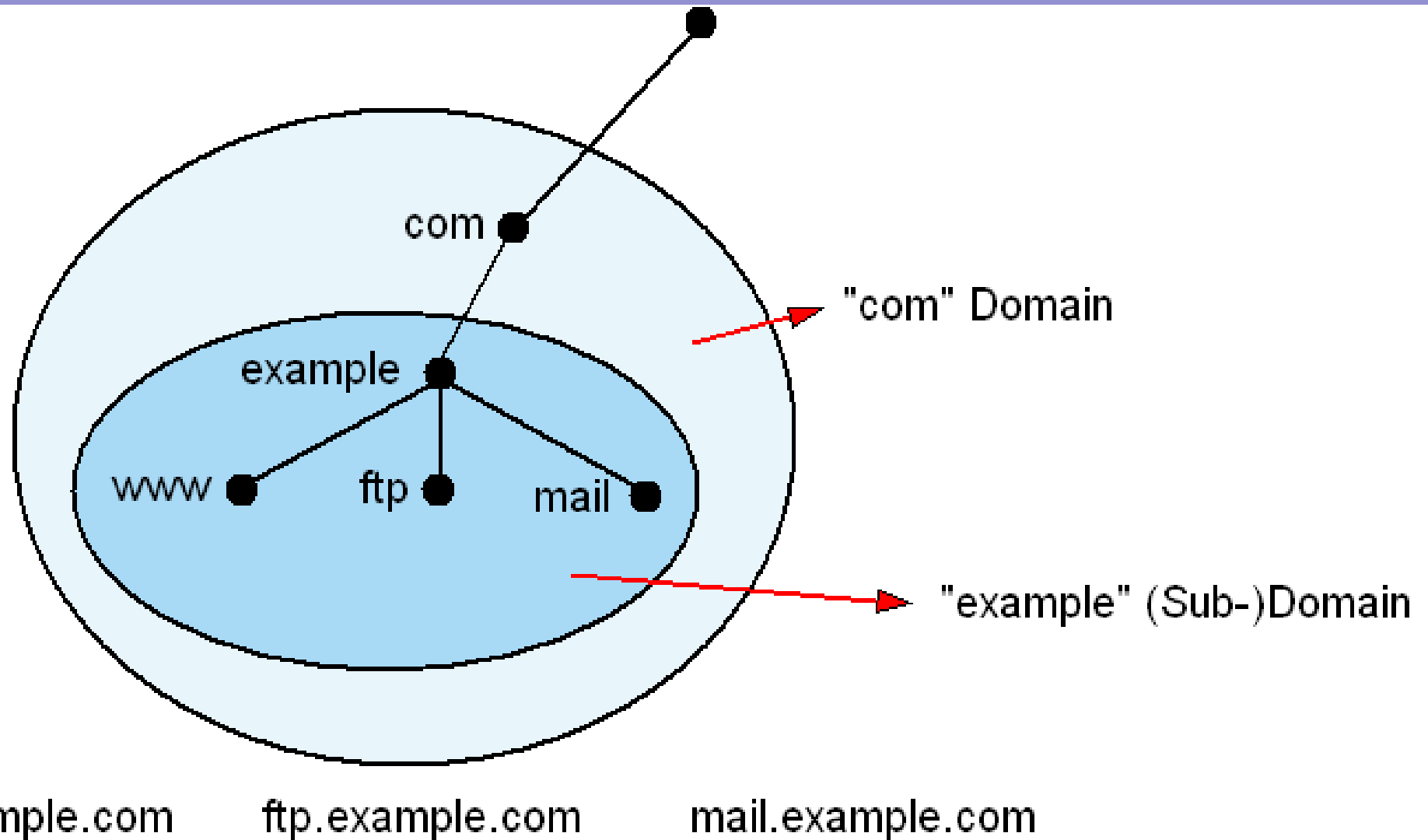
Sub-Domain

Eine Domain in einer Domain (relativer Begriff)

Name Server

Ein Server, der Informationen über Domains speichert  
(idR BIND)

