Das Domain Name System

Am Beispiel von BIND

Dirk Geschke

Linux User Group Erding

22. November 2006

Gliederung

Die Anfänge des Internets.

Adressierung von Computern

- ARPANET: Zuordnung Namen -> Adressen über die Datei HOSTS.TXT
- Unix-System: `/etc/hosts`
- SRI's Network Information Center (NIC) verwaltete die Datei HOSTS.TXT
- 80'er Jahre: Einführung von TCP/IP in Berkeley (BSD)
- explosionsartiger Zuwachs der Systeme

Die Anfänge des Internets.

Adressierung von Computern

- ARPANET: Zuordnung Namen -> Adressen über die Datei HOSTS.TXT
- Unix-System: `/etc/hosts`
- SRI's Network Information Center (NIC) verwaltete die Datei HOSTS.TXT
- 80'er Jahre: Einführung von TCP/IP in Berkeley (BSD)
- explosionsartiger Zuwachs der Systeme

Die Anfänge des Internets.

Adressierung von Computern

- ARPANET: Zuordnung Namen -> Adressen über die Datei HOSTS.TXT
- Unix-System: `/etc/hosts`
- SRI's Network Information Center (NIC) verwaltete die Datei HOSTS.TXT
- 80'er Jahre: Einführung von TCP/IP in Berkeley (BSD)
- explosionsartiger Zuwachs der Systeme

Die Anfänge des Internets.

Adressierung von Computern

- ARPANET: Zuordnung Namen -> Adressen über die Datei HOSTS.TXT
- Unix-System: `/etc/hosts`
- SRI's Network Information Center (NIC) verwaltete die Datei HOSTS.TXT
- 80'er Jahre: Einführung von TCP/IP in Berkeley (BSD)
- explosionsartiger Zuwachs der Systeme

Die Anfänge des Internets.

Adressierung von Computern

- ARPANET: Zuordnung Namen -> Adressen über die Datei HOSTS.TXT
- Unix-System: `/etc/hosts`
- SRI's Network Information Center (NIC) verwaltete die Datei HOSTS.TXT
- 80'er Jahre: Einführung von TCP/IP in Berkeley (BSD)
- explosionsartiger Zuwachs der Systeme

Zahl der Internetsysteme

Aus RFC 1296:

Datum	Systeme	Hosttabelle
08/81	213	Host table #152
08/83	562	Host table #300
10/84	1,024	Host table #392
10/85	1,961	Host table #485
11/86	5,089	
12/87	28,174	
10/88	56,000	
10/89	159,000	
10/90	313,000	

Die Anfänge des Internets.
Die Probleme mit HOSTS.TXT

- hohe Last und Traffic bei SRI-NIC
- Namenskollisionen: kein Eintrag durfte doppelt vorkommen
- Konsistenzprobleme: Update aller HOSTS.TXT sehr schwierig

$\Longrightarrow$ Lösung: Domain Name System

Die Anfänge des Internets.
Die Probleme mit HOSTS.TXT

- hohe Last und Traffic bei SRI-NIC
- Namenskollisionen: kein Eintrag durfte doppelt vorkommen
- Konsistenzprobleme: Update aller HOSTS.TXT sehr schwierig

$\Longrightarrow$ Lösung: Domain Name System

Die Anfänge des Internets.

Die Probleme mit HOSTS.TXT

- hohe Last und Traffic bei SRI-NIC
- Namenskollisionen: kein Eintrag durfte doppelt vorkommen
- Konsistenzprobleme: Update aller HOSTS.TXT sehr schwierig

$\Longrightarrow$ Lösung: Domain Name System

Die Anfänge des Internets.

Die Probleme mit HOSTS.TXT

- hohe Last und Traffic bei SRI-NIC
- Namenskollisionen: kein Eintrag durfte doppelt vorkommen
- Konsistenzprobleme: Update aller HOSTS.TXT sehr schwierig

$\Longrightarrow$ Lösung: Domain Name System

DNS

- verteilte Datenbank $\longrightarrow$ dezentrale Verwaltung
- Namenskollisionen nur noch innerhalb einer Domain möglich
- 2 Dienste:
  - nameserver (Serverdienst)
  - resolver (Clientdienst)

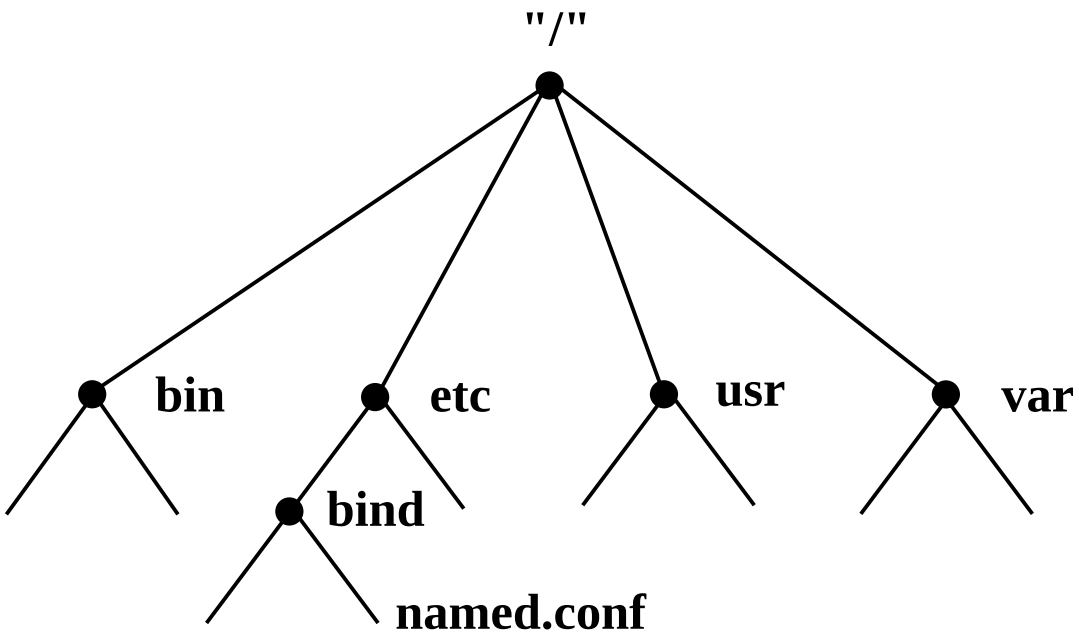
DNS

- verteilte Datenbank $\longrightarrow$ dezentrale Verwaltung
- Namenskollisionen nur noch innerhalb einer Domain möglich
- 2 Dienste:
  - nameserver (Serverdienst)
  - resolver (Clientdienst)

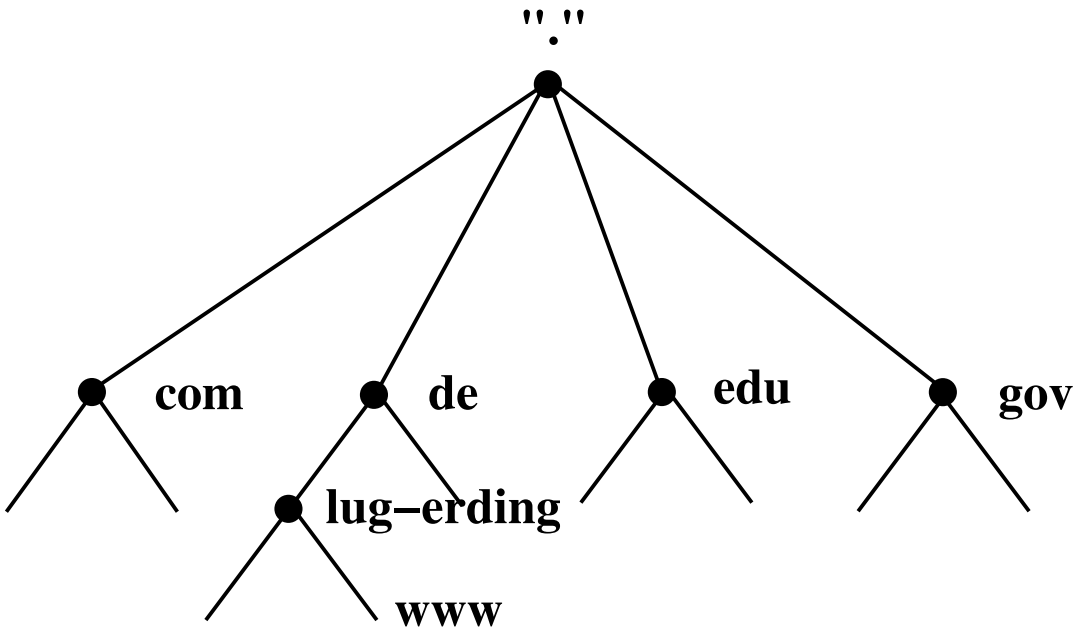
DNS

- verteilte Datenbank $\longrightarrow$ dezentrale Verwaltung
- Namenskollisionen nur noch innerhalb einer Domain möglich
- 2 Dienste:
  - nameserver (Serverdienst)
  - resolver (Clientdienst)

Struktur



Struktur



Unterschiede

- Dateisystem: **/etc/bind/named.conf**
- DNS: **www.lug-erding.de.**

$\Longrightarrow$ **Reihenfolge** ist entgegengesetzt!

ursprüngliche Topleveldomains

- **com**: kommerzielle Organisationen
- **edu**: educational → Schulen, Universitäten
- **gov**: government → Regierungsstellen, z.B. NASA.gov
- **mil**: militärische Einrichtungen
- **net**: Netzwerkbetreiber, seit 1996 frei
- **org**: nicht-kommerzielle Einrichtungen, seit 1996 frei
- **int**: internationale Einrichtungen, z.B. NATO.int
- **arpa**: Übergangsdomain HOSTS.TXT → DNS

Länderkürzel

- definiert in **ISO3166**: 2 Buchstaben pro Land
- Ausnahme: Großbritannien nutzt **uk** statt **gb**
- **de**: Deutschland
- **us**: USA, selten benutzt
- **by**: Weißrußland, nicht Bayern!
- **tv**: Tuvalu
- Abbildung der klassischen Toplevel unterhalb der Länderkürzel, z.B. **co.uk**

Länderkürzel

- definiert in **ISO3166**: 2 Buchstaben pro Land
- Ausnahme: Großbritannien nutzt **uk** statt **gb**
- **de**: Deutschland
- **us**: USA, selten benutzt
- **by**: Weißrußland, nicht Bayern!
- **tv**: Tuvalu
- Abbildung der klassischen Toplevel unterhalb der Länderkürzel, z.B. **co.uk**

Länderkürzel

- definiert in **ISO3166**: 2 Buchstaben pro Land
- Ausnahme: Großbritannien nutzt **uk** statt **gb**
- **de**: Deutschland
- **us**: USA, selten benutzt
- **by**: Weißrußland, nicht Bayern!
- **tv**: Tuvalu
- Abbildung der klassischen Toplevel unterhalb der Länderkürzel, z.B. **co.uk**

Länderkürzel

- definiert in **ISO3166**: 2 Buchstaben pro Land
- Ausnahme: Großbritannien nutzt **uk** statt **gb**
- **de**: Deutschland
- **us**: USA, selten benutzt
- **by**: Weißrußland, nicht Bayern!
- **tv**: Tuvalu
- Abbildung der klassischen Toplevel unterhalb der Länderkürzel, z.B. **co.uk**

Länderkürzel

- definiert in **ISO3166**: 2 Buchstaben pro Land
- Ausnahme: Großbritannien nutzt **uk** statt **gb**
- **de**: Deutschland
- **us**: USA, selten benutzt
- **by**: Weißrußland, nicht Bayern!
- **tv**: Tuvalu
- Abbildung der klassischen Toplevel unterhalb der Länderkürzel, z.B. **co.uk**

Länderkürzel

- definiert in **ISO3166**: 2 Buchstaben pro Land
- Ausnahme: Großbritannien nutzt **uk** statt **gb**
- **de**: Deutschland
- **us**: USA, selten benutzt
- **by**: Weißrußland, nicht Bayern!
- **tv**: Tuvalu
- Abbildung der klassischen Toplevel unterhalb der Länderkürzel, z.B. **co.uk**

Länderkürzel

- definiert in **ISO3166**: 2 Buchstaben pro Land
- Ausnahme: Großbritannien nutzt **uk** statt **gb**
- **de**: Deutschland
- **us**: USA, selten benutzt
- **by**: Weißrußland, nicht Bayern!
- **tv**: Tuvalu
- Abbildung der klassischen Toplevel unterhalb der Länderkürzel, z.B. **co.uk**

neue Domains

- **aero**: Luftfahrt, gesponsert
- **biz**: business, generisch
- **coop**: generisch, Kooperationen
- **info**: generisch
- **museum**: Museen, gesponsert
- **name**: Privatpersonen, generisch
- **pro**: Professionelle Anbieter, generisch
- **mobi**: Mobilfunk, gesponsert
- **jobs**: Stellenvermittlungen, gesponsert
- **travel**: Reiseunternehmen, gesponsert

Delegation:

- DNS ist eine dezentrale Datenbank
- Verlinkung $\longrightarrow$ Delegation
- *Unterdomain* heißt Subdomain
- Subdomains können an andere Nameserver delegiert werden
- dies muß nicht sein!

