Domain Name System

Peter Schmid

Hochschule für Technik Zürich
MAS Informatik, Verteilte Systeme

17.8.2010

Outline

Weshalb befassen wir uns mit DNS?

- Jeder der Internet benutzt, braucht DNS
- Gutes Beispiel für ein verteiltes System (hierarchische verteilte Datenbank)
- Einfach aufgebaut
- Gut geeignet um selber Hand anzulegen $\Rightarrow$ Uebungen
- Die Netzwerkaspekte werden im Modul Netzwerke genauer betrachtet
- DNS ist eine der vier Hauptkomponenten von Active Directory
- Leichter Einstieg in Verzeichnisdienste

Weshalb befassen wir uns mit DNS?

- Jeder der Internet benutzt, braucht DNS
- Gutes Beispiel für ein verteiltes System (hierarchische verteilte Datenbank)
- Einfach aufgebaut
- Gut geeignet um selber Hand anzulegen $\Rightarrow$ Uebungen
- Die Netzwerkaspekte werden im Modul Netzwerke genauer betrachtet
- DNS ist eine der vier Hauptkomponenten von Active Directory
- Leichter Einstieg in Verzeichnisdienste

Weshalb befassen wir uns mit DNS?

- Jeder der Internet benutzt, braucht DNS
- Gutes Beispiel für ein verteiltes System (hierarchische verteilte Datenbank)
- Einfach aufgebaut
- Gut geeignet um selber Hand anzulegen $\Rightarrow$ Uebungen
- Die Netzwerkaspekte werden im Modul Netzwerke genauer betrachtet
- DNS ist eine der vier Hauptkomponenten von Active Directory
- Leichter Einstieg in Verzeichnisdienste

Weshalb befassen wir uns mit DNS?

- Jeder der Internet benutzt, braucht DNS
- Gutes Beispiel für ein verteiltes System (hierarchische verteilte Datenbank)
- Einfach aufgebaut
- Gut geeignet um selber Hand anzulegen $\Rightarrow$ Uebungen
- Die Netzwerkaspekte werden im Modul Netzwerke genauer betrachtet
- DNS ist eine der vier Hauptkomponenten von Active Directory
- Leichter Einstieg in Verzeichnisdienste

Weshalb befassen wir uns mit DNS?

- Jeder der Internet benutzt, braucht DNS
- Gutes Beispiel für ein verteiltes System (hierarchische verteilte Datenbank)
- Einfach aufgebaut
- Gut geeignet um selber Hand anzulegen $\Rightarrow$ Uebungen
- Die Netzwerkaspekte werden im Modul Netzwerke genauer betrachtet
- DNS ist eine der vier Hauptkomponenten von Active Directory
- Leichter Einstieg in Verzeichnisdienste

Weshalb befassen wir uns mit DNS?

- Jeder der Internet benutzt, braucht DNS
- Gutes Beispiel für ein verteiltes System (hierarchische verteilte Datenbank)
- Einfach aufgebaut
- Gut geeignet um selber Hand anzulegen $\Rightarrow$ Uebungen
- Die Netzwerkaspekte werden im Modul Netzwerke genauer betrachtet
- DNS ist eine der vier Hauptkomponenten von Active Directory
- Leichter Einstieg in Verzeichnisdienste

Weshalb befassen wir uns mit DNS?

- Jeder der Internet benutzt, braucht DNS
- Gutes Beispiel für ein verteiltes System (hierarchische verteilte Datenbank)
- Einfach aufgebaut
- Gut geeignet um selber Hand anzulegen $\Rightarrow$ Uebungen
- Die Netzwerkaspekte werden im Modul Netzwerke genauer betrachtet
- DNS ist eine der vier Hauptkomponenten von Active Directory
- Leichter Einstieg in Verzeichnisdienste

Geschichtlicher Rückblick

- ARPAnet des US Departement of Defense setzte in den 1960er Jahre erstes WAN ein.
- Alle Hosts waren in einem File `HOSTS.TXT` verzeichnet, mit FTP herunter ladbar.
- In den 1970er waren es erst ein paar Hundert Hosts
- Mit dem Aufkommen von TCP/IP in den 1980er änderte sich das schlagartig (zigtausend Hosts)
- BSD Unix `/etc/hosts`
- 1983 von Paul Mockapetris spezifiziert und als JEEVES auf DEC implementiert
- Weiterentwicklung als BIND an Berkeley (Berkeley Internet Name Domain)

HOCHSCHULE FÜR TECHNIK ZÜRICH

Geschichtlicher Rückblick

- ARPAnet des US Departement of Defense setzte in den 1960er Jahre erstes WAN ein.
- Alle Hosts waren in einem File `HOSTS.TXT` verzeichnet, mit FTP herunter ladbar.
- In den 1970er waren es erst ein paar Hundert Hosts
- Mit dem Aufkommen von TCP/IP in den 1980er änderte sich das schlagartig (zigtausend Hosts)
- BSD Unix `/etc/hosts`
- 1983 von Paul Mockapetris spezifiziert und als JEEVES auf DEC implementiert
- Weiterentwicklung als BIND an Berkeley (Berkeley Internet Name Domain)

HOCHSCHULE FÜR TECHNIK ZÜRICH

Geschichtlicher Rückblick

- ARPAnet des US Departement of Defense setzte in den 1960er Jahre erstes WAN ein.
- Alle Hosts waren in einem File `HOSTS.TXT` verzeichnet, mit FTP herunter ladbar.
- In den 1970er waren es erst ein paar Hundert Hosts
- Mit dem Aufkommen von TCP/IP in den 1980er änderte sich das schlagartig (zigtausend Hosts)
- BSD Unix `/etc/hosts`
- 1983 von Paul Mockapetris spezifiziert und als JEEVES auf DEC implementiert
- Weiterentwicklung als BIND an Berkeley (Berkeley Internet Name Domain)

Geschichtlicher Rückblick

- ARPAnet des US Departement of Defense setzte in den 1960er Jahre erstes WAN ein.
- Alle Hosts waren in einem File `HOSTS.TXT` verzeichnet, mit FTP herunter ladbar.
- In den 1970er waren es erst ein paar Hundert Hosts
- Mit dem Aufkommen von TCP/IP in den 1980er änderte sich das schlagartig (zigtausend Hosts)
- BSD Unix `/etc/hosts`
- 1983 von Paul Mockapetris spezifiziert und als JEEVES auf DEC implementiert
- Weiterentwicklung als BIND an Berkeley (Berkeley Internet Name Domain)

Geschichtlicher Rückblick

- ARPAnet des US Departement of Defense setzte in den 1960er Jahre erstes WAN ein.
- Alle Hosts waren in einem File `HOSTS.TXT` verzeichnet, mit FTP herunter ladbar.
- In den 1970er waren es erst ein paar Hundert Hosts
- Mit dem Aufkommen von TCP/IP in den 1980er änderte sich das schlagartig (zigtausend Hosts)
- BSD Unix `/etc/hosts`
- 1983 von Paul Mockapetris spezifiziert und als JEEVES auf DEC implementiert
- Weiterentwicklung als BIND an Berkeley (Berkeley Internet Name Domain)

Geschichtlicher Rückblick

- ARPAnet des US Departement of Defense setzte in den 1960er Jahre erstes WAN ein.
- Alle Hosts waren in einem File `HOSTS.TXT` verzeichnet, mit FTP herunter ladbar.
- In den 1970er waren es erst ein paar Hundert Hosts
- Mit dem Aufkommen von TCP/IP in den 1980er änderte sich das schlagartig (zigtausend Hosts)
- BSD Unix `/etc/hosts`
- 1983 von Paul Mockapetris spezifiziert und als JEEVES auf DEC implementiert
- Weiterentwicklung als BIND an Berkeley (Berkeley Internet Name Domain)

Geschichtlicher Rückblick

- ARPAnet des US Departement of Defense setzte in den 1960er Jahre erstes WAN ein.
- Alle Hosts waren in einem File `HOSTS.TXT` verzeichnet, mit FTP herunter ladbar.
- In den 1970er waren es erst ein paar Hundert Hosts
- Mit dem Aufkommen von TCP/IP in den 1980er änderte sich das schlagartig (zigtausend Hosts)
- BSD Unix `/etc/hosts`
- 1983 von Paul Mockapetris spezifiziert und als JEEVES auf DEC implementiert
- Weiterentwicklung als BIND an Berkeley (Berkeley Internet Name Domain)

Ziel für neues System

- Lokal administrierbar
- Daten global verfügbar
- Hoch verfügbar (Master/Slave, Primary/Secondary/..)
- Drei Hauptkomponenten: Domain Name Space (hierarchisch), Name Servers, Resolvers (Clients).
- 1984 in RFC 882 und 883 formuliert
- Sicherheit war kein Design-Kriterium