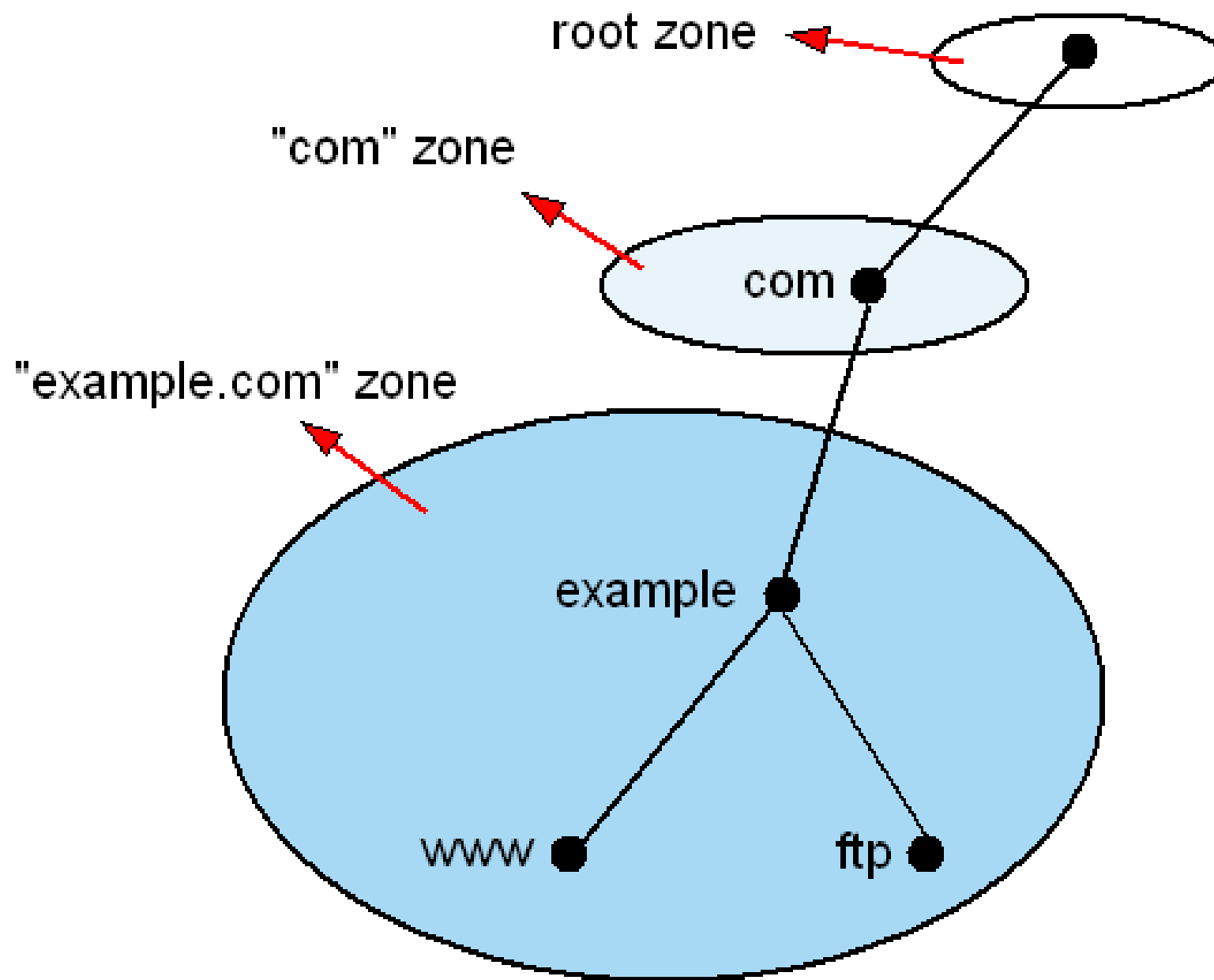
**Ein paar Grundbegriffe**



## Domain & Sub-Domain

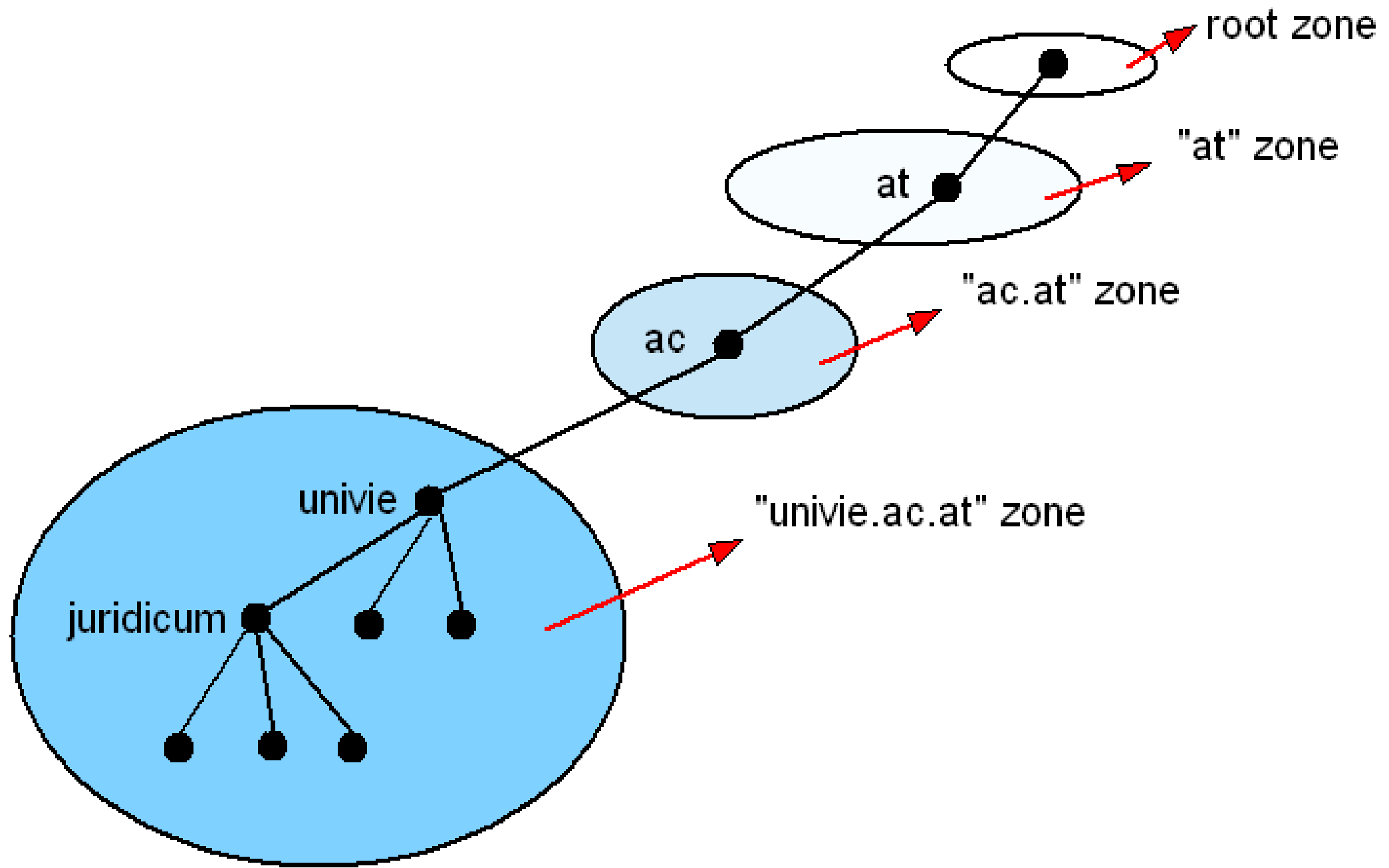
# DNS: Dezentrale Verwaltung durch Delegation