Delegation:

- DNS ist eine dezentrale Datenbank
- Verlinkung $\longrightarrow$ Delegation
- *Unterdomain* heißt Subdomain
- Subdomains können an andere Nameserver delegiert werden
- dies muß nicht sein!

Delegation:

- DNS ist eine dezentrale Datenbank
- Verlinkung $\longrightarrow$ Delegation
- *Unterdomain* heißt Subdomain
- Subdomains können an andere Nameserver delegiert werden
- dies muß nicht sein!

Delegation:

- DNS ist eine dezentrale Datenbank
- Verlinkung $\longrightarrow$ Delegation
- *Unterdomain* heißt Subdomain
- Subdomains können an andere Nameserver delegiert werden
- dies muß nicht sein!

Delegation:

- DNS ist eine dezentrale Datenbank
- Verlinkung $\longrightarrow$ Delegation
- *Unterdomain* heißt Subdomain
- Subdomains können an andere Nameserver delegiert werden
- dies muß nicht sein!

Einträge in lug-erding.de nicht-delegiert

www	IN	A	89.110.147.240
www.debian	IN	A	...
www.gentoo	IN	A	...
www.suse	IN	A	...
www.redhat	IN	A	...

Einträge in lug-erding.de delegiert, glue-Einträge

www	IN	A	89.110.147.240
debian	IN	NS	ns1.debian.lug-erding.de.
debian	IN	NS	ns2.debian.lug-erding.de.
gentoo	IN	NS	ns1.gentoo.lug-erding.de.
gentoo	IN	NS	ns2.gentoo.lug-erding.de.
ns1.debian	IN	A	...
ns2.debian	IN	A	...
ns1.gentoo	IN	A	...
ns2.gentoo	IN	A	...

Einträge in lug-erding.de delegiert, glue-Einträge

www	IN	A	89.110.147.240
debian	IN	NS	ns1.debian.lug-erding.de.
debian	IN	NS	ns2.debian.lug-erding.de.
gentoo	IN	NS	ns1.gentoo.lug-erding.de.
gentoo	IN	NS	ns2.gentoo.lug-erding.de.
ns1.debian	IN	A	...
ns2.debian	IN	A	...
ns1.gentoo	IN	A	...
ns2.gentoo	IN	A	...

Arten von Nameservern

historisch:

- **primary** master nameserver
- **secondary** master nameserver

heute:

- **master** Nameserver
- **slave** Nameserver

Arten von Nameservern

historisch:

- **primary** master nameserver
- **secondary** master nameserver

heute:

- **master** Nameserver
- **slave** Nameserver

Zonentransfer

- Abgleich der **Zonendaten**, dem Inhalt einer Domain
- gewöhnlich vom **master** zum **slave**
- es ist auch ein Abgleich vom **slave** zum **slave**

Zonentransfer

- Abgleich der **Zonendaten**, dem Inhalt einer Domain
- gewöhnlich vom **master** zum **slave**
- es ist auch ein Abgleich vom **slave** zum **slave**

Zonentransfer

- Abgleich der **Zonendaten**, dem Inhalt einer Domain
- gewöhnlich vom **master** zum **slave**
- es ist auch ein Abgleich vom **slave** zum **slave**

Resolver

- befragt Nameserver
- wertet die Antworten aus, z.B. IP-Adresse, DNS-Namen, Fehler
- liefert das Ergebnis zurück

iterative Namensauflösung

iterative Auflösung von z.B. **www.lug-erding.de**:

1. Root-Nameserver liefert Delegation zu DeNIC für **.de**

```
de.                     IN  NS  A.NIC.de.
de.                     IN  NS  F.NIC.de.
...
```

2. DeNIC liefert Delegation für **.lug-erding.de**:

```
lug-erding.de.      IN  NS  ns1.partnerpanel.de.
lug-erding.de.      IN  NS  sns1.partnerpanel.de.
```

3. **ns1.partnerpanel.de** liefert das Ergebnis zurück:

```
www.lug-erding.de. IN  A   89.110.147.240
```

iterative Namensauflösung

iterative Auflösung von z.B. **www.lug-erding.de**:

1. Root-Nameserver liefert Delegation zu DeNIC für **.de**

```
de.                     IN  NS  A.NIC.de.
de.                     IN  NS  F.NIC.de.
...
```

2. DeNIC liefert Delegation für **.lug-erding.de**:

```
lug-erding.de.      IN  NS  ns1.partnerpanel.de.
lug-erding.de.      IN  NS  sns1.partnerpanel.de.
```

3. **ns1.partnerpanel.de** liefert das Ergebnis zurück:

```
www.lug-erding.de. IN  A   89.110.147.240
```

iterative Namensauflösung

iterative Auflösung von z.B. **www.lug-erding.de**:

1. Root-Nameserver liefert Delegation zu DeNIC für **.de**

```
de.                     IN  NS  A.NIC.de.
de.                     IN  NS  F.NIC.de.
...
```

2. DeNIC liefert Delegation für **.lug-erding.de**:

```
lug-erding.de.      IN  NS  ns1.partnerpanel.de.
lug-erding.de.      IN  NS  sns1.partnerpanel.de.
```

3. **ns1.partnerpanel.de** liefert das Ergebnis zurück:

```
www.lug-erding.de. IN  A   89.110.147.240
```

iterative Namensauflösung

iterative Auflösung von z.B. **www.lug-erding.de**:

1. Root-Nameserver liefert Delegation zu DeNIC für **.de**

```
de.                     IN  NS  A.NIC.de.
de.                     IN  NS  F.NIC.de.
...
```

2. DeNIC liefert Delegation für **.lug-erding.de**:

```
lug-erding.de.      IN  NS  ns1.partnerpanel.de.
lug-erding.de.      IN  NS  sns1.partnerpanel.de.
```

3. **ns1.partnerpanel.de** liefert das Ergebnis zurück:

```
www.lug-erding.de. IN  A   89.110.147.240
```

iterative Namensauflösung

- Verfahren für einfache Resolver viel zu aufwendig!
- Lösung: rekursive Anfragen an **einen** Nameserver

$\Longrightarrow$ Problem zum Nameserver verlagert!

rekursive Namensauflösung

1. Resolver befragt *seinen* Nameserver aus `/etc/resolv.conf`
2. weiß dieser die Antwort, so liefert er diese aus
3. wenn nicht: Nameserver startet iterativen Anfrage
4. der Nameserver liefert die Antwort aus
5. Caching der erhaltenen Informationen inklusive Delegationen ist sehr wichtig!

rekursive Namensauflösung

1. Resolver befragt *seinen* Nameserver aus `/etc/resolv.conf`
2. weiß dieser die Antwort, so liefert er diese aus
3. wenn nicht: Nameserver startet iterativen Anfrage
4. der Nameserver liefert die Antwort aus
5. Caching der erhaltenen Informationen inklusive Delegationen ist sehr wichtig!

rekursive Namensauflösung

1. Resolver befragt *seinen* Nameserver aus `/etc/resolv.conf`
2. weiß dieser die Antwort, so liefert er diese aus
3. wenn nicht: Nameserver startet iterativen Anfrage
4. der Nameserver liefert die Antwort aus
5. Caching der erhaltenen Informationen inklusive Delegationen ist sehr wichtig!

rekursive Namensauflösung

1. Resolver befragt *seinen* Nameserver aus `/etc/resolv.conf`
2. weiß dieser die Antwort, so liefert er diese aus
3. wenn nicht: Nameserver startet iterativen Anfrage
4. der Nameserver liefert die Antwort aus
5. Caching der erhaltenen Informationen inklusive Delegationen ist sehr wichtig!

rekursive Namensauflösung

1. Resolver befragt *seinen* Nameserver aus `/etc/resolv.conf`
2. weiß dieser die Antwort, so liefert er diese aus
3. wenn nicht: Nameserver startet iterativen Anfrage
4. der Nameserver liefert die Antwort aus
5. Caching der erhaltenen Informationen inklusive Delegationen ist sehr wichtig!

reverse DNS

- DNS: Zuordnung von Namen → IP-Adresse
- reverse DNS: Zuordnung von IP-Adressen → Namen
- Problem: Es existiert keine direkte Kopplung!
- Eintrag in Domain **lug-erding.de** bewirkt noch kein funktionierendes reverse DNS für diesen Namen
- Lösung: eigene Domain für reverse DNS!

reverse DNS

- DNS: Zuordnung von Namen → IP-Adresse
- reverse DNS: Zuordnung von IP-Adressen → Namen
- Problem: Es existiert keine direkte Kopplung!
- Eintrag in Domain **lug-erding.de** bewirkt noch kein funktionierendes reverse DNS für diesen Namen
- Lösung: eigene Domain für reverse DNS!

reverse DNS

- DNS: Zuordnung von Namen → IP-Adresse
- reverse DNS: Zuordnung von IP-Adressen → Namen
- Problem: Es existiert keine direkte Kopplung!
- Eintrag in Domain **lug-erding.de** bewirkt noch kein funktionierendes **reverse** DNS für diesen Namen
- Lösung: eigene Domain für reverse DNS!

reverse DNS

- DNS: Zuordnung von Namen → IP-Adresse
- reverse DNS: Zuordnung von IP-Adressen → Namen
- Problem: Es existiert keine direkte Kopplung!
- Eintrag in Domain **lug-erding.de** bewirkt noch kein funktionierendes reverse DNS für diesen Namen
- Lösung: eigene Domain für reverse DNS!

reverse DNS

- Domain für IP-mapping: in-addr.arpa
- erneutes Problem: Aufteilung?
- DNS-Namen von *rechts* nach *links*:
  de → lug-erding → www
- IP-Adressen von *links* nach *rechts*:
  89 → 110 → 147 → 240
- Lösung: Umdrehen der IP-Blöcke!

reverse DNS

- Domain für IP-mapping: in-addr.arpa
- erneutes Problem: Aufteilung?
- DNS-Namen von *rechts* nach *links*:
  de → lug-erding → www
- IP-Adressen von *links* nach *rechts*:
  89 → 110 → 147 → 240
- Lösung: Umdrehen der IP-Blöcke!

reverse DNS

- Domain für IP-mapping: in-addr.arpa
- erneutes Problem: Aufteilung?
- DNS-Namen von *rechts* nach *links*:
  de $\rightarrow$ lug-erding $\rightarrow$ www
- IP-Adressen von *links* nach *rechts*:
  89 $\longrightarrow$ 110 $\longrightarrow$ 147 $\longrightarrow$ 240
- Lösung: Umdrehen der IP-Blöcke!

reverse DNS

- Domain für IP-mapping: in-addr.arpa
- erneutes Problem: Aufteilung?
- DNS-Namen von *rechts* nach *links*:
  de $\rightarrow$ lug-erding $\rightarrow$ www
- IP-Adressen von *links* nach *rechts*:
  89 $\longrightarrow$ 110 $\longrightarrow$ 147 $\longrightarrow$ 240
- Lösung: Umdrehen der IP-Blöcke!

Beispiel: reverse DNS

- Finden des Nameservers für 89.110.147.240

```
147.110.89.in-addr.arpa. IN NS ns1.netcl.de.
147.110.89.in-addr.arpa. IN NS sns5.netcl.de.
147.110.89.in-addr.arpa. IN NS ns10.netcl.de.
```

- diese Nameserver liefern die Antwort

```
240.147.110.89.in-addr.arpa. IN \
                          PTR mail.lug-erding.de.
```

- erneutes Problem: Blockgrenzen!
  $\Longrightarrow$ **classless Delegation**

Beispiel: reverse DNS

- Finden des Nameservers für 89.110.147.240

```
147.110.89.in-addr.arpa. IN NS ns1.netcl.de.
147.110.89.in-addr.arpa. IN NS sns5.netcl.de.
147.110.89.in-addr.arpa. IN NS ns10.netcl.de.
```

- diese Nameserver liefern die Antwort

```
240.147.110.89.in-addr.arpa. IN \
                              PTR mail.lug-erding.de.
```

- erneutes Problem: Blockgrenzen!

  $\Longrightarrow$ **classless Delegation**

Beispiel: reverse DNS

- Finden des Nameservers für 89.110.147.240

```
147.110.89.in-addr.arpa. IN NS ns1.netcl.de.
147.110.89.in-addr.arpa. IN NS sns5.netcl.de.
147.110.89.in-addr.arpa. IN NS ns10.netcl.de.
```

- diese Nameserver liefern die Antwort

```
240.147.110.89.in-addr.arpa. IN \
                              PTR mail.lug-erding.de.
```

- erneutes Problem: Blockgrenzen!