Ziel für neues System

- Lokal administrierbar
- Daten global verfügbar
- Hoch verfügbar (Master/Slave, Primary/Secondary/..)
- Drei Hauptkomponenten: Domain Name Space (hierarchisch), Name Servers, Resolvers (Clients).
- 1984 in RFC 882 und 883 formuliert
- Sicherheit war kein Design-Kriterium

Ziel für neues System

- Lokal administrierbar
- Daten global verfügbar
- Hoch verfügbar (Master/Slave, Primary/Secondary/..)
- Drei Hauptkomponenten: Domain Name Space (hierarchisch), Name Servers, Resolvers (Clients).
- 1984 in RFC 882 und 883 formuliert
- Sicherheit war kein Design-Kriterium

Ziel für neues System

- Lokal administrierbar
- Daten global verfügbar
- Hoch verfügbar (Master/Slave, Primary/Secondary/..)
- Drei Hauptkomponenten: Domain Name Space (hierarchisch), Name Servers, Resolvers (Clients).
- 1984 in RFC 882 und 883 formuliert
- Sicherheit war kein Design-Kriterium

Ziel für neues System

- Lokal administrierbar
- Daten global verfügbar
- Hoch verfügbar (Master/Slave, Primary/Secondary/..)
- Drei Hauptkomponenten: Domain Name Space (hierarchisch), Name Servers, Resolvers (Clients).
- 1984 in RFC 882 und 883 formuliert
- Sicherheit war kein Design-Kriterium

Ziel für neues System

- Lokal administrierbar
- Daten global verfügbar
- Hoch verfügbar (Master/Slave, Primary/Secondary/..)
- Drei Hauptkomponenten: Domain Name Space (hierarchisch), Name Servers, Resolvers (Clients).
- 1984 in RFC 882 und 883 formuliert
- Sicherheit war kein Design-Kriterium

Domain Names

- Jeder Knoten (Domain) hat ein Text-Label ausser der Top-Node (root) ist leer
- Der Domain Name eines Knoten ist die Aneinanderreihung von allen Text-Labels des Pfades zum Top-Node (root). Von rechts nach links abgetrennt durch "."
- Top-Level Domain TLD:
  - generic TLD `gov`, `edu`, `com`, `mil`, `org`, `int` und `net`
  - country-code TLD `ch`, `cn`, `us`, `de`, `tv` etc. zwei Zeichen nach ISO 3166
  - erweiterte generic TLD `name`, `biz`, `info` etc.
- LDH rule d.h. Letter, Digits und Hyphen (-)
- Klein- und Grossschreibung wird nicht unterschieden
- max. 63 Zeichen lang, max. 253 Zeichen für FQDN
- max. 127 Levels

Domain Names

- Jeder Knoten (Domain) hat ein Text-Label ausser der Top-Node (root) ist leer
- Der Domain Name eines Knoten ist die Aneinanderreihung von allen Text-Labels des Pfades zum Top-Node (root). Von rechts nach links abgetrennt durch "."
- Top-Level Domain TLD:
  - generic TLD `gov`, `edu`, `com`, `mil`, `org`, `int` und `net`
  - country-code TLD `ch`, `cn`, `us`, `de`, `tv` etc. zwei Zeichen nach ISO 3166
  - erweiterte generic TLD `name`, `biz`, `info` etc.
- LDH rule d.h. Letter, Digits und Hyphen (-)
- Klein- und Grossschreibung wird nicht unterschieden
- max. 63 Zeichen lang, max. 253 Zeichen für FQDN
- max. 127 Levels

Domain Names

- Jeder Knoten (Domain) hat ein Text-Label ausser der Top-Node (root) ist leer
- Der Domain Name eines Knoten ist die Aneinanderreihung von allen Text-Labels des Pfades zum Top-Node (root). Von rechts nach links abgetrennt durch "."
- Top-Level Domain TLD:
  - generic TLD `gov`, `edu`, `com`, `mil`, `org`, `int` und `net`
  - country-code TLD `ch`, `cn`, `us`, `de`, `tv` etc. zwei Zeichen nach ISO 3166
  - erweiterte generic TLD `name`, `biz`, `info` etc.
- LDH rule d.h. Letter, Digits und Hyphen (-)
- Klein- und Grossschreibung wird nicht unterschieden
- max. 63 Zeichen lang, max. 253 Zeichen für FQDN
- max. 127 Levels

Domain Names

- Jeder Knoten (Domain) hat ein Text-Label ausser der Top-Node (root) ist leer
- Der Domain Name eines Knoten ist die Aneinanderreihung von allen Text-Labels des Pfades zum Top-Node (root). Von rechts nach links abgetrennt durch "."
- Top-Level Domain TLD:
  - generic TLD `gov`, `edu`, `com`, `mil`, `org`, `int` und `net`
  - country-code TLD `ch`, `cn`, `us`, `de`, `tv` etc. zwei Zeichen nach ISO 3166
  - erweiterte generic TLD `name`, `biz`, `info` etc.
- LDH rule d.h. Letter, Digits und Hyphen (-)
- Klein- und Grossschreibung wird nicht unterschieden
- max. 63 Zeichen lang, max. 253 Zeichen für FQDN
- max. 127 Levels

Domain Names

- Jeder Knoten (Domain) hat ein Text-Label ausser der Top-Node (root) ist leer
- Der Domain Name eines Knoten ist die Aneinanderreihung von allen Text-Labels des Pfades zum Top-Node (root). Von rechts nach links abgetrennt durch "."
- Top-Level Domain TLD:
  - generic TLD `gov`, `edu`, `com`, `mil`, `org`, `int` und `net`
  - country-code TLD `ch`, `cn`, `us`, `de`, `tv` etc. zwei Zeichen nach ISO 3166
  - erweiterte generic TLD `name`, `biz`, `info` etc.
- LDH rule d.h. Letter, Digits und Hyphen (-)
- Klein- und Grossschreibung wird nicht unterschieden
- max. 63 Zeichen lang, max. 253 Zeichen für FQDN
- max. 127 Levels