Ein Name-Server hat nur für seine Zones „Autorität“

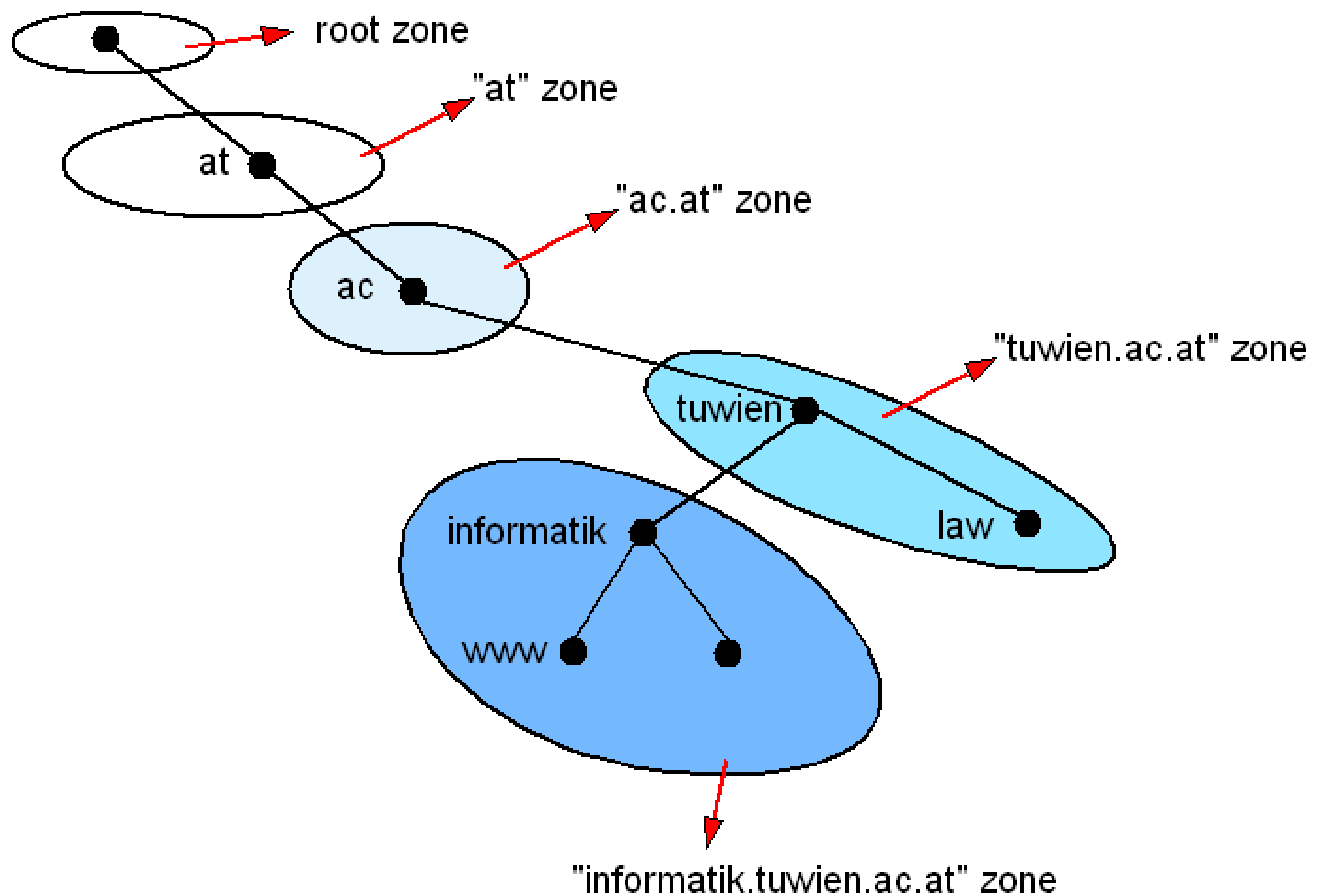


**Die Zone „com“ hat nur einen Verweis auf die Zone „example.com“**

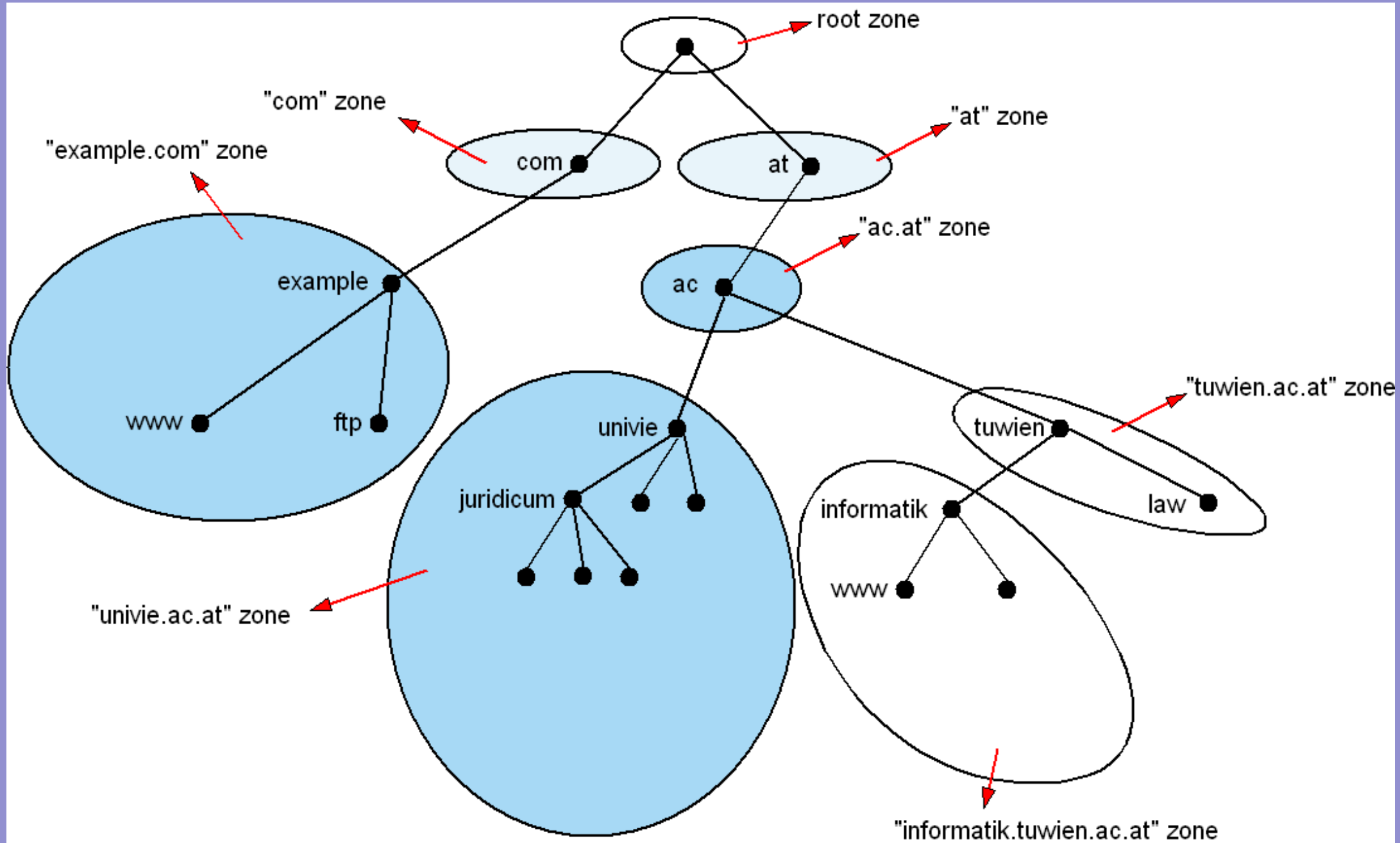




**Zones: „root“, at, ac.at & univie.ac.at**



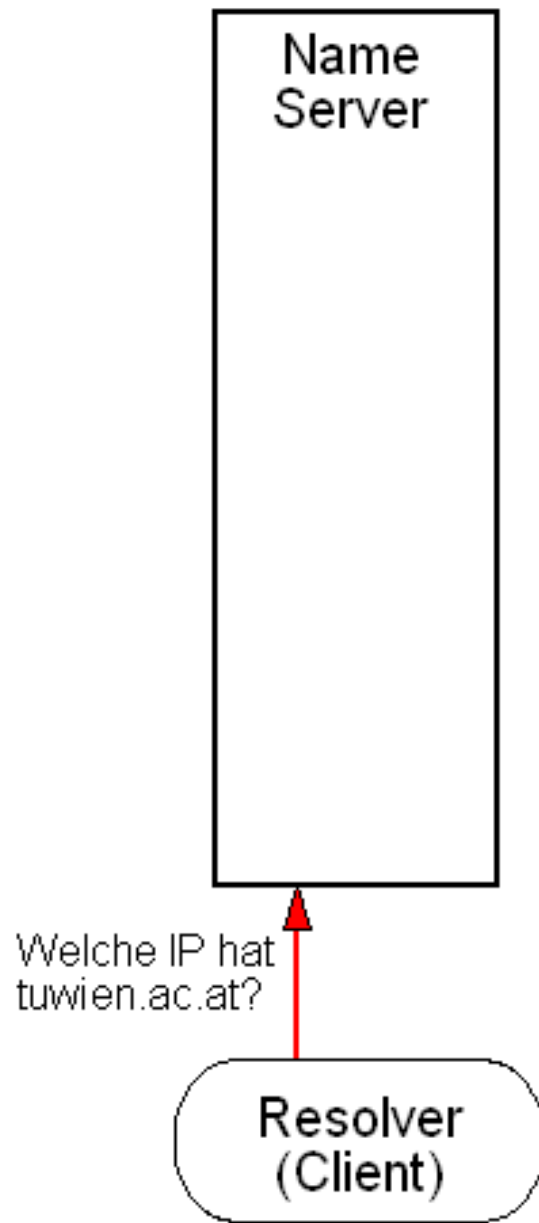
**Zones: „root“, at, ac.at, tuwien.ac.at, informatik.tuwien.ac.at**