  $\Longrightarrow$ classless Delegation

Beispiel: reverse DNS

- Finden des Nameservers für 89.110.147.240

```
147.110.89.in-addr.arpa. IN NS ns1.netcl.de.
147.110.89.in-addr.arpa. IN NS sns5.netcl.de.
147.110.89.in-addr.arpa. IN NS ns10.netcl.de.
```

- diese Nameserver liefern die Antwort

```
240.147.110.89.in-addr.arpa. IN \
                              PTR mail.lug-erding.de.
```

- erneutes Problem: Blockgrenzen!
  $\Longrightarrow$ **classless Delegation**

Abkürzungen

Ein **Record** bezeichnet einen Eintrag in der Zonendatei
Alle Einträge zu DNS-Eintrag heißen **Resource-Record** (**RR**)
Einige Typen aus Zonendateien:

- **SOA**: **S**tart **O**f **A**uthorization
- **NS**: zuständiger Nameserver, Delegation
- **A**: Address-Record, liefert IP-Adresse
- **PTR**: liefert reversen Eintrag, d.h. Namen zu IP-Adresse
- **CNAME**: **c**anonical **name**
- **MX**: **m**ail e**x**changer, Mailserver
- ...

TTL

Die **TTL**, **T**ime **t**o **L**ive gibt die Dauer an wie lange ein Nameserver einen gültigen Eintrag zwischenspeichern (cachen) darf.

- bei aktuellen BIND Versionen wird diese über $TTL angegeben.
- ältere verwenden den 5. Wert aus der **SOA**

SOA

SOA, **S**tart **o**f **A**uthorization, erster Eintrag in Zonendatei.
Für die LUG-Erding gilt:

```
lug-erding.de.  IN  SOA  ns1.partnerpanel.de.  \
   hostmaster.lug-erding.de.  \
   2006100401 10000 1800 604800 480
```

1. Domainname
2. **IN**: Internet-Klasse
3. **SOA**: Eintrag ist vom Typ SOA
4. Nameserver: Der primäre Nameserver der Domain
5. Mailadresse: Wer ist für die Domain zuständig (@ → .)?

SOA

SOA, **S**tart **o**f **A**uthorization, erster Eintrag in Zonendatei.
Für die LUG-Erding gilt:

```
lug-erding.de.  IN  SOA  ns1.partnerpanel.de.  \
   hostmaster.lug-erding.de.  \
   2006100401 10000 1800 604800 480
```

1. Domainname
2. **IN**: Internet-Klasse
3. **SOA**: Eintrag ist vom Typ SOA
4. Nameserver: Der primäre Nameserver der Domain
5. Mailadresse: Wer ist für die Domain zuständig (@ → .)?

SOA

SOA, **S**tart **o**f **A**uthorization, erster Eintrag in Zonendatei.
Für die LUG-Erding gilt:

```
lug-erding.de.  IN  SOA  ns1.partnerpanel.de.  \
   hostmaster.lug-erding.de.  \
   2006100401 10000 1800 604800 480
```

1. Domainname
2. **IN**: Internet-Klasse
3. **SOA**: Eintrag ist vom Typ SOA
4. Nameserver: Der primäre Nameserver der Domain
5. Mailadresse: Wer ist für die Domain zuständig (@ → .)?

SOA

SOA, **S**tart **o**f **A**uthorization, erster Eintrag in Zonendatei.
Für die LUG-Erding gilt:

```
lug-erding.de.  IN  SOA  ns1.partnerpanel.de.  \
   hostmaster.lug-erding.de.  \
   2006100401 10000 1800 604800 480
```

1. Domainname
2. **IN**: Internet-Klasse
3. **SOA**: Eintrag ist vom Typ SOA
4. Nameserver: Der primäre Nameserver der Domain
5. Mailadresse: Wer ist für die Domain zuständig (@ → .)?

SOA

SOA, **S**tart **o**f **A**uthorization, erster Eintrag in Zonendatei.
Für die LUG-Erding gilt:

```
lug-erding.de.  IN  SOA  ns1.partnerpanel.de.  \
   hostmaster.lug-erding.de.  \
   2006100401 10000 1800 604800 480
```

1. Domainname
2. **IN**: Internet-Klasse
3. **SOA**: Eintrag ist vom Typ SOA
4. Nameserver: Der primäre Nameserver der Domain
5. Mailadresse: Wer ist für die Domain zuständig (@ → .)?

SOA

SOA, **S**tart **o**f **A**uthorization, erster Eintrag in Zonendatei.
Für die LUG-Erding gilt:

```
lug-erding.de.  IN  SOA  ns1.partnerpanel.de.  \
   hostmaster.lug-erding.de.  \
   2006100401 10000 1800 604800 480
```

1. Domainname
2. **IN**: Internet-Klasse
3. **SOA**: Eintrag ist vom Typ SOA
4. Nameserver: Der primäre Nameserver der Domain
5. Mailadresse: Wer ist für die Domain zuständig (@ → .)?

SOA in BIND-Zonendatei

Das Gleiche in BIND-Syntax:

```
lug-erding.de.  IN  SOA  ns1.partnerpanel.de.  \
   hostmaster.lug-erding.de.  (
   2006100401 ; serial
        10000 ; refresh
         1800 ; retry
       604800 ; expire
          480 ; negative TTL
  )
```

SOA-Werte

Die Zahlenwerte in **SOA**, **S**tart **o**f **A**uthorization, bezeichnen:

1. **serial**: Wird bei jeder Änderung der Daten erhöht.
2. **refresh**: Wann soll ein Slave die **serial** prüfen?
3. **retry**: Wann erneut versuchen wenn **refresh** fehlschlägt?
4. **expire**: Ab hier sind alle Werte ungültig
5. **neg. TTL**: Wie lange dürfen nicht-existierende Einträge im Cache bleiben?

SOA-Werte

Die Zahlenwerte in **SOA**, **S**tart **o**f **A**uthorization, bezeichnen:

1. **serial**: Wird bei jeder Änderung der Daten erhöht.
2. **refresh**: Wann soll ein Slave die **serial** prüfen?
3. **retry**: Wann erneut versuchen wenn **refresh** fehlschlägt?
4. **expire**: Ab hier sind alle Werte ungültig
5. **neg. TTL**: Wie lange dürfen nicht-existierende Einträge im Cache bleiben?

SOA-Werte

Die Zahlenwerte in **SOA**, **S**tart **o**f **A**uthorization, bezeichnen:

1. **serial**: Wird bei jeder Änderung der Daten erhöht.
2. **refresh**: Wann soll ein Slave die **serial** prüfen?
3. **retry**: Wann erneut versuchen wenn **refresh** fehlschlägt?
4. **expire**: Ab hier sind alle Werte ungültig
5. **neg. TTL**: Wie lange dürfen nicht-existierende Einträge im Cache bleiben?

SOA-Werte

Die Zahlenwerte in **SOA**, **S**tart **o**f **A**uthorization, bezeichnen:

1. **serial**: Wird bei jeder Änderung der Daten erhöht.
2. **refresh**: Wann soll ein Slave die **serial** prüfen?
3. **retry**: Wann erneut versuchen wenn **refresh** fehlschlägt?
4. **expire**: Ab hier sind alle Werte ungültig
5. **neg. TTL**: Wie lange dürfen nicht-existierende Einträge im Cache bleiben?

SOA-Werte

Die Zahlenwerte in **SOA**, **S**tart **o**f **A**uthorization, bezeichnen:

1. **serial**: Wird bei jeder Änderung der Daten erhöht.
2. **refresh**: Wann soll ein Slave die **serial** prüfen?
3. **retry**: Wann erneut versuchen wenn **refresh** fehlschlägt?
4. **expire**: Ab hier sind alle Werte ungültig
5. **neg. TTL**: Wie lange dürfen nicht-existierende Einträge im Cache bleiben?

Nameserver bei Delegationen

- klassisch:

```
lug-erding.de.     IN  NS  ns1.partnerpanel.de.
lug-erding.de.     IN  NS  sns1.partnerpanel.de.
```

- Bei BIND geht es kürzer:

```
@      IN  NS  ns1.partnerpanel.de.
@      IN  NS  sns1.partnerpanel.de.
```

- und noch kürzer:

```
@      IN  NS  ns1.partnerpanel.de.
       IN  NS  sns1.partnerpanel.de.
```

Nameserver bei Delegationen

- klassisch:

```
lug-erding.de.     IN  NS  ns1.partnerpanel.de.
lug-erding.de.     IN  NS  sns1.partnerpanel.de.
```

- Bei BIND geht es kürzer:

```
@      IN  NS  ns1.partnerpanel.de.
@      IN  NS  sns1.partnerpanel.de.
```

- und noch kürzer:

```
@      IN  NS  ns1.partnerpanel.de.
       IN  NS  sns1.partnerpanel.de.
```

Nameserver bei Delegationen

- klassisch:

```
lug-erding.de.     IN  NS  ns1.partnerpanel.de.
lug-erding.de.     IN  NS  sns1.partnerpanel.de.
```

- Bei BIND geht es kürzer:

```
@      IN  NS  ns1.partnerpanel.de.
@      IN  NS  sns1.partnerpanel.de.
```

- und noch kürzer:

```
@      IN  NS  ns1.partnerpanel.de.
       IN  NS  sns1.partnerpanel.de.
```

Adresseinträge

Resource-Records:

- **A**ddress-Records:

```
www       IN   A    89.110.147.240
mail      IN   A    89.110.147.240
@         IN   A    89.110.147.240
```

- **C**anonical **name**:

```
www.mglug     IN    CNAME   www.mglug.de
```

- **DNAME** bildet komplette Zone via **CNAME** ab
- reverse Einträge (**p**oin**t**e**r**):

```
  240     IN    PTR    mail.lug-erding.de
```

(steht in der Zone 147.110.89.in-addr.arpa)

Adresseinträge

Resource-Records:

- **A**ddress-Records:

```
www       IN   A    89.110.147.240
mail      IN   A    89.110.147.240
@         IN   A    89.110.147.240
```

- **C**anonical **name**:

```
www.mglug      IN    CNAME   www.mglug.de
```

- **DNAME** bildet komplette Zone via **CNAME** ab
- reverse Einträge (**p**oin**t**e**r**):

```
240      IN    PTR    mail.lug-erding.de
```

(steht in der Zone 147.110.89.in-addr.arpa)

Adresseinträge

Resource-Records:

- **A**ddress-Records:

```
www       IN   A    89.110.147.240
mail      IN   A    89.110.147.240
@         IN   A    89.110.147.240
```

- **C**anonical **name**:

```
www.mglug      IN    CNAME   www.mglug.de
```

- **DNAME** bildet komplette Zone via **CNAME** ab
- reverse Einträge (**p**oin**t**e**r**):

```
240      IN    PTR    mail.lug-erding.de
```

(steht in der Zone 147.110.89.in-addr.arpa)

Adresseinträge

Resource-Records:

- **A**ddress-Records:

```
www       IN   A    89.110.147.240
mail      IN   A    89.110.147.240
@         IN   A    89.110.147.240
```

- **C**anonical **name**:

```
www.mglug      IN    CNAME   www.mglug.de
```

- **DNAME** bildet komplette Zone via **CNAME** ab
- reverse Einträge (**p**oin**t**e**r**):

```
  240      IN    PTR    mail.lug-erding.de
```

(steht in der Zone 147.110.89.in-addr.arpa)

Klassen für Zonendaten

- **IN**: Internet
- **CS**: CSNet
- **CH**: Chaos, BIND-Versionsnummer und Autoren
- **HS**: Hesiod

Typen für Zonendaten

- **A**: **A**dresseneintrag, IP-Adresse
- **CNAME**: **c**anonischer **Name**, Alias
- **HINFO**: **h**ost **info**, zusätzliche Angaben zum System
- **MX**: **m**ail e**x**changer, wohin soll E-Mail gesendet werden
- **NS**: **N**ame**s**erver
- **PTR**: reverse Einträge (**p**oin**t**e**r**)
- **SOA**: **S**tart **o**f **A**uthorization
- **TXT**: zusätzlicher **Text**eintrag
- **WKS**: **W**ell **K**nown **S**ervices, gibt Serverdienste an

neue Typen für Zonendaten

- **AFSDB**: **A**ndrew **F**ile **S**ystem **D**ata **B**ase, experimentell
- **ISDN**: **I**ntegrated **S**ervices **D**igital **N**etwork, experimentell
- **RP**: **R**esponsible **P**erson, experimentell **II**
- **RT**: **R**oute **T**hrough, experimentell
- **X25**: X.25 Adressen, experimentell
- **PX**: X.400 / RFC Abbildungen
- **AAAA**: IPv6 Adressen
- **SRV**: Lokalisierung von **S**e**rv**erdiensten
- **NAPTR**: **N**aming **A**uthority **P**oin**t**e**r**

Besonderheiten

- localhost-Adresse (127.0.0.1) sollte in jeder Zone existieren
- Reverse-Zone für Loopback sollte jeder Nameserver selber bereitstellen
- jeder Server benötigt Hints-Datei:
  - Liste der Root-Nameserver (BIND: `db.cache`)
  - zu beziehen via FTP: `ftp.rs.internic.net` (198.41.0.6)
  - Aktualisierung: Manuell!
  - BIND-Version 9 hat die Liste eincompiliert!