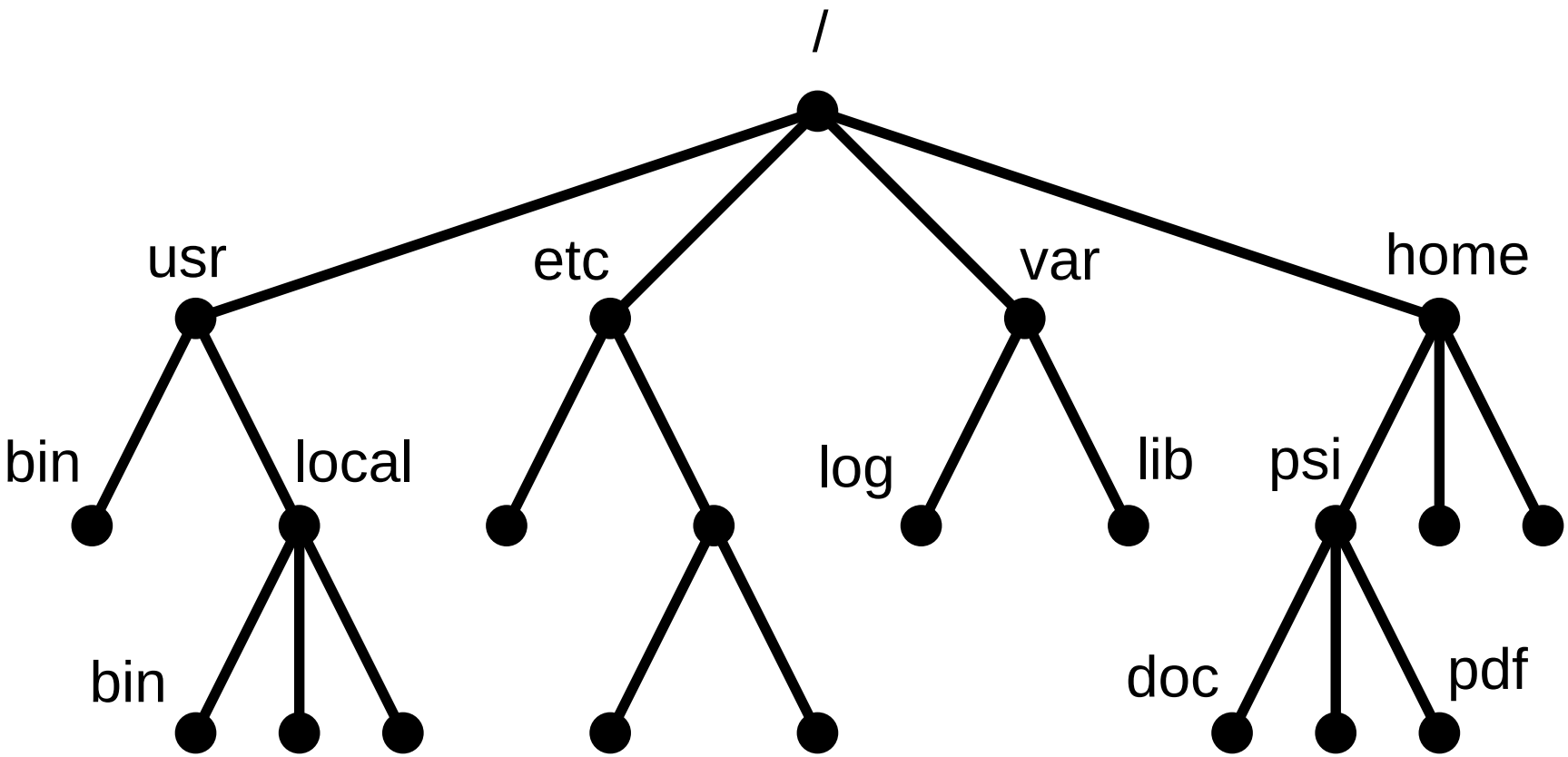
Domain Names

- Jeder Knoten (Domain) hat ein Text-Label ausser der Top-Node (root) ist leer
- Der Domain Name eines Knoten ist die Aneinanderreihung von allen Text-Labels des Pfades zum Top-Node (root). Von rechts nach links abgetrennt durch "."
- Top-Level Domain TLD:
  - generic TLD `gov`, `edu`, `com`, `mil`, `org`, `int` und `net`
  - country-code TLD `ch`, `cn`, `us`, `de`, `tv` etc. zwei Zeichen nach ISO 3166
  - erweiterte generic TLD `name`, `biz`, `info` etc.
- LDH rule d.h. Letter, Digits und Hyphen (-)
- Klein- und Grossschreibung wird nicht unterschieden
- max. 63 Zeichen lang, max. 253 Zeichen für FQDN
- max. 127 Levels

Domain Names

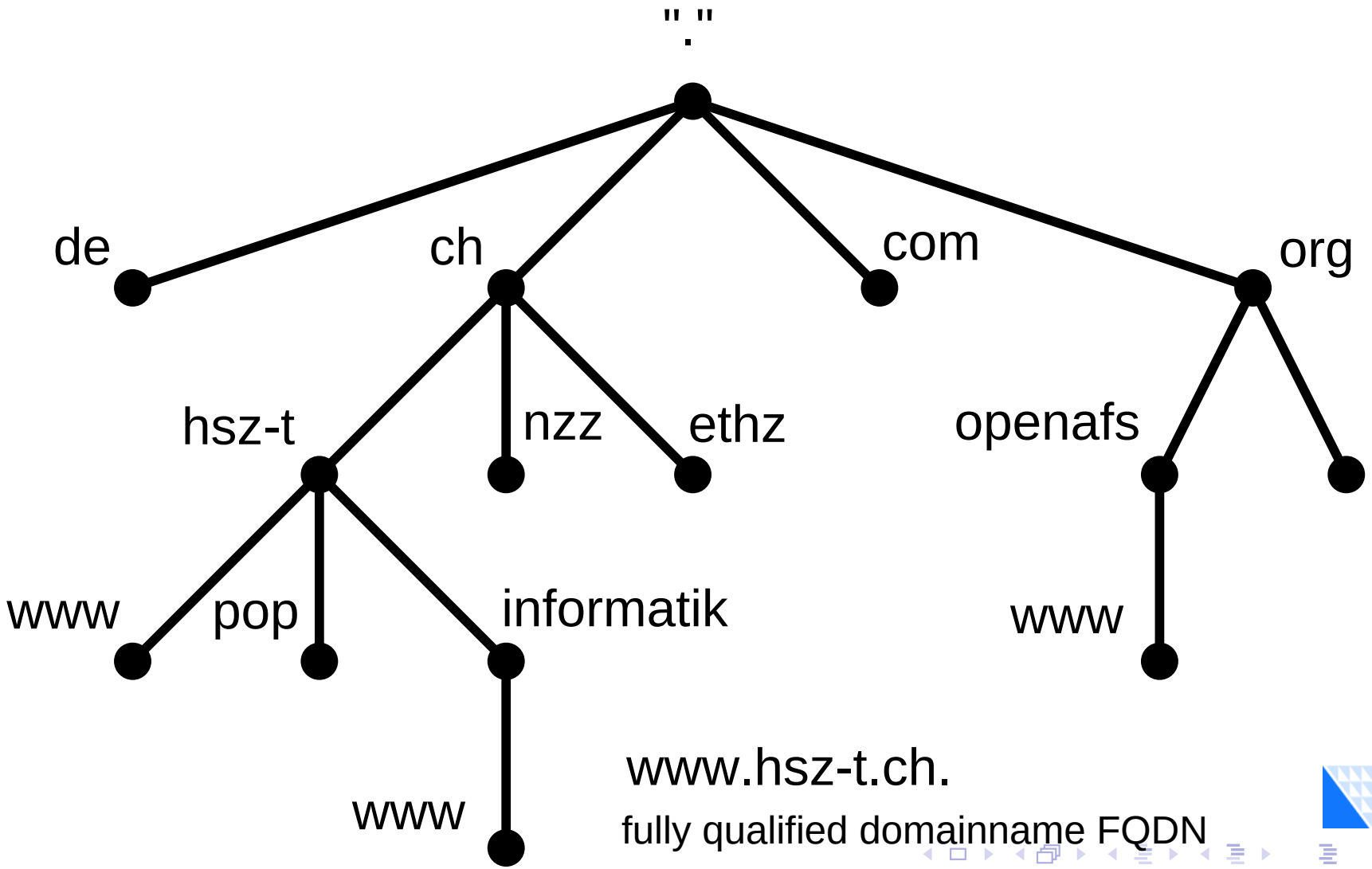
- Jeder Knoten (Domain) hat ein Text-Label ausser der Top-Node (root) ist leer
- Der Domain Name eines Knoten ist die Aneinanderreihung von allen Text-Labels des Pfades zum Top-Node (root). Von rechts nach links abgetrennt durch "."
- Top-Level Domain TLD:
  - generic TLD `gov`, `edu`, `com`, `mil`, `org`, `int` und `net`
  - country-code TLD `ch`, `cn`, `us`, `de`, `tv` etc. zwei Zeichen nach ISO 3166
  - erweiterte generic TLD `name`, `biz`, `info` etc.
- LDH rule d.h. Letter, Digits und Hyphen (-)
- Klein- und Grossschreibung wird nicht unterschieden
- max. 63 Zeichen lang, max. 253 Zeichen für FQDN
- max. 127 Levels