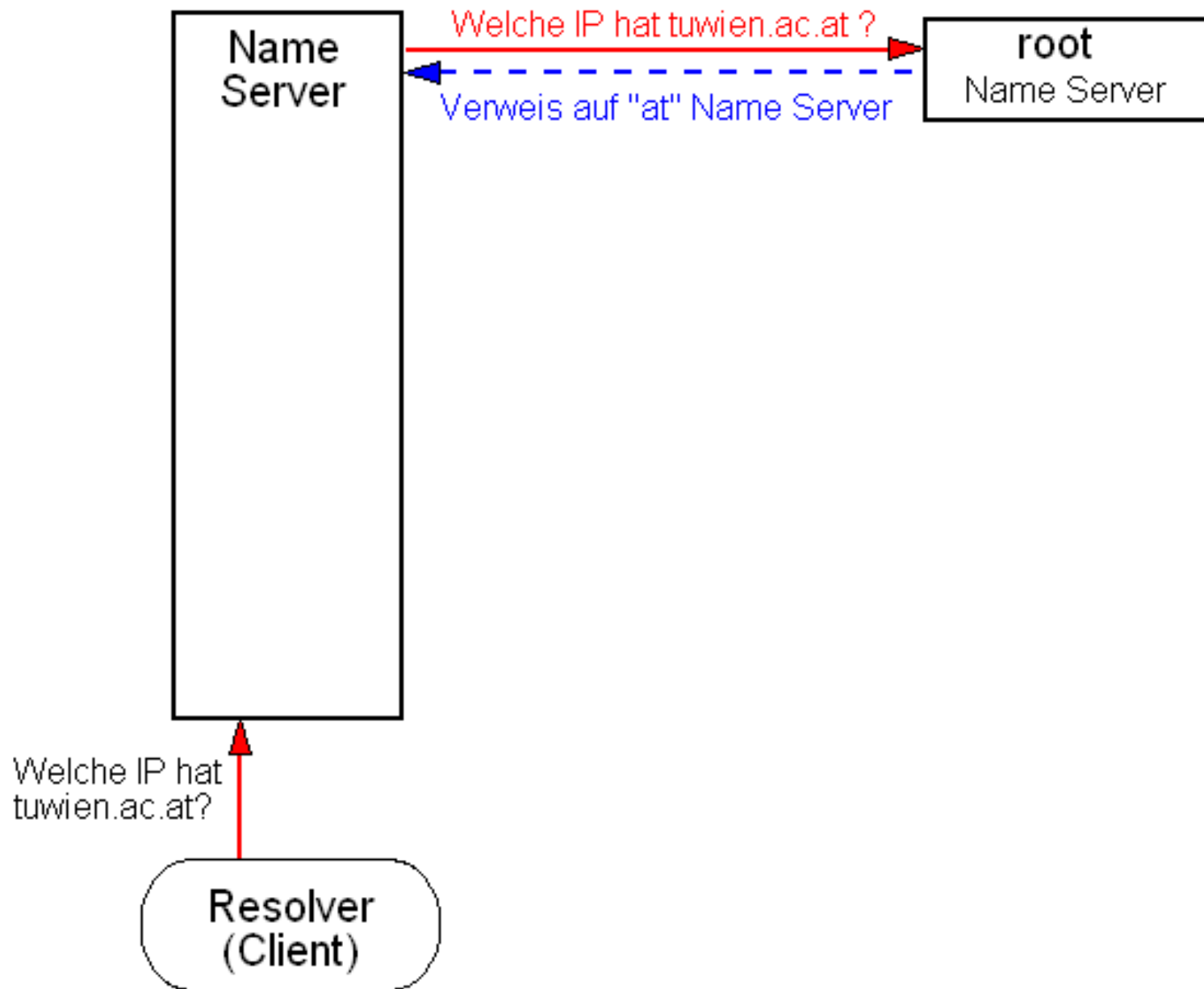
**Dezentrale Verwaltung!**

# Name Resolution

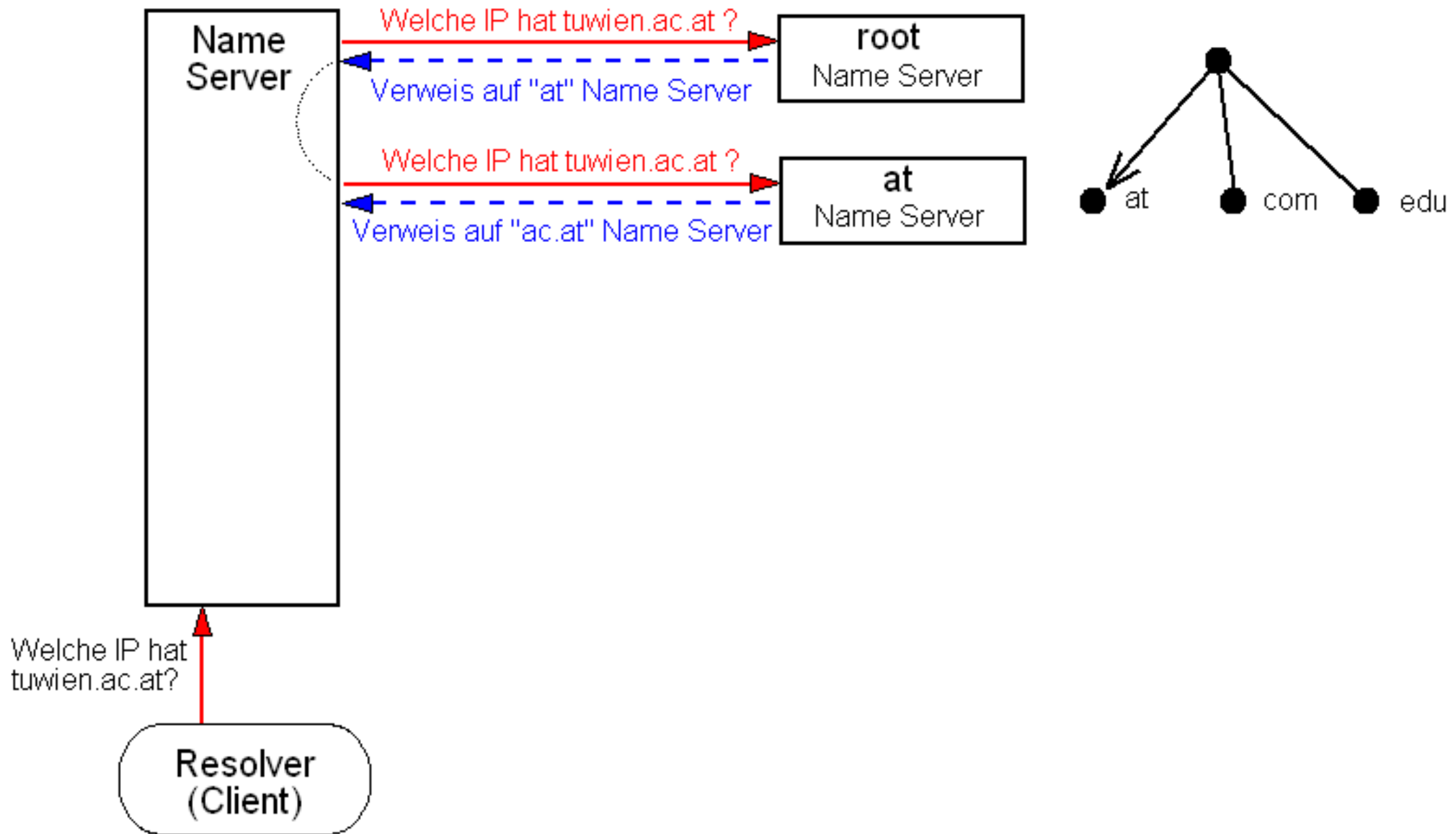
Einen Domain-Namen in eine IP-Adresse auflösen