Besonderheiten

- localhost-Adresse (127.0.0.1) sollte in jeder Zone existieren
- Reverse-Zone für Loopback sollte jeder Nameserver selber bereitstellen
- jeder Server benötigt Hints-Datei:
  - Liste der Root-Nameserver (BIND: `db.cache`)
  - zu beziehen via FTP: `ftp.rs.internic.net` (198.41.0.6)
  - Aktualisierung: Manuell!
  - BIND-Version 9 hat die Liste eincompiliert!

Besonderheiten

- localhost-Adresse (127.0.0.1) sollte in jeder Zone existieren
- Reverse-Zone für Loopback sollte jeder Nameserver selber bereitstellen
- jeder Server benötigt Hints-Datei:
  - Liste der Root-Nameserver (BIND: `db.cache`)
  - zu beziehen via FTP: `ftp.rs.internic.net` (198.41.0.6)
  - Aktualisierung: Manuell!
  - BIND-Version 9 hat die Liste eincompiliert!

DNS-Format

Eine DNS-Nachricht besteht aus 5 Teilen:

1. Header: rekursiv, authoritative, truncation, inverse query,. . .
2. Query, ursprüngliche Frage
3. Answer, passend zur Frage
4. Authority, welcher Server ist zuständig?
5. Additional, z.B. IP-Adressen der zuständigen Nameserver

DNS-Format

Eine DNS-Nachricht besteht aus 5 Teilen:

1. Header: rekursiv, authoritative, truncation, inverse query,. . .
2. Query, ursprüngliche Frage
3. Answer, passend zur Frage
4. Authority, welcher Server ist zuständig?
5. Additional, z.B. IP-Adressen der zuständigen Nameserver

DNS-Format

Eine DNS-Nachricht besteht aus 5 Teilen:

1. Header: rekursiv, authoritative, truncation, inverse query,. . .
2. Query, ursprüngliche Frage
3. Answer, passend zur Frage
4. Authority, welcher Server ist zuständig?
5. Additional, z.B. IP-Adressen der zuständigen Nameserver

DNS-Format

Eine DNS-Nachricht besteht aus 5 Teilen:

1. Header: rekursiv, authoritative, truncation, inverse query,. . .
2. Query, ursprüngliche Frage
3. Answer, passend zur Frage
4. Authority, welcher Server ist zuständig?
5. Additional, z.B. IP-Adressen der zuständigen Nameserver

DNS-Format

Eine DNS-Nachricht besteht aus 5 Teilen:

1. Header: rekursiv, authoritative, truncation, inverse query,. . .
2. Query, ursprüngliche Frage
3. Answer, passend zur Frage
4. Authority, welcher Server ist zuständig?
5. Additional, z.B. IP-Adressen der zuständigen Nameserver

Besonderheiten aufgrund des Netzwerkes

- Maximale Größe für Host- bzw. Domainanteil ist 63 Bytes
- Maximale Größe eines DNS–UDP-Paketes sind 512 Bytes (Ausnahme: EDNS0)
- Kompression ist möglich, Verwendung der Endung anderer Einträge
  $\longrightarrow$ 13 Root-Nameserver alle in gleicher Domain!
- normale DNS-Requests erfolgen via UDP Port 53
- Zonenabgleiche erfolgen via TCP Port 53

Besonderheiten aufgrund des Netzwerkes

- Maximale Größe für Host- bzw. Domainanteil ist 63 Bytes
- Maximale Größe eines DNS–**UDP**-Paketes sind 512 Bytes (Ausnahme: EDNS0)
- Kompression ist möglich, Verwendung der Endung anderer Einträge
  $\longrightarrow$ 13 Root-Nameserver alle in gleicher Domain!
- normale DNS-Requests erfolgen via UDP Port 53
- Zonenabgleiche erfolgen via TCP Port 53

Besonderheiten aufgrund des Netzwerkes

- Maximale Größe für Host- bzw. Domainanteil ist 63 Bytes
- Maximale Größe eines DNS–UDP-Paketes sind 512 Bytes (Ausnahme: EDNS0)
- Kompression ist möglich, Verwendung der Endung anderer Einträge
  $\longrightarrow$ 13 Root-Nameserver alle in gleicher Domain!
- normale DNS-Requests erfolgen via UDP Port 53
- Zonenabgleiche erfolgen via TCP Port 53

Besonderheiten aufgrund des Netzwerkes

- Maximale Größe für Host- bzw. Domainanteil ist 63 Bytes
- Maximale Größe eines DNS–UDP-Paketes sind 512 Bytes (Ausnahme: EDNS0)
- Kompression ist möglich, Verwendung der Endung anderer Einträge
  $\longrightarrow$ 13 Root-Nameserver alle in gleicher Domain!
- normale DNS-Requests erfolgen via UDP Port 53
- Zonenabgleiche erfolgen via TCP Port 53

Besonderheiten aufgrund des Netzwerkes

- Maximale Größe für Host- bzw. Domainanteil ist 63 Bytes
- Maximale Größe eines DNS–UDP-Paketes sind 512 Bytes (Ausnahme: EDNS0)
- Kompression ist möglich, Verwendung der Endung anderer Einträge
  $\longrightarrow$ 13 Root-Nameserver alle in gleicher Domain!
- normale DNS-Requests erfolgen via UDP Port 53
- Zonenabgleiche erfolgen via TCP Port 53

Finden von Mailservern

- E-Mails werden an **MX**-Record für den Domainnamen versendet:

```
lug-erding.de.    IN    MX  10 mail.lug-erding.de
```

- es werden Prioritäten (hier 10) vergeben, niedrigste Zahl bevorzugt
- Mailserver mit höherem Wert lagern E-Mails nur zwischen (**relaying**)
- existiert kein **MX** so wird an den **A**-Record gesendet
- es sind keine **CNAME** für den **MX**-Record erlaubt!

Finden von Mailservern

- E-Mails werden an **MX**-Record für den Domainnamen versendet:

  ```
  lug-erding.de.    IN    MX  10 mail.lug-erding.de
  ```

- es werden Prioritäten (hier 10) vergeben, niedrigste Zahl bevorzugt
- Mailserver mit höherem Wert lagern E-Mails nur zwischen (**relaying**)
- existiert kein **MX** so wird an den **A**-Record gesendet
- es sind keine **CNAME** für den **MX**-Record erlaubt!

Finden von Mailservern

- E-Mails werden an **MX**-Record für den Domainnamen versendet:

  ```
  lug-erding.de.   IN    MX  10 mail.lug-erding.de
  ```

- es werden Prioritäten (hier 10) vergeben, niedrigste Zahl bevorzugt
- Mailserver mit höherem Wert lagern E-Mails nur zwischen (**relaying**)
- existiert kein **MX** so wird an den **A**-Record gesendet
- es sind keine **CNAME** für den **MX**-Record erlaubt!

Finden von Mailservern

- E-Mails werden an **MX**-Record für den Domainnamen versendet:

  ```
  lug-erding.de.   IN    MX  10 mail.lug-erding.de
  ```

- es werden Prioritäten (hier 10) vergeben, niedrigste Zahl bevorzugt
- Mailserver mit höherem Wert lagern E-Mails nur zwischen (**relaying**)
- existiert kein **MX** so wird an den **A**-Record gesendet
- es sind keine **CNAME** für den **MX**-Record erlaubt!

Finden von Mailservern

- E-Mails werden an **MX**-Record für den Domainnamen versendet:

  ```
  lug-erding.de.   IN    MX  10 mail.lug-erding.de
  ```

- es werden Prioritäten (hier 10) vergeben, niedrigste Zahl bevorzugt
- Mailserver mit höherem Wert lagern E-Mails nur zwischen (**relaying**)
- existiert kein **MX** so wird an den **A**-Record gesendet
- es sind keine **CNAME** für den **MX**-Record erlaubt!

Sender Policy Framework, Spambekämpfung

- **SPF**: Möglichkeit über DNS festzulegen wer E-Mails einer Domain versenden darf
- wird als **TXT**-Typ implementiert:

```
oreilly.com. IN TXT  \
   "v=spf1 mx ptr include:oreillynet.com -all"
```

- **v=spf1**: markiert einen SPF-Eintrag (Version 1)
- **mx**: alle MX-Records der sendenden Domain sind erlaubt
- **ptr**: PTR-Record muß existieren
- **include**: es gelten auch die SPF-Werte der zusätzlichen Domain
- **-all**: alle anderen Server dürfen keine E-Mails mit dem Absender `@oreilly.com` versenden

Sender Policy Framework, Spambekämpfung

- **SPF**: Möglichkeit über DNS festzulegen wer E-Mails einer Domain versenden darf
- wird als **TXT**-Typ implementiert:

```
oreilly.com. IN TXT  \
  "v=spf1 mx ptr include:oreillynet.com -all"
```

- **v=spf1**: markiert einen SPF-Eintrag (Version 1)
- **mx**: alle MX-Records der sendenden Domain sind erlaubt
- **ptr**: PTR-Record muß existieren
- **include**: es gelten auch die SPF-Werte der zusätzlichen Domain
- **-all**: alle anderen Server dürfen keine E-Mails mit dem Absender `@oreilly.com` versenden

Sender Policy Framework, Spambekämpfung

- **SPF**: Möglichkeit über DNS festzulegen wer E-Mails einer Domain versenden darf
- wird als **TXT**-Typ implementiert:

```
oreilly.com. IN TXT  \
  "v=spf1 mx ptr include:oreillynet.com -all"
```

- **v=spf1**: markiert einen SPF-Eintrag (Version 1)
- **mx**: alle MX-Records der sendenden Domain sind erlaubt
- **ptr**: PTR-Record muß existieren
- **include**: es gelten auch die SPF-Werte der zusätzlichen Domain
- **-all**: alle anderen Server dürfen keine E-Mails mit dem Absender `@oreilly.com` versenden

Sender Policy Framework, Spambekämpfung

- **SPF**: Möglichkeit über DNS festzulegen wer E-Mails einer Domain versenden darf
- wird als **TXT**-Typ implementiert:

```
oreilly.com. IN TXT  \
  "v=spf1 mx ptr include:oreillynet.com -all"
```

- **v=spf1**: markiert einen SPF-Eintrag (Version 1)
- **mx**: alle MX-Records der sendenden Domain sind erlaubt
- **ptr**: PTR-Record muß existieren
- **include**: es gelten auch die SPF-Werte der zusätzlichen Domain
- **-all**: alle anderen Server dürfen keine E-Mails mit dem Absender `@oreilly.com` versenden

Sender Policy Framework, Spambekämpfung

- **SPF**: Möglichkeit über DNS festzulegen wer E-Mails einer Domain versenden darf
- wird als **TXT**-Typ implementiert:

```
oreilly.com. IN TXT  \
  "v=spf1 mx ptr include:oreillynet.com -all"
```

- **v=spf1**: markiert einen SPF-Eintrag (Version 1)
- **mx**: alle MX-Records der sendenden Domain sind erlaubt
- **ptr**: PTR-Record muß existieren
- **include**: es gelten auch die SPF-Werte der zusätzlichen Domain
- **-all**: alle anderen Server dürfen keine E-Mails mit dem Absender `@oreilly.com` versenden

Sender Policy Framework, Spambekämpfung

- **SPF**: Möglichkeit über DNS festzulegen wer E-Mails einer Domain versenden darf
- wird als **TXT**-Typ implementiert:

```
oreilly.com. IN TXT  \
  "v=spf1 mx ptr include:oreillynet.com -all"
```

- **v=spf1**: markiert einen SPF-Eintrag (Version 1)
- **mx**: alle MX-Records der sendenden Domain sind erlaubt
- **ptr**: PTR-Record muß existieren
- **include**: es gelten auch die SPF-Werte der zusätzlichen Domain
- **-all**: alle anderen Server dürfen keine E-Mails mit dem Absender `@oreilly.com` versenden

Sender Policy Framework, Spambekämpfung

- **SPF**: Möglichkeit über DNS festzulegen wer E-Mails einer Domain versenden darf
- wird als **TXT**-Typ implementiert:

```
oreilly.com. IN TXT  \
  "v=spf1 mx ptr include:oreillynet.com -all"
```

- **v=spf1**: markiert einen SPF-Eintrag (Version 1)
- **mx**: alle MX-Records der sendenden Domain sind erlaubt
- **ptr**: PTR-Record muß existieren
- **include**: es gelten auch die SPF-Werte der zusätzlichen Domain
- **-all**: alle anderen Server dürfen keine E-Mails mit dem Absender `@oreilly.com` versenden

Sender Policy Framework, Spambekämpfung

Mögliche Qualifier:

- **+**: erlaubt für Mailversand
- **-**: verboten für Mailversand
- **~**: weiches Verbot, Mailserver sollte E-Mails prüfen
- **?**: neutral, kein Effekt

Sender Policy Framework, Spambekämpfung

Mögliche Mechanismen:

- **a**: Domainnamen die erlaubt sind zu senden
- **mx**: MX für Domain dürfen versenden
- **ip4**: betrifft IPv4-Adressen
- **ip6**: betrifft IPv6-Adressen
- **ptr**: PTR-Record muß existieren und auf Domain mappen