Filesystem Tree



/usr/local/bin absolute path

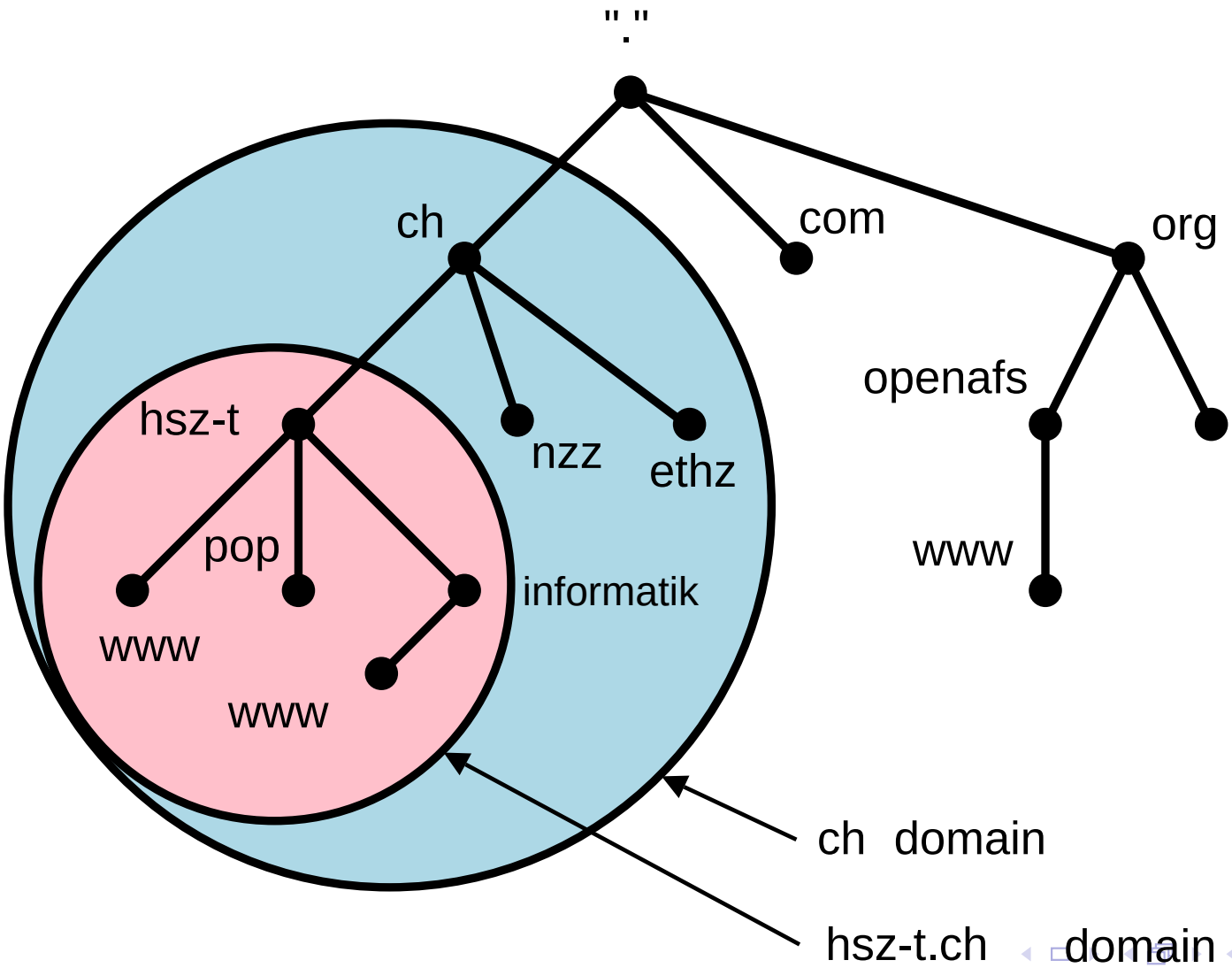
Domain Name Space



www.hsz-t.ch.

fully qualified domainname FQDN

Domains


"."
ch
com
org
openafs
hsz-t
nzz
ethz
pop
informatik
www
www
www
ch domain
hsz-t.ch domain

Zonen

- Der Namensraum (Domain Name Space) ist in Zonen aufgeteilt
- Die Zonen beginnen an einem Knoten und enden an einem Blatt oder einem Knoten wo eine neue Zone beginnt.
- Die Daten einer Zone werden von einem Name Server gespeichert und verwaltet
- Zonen und Domains sind nicht dasselbe. Domains und Subdomains überschneiden sich, Zonen überschneiden sich nie.
- *Domain* Name Server verwalten Zonen und nicht Domains!

Zonen

- Der Namensraum (Domain Name Space) ist in Zonen aufgeteilt
- Die Zonen beginnen an einem Knoten und enden an einem Blatt oder einem Knoten wo eine neue Zone beginnt.
- Die Daten einer Zone werden von einem Name Server gespeichert und verwaltet
- Zonen und Domains sind nicht dasselbe. Domains und Subdomains überschneiden sich, Zonen überschneiden sich nie.
- *Domain* Name Server verwalten Zonen und nicht Domains!

Zonen

- Der Namensraum (Domain Name Space) ist in Zonen aufgeteilt
- Die Zonen beginnen an einem Knoten und enden an einem Blatt oder einem Knoten wo eine neue Zone beginnt.
- Die Daten einer Zone werden von einem Name Server gespeichert und verwaltet
- Zonen und Domains sind nicht dasselbe. Domains und Subdomains überschneiden sich, Zonen überschneiden sich nie.
- *Domain* Name Server verwalten Zonen und nicht Domains!

Zonen

- Der Namensraum (Domain Name Space) ist in Zonen aufgeteilt
- Die Zonen beginnen an einem Knoten und enden an einem Blatt oder einem Knoten wo eine neue Zone beginnt.
- Die Daten einer Zone werden von einem Name Server gespeichert und verwaltet
- Zonen und Domains sind nicht dasselbe. Domains und Subdomains überschneiden sich, Zonen überschneiden sich nie.
- *Domain* Name Server verwalten Zonen und nicht Domains!

Zonen

- Der Namensraum (Domain Name Space) ist in Zonen aufgeteilt
- Die Zonen beginnen an einem Knoten und enden an einem Blatt oder einem Knoten wo eine neue Zone beginnt.
- Die Daten einer Zone werden von einem Name Server gespeichert und verwaltet
- Zonen und Domains sind nicht dasselbe. Domains und Subdomains überschneiden sich, Zonen überschneiden sich nie.
- *Domain* Name Server verwalten Zonen und nicht Domains!

Delegation

- Name Server können Zonen delegieren, d.h. die Verantwortung für die Auflösung hat dann der delegierte Server
- Ein Delegationspunkt wird mit einem oder mehreren *NS Records* gekennzeichnet
- Eine Zone kann z.B. einfach nur zwei Delegationen haben `aaa.beispiel.ch` und `bbb.beispiel.ch`

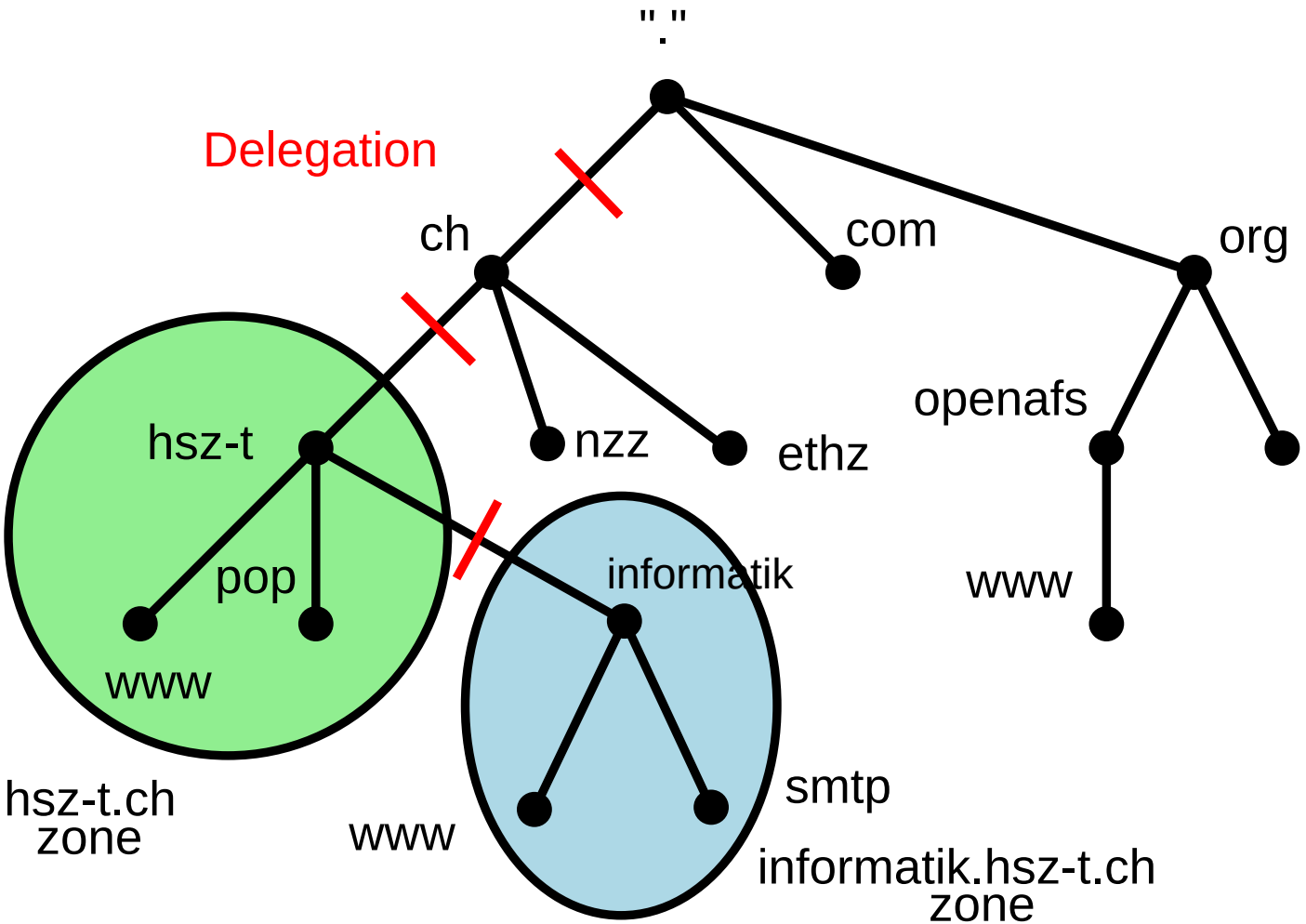
Delegation

- Name Server können Zonen delegieren, d.h. die Verantwortung für die Auflösung hat dann der delegierte Server
- Ein Delegationspunkt wird mit einem oder mehreren *NS Records* gekennzeichnet
- Eine Zone kann z.B. einfach nur zwei Delegationen haben `aaa.beispiel.ch` und `bbb.beispiel.ch`