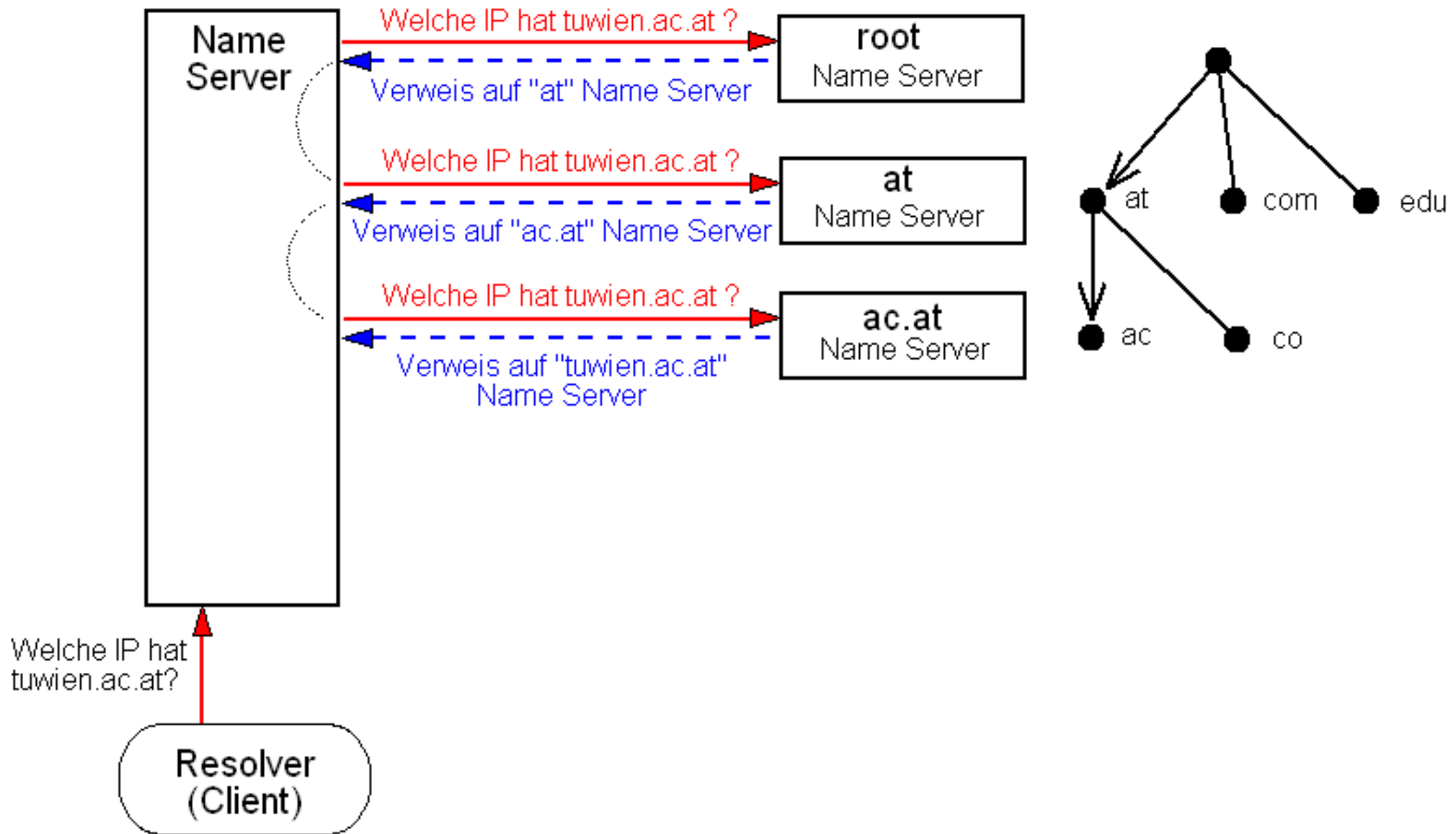
**Welche IP-Adresse hat tuwien.ac.at?**



**Der root Name-Server verweist auf den Name-Server der „at“-Domain**

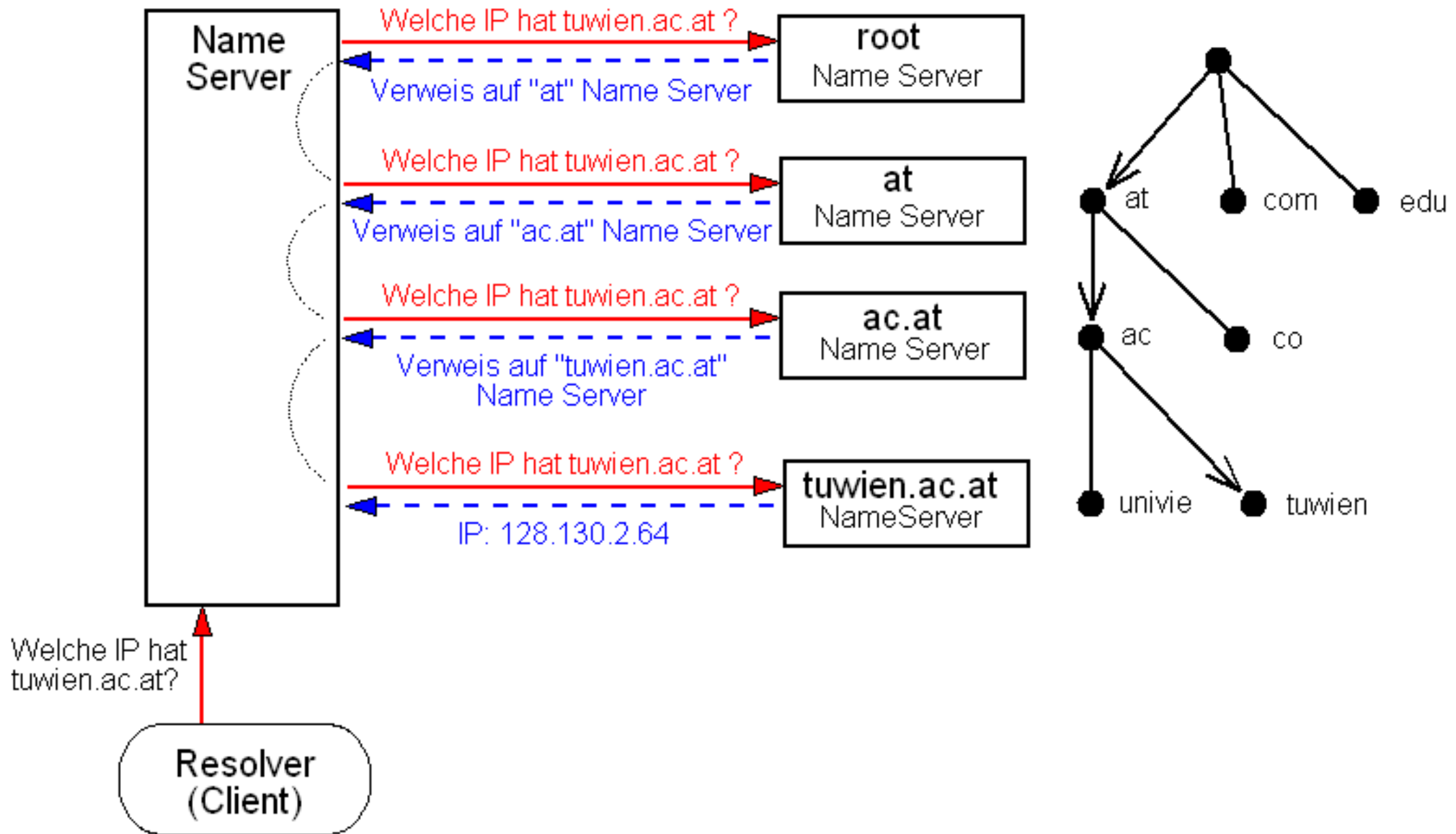


**Der NS der „at“-Domain verweist auf den NS der „ac.at“-Domain**

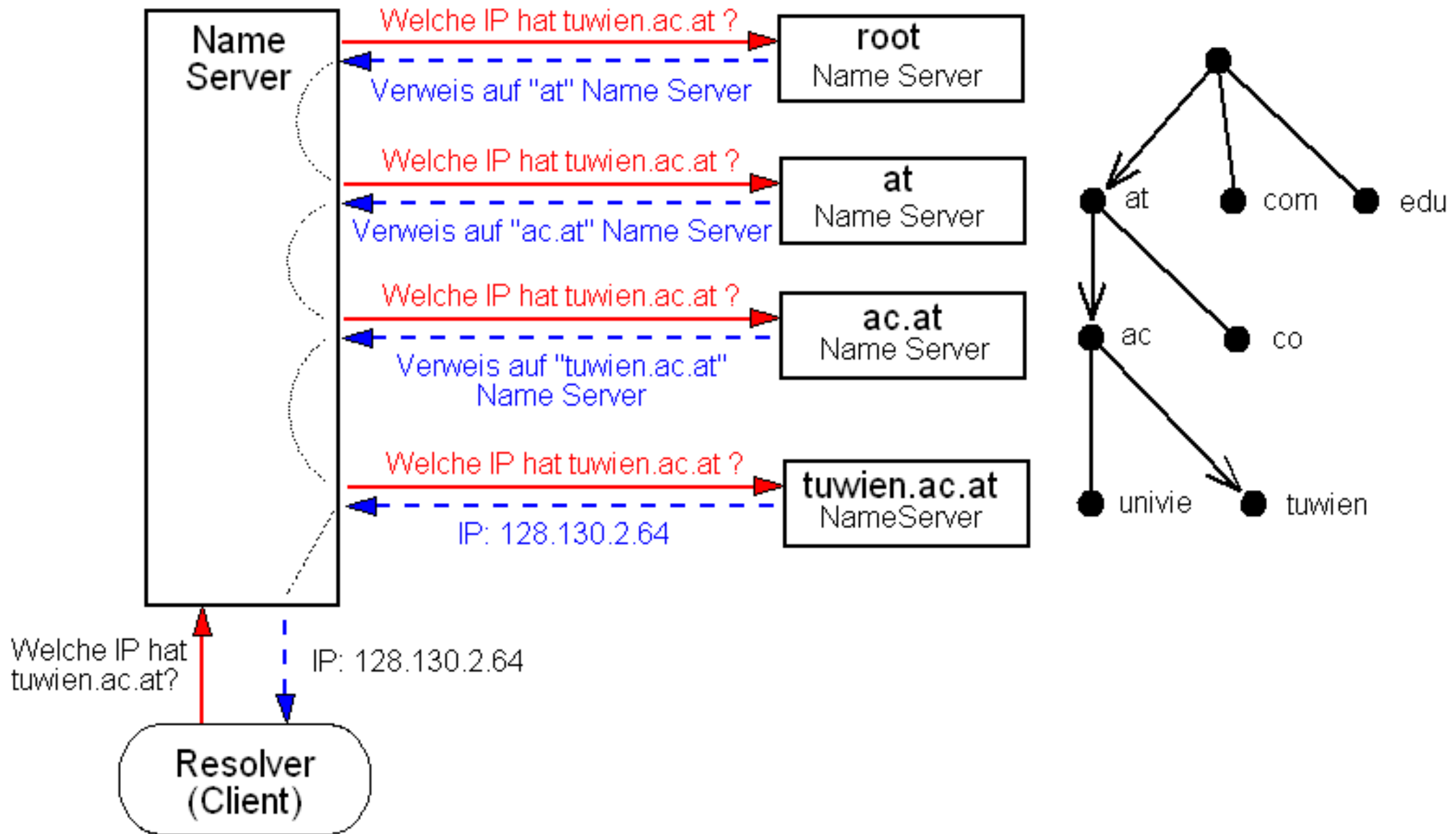