Sender Policy Framework, Spambekämpfung

Probleme:

- setzt funktionierendes DNS voraus
- relaying über andere Server nur bedingt möglich
- Absender muß SPF im DNS einbauen
- Empfänger muß SPF überprüfen

Sender Policy Framework, Spambekämpfung

Probleme:

- setzt funktionierendes DNS voraus
- relaying über andere Server nur bedingt möglich
- Absender muß SPF im DNS einbauen
- Empfänger muß SPF überprüfen

Sender Policy Framework, Spambekämpfung

Probleme:

- setzt funktionierendes DNS voraus
- relaying über andere Server nur bedingt möglich
- Absender muß SPF im DNS einbauen
- Empfänger muß SPF überprüfen

Sender Policy Framework, Spambekämpfung

Probleme:

- setzt funktionierendes DNS voraus
- relaying über andere Server nur bedingt möglich
- Absender muß SPF im DNS einbauen
- Empfänger muß SPF überprüfen

Allgemeines

- Adresslänge ist 128 Bits (IPv4: **32** Bits)
- 340282366920938463463374607431768211456 bzw. $3.4 * 10^{38}$ Adressen
- Schreibweise in 4er Blöcken getrennt durch Doppelpunkte: 2001:db80:0123:4567:89ab:cdef:0123:4567
- führende Nullen können weggelassen werden
- Abkürzung nachfolgender 0er-Blöcke durch **::**
- Loopbackadresse in IPv6: **::1**

Allgemeines

- Adresslänge ist 128 Bits (IPv4: 32 Bits)
- 340282366920938463463374607431768211456 bzw. $3.4 * 10^{38}$ Adressen
- Schreibweise in 4er Blöcken getrennt durch Doppelpunkte: 2001:db80:0123:4567:89ab:cdef:0123:4567
- führende Nullen können weggelassen werden
- Abkürzung nachfolgender 0er-Blöcke durch **::**
- Loopbackadresse in IPv6: **::1**

Allgemeines

- Adresslänge ist 128 Bits (IPv4: 32 Bits)
- 340282366920938463463374607431768211456 bzw. $3.4 * 10^{38}$ Adressen
- Schreibweise in 4er Blöcken getrennt durch Doppelpunkte: 2001:db80:0123:4567:89ab:cdef:0123:4567
- führende Nullen können weggelassen werden
- Abkürzung nachfolgender 0er-Blöcke durch **::**
- Loopbackadresse in IPv6: **::1**

Allgemeines

- Adresslänge ist 128 Bits (IPv4: 32 Bits)
- 340282366920938463463374607431768211456 bzw. $3.4 * 10^{38}$ Adressen
- Schreibweise in 4er Blöcken getrennt durch Doppelpunkte: 2001:db80:0123:4567:89ab:cdef:0123:4567
- führende Nullen können weggelassen werden
- Abkürzung nachfolgender 0er-Blöcke durch **::**
- Loopbackadresse in IPv6: **::1**

Allgemeines

- Adresslänge ist 128 Bits (IPv4: 32 Bits)
- 340282366920938463463374607431768211456 bzw. $3.4 * 10^{38}$ Adressen
- Schreibweise in 4er Blöcken getrennt durch Doppelpunkte: 2001:db80:0123:4567:89ab:cdef:0123:4567
- führende Nullen können weggelassen werden
- Abkürzung nachfolgender 0er-Blöcke durch **::**
- Loopbackadresse in IPv6: **::1**

Allgemeines

- Adresslänge ist 128 Bits (IPv4: 32 Bits)
- 340282366920938463463374607431768211456 bzw. $3.4 * 10^{38}$ Adressen
- Schreibweise in 4er Blöcken getrennt durch Doppelpunkte: 2001:db80:0123:4567:89ab:cdef:0123:4567
- führende Nullen können weggelassen werden
- Abkürzung nachfolgender 0er-Blöcke durch **::**
- Loopbackadresse in IPv6: **::1**

Aufteilung

- Aufteilung der Adresse in 3 Blöcke:
  1. global routing prefix
  2. subnet ID
  3. interface ID
- **EDNS0**: **E**xtended mechanism for **DNS**, version **0**
  $\longrightarrow$ Möglichkeit UDP-Pakete größer 512 Bytes zu verwenden

Aufteilung

- Aufteilung der Adresse in 3 Blöcke:
  1. global routing prefix
  2. subnet ID
  3. interface ID
- **EDNS0**: **E**xtended mechanism for **DNS**, version **0**
  $\longrightarrow$ Möglichkeit UDP-Pakete größer 512 Bytes zu verwenden

1. Ansatz

- **A**-Record hat 32-bit Adressen $\longrightarrow$ **AAAA**-Record für IPv6:

```
ipv6-host IN AAAA 2001:db80:1:2:3:4:567:89ab
```

- reverse mapping via **ip6.int**, jetzt **ip6.arpa**:

```
b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2...ip6.arpa.\
                                IN PTR ipv6-host
```

- **AAAA** ist das alte und wieder aktuelle Verfahren für IPv6

1. Ansatz

- **A**-Record hat 32-bit Adressen $\longrightarrow$ **AAAA**-Record für IPv6:

```
ipv6-host IN AAAA 2001:db80:1:2:3:4:567:89ab
```

- reverse mapping via **ip6.int**, jetzt **ip6.arpa**:

```
b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2...ip6.arpa.\
                                    IN PTR ipv6-host
```

- **AAAA** ist das alte und wieder aktuelle Verfahren für IPv6

1. Ansatz

- **A**-Record hat 32-bit Adressen $\longrightarrow$ **AAAA**-Record für IPv6:

```
ipv6-host IN AAAA 2001:db80:1:2:3:4:567:89ab
```

- reverse mapping via **ip6.int**, jetzt **ip6.arpa**:

```
b.a.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2...ip6.arpa.\
                                    IN PTR ipv6-host
```

- **AAAA** ist das alte und wieder aktuelle Verfahren für IPv6

2. Ansatz

- soll den Adresswechsel von ISPs erleichtern
- **A6**-Records spezifizieren nur einen Teil:

```
$ORIGIN movie.edu.
drunkenmaster   IN   A6   64  \
  ::0210:4bff:fe10:0d24 subnet1.v6.movie.edu.
```

- 64 gibt an, daß die ersten 64 Bits fehlen
- diese werden durch `subnet1.v6.movie.edu` bestimmt:

```
$ORIGIN v6.movie.edu.
subnet1 IN A6 48 0:0:0:1:: movie-u.isp-a.net.
subnet1 IN A6 48 0:0:0:1:: movie-u.isp-b.net.
```

2. Ansatz

- soll den Adresswechsel von ISPs erleichtern
- **A6**-Records spezifizieren nur einen Teil:

```
$ORIGIN movie.edu.
drunkenmaster   IN   A6    64  \
  ::0210:4bff:fe10:0d24 subnet1.v6.movie.edu.
```

- 64 gibt an, daß die ersten 64 Bits fehlen
- diese werden durch `subnet1.v6.movie.edu` bestimmt:

```
$ORIGIN v6.movie.edu.
subnet1 IN A6 48 0:0:0:1:: movie-u.isp-a.net.
subnet1 IN A6 48 0:0:0:1:: movie-u.isp-b.net.
```

2. Ansatz

- soll den Adresswechsel von ISPs erleichtern
- **A6**-Records spezifizieren nur einen Teil:

```
$ORIGIN movie.edu.
drunkenmaster   IN   A6   64   \
  ::0210:4bff:fe10:0d24 subnet1.v6.movie.edu.
```

- 64 gibt an, daß die ersten 64 Bits fehlen
- diese werden durch `subnet1.v6.movie.edu` bestimmt:

```
$ORIGIN v6.movie.edu.
subnet1 IN A6 48 0:0:0:1:: movie-u.isp-a.net.
subnet1 IN A6 48 0:0:0:1:: movie-u.isp-b.net.
```

2. Ansatz

- soll den Adresswechsel von ISPs erleichtern
- **A6**-Records spezifizieren nur einen Teil:

```
$ORIGIN movie.edu.
drunkenmaster   IN   A6   64  \
  ::0210:4bff:fe10:0d24 subnet1.v6.movie.edu.
```

- 64 gibt an, daß die ersten 64 Bits fehlen
- diese werden durch `subnet1.v6.movie.edu` bestimmt:

```
$ORIGIN v6.movie.edu.
subnet1 IN A6 48 0:0:0:1:: movie-u.isp-a.net.
subnet1 IN A6 48 0:0:0:1:: movie-u.isp-b.net.
```

2. Ansatz

- hier sind 2 ISPs verbunden
- der erste ISP hat z.B.:

```
$ORIGIN isp-a.net.
movie-u   IN   A6   40 0:0:21:: isp-a.rir-1.net.
```

- dieser hat seine Registrar-Adresse:

```
$ORIGIN rir-1.net.
isp-a IN A6 36 0:0:0500:: rir2.top-level-v6.net.
```

- und der Toplevel-Registrar hat:

```
$ORIGIN top-level.v6.net.
rir2  IN   A6     0 2001:db80::2
```

2. Ansatz

- hier sind 2 ISPs verbunden
- der erste ISP hat z.B.:

```
$ORIGIN isp-a.net.
movie-u   IN   A6   40 0:0:21:: isp-a.rir-1.net.
```

- dieser hat seine Registrar-Adresse:

```
$ORIGIN rir-1.net.
isp-a IN A6 36 0:0:0500:: rir2.top-level-v6.net.
```

- und der Toplevel-Registrar hat:

```
$ORIGIN top-level.v6.net.
rir2  IN   A6     0 2001:db80::2
```

2. Ansatz

- hier sind 2 ISPs verbunden
- der erste ISP hat z.B.:

```
$ORIGIN isp-a.net.
movie-u   IN   A6   40 0:0:21:: isp-a.rir-1.net.
```

- dieser hat seine Registrar-Adresse:

```
$ORIGIN rir-1.net.
isp-a IN A6 36 0:0:0500:: rir2.top-level-v6.net.
```

- und der Toplevel-Registrar hat:

```
$ORIGIN top-level.v6.net.
rir2  IN   A6     0 2001:db80::2
```

2. Ansatz

- hier sind 2 ISPs verbunden
- der erste ISP hat z.B.:

```
$ORIGIN isp-a.net.
movie-u   IN   A6   40 0:0:21:: isp-a.rir-1.net.
```

- dieser hat seine Registrar-Adresse:

```
$ORIGIN rir-1.net.
isp-a IN A6 36 0:0:0500:: rir2.top-level-v6.net.
```

- und der Toplevel-Registrar hat:

```
$ORIGIN top-level.v6.net.
rir2  IN   A6     0 2001:db80::2
```

aktueller Stand

- der zweite Ansatz konnte sehr viele DNS-Anfragen bedeuten
- dieser wurde als zu aufwendig verworfen
- die erste Variante ist wieder aktuell
- allerdings mit **ip6.arpa** statt **ip6.int**

Entstehung von BIND

- die **B**erkeley **I**nternet **N**ame **D**omain Software entstand als *graduate student project* in Berkeley
- wurde von **DARPA** gesponsert
- weiterentwickelt bei **DEC** (Paul Vixie)
- May 1997: BIND Version 8 erschien als Produktionsversion
- September 2000: Bind Version 9 erschien, fast ein kompletter rewrite der Software
- wird vom **I**nternet **S**ystem **C**onsortium vertrieben

Entstehung von BIND

- die **B**erkeley **I**nternet **N**ame **D**omain Software entstand als *graduate student project* in Berkeley
- wurde von **DARPA** gesponsert
- weiterentwickelt bei **DEC** (Paul Vixie)
- May 1997: BIND Version 8 erschien als Produktionsversion
- September 2000: Bind Version 9 erschien, fast ein kompletter rewrite der Software
- wird vom **I**nternet **S**ystem **C**onsortium vertrieben

Entstehung von BIND

- die **B**erkeley **I**nternet **N**ame **D**omain Software entstand als *graduate student project* in Berkeley
- wurde von **DARPA** gesponsert
- weiterentwickelt bei **DEC** (Paul Vixie)
- May 1997: BIND Version 8 erschien als Produktionsversion
- September 2000: Bind Version 9 erschien, fast ein kompletter rewrite der Software
- wird vom **I**nternet **S**ystem **C**onsortium vertrieben

Entstehung von BIND

- die **B**erkeley **I**nternet **N**ame **D**omain Software entstand als *graduate student project* in Berkeley
- wurde von **DARPA** gesponsert
- weiterentwickelt bei **DEC** (Paul Vixie)
- May 1997: BIND Version 8 erschien als Produktionsversion
- September 2000: Bind Version 9 erschien, fast ein kompletter rewrite der Software
- wird vom **I**nternet **S**ystem **C**onsortium vertrieben

Konfigurationsdatei

- Typischer Name ist `named.conf`
- Optionen können global oder für Zonen gelten (BIND 9)
- globale Optionen:

```
options { directory “/var/named”; };
```

- Zoneneinträge

```
zone “lug-erding.de” in {
 type master;
 file “db.lug-erding.de”;
 };
```

Konfigurationsdatei

- Typischer Name ist `named.conf`
- Optionen können global oder für Zonen gelten (BIND 9)
- globale Optionen:

```
options { directory “/var/named”; };
```

- Zoneneinträge

```
zone “lug-erding.de” in {
 type master;
 file “db.lug-erding.de”;
 };
```

Konfigurationsdatei

- Typischer Name ist `named.conf`
- Optionen können global oder für Zonen gelten (BIND 9)
- globale Optionen:

```
options { directory “/var/named”; };
```

- Zoneneinträge

```
zone “lug-erding.de” in {
 type master;
 file “db.lug-erding.de”;
 };
```

Konfigurationsdatei

- Typischer Name ist `named.conf`
- Optionen können global oder für Zonen gelten (BIND 9)
- globale Optionen:

```
options { directory “/var/named”; };
```

- Zoneneinträge

```
zone “lug-erding.de” in {
 type master;
 file “db.lug-erding.de”;
 };
```

Slaveserver

- Eintrag für einen slave-Nameserver:

```
zone “mglug.de” in {
 type slave;
 file “SEC.mglug.de”;
 masters { ns1.croup.de; ns2.croup.de; };
 };
```

- mehrere Server sind möglich
- Abgleich der Zonendateien muß erlaubt sein!