Delegation

- Name Server können Zonen delegieren, d.h. die Verantwortung für die Auflösung hat dann der delegierte Server
- Ein Delegationspunkt wird mit einem oder mehreren *NS Records* gekennzeichnet
- Eine Zone kann z.B. einfach nur zwei Delegationen haben `aaa.beispiel.ch` und `bbb.beispiel.ch`

Beispiel Zonen



Authorative Name Servers

- Authorative Name Server sind verantwortlich für die Auflösung von Namen und geben die Anfragen nicht weiter
- *Primary Master* oder einfach nur *Primary* Server holt die Zonen-Daten von lokalen Files (Zonen-File).
- *Slave* oder *Secondary* Server holen die Daten vom Primary (Zonen-Transfer) oder auch von einem anderen Secondary und halten die Daten dann lokal.
- *Stealth* Server eine art versteckter Primary. Ist nicht eingetragen in NS Records. Kann hinter Firewall stehen $\Rightarrow$ mehr Sicherheit

Authorative Name Servers

- Authorative Name Server sind verantwortlich für die Auflösung von Namen und geben die Anfragen nicht weiter
- *Primary Master* oder einfach nur *Primary* Server holt die Zonen-Daten von lokalen Files (Zonen-File).
- *Slave* oder *Secondary* Server holen die Daten vom Primary (Zonen-Transfer) oder auch von einem anderen Secondary und halten die Daten dann lokal.
- *Stealth* Server eine art versteckter Primary. Ist nicht eingetragen in NS Records. Kann hinter Firewall stehen $\Rightarrow$ mehr Sicherheit

Authorative Name Servers

- Authorative Name Server sind verantwortlich für die Auflösung von Namen und geben die Anfragen nicht weiter
- *Primary Master* oder einfach nur *Primary* Server holt die Zonen-Daten von lokalen Files (Zonen-File).
- *Slave* oder *Secondary* Server holen die Daten vom Primary (Zonen-Transfer) oder auch von einem anderen Secondary und halten die Daten dann lokal.
- *Stealth* Server eine art versteckter Primary. Ist nicht eingetragen in NS Records. Kann hinter Firewall stehen $\Rightarrow$ mehr Sicherheit

Authorative Name Servers

- Authorative Name Server sind verantwortlich für die Auflösung von Namen und geben die Anfragen nicht weiter
- *Primary Master* oder einfach nur *Primary* Server holt die Zonen-Daten von lokalen Files (Zonen-File).
- *Slave* oder *Secondary* Server holen die Daten vom Primary (Zonen-Transfer) oder auch von einem anderen Secondary und halten die Daten dann lokal.
- *Stealth* Server eine art versteckter Primary. Ist nicht eingetragen in NS Records. Kann hinter Firewall stehen $\Rightarrow$ mehr Sicherheit

Caching Name Servers

- Namen die eine Name Server nicht auflösen kann, versucht er rekursiv aufzulösen. Die aufgelösten Namen werden zwischengespeichert (Cache).
- Caching Server und Recursive Server werden oft synonym verwendet
- Die Dauer der Zwischenspeicherung hängt vom Time To Live (TTL) Feld ab,
- Der Suchvorgang nennt man *Recursive Lookup*
- Der Name Server kann den Suchvorgang weiterreichen $\Rightarrow$ *Query Forward*

Caching Name Servers

- Namen die eine Name Server nicht auflösen kann, versucht er rekursiv aufzulösen. Die aufgelösten Namen werden zwischengespeichert (Cache).
- Caching Server und Recursive Server werden oft synonym verwendet
- Die Dauer der Zwischenspeicherung hängt vom Time To Live (TTL) Feld ab,
- Der Suchvorgang nennt man *Recursive Lookup*
- Der Name Server kann den Suchvorgang weiterreichen ⇒ *Query Forward*

Caching Name Servers

- Namen die eine Name Server nicht auflösen kann, versucht er rekursiv aufzulösen. Die aufgelösten Namen werden zwischengespeichert (Cache).
- Caching Server und Recursive Server werden oft synonym verwendet
- Die Dauer der Zwischenspeicherung hängt vom Time To Live (TTL) Feld ab,
- Der Suchvorgang nennt man *Recursive Lookup*
- Der Name Server kann den Suchvorgang weiterreichen ⇒ *Query Forward*

Caching Name Servers

- Namen die eine Name Server nicht auflösen kann, versucht er rekursiv aufzulösen. Die aufgelösten Namen werden zwischengespeichert (Cache).
- Caching Server und Recursive Server werden oft synonym verwendet
- Die Dauer der Zwischenspeicherung hängt vom Time To Live (TTL) Feld ab,
- Der Suchvorgang nennt man *Recursive Lookup*
- Der Name Server kann den Suchvorgang weiterreichen ⇒ *Query Forward*

Caching Name Servers

- Namen die eine Name Server nicht auflösen kann, versucht er rekursiv aufzulösen. Die aufgelösten Namen werden zwischengespeichert (Cache).
- Caching Server und Recursive Server werden oft synonym verwendet
- Die Dauer der Zwischenspeicherung hängt vom Time To Live (TTL) Feld ab,
- Der Suchvorgang nennt man *Recursive Lookup*
- Der Name Server kann den Suchvorgang weiterreichen $\Rightarrow$ *Query Forward*

Name Server in verschiedenen Rollen

- Ein Name Server kann gleichzeitig für einzelne Zonen Master sein, für andere Slave und auch Caching (recursive) für lokale Clients.
- v.a. bei grösseren Installationen macht es Sinn, diese Rollen zu trennen:
  - Authoritative-only Server (aus Sicherheitsgründen Recursive gesperrt)
  - Caching-only Server für lokale Clients. Zugriff von ausserhalb (Internet) gesperrt

Name Server in verschiedenen Rollen

- Ein Name Server kann gleichzeitig für einzelne Zonen Master sein, für andere Slave und auch Caching (recursive) für lokale Clients.
- v.a. bei grösseren Installationen macht es Sinn, diese Rollen zu trennen:
  - Authoritative-only Server (aus Sicherheitsgründen Recursive gesperrt)
  - Caching-only Server für lokale Clients. Zugriff von ausserhalb (Internet) gesperrt

Name Server in verschiedenen Rollen

- Ein Name Server kann gleichzeitig für einzelne Zonen Master sein, für andere Slave und auch Caching (recursive) für lokale Clients.
- v.a. bei grösseren Installationen macht es Sinn, diese Rollen zu trennen:
  - Authoritative-only Server (aus Sicherheitsgründen Recursive gesperrt)
  - Caching-only Server für lokale Clients. Zugriff von ausserhalb (Internet) gesperrt

Name Server in verschiedenen Rollen

- Ein Name Server kann gleichzeitig für einzelne Zonen Master sein, für andere Slave und auch Caching (recursive) für lokale Clients.
- v.a. bei grösseren Installationen macht es Sinn, diese Rollen zu trennen:
  - Authoritative-only Server (aus Sicherheitsgründen Recursive gesperrt)
  - Caching-only Server für lokale Clients. Zugriff von ausserhalb (Internet) gesperrt

Zonentransfer

- Der Abgleich der Daten zwischen Primary (Master) und den Secondaries (Slaves) geschieht durch Zonentransfers
- Erhöhung der Ausfallsicherheit und Lastausgleich
- Notify-Verfahren: Master benachrichtigt alle Slaves einer Zone, Slave fordert Daten an (ganz oder inkrementell)
- Slave-Hol-Verfahren: Slave holt in bestimmten Abständen den SOA Resource Record, ist die Seriennummer des SOA-RRs des Masters $>$ Slaves, dann fordert Slave die Daten an (ganz oder inkrementell)

Zonentransfer

- Der Abgleich der Daten zwischen Primary (Master) und den Secondaries (Slaves) geschieht durch Zonentransfers
- Erhöhung der Ausfallsicherheit und Lastausgleich
- Notify-Verfahren: Master benachrichtigt alle Slaves einer Zone, Slave fordert Daten an (ganz oder inkrementell)
- Slave-Hol-Verfahren: Slave holt in bestimmten Abständen den SOA Resource Record, ist die Seriennummer des SOA-RRs des Masters $>$ Slaves, dann fordert Slave die Daten an (ganz oder inkrementell)

Zonentransfer

- Der Abgleich der Daten zwischen Primary (Master) und den Secondaries (Slaves) geschieht durch Zonentransfers
- Erhöhung der Ausfallsicherheit und Lastausgleich
- Notify-Verfahren: Master benachrichtigt alle Slaves einer Zone, Slave fordert Daten an (ganz oder inkrementell)
- Slave-Hol-Verfahren: Slave holt in bestimmten Abständen den SOA Resource Record, ist die Seriennummer des SOA-RRs des Masters $>$ Slaves, dann fordert Slave die Daten an (ganz oder inkrementell)