**Der NS der „ac.at“-Domain verweist auf den „tuwien.ac.at“ NS**





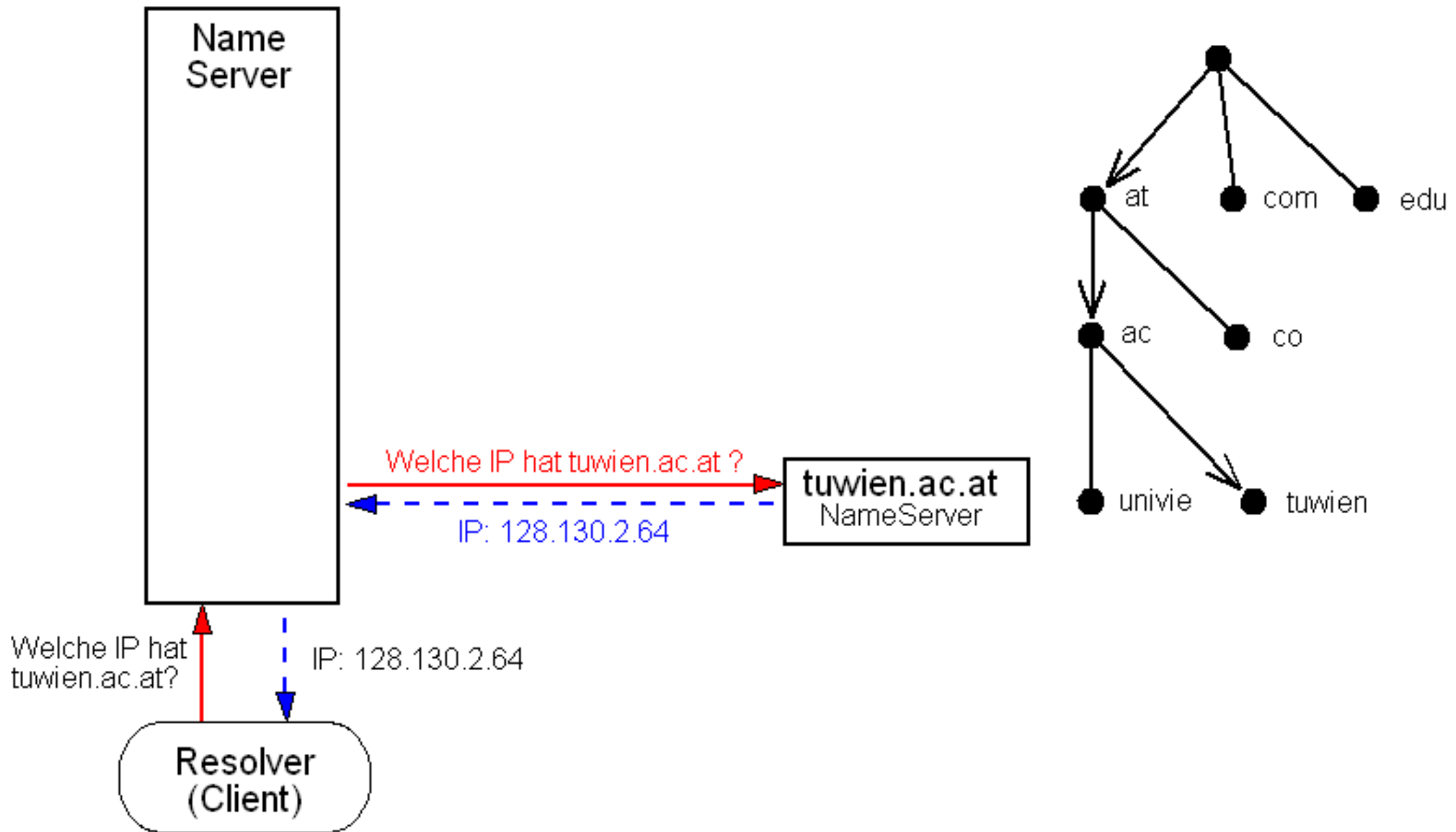
**Der NS der Domain „tuwien.ac.at“ kennt die IP und antwortet**



**Der Resolver erhält die IP 128.130.2.64**

## DNS: Caching

Jede Zone hat eine „Time To Live“ (TTL); erst nach deren Überschreiten erfolgt eine neue Abfrage.