Slaveserver

- Eintrag für einen slave-Nameserver:

```
zone “mglug.de” in {
 type slave;
 file “SEC.mglug.de”;
 masters { ns1.croup.de; ns2.croup.de; };
 };
```

- mehrere Server sind möglich
- Abgleich der Zonendateien muß erlaubt sein!

Slaveserver

- Eintrag für einen slave-Nameserver:

```
zone “mglug.de” in {
 type slave;
 file “SEC.mglug.de”;
 masters { ns1.croup.de; ns2.croup.de; };
 };
```

- mehrere Server sind möglich
- Abgleich der Zonendateien muß erlaubt sein!

Resolverdetails

- Verhalten bestimmt durch `/etc/resolv.conf`
- bis zu 3 Nameserver-Einträge (`nameserver`)
- Suchliste (`search`) kann verwendet werden
- Domainname (`domain`) ist mittlerweile obsolet
- Adressen können nach Präferenzen sortiert werden (`sortlist`)
- viele Optionen (`options`)

Resolverdetails

- Verhalten bestimmt durch `/etc/resolv.conf`
- bis zu 3 Nameserver-Einträge (`nameserver`)
- Suchliste (`search`) kann verwendet werden
- Domainname (`domain`) ist mittlerweile obsolet
- Adressen können nach Präferenzen sortiert werden (`sortlist`)
- viele Optionen (`options`)

Resolverdetails

- Verhalten bestimmt durch `/etc/resolv.conf`
- bis zu 3 Nameserver-Einträge (`nameserver`)
- Suchliste (`search`) kann verwendet werden
- Domainname (`domain`) ist mittlerweile obsolet
- Adressen können nach Präferenzen sortiert werden (`sortlist`)
- viele Optionen (`options`)

Resolverdetails

- Verhalten bestimmt durch `/etc/resolv.conf`
- bis zu 3 Nameserver-Einträge (`nameserver`)
- Suchliste (`search`) kann verwendet werden
- Domainname (`domain`) ist mittlerweile obsolet
- Adressen können nach Präferenzen sortiert werden (`sortlist`)
- viele Optionen (`options`)

Resolverdetails

- Verhalten bestimmt durch `/etc/resolv.conf`
- bis zu 3 Nameserver-Einträge (`nameserver`)
- Suchliste (`search`) kann verwendet werden
- Domainname (`domain`) ist mittlerweile obsolet
- Adressen können nach Präferenzen sortiert werden (`sortlist`)
- viele Optionen (`options`)

Resolverdetails

- Verhalten bestimmt durch `/etc/resolv.conf`
- bis zu 3 Nameserver-Einträge (`nameserver`)
- Suchliste (`search`) kann verwendet werden
- Domainname (`domain`) ist mittlerweile obsolet
- Adressen können nach Präferenzen sortiert werden (`sortlist`)
- viele Optionen (`options`)

Optionen

- **debug**
- **ndots**:n - ab n Punkten im Namen wird dieser zuerst versucht, default 1
- **timeout**:n - Der Timeout in Sekunden, default 5
- **attempts**:n - Zahl der Versuche
- **rotate** - wechselweise Verwendung der Nameserver
- **no-check-names** - keinen Namenschecks vor der Abfrage
- **inet6** - zuerst IPv6 (AAAA) Anfragen versuchen

Timeouts des Resolvers

Resolver-Timeouts ab BIND 8.2.1

	Nameserver		
retry	1	2	3
0	5s	(2x) 5s	(3x) 5s
1	10s	(2x) 5s	(3x) 3s
total	15s	20s	24s

Der Timeout wird verdoppelt und durch die Zahl der Nameserver dividiert.

nslookup

- eigene resolver-Bibliothek, nicht die des Systems!
- nur 1 Nameserver kann befragt werden
- Zonentransfer möglich
- interaktiver und nicht-interaktiver Modus

nslookup

- eigene resolver-Bibliothek, nicht die des Systems!
- nur 1 Nameserver kann befragt werden
- Zonentransfer möglich
- interaktiver und nicht-interaktiver Modus

nslookup

- eigene resolver-Bibliothek, nicht die des Systems!
- nur 1 Nameserver kann befragt werden
- Zonentransfer möglich
- interaktiver und nicht-interaktiver Modus

nslookup

- eigene resolver-Bibliothek, nicht die des Systems!
- nur 1 Nameserver kann befragt werden
- Zonentransfer möglich
- interaktiver und nicht-interaktiver Modus

nslookup – Optionen

- Optionen werden durch ein Minus-Zeichen auf der Kommandozeile angegeben (nicht-interaktiver Modus)
- oder via **set** (interaktiver Modus)
- Optionen können durch vorgestelltes **no** deaktiviert werden
- Zonentransfer ist mit **ls -d** *Zonenname* möglich

nslookup – Optionen

- **all** – zeige alle Werte
- **class**=IN
- **[no]debug**
- **[no]d2** – erweitertes debugging
- **[no]search** – Anwendung der Suchliste
- **[no]recursive**
- **[no]vc** – verwende TCP statt UDP (**v**irtual **c**ircuit)

nslookup – Optionen

- **port**=53 – setzen des Zielports
- **querytype**=A
- **type**=A
- **retry**=2
- **timeout**=5
- **server**=a.root-dns.net – Setzen des Nameservers

dig

- **D**omain **I**nformation **G**roper (reverse engineered acronym)
- keine searchlist
- zeigt alle Sektionen an, Frage, Antwort, authority, additional
- einfacher zu bedienen
- Beispiel:
  dig @majestix.physik.home ns www.lug-erding.de
- erkennt den Typen recht zuverlässig
- für reverse DNS muß die Option **-x** verwendet werden

dig

- **D**omain **I**nformation **G**roper (reverse engineered acronym)
- keine searchlist
- zeigt alle Sektionen an, Frage, Antwort, authority, additional
- einfacher zu bedienen
- Beispiel:
  dig @majestix.physik.home ns www.lug-erding.de
- erkennt den Typen recht zuverlässig
- für reverse DNS muß die Option -x verwendet werden

dig

- **D**omain **I**nformation **G**roper (reverse engineered acronym)
- keine searchlist
- zeigt alle Sektionen an, Frage, Antwort, authority, additional
- einfacher zu bedienen
- Beispiel:
  dig @majestix.physik.home ns www.lug-erding.de
- erkennt den Typen recht zuverlässig
- für reverse DNS muß die Option -x verwendet werden

dig

- **D**omain **I**nformation **G**roper (reverse engineered acronym)
- keine searchlist
- zeigt alle Sektionen an, Frage, Antwort, authority, additional
- einfacher zu bedienen
- Beispiel:
  dig @majestix.physik.home ns www.lug-erding.de
- erkennt den Typen recht zuverlässig
- für reverse DNS muß die Option -x verwendet werden

dig

- **D**omain **I**nformation **G**roper (reverse engineered acronym)
- keine searchlist
- zeigt alle Sektionen an, Frage, Antwort, authority, additional
- einfacher zu bedienen
- Beispiel:
  dig @majestix.physik.home ns www.lug-erding.de
- erkennt den Typen recht zuverlässig
- für reverse DNS muß die Option **-x** verwendet werden

dig

- **D**omain **I**nformation **G**roper (reverse engineered acronym)
- keine searchlist
- zeigt alle Sektionen an, Frage, Antwort, authority, additional
- einfacher zu bedienen
- Beispiel:
  dig @majestix.physik.home ns www.lug-erding.de
- erkennt den Typen recht zuverlässig
- für reverse DNS muß die Option -x verwendet werden

dig

- **D**omain **I**nformation **G**roper (reverse engineered acronym)
- keine searchlist
- zeigt alle Sektionen an, Frage, Antwort, authority, additional
- einfacher zu bedienen
- Beispiel:
  dig @majestix.physik.home ns www.lug-erding.de
- erkennt den Typen recht zuverlässig
- für reverse DNS muß die Option **-x** verwendet werden

TSIG

- **t**ransaction **Sig**nature, one-way hash Funktion mit shared secret
- dient der Absicherung von Zonentransfers
- und dynamischen Zonenupdates (dyn-DNS)

DNSSec

- public-key Verfahren
- chain of trust via parent
- benötigt **EDNS** wegen größerer Pakete
- Root-Nameserver unterstützen kein **DNSSec**
- Einzige Länderdomain ist Schweden, **se**
- $\Longrightarrow$ Performanceproblem!

DNSSec

- public-key Verfahren
- chain of trust via parent
- benötigt **EDNS** wegen größerer Pakete
- Root-Nameserver unterstützen kein **DNSSec**
- Einzige Länderdomain ist Schweden, **se**
- $\Longrightarrow$ Performanceproblem!

DNSSec

- public-key Verfahren
- chain of trust via parent
- benötigt **EDNS** wegen größerer Pakete
- Root-Nameserver unterstützen kein **DNSSec**
- Einzige Länderdomain ist Schweden, **se**
- $\Longrightarrow$ Performanceproblem!

DNSSec

- public-key Verfahren
- chain of trust via parent
- benötigt **EDNS** wegen größerer Pakete
- Root-Nameserver unterstützen kein **DNSSec**
- Einzige Länderdomain ist Schweden, **se**
- $\Longrightarrow$ Performanceproblem!

DNSSec

- public-key Verfahren
- chain of trust via parent
- benötigt **EDNS** wegen größerer Pakete
- Root-Nameserver unterstützen kein **DNSSec**
- Einzige Länderdomain ist Schweden, **se**
- $\Longrightarrow$ Performanceproblem!

DNSSec

- public-key Verfahren
- chain of trust via parent
- benötigt **EDNS** wegen größerer Pakete
- Root-Nameserver unterstützen kein **DNSSec**
- Einzige Länderdomain ist Schweden, **se**
- $\Longrightarrow$ Performanceproblem!

BIND Optionen zur Sicherheit

- **allow-transfer**: wer darf eine Zone beziehen
- **allow-recursion**: Einschränkung wer rekursive Anfragen stellen darf
- **forwarders, forward-only**: Anfragen werden an Upstream Nameserver weitergeleitet → **Firewall**
- BIND Version 9 erlaubt auch **forwarders** für einzelne Zonen → interne Nameserver
- **views**: verschiedene Zoneneinträge bei unterschiedlichen Sourcen

BIND Optionen zur Sicherheit

- **allow-transfer**: wer darf eine Zone beziehen
- **allow-recursion**: Einschränkung wer rekursive Anfragen stellen darf
- **forwarders, forward-only**: Anfragen werden an Upstream Nameserver weitergeleitet → **Firewall**
- BIND Version 9 erlaubt auch **forwarders** für einzelne Zonen → interne Nameserver
- **views**: verschiedene Zoneneinträge bei unterschiedlichen Sourcen

BIND Optionen zur Sicherheit

- **allow-transfer**: wer darf eine Zone beziehen
- **allow-recursion**: Einschränkung wer rekursive Anfragen stellen darf
- **forwarders, forward-only**: Anfragen werden an Upstream Nameserver weitergeleitet → **Firewall**
- BIND Version 9 erlaubt auch **forwarders** für einzelne Zonen → interne Nameserver
- **views**: verschiedene Zoneneinträge bei unterschiedlichen Sourcen

BIND Optionen zur Sicherheit

- **allow-transfer**: wer darf eine Zone beziehen
- **allow-recursion**: Einschränkung wer rekursive Anfragen stellen darf
- **forwarders, forward-only**: Anfragen werden an Upstream Nameserver weitergeleitet → **Firewall**
- BIND Version 9 erlaubt auch **forwarders** für einzelne Zonen → **interne** Nameserver
- **views**: verschiedene Zoneneinträge bei unterschiedlichen Sourcen

BIND Optionen zur Sicherheit

- **allow-transfer**: wer darf eine Zone beziehen
- **allow-recursion**: Einschränkung wer rekursive Anfragen stellen darf
- **forwarders, forward-only**: Anfragen werden an Upstream Nameserver weitergeleitet $\rightarrow$ **Firewall**
- BIND Version 9 erlaubt auch **forwarders** für einzelne Zonen $\rightarrow$ interne Nameserver
- **views**: verschiedene Zoneneinträge bei unterschiedlichen Sourcen

BIND Optionen zur Sicherheit

- **query-source-address**: festlegen der Absendeparameter $\longrightarrow$ **Firewall**
- **-t** Verzeichnis: chroot() in Verzeichnis nach dem Start
- **-u** user: setuid() zu user nachdem keine priveligierten Rechte mehr benötigt werden. Achtung: Schreibrechte für slave und dynamisches DNS!