Zonentransfer

- Der Abgleich der Daten zwischen Primary (Master) und den Secondaries (Slaves) geschieht durch Zonentransfers
- Erhöhung der Ausfallsicherheit und Lastausgleich
- Notify-Verfahren: Master benachrichtigt alle Slaves einer Zone, Slave fordert Daten an (ganz oder inkrementell)
- Slave-Hol-Verfahren: Slave holt in bestimmten Abständen den SOA Resource Record, ist die Seriennummer des SOA-RRs des Masters $>$ Slaves, dann fordert Slave die Daten an (ganz oder inkrementell)

Registrierung von Domain-Names

- Um DNS-Namen im Internet bekannt zu machen, muss der Besitzer die Domain registrieren
- Sicherstellung, dass formale Regeln eingehalten werden und dass Namen weltweit eindeutig sind
- Domain-Registrierungen werden von IANA bzw. ICANN autorisierten Organisationen (Registrars) vorgenommen
- Meistens unter `nic.*` erreichbar. Für CH und LI ist es die Switch die `nic.ch` verwaltet.
- Fast immer kostenpflichtig, ausser z.B. `org` und `edu`

Registrierung von Domain-Names

- Um DNS-Namen im Internet bekannt zu machen, muss der Besitzer die Domain registrieren
- Sicherstellung, dass formale Regeln eingehalten werden und dass Namen weltweit eindeutig sind
- Domain-Registrierungen werden von IANA bzw. ICANN autorisierten Organisationen (Registrars) vorgenommen
- Meistens unter `nic.*` erreichbar. Für CH und LI ist es die Switch die `nic.ch` verwaltet.
- Fast immer kostenpflichtig, ausser z.B. `org` und `edu`

Registrierung von Domain-Names

- Um DNS-Namen im Internet bekannt zu machen, muss der Besitzer die Domain registrieren
- Sicherstellung, dass formale Regeln eingehalten werden und dass Namen weltweit eindeutig sind
- Domain-Registrierungen werden von IANA bzw. ICANN autorisierten Organisationen (Registrars) vorgenommen
- Meistens unter `nic.*` erreichbar. Für CH und LI ist es die Switch die `nic.ch` verwaltet.
- Fast immer kostenpflichtig, ausser z.B. `org` und `edu`

Registrierung von Domain-Names

- Um DNS-Namen im Internet bekannt zu machen, muss der Besitzer die Domain registrieren
- Sicherstellung, dass formale Regeln eingehalten werden und dass Namen weltweit eindeutig sind
- Domain-Registrierungen werden von IANA bzw. ICANN autorisierten Organisationen (Registrars) vorgenommen
- Meistens unter `nic.*` erreichbar. Für CH und LI ist es die Switch die `nic.ch` verwaltet.
- Fast immer kostenpflichtig, ausser z.B. `org` und `edu`

Registrierung von Domain-Names

- Um DNS-Namen im Internet bekannt zu machen, muss der Besitzer die Domain registrieren
- Sicherstellung, dass formale Regeln eingehalten werden und dass Namen weltweit eindeutig sind
- Domain-Registrierungen werden von IANA bzw. ICANN autorisierten Organisationen (Registrars) vorgenommen
- Meistens unter `nic.*` erreichbar. Für CH und LI ist es die Switch die `nic.ch` verwaltet.
- Fast immer kostenpflichtig, ausser z.B. `org` und `edu`

Namensauflösung Client

- Der Client kann üblicherweise nicht selbständig Namen auflösen
- Dazu fragt er einen Caching Name Server an
- Nameservereinstellungen vielfach über DHCP
- Bei UNIX/Linux findet man die Konfiguration in `/etc/resolv.conf`:

```
domain hsz-t.ch
nameserver 193.5.54.121
nameserver 193.5.54.30
search hsz-t.ch isz.ch juventus.ch
```

Namensauflösung Client

- Der Client kann üblicherweise nicht selbständig Namen auflösen
- Dazu fragt er einen Caching Name Server an
- Nameservereinstellungen vielfach über DHCP
- Bei UNIX/Linux findet man die Konfiguration in `/etc/resolv.conf`:

```
domain hsz-t.ch
nameserver 193.5.54.121
nameserver 193.5.54.30
search hsz-t.ch isz.ch juventus.ch
```

Namensauflösung Client

- Der Client kann üblicherweise nicht selbständig Namen auflösen
- Dazu fragt er einen Caching Name Server an
- Nameservereinstellungen vielfach über DHCP
- Bei UNIX/Linux findet man die Konfiguration in `/etc/resolv.conf`:

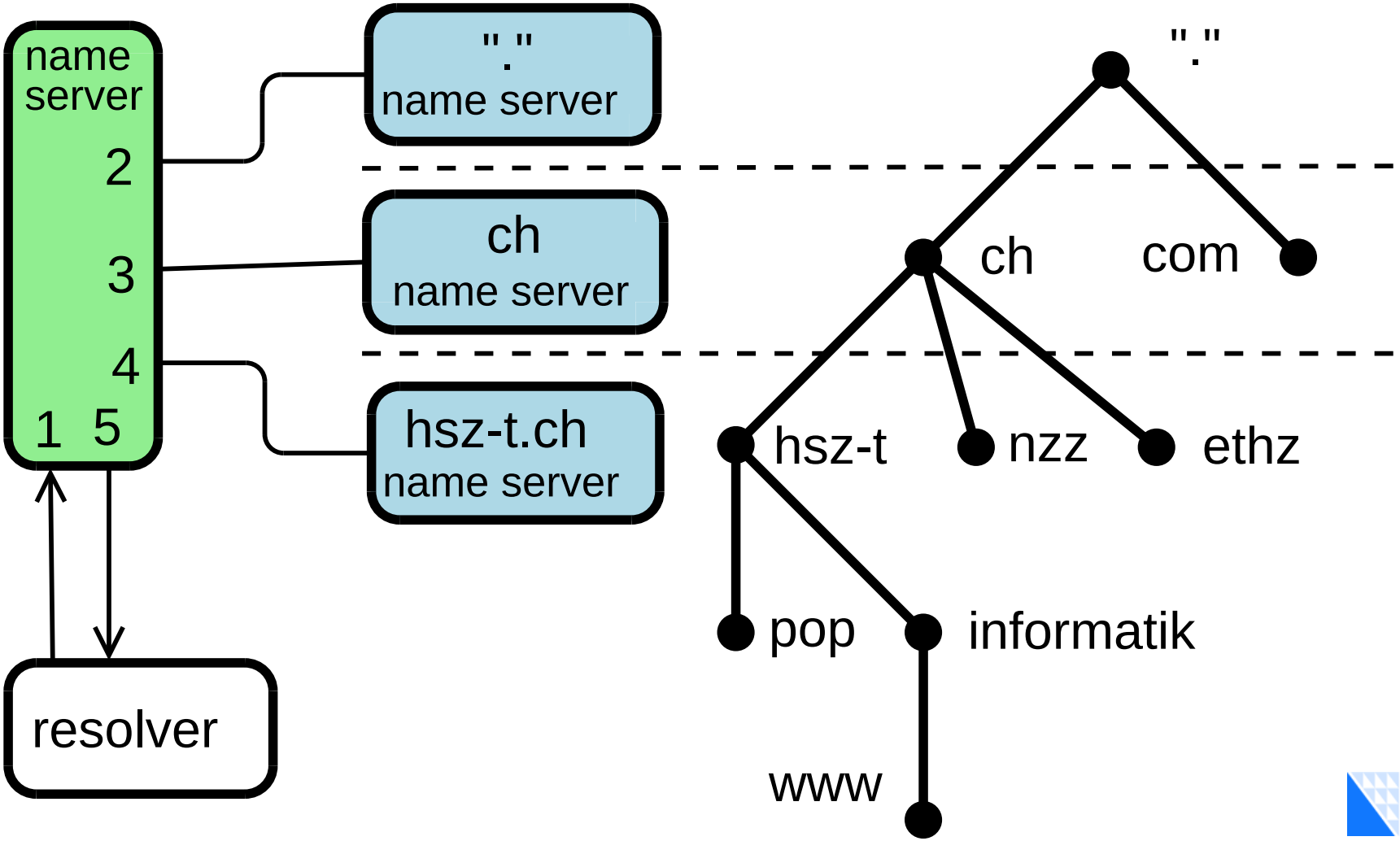
```
domain hsz-t.ch
nameserver 193.5.54.121
nameserver 193.5.54.30
search hsz-t.ch isz.ch juventus.ch
```

Namensauflösung Client

- Der Client kann üblicherweise nicht selbständig Namen auflösen
- Dazu fragt er einen Caching Name Server an
- Nameservereinstellungen vielfach über DHCP
- Bei UNIX/Linux findet man die Konfiguration in `/etc/resolv.conf`:

```
domain hsz-t.ch
nameserver 193.5.54.121
nameserver 193.5.54.30
search hsz-t.ch isz.ch juventus.ch
```

Ablauf Namensauflösung



Ablauf Namensauflösung cnt.