**Wesentlich bessere Performance dank Caching**

# Das command line tool nslookup

Parameter: Domain-Name, der in IP aufgelöst werden soll

z.B: nslookup www.sae.edu

**„Händisch“ einen Domain-Name in eine IP-Adresse auflösen**

# E-Mail (SMTP, POP3, IMAP)

Eine der ältesten aber auch wirtschaftlich wichtigsten Anwendungen des Internets.

Envelope	Header	Body
MAIL FROM: != From Header  RCPT TO: != To Header	Subject Betreff  From Absender (user@host.tld)  To Empfänger (user@host.tld)  Cc Carbon Copy  Bcc Blind carbon copy	Inhalt der Nachricht  Seit MIME (Multi-purpose Internet Mail Extension) in beliebigem Format

**Woraus besteht ein E-Mail?**

SMTP-Server  
mail.unet.univie.ac.at



SMTP-Server  
mail.empoweredmedia.com



POP3-Server  
mail.empoweredmedia.com

From: a0201227@unet.univie.ac.at  
To: lukas@empoweredmedia.com



Outlook/Thunderbird/...  
als SMTP-Client



Outlook/Thunderbird/...  
als POP3-Client

**Der typische Weg eines E-Mails (simplifiziert)**



Woher weiß mail.unet.univie.ac.at welcher Server für lukas@empoweredmedia.com zuständig ist?

Mittels DNS kann für jede Domain ein Mail-Server definiert werden  
z.B: für empoweredmedia.com: mail.empoweredmedia.com  
für sae.at: mail.nextra.at

mit nslookup:

```
nslookup -type=MX empoweredmedia.com
```

Selbst ein E-Mail via telnet versenden:

```
telnet smtp.chello.at 25  
> HELO ROCKET  
> MAIL FROM: lukas.feiler@chello.at  
> RCPT TO: saetest@chello.at  
> DATA  
> From: me  
> To: you  
> Subject: test  
>  
> This is just a test.  
> .  
> quit
```

**Do you speak SMTP?**

POP3: Mails werden am Server nur zwischengespeichert  
jedes heruntergeladene Mail wird idR am Server gelöscht

IMAP: Mails bleiben am Server

Selbst E-Mails via telnet lesen:

```
telnet pop.chello.at 110  
>USER saetest@chello.at  
>PASS *****  
>LIST  
>TOP 1 0  
>RETR 1  
>DELE 1  
>LIST  
>quit
```

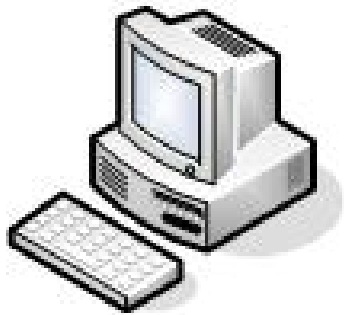
**Do you speak POP3?**

# FTP (File Transfer Protocol)

Läuft über TCP

Standard-Port ist 21

FTP-Client  
(z.B: CuteFTP)



FTP-Server  
(idR auch Web-Server)



Authentifizierung mit  
Username & Passwort

**FTP wird meist verwendet um auf Web-Server zuzugreifen**

# GET

Eine Datei vom FTP-Server zum FTP-Client kopieren

# PUT

Eine Datei vom FTP-Client zum FTP-Server kopieren

Weiters:

Ordner am Server erstellen

Dateien/Ordner am Server umbenennen und verschieben

binary/ascii

binär: images, flash, sound-files

ascii: html, txt, php

control connection & data connection

eigene Verbindung für GET/PUT Operationen

active FTP

data connection wird vom FTP-Server (aktiv) zum FTP-Client aufgebaut

passive FTP

data connection wird vom FTP-Client zu FTP-Server aufgebaut;  
der FTP-Server wartet passiv auf die Verbindungsanfrage durch  
den FTP-Client

**Weitere Punkte zu FTP**



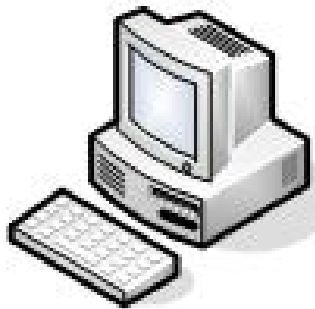
# HTTP (Hyper-Text Transfer Protocol)