BIND Optionen zur Sicherheit

- **query-source-address**: festlegen der Absendeparameter $\longrightarrow$ **Firewall**
- **-t** Verzeichnis: chroot() in Verzeichnis nach dem Start
- **-u** user: setuid() zu user nachdem keine priveligierten Rechte mehr benötigt werden. Achtung: Schreibrechte für slave und dynamisches DNS!

BIND Optionen zur Sicherheit

- **query-source-address**: festlegen der Absendeparameter → **Firewall**
- **-t** Verzeichnis: chroot() in Verzeichnis nach dem Start
- **-u** user: setuid() zu user nachdem keine priveligierten Rechte mehr benötigt werden. Achtung: Schreibrechte für slave und dynamisches DNS!

Classless Delegation

- Delegation bei reverse-DNS nur als Class-C Netz (256 Adressen)
- oder eigene reverse Domain pro Adresse
- Trick: Verwendung von **CNAME**s
- Beispiel: 0-63 aus Class-C Netz 10.1.2.0/24

Classless Delegation

- Delegation bei reverse-DNS nur als Class-C Netz (256 Adressen)
- oder eigene reverse Domain pro Adresse
- Trick: Verwendung von **CNAME**s
- Beispiel: 0-63 aus Class-C Netz 10.1.2.0/24

Classless Delegation

- Delegation bei reverse-DNS nur als Class-C Netz (256 Adressen)
- oder eigene reverse Domain pro Adresse
- Trick: Verwendung von **CNAME**s
- Beispiel: 0-63 aus Class-C Netz 10.1.2.0/24

Classless Delegation

- Delegation bei reverse-DNS nur als Class-C Netz (256 Adressen)
- oder eigene reverse Domain pro Adresse
- Trick: Verwendung von **CNAME**s
- Beispiel: 0-63 aus Class-C Netz 10.1.2.0/24

Beispiel: Delegation von 64 Adressen

- Datei db.10.1.2:

```
1.2.1.10  IN CNAME 1.0-63.2.1.10.in-addr.arpa.
2.2.1.10  IN CNAME 2.0-63.2.1.10.in-addr.arpa.
...
63.2.1.10 IN CNAME 63.0-63.2.1.10.in-addr.arpa.

0-63.1.10 IN NS ns1.foo.com.
          IN NS ns2.foo.com.
```

- Dadurch ist wieder ein normaler Zoneneintrag möglich

Beispiel: Delegation von 64 Adressen

- delegierte Zone kann normale **PTR**-Records vornehmen
- Datei db.10.1.2.0-63:

```
@    IN    SOA ...
     IN    NS ns1.foo.com.
     IN    NS ns2.foo.com.

1    IN     PTR    host1.foo.com.
2    IN     PTR    host2.foo.com.
...
```

Internationale Domains — Domains mit Umlauten

- **ACE**: **A**SCII **c**ompatible **e**ncoding
- Puny-Code mit Delta-Kodierung (RFC 3490)
- spezieller Prefix: **xn**−−
- Umlaute werden weggelassen und punycodiert angehängt
- Umwandlung Umlaute in Puny-Code durch Applikation
- Darstellung in Unicode
- Beispiel: www.etwas−ähnlich.de
  ⟶ www.xn−−etwas−hnlich−lcb.de

Internationale Domains — Domains mit Umlauten

- **ACE**: **A**SCII **c**ompatible **e**ncoding
- Puny-Code mit Delta-Kodierung (RFC 3490)
- spezieller Prefix: **xn**−−
- Umlaute werden weggelassen und punycodiert angehängt
- Umwandlung Umlaute in Puny-Code durch Applikation
- Darstellung in Unicode
- Beispiel: www.etwas−ähnlich.de
  ⟶ www.xn−−etwas−hnlich−lcb.de

Internationale Domains — Domains mit Umlauten

- **ACE**: **A**SCII **c**ompatible **e**ncoding
- Puny-Code mit Delta-Kodierung (RFC 3490)
- spezieller Prefix: **xn−−**
- Umlaute werden weggelassen und punycodiert angehängt
- Umwandlung Umlaute in Puny-Code durch Applikation
- Darstellung in Unicode
- Beispiel: www.etwas−ähnlich.de
  $\longrightarrow$ www.xn−−etwas−hnlich−lcb.de

Internationale Domains — Domains mit Umlauten

- **ACE**: **A**SCII **c**ompatible **e**ncoding
- Puny-Code mit Delta-Kodierung (RFC 3490)
- spezieller Prefix: **xn**−−
- Umlaute werden weggelassen und punycodiert angehängt
- Umwandlung Umlaute in Puny-Code durch Applikation
- Darstellung in Unicode
- Beispiel: www.etwas−ähnlich.de
  ⟶ www.xn−−etwas−hnlich−lcb.de

Internationale Domains — Domains mit Umlauten

- **ACE**: **A**SCII **c**ompatible **e**ncoding
- Puny-Code mit Delta-Kodierung (RFC 3490)
- spezieller Prefix: **xn**−−
- Umlaute werden weggelassen und punycodiert angehängt
- Umwandlung Umlaute in Puny-Code durch Applikation
- Darstellung in Unicode
- Beispiel: www.etwas−ähnlich.de
  ⟶ www.xn−−etwas−hnlich−lcb.de

Internationale Domains — Domains mit Umlauten

- **ACE**: **A**SCII **c**ompatible **e**ncoding
- Puny-Code mit Delta-Kodierung (RFC 3490)
- spezieller Prefix: **xn**−−
- Umlaute werden weggelassen und punycodiert angehängt
- Umwandlung Umlaute in Puny-Code durch Applikation
- Darstellung in Unicode
- Beispiel: www.etwas−ähnlich.de
  ⟶ www.xn−−etwas−hnlich−lcb.de

Internationale Domains — Domains mit Umlauten

- **ACE**: **A**SCII **c**ompatible **e**ncoding
- Puny-Code mit Delta-Kodierung (RFC 3490)
- spezieller Prefix: **xn**−−
- Umlaute werden weggelassen und punycodiert angehängt
- Umwandlung Umlaute in Puny-Code durch Applikation
- Darstellung in Unicode
- Beispiel: www.etwas−ähnlich.de
  $\longrightarrow$ www.xn−−etwas−hnlich−lcb.de

dynamisches DNS

- dynamische Updates sind möglich, meist über **DHCP**-Server
- Eintrag auch in anderen (höheren) Nameservern möglich
- Absicherung notwendig (**TSIG**)!
- Nameserver legt Journaldateien (**.jnl**) an
- diese wachsen stetig
- inkrementelle Zonentransfers sind möglich

dynamisches DNS

- dynamische Updates sind möglich, meist über **DHCP**-Server
- Eintrag auch in anderen (höheren) Nameservern möglich
- Absicherung notwendig (**TSIG**)!
- Nameserver legt Journaldateien (**.jnl**) an
- diese wachsen stetig
- inkrementelle Zonentransfers sind möglich

dynamisches DNS

- dynamische Updates sind möglich, meist über **DHCP**-Server
- Eintrag auch in anderen (höheren) Nameservern möglich
- Absicherung notwendig (**TSIG**)!
- Nameserver legt Journaldateien (**.jnl**) an
- diese wachsen stetig
- inkrementelle Zonentransfers sind möglich

dynamisches DNS

- dynamische Updates sind möglich, meist über **DHCP**-Server
- Eintrag auch in anderen (höheren) Nameservern möglich
- Absicherung notwendig (**TSIG**)!
- Nameserver legt Journaldateien (**.jnl**) an
- diese wachsen stetig
- inkrementelle Zonentransfers sind möglich

dynamisches DNS

- dynamische Updates sind möglich, meist über **DHCP**-Server
- Eintrag auch in anderen (höheren) Nameservern möglich
- Absicherung notwendig (**TSIG**)!
- Nameserver legt Journaldateien (**.jnl**) an
- diese wachsen stetig
- inkrementelle Zonentransfers sind möglich

dynamisches DNS

- dynamische Updates sind möglich, meist über **DHCP**-Server
- Eintrag auch in anderen (höheren) Nameservern möglich
- Absicherung notwendig (**TSIG**)!
- Nameserver legt Journaldateien (**.jnl**) an
- diese wachsen stetig
- inkrementelle Zonentransfers sind möglich

Sonstiges zu BIND/DNS

- **Notify**: Benachrichtigung bei Änderung der **serial** Nummer
- **RoundRobin**: wechseln der Reihenfolge der Einträge
- **sortlist**: bevorzugte Antworten bei Fragen aus bestimmten Netzen
- **Multiple CNAMEs**: möglich aber nicht ratsam (**CNAME** auf **CNAME**)
- **ndc, rndc**: Tool zur Kommunikation mit **named**

Sonstiges zu BIND/DNS

- **Notify**: Benachrichtigung bei Änderung der **serial** Nummer
- **RoundRobin**: wechseln der Reihenfolge der Einträge
- **sortlist**: bevorzugte Antworten bei Fragen aus bestimmten Netzen
- **Multiple CNAMEs**: möglich aber nicht ratsam (**CNAME** auf **CNAME**)
- **ndc, rndc**: Tool zur Kommunikation mit **named**

Sonstiges zu BIND/DNS

- **Notify**: Benachrichtigung bei Änderung der **serial** Nummer
- **RoundRobin**: wechseln der Reihenfolge der Einträge
- **sortlist**: bevorzugte Antworten bei Fragen aus bestimmten Netzen
- **Multiple CNAMEs**: möglich aber nicht ratsam (**CNAME** auf **CNAME**)
- **ndc, rndc**: Tool zur Kommunikation mit **named**

Sonstiges zu BIND/DNS

- **Notify**: Benachrichtigung bei Änderung der **serial** Nummer
- **RoundRobin**: wechseln der Reihenfolge der Einträge
- **sortlist**: bevorzugte Antworten bei Fragen aus bestimmten Netzen
- **Multiple CNAMEs**: möglich aber nicht ratsam (**CNAME** auf **CNAME**)
- **ndc, rndc**: Tool zur Kommunikation mit **named**

Sonstiges zu BIND/DNS

- **Notify**: Benachrichtigung bei Änderung der **serial** Nummer
- **RoundRobin**: wechseln der Reihenfolge der Einträge
- **sortlist**: bevorzugte Antworten bei Fragen aus bestimmten Netzen
- **Multiple CNAMEs**: möglich aber nicht ratsam (**CNAME** auf **CNAME**)
- **ndc, rndc**: Tool zur Kommunikation mit **named**

named Debugging

- Option -d <Debuglevel>
- **syslog**, facility `daemon`
- Option -g: named bleibt im Vordergrund und output nach **stderr**
- Kommando: **rndc trace** <Debuglevel>
- Kommando: **rndc dumpdb -all**, dumpen der Datenbank

typische Fehler

- Seriennummer nicht erhöht
- **reload** vergessen
- IP-Adresswechsel
  → Slaves bekommen keine neuen Zonendaten
  → Zonendaten expiren!
- Zonentransfers mit **dig** oder **nslookup** testen

typische Fehler

- Seriennummer nicht erhöht
- **reload** vergessen
- IP-Adresswechsel
  → Slaves bekommen keine neuen Zonendaten
  → Zonendaten expiren!
- Zonentransfers mit **dig** oder **nslookup** testen

typische Fehler

- Seriennummer nicht erhöht
- **reload** vergessen
- IP-Adresswechsel
  → Slaves bekommen keine neuen Zonendaten
  → Zonendaten expiren!
- Zonentransfers mit **dig** oder **nslookup** testen

typische Fehler

- Seriennummer nicht erhöht
- **reload** vergessen
- IP-Adresswechsel
  → Slaves bekommen keine neuen Zonendaten
  → Zonendaten expiren!
- Zonentransfers mit **dig** oder **nslookup** testen

typische Fehler

- Seriennummer nicht erhöht
- **reload** vergessen
- IP-Adresswechsel
  → Slaves bekommen keine neuen Zonendaten
  → Zonendaten expiren!
- Zonentransfers mit **dig** oder **nslookup** testen

typische Fehler

- Seriennummer nicht erhöht
- **reload** vergessen
- IP-Adresswechsel
  $\longrightarrow$ Slaves bekommen keine neuen Zonendaten
  $\longrightarrow$ Zonendaten expiren!
- Zonentransfers mit **dig** oder **nslookup** testen