Auflösung für `pop.hsz-t.ch.`:

1. Resolver Query `pop.hsz-t.ch.` (rekursive Anfrage)
2. Query `pop.hsz-t.ch.` to root server (iterative Anfrage)
   Answer referral to `ch` name servers
3. Query `pop.hsz-t.ch.` to `ch` server (iterative Anfrage)
   Answer referral to `hsz-t.ch` name servers
4. Query `pop.hsz-t.ch.` to `hsz-t.ch` server (iterative Anfrage)
   Answer IP-Address `193.5.54.125`
5. Answer IP-Address `193.5.54.125`

Ablauf Namensauflösung cnt.

Auflösung für `pop.hsz-t.ch.`:

1. Resolver Query `pop.hsz-t.ch.` (rekursive Anfrage)
2. Query `pop.hsz-t.ch.` to root server (iterative Anfrage)
   Answer referral to `ch` name servers
3. Query `pop.hsz-t.ch.` to `ch` server (iterative Anfrage)
   Answer referral to `hsz-t.ch` name servers
4. Query `pop.hsz-t.ch.` to `hsz-t.ch` server (iterative Anfrage)
   Answer IP-Address `193.5.54.125`
5. Answer IP-Address `193.5.54.125`

Ablauf Namensauflösung cnt.

Auflösung für `pop.hsz-t.ch.`:

1. Resolver Query `pop.hsz-t.ch.` (rekursive Anfrage)
2. Query `pop.hsz-t.ch.` to root server (iterative Anfrage)
   Answer referral to `ch` name servers
3. Query `pop.hsz-t.ch.` to `ch` server (iterative Anfrage)
   Answer referral to `hsz-t.ch` name servers
4. Query `pop.hsz-t.ch.` to `hsz-t.ch` server (iterative Anfrage)
   Answer IP-Address `193.5.54.125`
5. Answer IP-Address `193.5.54.125`

Ablauf Namensauflösung cnt.

Auflösung für `pop.hsz-t.ch.`:

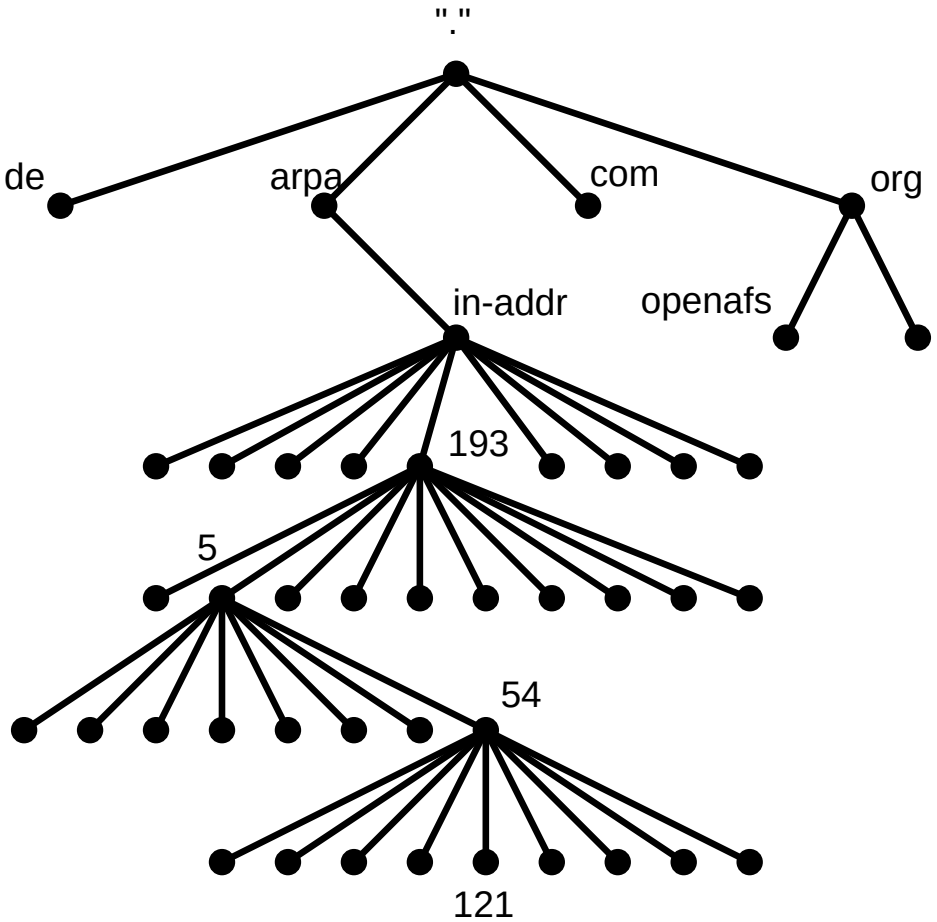
1. Resolver Query `pop.hsz-t.ch.` (rekursive Anfrage)
2. Query `pop.hsz-t.ch.` to root server (iterative Anfrage)
   Answer referral to `ch` name servers
3. Query `pop.hsz-t.ch.` to `ch` server (iterative Anfrage)
   Answer referral to `hsz-t.ch` name servers
4. Query `pop.hsz-t.ch.` to `hsz-t.ch` server (iterative Anfrage)
   Answer IP-Address `193.5.54.125`
5. Answer IP-Address `193.5.54.125`

Ablauf Namensauflösung cnt.

Auflösung für `pop.hsz-t.ch.`:

1. Resolver Query `pop.hsz-t.ch.` (rekursive Anfrage)
2. Query `pop.hsz-t.ch.` to root server (iterative Anfrage)
   Answer referral to `ch` name servers
3. Query `pop.hsz-t.ch.` to `ch` server (iterative Anfrage)
   Answer referral to `hsz-t.ch` name servers
4. Query `pop.hsz-t.ch.` to `hsz-t.ch` server (iterative Anfrage)
   Answer IP-Address `193.5.54.125`
5. Answer IP-Address `193.5.54.125`

Reverse Lookup, Inverse Queries



Aufloesung 193.5.54.12
Zuordnung IP-Adressen auf Namen (address-to-name mapping)

Dazu wird die Domain in-addr.arpa verwendet

DNS Resource Records (Aufbau Datenbank)

Wir beschränken uns hier auf die wichtigsten Resource Records:

SOA	Start Of Authority	Beginn der Zuständigkeit
NS	Name Server	Adresse der Name Servers
A	Address	Name nach Adresse
PTR	Pointer	Adresse nach Name
CNAME	Canonical Name	Alias
MX	Mail Exchange	Adresse des Mailservers

Time to Live TTL

- Name Servers cachen die Daten nicht für immer
- Bei Aenderungen würden sonst diese die Clients nie erreichen
- TTL ist die Zeit, die der Name Server die Daten im Cache behalten darf
- Abwägung zwischen Performance und Konsistenz

Time to Live TTL

- Name Servers cachen die Daten nicht für immer
- Bei Aenderungen würden sonst diese die Clients nie erreichen
- TTL ist die Zeit, die der Name Server die Daten im Cache behalten darf
- Abwägung zwischen Performance und Konsistenz

Time to Live TTL

- Name Servers cachen die Daten nicht für immer
- Bei Aenderungen würden sonst diese die Clients nie erreichen
- TTL ist die Zeit, die der Name Server die Daten im Cache behalten darf
- Abwägung zwischen Performance und Konsistenz

Time to Live TTL

- Name Servers cachen die Daten nicht für immer
- Bei Aenderungen würden sonst diese die Clients nie erreichen
- TTL ist die Zeit, die der Name Server die Daten im Cache behalten darf
- Abwägung zwischen Performance und Konsistenz

Wieso BIND?

- Noch immer etwa 70 % aller Namen werden mit BIND Server verwaltet
- Ist auf allen Systemen verfügbar
- Einfach über Textdateien konfigurierbar
- Es gibt auch GUI Frontends
- Gute Dokumentation

Wieso BIND?

- Noch immer etwa 70 % aller Namen werden mit BIND Server verwaltet
- Ist auf allen Systemen verfügbar
- Einfach über Textdateien konfigurierbar
- Es gibt auch GUI Frontends
- Gute Dokumentation

Wieso BIND?

- Noch immer etwa 70 % aller Namen werden mit BIND Server verwaltet
- Ist auf allen Systemen verfügbar
- Einfach über Textdateien konfigurierbar
- Es gibt auch GUI Frontends
- Gute Dokumentation

Wieso BIND?

- Noch immer etwa 70 % aller Namen werden mit BIND Server verwaltet
- Ist auf allen Systemen verfügbar
- Einfach über Textdateien konfigurierbar
- Es gibt auch GUI Frontends
- Gute Dokumentation

Wieso BIND?