Läuft über TCP

Standard-Port ist 80

Browser (z.B IE)  
als HTTP-Client

Web-Server (z.B Apache)  
als HTTP-Server



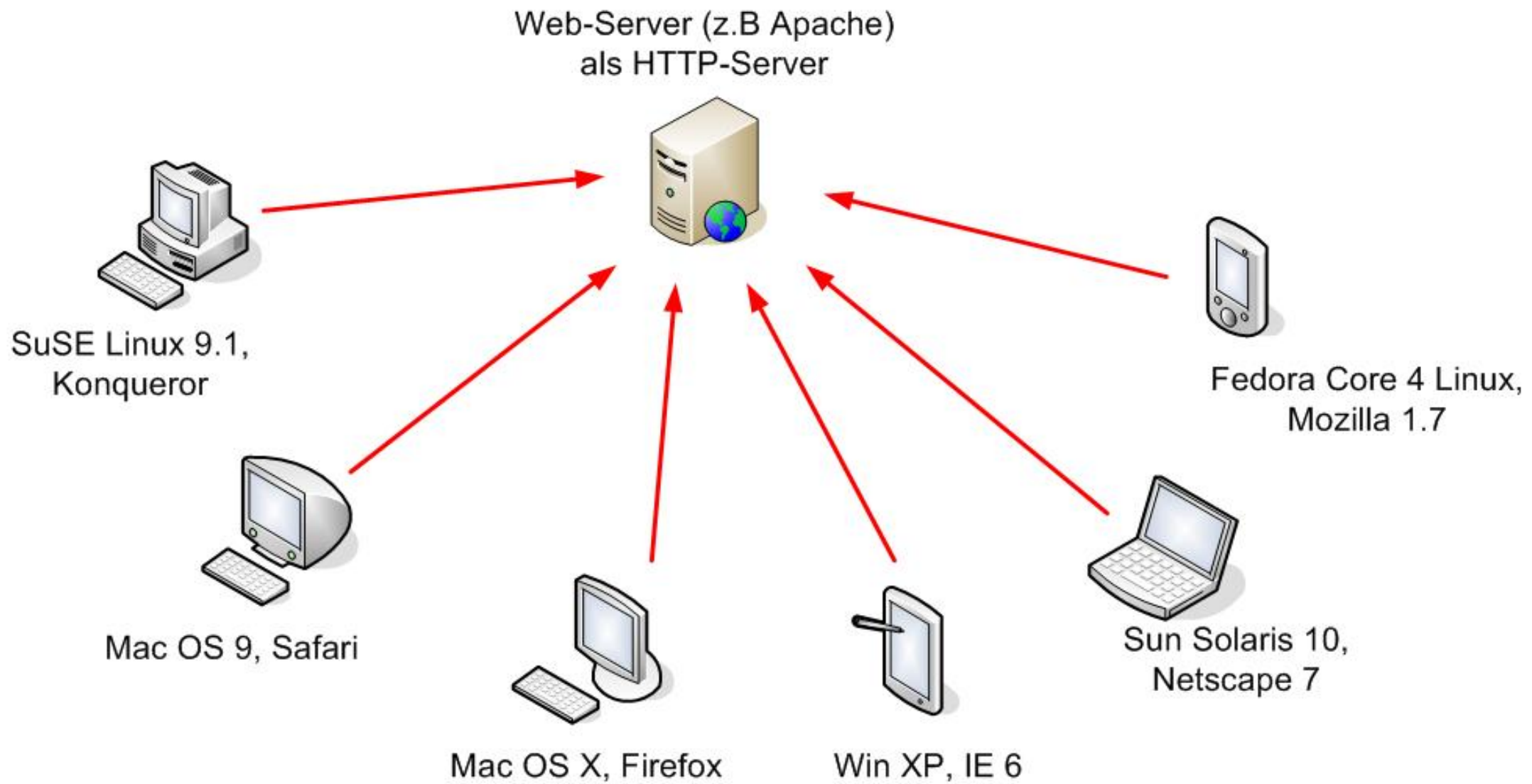
HTTP Request



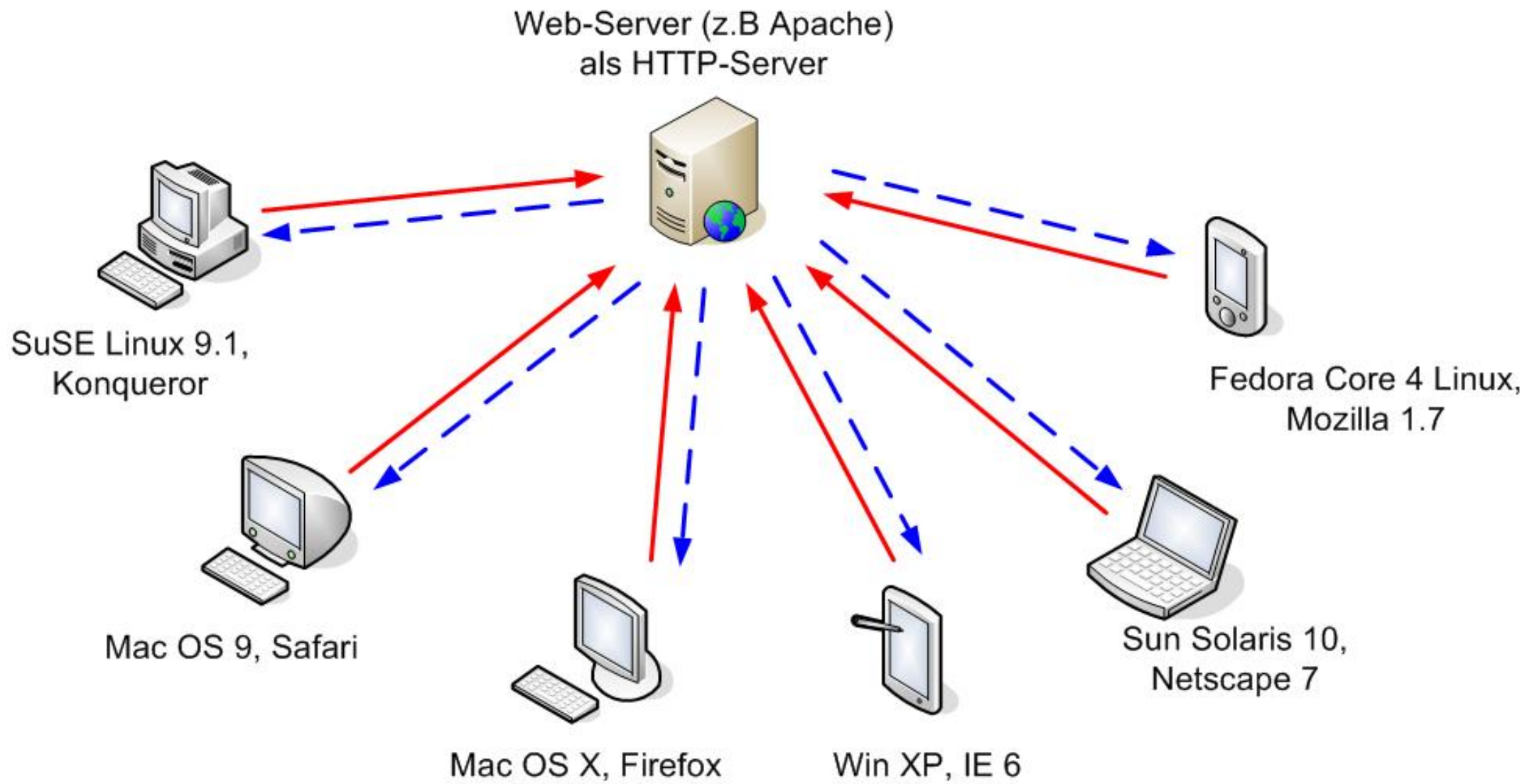
HTTP Response



a) Browser fordert Daten an    b) Server sendet Daten



**Viele gleichzeitige Anfragen von verschiedenen Clients**



**Alle Anfragen müssen beantwortet werden**

# Der HTTP Request

Was wird angefordert bzw. mitgesendet?

# Der HTTP Request

Was wird angefordert bzw. mitgesendet?

- URL
- Request Method: GET od. POST

# URL - Uniform Resource Locator

`http://www.univie.ac.at`

`http://www.univie.ac.at/strafrecht/`

`http://www.univie.ac.at/strafrecht/index.html`

`http://www.univie.ac.at/strafrecht/news.php`

`http://mydomain.com/folder/search.php?keyword=SAE`

Request für `http://www.google.at/index.html`

`telnet www.google.at 80`

`>GET /index.html HTTP/1.0`

`>`

`>`

`[zwei Mal Enter]`

# Request Methods:

## GET (bekommen)

Jeder „normale“ URL-Aufruf

Angehängter Query String in der Form

*?variable1=wert1&variable2=wert2*

möglich

z.B: <http://mydomain.com/folder/search.php?keyword=SAE>

## POST (verschicken)

Query String wird nicht wie bei GET an URL angehängt

In eigenem HTTP-Header (kann sehr groß sein)

Mit beiden können Daten aus einer HTML-Form übermittelt werden

## Request Method



## Der HTTP Response

HTTP Server liefert die angeforderten Daten bzw. einen Error Code

# HTTP Status Codes

100 bis 199: nur informativ; selten

200 bis 299: Erfolgreicher Request; va 200 oft verwendet

300 bis 399: Warnung (aber noch kein Fehler)

400 bis 499: Client Error (fehlerhafter Request)

500 bis 599: Server Error (Request war gültig)

# HTTP Keep Alive

Seit HTTP 1.1

Um langen TCP-Verbindungsaufbau f. jedes Bild (!) zu umgehen

# HTTPS (HyperText Transfer Protocol Secure)

Läuft ebenso über TCP  
Standard-Port ist 443

# Funktion von HTTPS

Verschlüsselung mit SSL

Sicherstellung der Identität des Servers

## HTTPS-URL

<https://mydomain.com/index.php>

# SSH (Secure Shell)

Verschlüsselte remote Shell