typische Fehler

- PTR-Eintrag fehlt $\longrightarrow$ Probleme z.B. mit E-Mail oder FTP
- Fehler in der Konfigurations- oder Zonendatei $\longrightarrow$ **syslog**
- abschließender Punkt in der Zone vergessen $\longrightarrow$ doppelte Domainendung wie z.B. **www.lug-erding.de.lug-erding.de**
- fehlende root-hints Datei
  $\longrightarrow$ Nameserver funktioniert nur für eigene Zonen
  $\longrightarrow$ syslog: `No root servers for class IN`

typische Fehler

- PTR-Eintrag fehlt $\longrightarrow$ Probleme z.B. mit E-Mail oder FTP
- Fehler in der Konfigurations- oder Zonendatei $\longrightarrow$ **syslog**
- abschließender Punkt in der Zone vergessen $\longrightarrow$ doppelte Domainendung wie z.B.
  www.lug-erding.de.lug-erding.de
- fehlende root-hints Datei
  $\longrightarrow$ Nameserver funktioniert nur für eigene Zonen
  $\longrightarrow$ syslog: `No root servers for class IN`

typische Fehler

- PTR-Eintrag fehlt $\longrightarrow$ Probleme z.B. mit E-Mail oder FTP
- Fehler in der Konfigurations- oder Zonendatei $\longrightarrow$ **syslog**
- abschließender Punkt in der Zone vergessen $\longrightarrow$ doppelte Domainendung wie z.B. **www.lug-erding.de.lug-erding.de**
- fehlende root-hints Datei
  $\longrightarrow$ Nameserver funktioniert nur für eigene Zonen
  $\longrightarrow$ syslog: `No root servers for class IN`

typische Fehler

- PTR-Eintrag fehlt $\longrightarrow$ Probleme z.B. mit E-Mail oder FTP
- Fehler in der Konfigurations- oder Zonendatei $\longrightarrow$ **syslog**
- abschließender Punkt in der Zone vergessen $\longrightarrow$ doppelte Domainendung wie z.B. **www.lug-erding.de.lug-erding.de**
- fehlende root-hints Datei
  $\longrightarrow$ Nameserver funktioniert nur für eigene Zonen
  $\longrightarrow$ syslog: `No root servers for class IN`

typische Fehler

- PTR-Eintrag fehlt $\longrightarrow$ Probleme z.B. mit E-Mail oder FTP
- Fehler in der Konfigurations- oder Zonendatei $\longrightarrow$ **syslog**
- abschließender Punkt in der Zone vergessen $\longrightarrow$ doppelte Domainendung wie z.B. **www.lug-erding.de.lug-erding.de**
- fehlende root-hints Datei
  $\longrightarrow$ Nameserver funktioniert nur für eigene Zonen
  $\longrightarrow$ syslog: `No root servers for class IN`

typische Fehler

- PTR-Eintrag fehlt $\longrightarrow$ Probleme z.B. mit E-Mail oder FTP
- Fehler in der Konfigurations- oder Zonendatei $\longrightarrow$ **syslog**
- abschließender Punkt in der Zone vergessen $\longrightarrow$ doppelte Domainendung wie z.B. **www.lug-erding.de.lug-erding.de**
- fehlende root-hints Datei
  $\longrightarrow$ Nameserver funktioniert nur für eigene Zonen
  $\longrightarrow$ syslog: `No root servers for class IN`

typische Fehler

- Netzwerk unterbrochen
  $\longrightarrow$ `request ...timed out`
  $\longrightarrow$ Test mit **ping** oder **tcpdump** (**ICMP unreachable**)
- fehlende Delegation, d.h. der authoritative Nameserver antwortet und der parent behauptet `Non-existent Domain`
- fehlerhafte Delegation $\longrightarrow$ `lame server`, der delegierte Server ist nicht zuständig.

typische Fehler

- Netzwerk unterbrochen
  $\longrightarrow$ `request ...timed out`
  $\longrightarrow$ Test mit **ping** oder **tcpdump** (**ICMP unreachable**)
- fehlende Delegation, d.h. der authoritative Nameserver antwortet und der parent behauptet `Non-existent Domain`
- fehlerhafte Delegation $\longrightarrow$ `lame server`, der delegierte Server ist nicht zuständig.

typische Fehler

- Netzwerk unterbrochen
  $\longrightarrow$ `request ...timed out`
  $\longrightarrow$ Test mit **ping** oder **tcpdump** (**ICMP unreachable**)
- fehlende Delegation, d.h. der authoritative Nameserver antwortet und der parent behauptet `Non-existent Domain`
- fehlerhafte Delegation $\longrightarrow$ `lame server`, der delegierte Server ist nicht zuständig.

typische Fehler

- Netzwerk unterbrochen
  $\longrightarrow$ `request ...timed out`
  $\longrightarrow$ Test mit **ping** oder **tcpdump** (**ICMP unreachable**)
- fehlende Delegation, d.h. der authoritative Nameserver antwortet und der parent behauptet `Non-existent Domain`
- fehlerhafte Delegation $\longrightarrow$ `lame server`, der delegierte Server ist nicht zuständig.

typische Fehler

- Netzwerk unterbrochen
  $\longrightarrow$ `request ...timed out`
  $\longrightarrow$ Test mit **ping** oder **tcpdump** (**ICMP unreachable**)
- fehlende Delegation, d.h. der authoritative Nameserver antwortet und der parent behauptet `Non-existent Domain`
- fehlerhafte Delegation $\longrightarrow$ `lame server`, der delegierte Server ist nicht zuständig.

typische Fehler

- Fehler in resolv.conf
  $\longrightarrow$ Test mit **nslookup** und der Option `set all`
- fehlende **search**list, kein Domainname gesetzt?
- Antwort von unbekannter Quelle
  (`response from unexpected source`)
  $\longrightarrow$ Antwort von falschem Interface
  $\longrightarrow$ spoofing Angriff
- out-of-zone data: Daten die in delegierte Zone gehören!

typische Fehler

- Fehler in resolv.conf
  $\longrightarrow$ Test mit **nslookup** und der Option `set all`
- fehlende **search**list, kein Domainname gesetzt?
- Antwort von unbekannter Quelle
  (`response from unexpected source`)
  $\longrightarrow$ Antwort von falschem Interface
  $\longrightarrow$ spoofing Angriff
- out-of-zone data: Daten die in delegierte Zone gehören!

typische Fehler

- Fehler in resolv.conf
  $\longrightarrow$ Test mit **nslookup** und der Option `set all`
- fehlende **search**list, kein Domainname gesetzt?
- Antwort von unbekannter Quelle
  (`response from unexpected source`)
  $\longrightarrow$ Antwort von falschem Interface
  $\longrightarrow$ spoofing Angriff
- out-of-zone data: Daten die in delegierte Zone gehören!

typische Fehler

- Fehler in resolv.conf
  $\longrightarrow$ Test mit **nslookup** und der Option `set all`
- fehlende **search**list, kein Domainname gesetzt?
- Antwort von unbekannter Quelle
  (`response from unexpected source`)
  $\longrightarrow$ Antwort von falschem Interface
  $\longrightarrow$ spoofing Angriff
- out-of-zone data: Daten die in delegierte Zone gehören!

typische Fehler

- Fehler in resolv.conf
  $\longrightarrow$ Test mit **nslookup** und der Option `set all`
- fehlende **search**list, kein Domainname gesetzt?
- Antwort von unbekannter Quelle
  (`response from unexpected source`)
  $\longrightarrow$ Antwort von falschem Interface
  $\longrightarrow$ spoofing Angriff
- out-of-zone data: Daten die in delegierte Zone gehören!

typische Fehler

- Fehler in resolv.conf
  $\longrightarrow$ Test mit **nslookup** und der Option `set all`
- fehlende **search**list, kein Domainname gesetzt?
- Antwort von unbekannter Quelle
  (`response from unexpected source`)
  $\longrightarrow$ Antwort von falschem Interface
  $\longrightarrow$ spoofing Angriff
- out-of-zone data: Daten die in delegierte Zone gehören!

typische Fehler

- Zonentransfer mit WINS-Einträgen funktionieren nicht!
- Andere Nameserver ignorieren negative gecachte Antworten
  $\longrightarrow$ häufig ein Problem bei Verwendung von forwardern
  $\longrightarrow$ diese versuchen dann die Root-Nameserver direkt zu befragen
  $\longrightarrow$ Problem mit Timeouts bei Firewalls, PAC-Dateien!
  $\longrightarrow$ sehr schwierig zu finden!

typische Fehler

- Zonentransfer mit WINS-Einträgen funktionieren nicht!
- Andere Nameserver ignorieren negative gecachte Antworten
  $\longrightarrow$ häufig ein Problem bei Verwendung von forwardern
  $\longrightarrow$ diese versuchen dann die Root-Nameserver direkt zu befragen
  $\longrightarrow$ Problem mit Timeouts bei Firewalls, PAC-Dateien!
  $\longrightarrow$ sehr schwierig zu finden!

typische Fehler

- Zonentransfer mit WINS-Einträgen funktionieren nicht!
- Andere Nameserver ignorieren negative gecachte Antworten
  $\longrightarrow$ häufig ein Problem bei Verwendung von forwardern
  $\longrightarrow$ diese versuchen dann die Root-Nameserver direkt zu befragen
  $\longrightarrow$ Problem mit Timeouts bei Firewalls, PAC-Dateien!
  $\longrightarrow$ sehr schwierig zu finden!

typische Fehler

- Zonentransfer mit WINS-Einträgen funktionieren nicht!
- Andere Nameserver ignorieren negative gecachte Antworten
  $\longrightarrow$ häufig ein Problem bei Verwendung von forwardern
  $\longrightarrow$ diese versuchen dann die Root-Nameserver direkt zu befragen
  $\longrightarrow$ Problem mit Timeouts bei Firewalls, PAC-Dateien!
  $\longrightarrow$ sehr schwierig zu finden!

typische Fehler

- Zonentransfer mit WINS-Einträgen funktionieren nicht!
- Andere Nameserver ignorieren negative gecachte Antworten
  $\longrightarrow$ häufig ein Problem bei Verwendung von forwardern
  $\longrightarrow$ diese versuchen dann die Root-Nameserver direkt zu befragen
  $\longrightarrow$ Problem mit Timeouts bei Firewalls, PAC-Dateien!
  $\longrightarrow$ sehr schwierig zu finden!

typische Fehler

- Zonentransfer mit WINS-Einträgen funktionieren nicht!
- Andere Nameserver ignorieren negative gecachte Antworten
  $\longrightarrow$ häufig ein Problem bei Verwendung von forwardern
  $\longrightarrow$ diese versuchen dann die Root-Nameserver direkt zu befragen
  $\longrightarrow$ Problem mit Timeouts bei Firewalls, PAC-Dateien!
  $\longrightarrow$ sehr schwierig zu finden!

Typische Fehler

- Internet ist träge
  $\longrightarrow$ oft bei WWW-Zugriffen beobachtbar
  $\longrightarrow$ 1. Nameserver in `/etc/resolv.conf` antwortet nicht.
- Oft ein Problem mit PAC-Dateien, benötigen DNS-Auflösung zur Auswahl des Proxies oder direkten Zugriff
- Noch schlimmer: nscd, **n**ame **s**ervice **c**ache **d**aemon

Typische Fehler

- Internet ist träge
  $\longrightarrow$ oft bei WWW-Zugriffen beobachtbar
  $\longrightarrow$ 1. Nameserver in `/etc/resolv.conf` antwortet nicht.
- Oft ein Problem mit PAC-Dateien, benötigen DNS-Auflösung zur Auswahl des Proxies oder direkten Zugriff
- Noch schlimmer: nscd, **n**ame **s**ervice **c**ache **d**aemon

Typische Fehler

- Internet ist träge
  $\longrightarrow$ oft bei WWW-Zugriffen beobachtbar
  $\longrightarrow$ 1. Nameserver in `/etc/resolv.conf` antwortet nicht.
- Oft ein Problem mit PAC-Dateien, benötigen DNS-Auflösung zur Auswahl des Proxies oder direkten Zugriff
- Noch schlimmer: nscd, **n**ame **s**ervice **c**ache **d**aemon

Typische Fehler

- Internet ist träge
  $\longrightarrow$ oft bei WWW-Zugriffen beobachtbar
  $\longrightarrow$ 1. Nameserver in `/etc/resolv.conf` antwortet nicht.
- Oft ein Problem mit PAC-Dateien, benötigen DNS-Auflösung zur Auswahl des Proxies oder direkten Zugriff
- Noch schlimmer: nscd, **n**ame **s**ervice **c**ache **d**aemon

Typische Fehler

- Internet ist träge
  $\longrightarrow$ oft bei WWW-Zugriffen beobachtbar
  $\longrightarrow$ 1. Nameserver in `/etc/resolv.conf` antwortet nicht.
- Oft ein Problem mit PAC-Dateien, benötigen DNS-Auflösung zur Auswahl des Proxies oder direkten Zugriff
- Noch schlimmer: nscd, **n**ame **s**ervice **c**ache **d**aemon

FIN

Noch Fragen?