- Noch immer etwa 70 % aller Namen werden mit BIND Server verwaltet
- Ist auf allen Systemen verfügbar
- Einfach über Textdateien konfigurierbar
- Es gibt auch GUI Frontends
- Gute Dokumentation

Zone Data Files

- Reine Textdateien (plain vanilla ASCII)
- Resource Records beginnen immer in der ersten Spalte
- RR sind nur eine Zeile lang, ausser sie enden mit einer Klammer `(` , v.a. bei `SOA`
- Kommentare beginnen mit `;` und enden mit dem Zeilenende, leere Zeilen send erlaubt.
- Konvention: geordnet nach `SOA`, `NS`, other, `A`, `PTR`, `CNAME`
- Konvention: Namen klein geschrieben
- Auf Knoppix/Debian zu finden unter `/etc/bind`, Konvention `db.zone`

Zone Data Files

- Reine Textdateien (plain vanilla ASCII)
- Resource Records beginnen immer in der ersten Spalte
- RR sind nur eine Zeile lang, ausser sie enden mit einer Klammer `(` , v.a. bei `SOA`
- Kommentare beginnen mit `;` und enden mit dem Zeilenende, leere Zeilen send erlaubt.
- Konvention: geordnet nach `SOA`, `NS`, other, `A`, `PTR`, `CNAME`
- Konvention: Namen klein geschrieben
- Auf Knoppix/Debian zu finden unter `/etc/bind`, Konvention `db.zone`

Zone Data Files

- Reine Textdateien (plain vanilla ASCII)
- Resource Records beginnen immer in der ersten Spalte
- RR sind nur eine Zeile lang, ausser sie enden mit einer Klammer `(` , v.a. bei `SOA`
- Kommentare beginnen mit `;` und enden mit dem Zeilenende, leere Zeilen send erlaubt.
- Konvention: geordnet nach `SOA`, `NS`, other, `A`, `PTR`, `CNAME`
- Konvention: Namen klein geschrieben
- Auf Knoppix/Debian zu finden unter `/etc/bind`, Konvention `db.zone`

Zone Data Files

- Reine Textdateien (plain vanilla ASCII)
- Resource Records beginnen immer in der ersten Spalte
- RR sind nur eine Zeile lang, ausser sie enden mit einer Klammer `(` , v.a. bei `SOA`
- Kommentare beginnen mit `;` und enden mit dem Zeilenende, leere Zeilen send erlaubt.
- Konvention: geordnet nach `SOA`, `NS`, other, `A`, `PTR`, `CNAME`
- Konvention: Namen klein geschrieben
- Auf Knoppix/Debian zu finden unter `/etc/bind`, Konvention `db.zone`

Zone Data Files

- Reine Textdateien (plain vanilla ASCII)
- Resource Records beginnen immer in der ersten Spalte
- RR sind nur eine Zeile lang, ausser sie enden mit einer Klammer `(` , v.a. bei `SOA`
- Kommentare beginnen mit `;` und enden mit dem Zeilenende, leere Zeilen send erlaubt.
- Konvention: geordnet nach `SOA`, `NS`, other, `A`, `PTR`, `CNAME`
- Konvention: Namen klein geschrieben
- Auf Knoppix/Debian zu finden unter `/etc/bind`, Konvention `db.zone`

Zone Data Files

- Reine Textdateien (plain vanilla ASCII)
- Resource Records beginnen immer in der ersten Spalte
- RR sind nur eine Zeile lang, ausser sie enden mit einer Klammer `(` , v.a. bei `SOA`
- Kommentare beginnen mit `;` und enden mit dem Zeilenende, leere Zeilen send erlaubt.
- Konvention: geordnet nach `SOA`, `NS`, other, `A`, `PTR`, `CNAME`
- Konvention: Namen klein geschrieben
- Auf Knoppix/Debian zu finden unter `/etc/bind`, Konvention `db.zone`

Zone Data Files

- Reine Textdateien (plain vanilla ASCII)
- Resource Records beginnen immer in der ersten Spalte
- RR sind nur eine Zeile lang, ausser sie enden mit einer Klammer `(` , v.a. bei `SOA`
- Kommentare beginnen mit `;` und enden mit dem Zeilenende, leere Zeilen send erlaubt.
- Konvention: geordnet nach `SOA`, `NS`, other, `A`, `PTR`, `CNAME`
- Konvention: Namen klein geschrieben
- Auf Knoppix/Debian zu finden unter `/etc/bind`, Konvention `db.`*`zone`*

SOA Ressource Record

Name	der Zone
IN	Zonenklasse (meist IN für Internet)
SOA	Start Of Authority
Primary	Primary Master für diese Zone
E-Mail	des Verantwortlichen für diese Zone
Serial	bei jeder Änderung inkrem. (meist YYYYMMDDVV)
Refresh	Zeit [s] in dem die Slaves Aenderungen abfragen
Retry	Zeit [s] Wiederholung von nicht beantw. Anfragen
Expire	Zeit [s] bis Domain dekativiert
TTL	negativ-Caching-TTL (für nicht exist. Domains)

SOA Beispiel

```
mydomain.ch. IN SOA ns.mydomain.ch. me.mydomain.ch. (
                    2010082901 ; serial
                    10800      ; refresh 3 h (24 h)
                    3600       ; retry 1 h   (2 h)
                    60480      ; expire 7 d  (30 d)
                    86400 )    ; TTL 1 d     (4 d)
```

NS Ressource Record

Domain	Name der der Zone
TTL	Time to Live [s], optional
IN	Class, optional
NS	Name Service
Server	Name des autorativen Name Servers

```
mydomain.ch. 3600 IN NS ns1.mydomain.ch.   ; Master
mydomain.ch. 3600 IN NS ns2.mydomain.ch.   ; Slave

# Delegation
other.mydomain.ch 3600 IN NS ns1.other.mydomain.
```

A Ressource Record

Name	veröffentlichter Name
TTL	Time to Live [s], optional
IN	Class, optional
A	Record Type
Address	Die IP-Adresse des Namens

Beispiele:

```
www.mydomain.ch. 3600 IN A 193.5.54.111
```

```
$TTL 3600
$ORIGIN mydomain.ch.
www A 193.5.54.111
```

PTR Ressource Record

IP-Address	IP-Adresse in umgekehrter Reihenfolge
TTL	Time to Live [s], optional
IN	Class, optional
PTR	Record Type
Name	der zugeordnete Name

Beispiele:

```
111.193.5.54.in-addr.arpa. 3600 IN PTR www.mydomain.ch.

$TTL 3600
$ORIGIN 193.5.54.in-addr.arpa.
111 IN PTR www.mydomain.ch.
```

CNAME Ressource Record

Alias	veröffentlichter Name (Alias)
TTL	Time to Live [s], optional
IN	Class, optional
CNAME	Type
Name	Kanonischer Name

Für gleichen Namen dürfen beliebig viele Aliase existieren. Verkettete Aliase sind nicht erlaubt (funktionieren aber..). Es dürfen natürlich auch Namen anderer Domänen verwendet werden.

```
intranet.mydomain.ch. 3600 IN CNAME www.mydomain.ch.

$TTL 3600
$ORIGIN mydomain.ch.
intranet CNAME www.hsz-t.ch.
```

Beispiel einer einfachen Zonendatei

```
$TTL      3h                    ; BIND ab 8.2 braucht TTL
$ORIGIN   hsz-t.ch.             ; Hier beginnt Zone

@         IN SOA ns.mydomain.ch. me.mydomain.ch. (
                      2010082901 ; serial
                      10800      ; refresh 3 h
                      3600       ; retry 1 h
                      60480      ; expire 7 d
                      86400 )    ; TTL 1 d

; Authorative Name Servers
@         IN NS   ns1.mydomain.ch.   ; Master
@         IN NS   ns2.mydomain.ch.   ; Slave
```

Beispiel einer einfachen Zonendatei, cont.

```
; Mailserver
@          IN MX 20 mail.mydomain.ch.
@          IN MX 50 mail2.mydomain.ch.

; Hosts
www        IN A    193.5.54.111
mail       IN A    193.5.54.100
mail2      IN A    193.5.54.101
ns1        IN A    193.5.54.102
ns2        IN A    201.18.13.20

; Aliases
intranet CNAME   www.hsz-t.ch.
```

Bereits eingerichtete db-Dateien

- `db.empty` Reverse db für leere rfc1918 Zone
- `db.0` und `db.255` Broadcast Zone (reverse)
- `db.127` und `db.local` Local Loopback Address

Bereits eingerichtete db-Dateien

- `db.empty` Reverse db für leere rfc1918 Zone
- `db.0` und `db.255` Broadcast Zone (reverse)
- `db.127` und `db.local` Local Loopback Address

Bereits eingerichtete db-Dateien

- `db.empty` Reverse db für leere rfc1918 Zone
- `db.0` und `db.255` Broadcast Zone (reverse)
- `db.127` und `db.local` Local Loopback Address

Root Hints Data (`db.root`)

- Der Name Server muss ja wissen, wo die Root Name Server zu finden sind
- Momentan existieren 13 Root-Server (A bis M).
- Ihre Namen haben die Form `buchstabe.root-servers.net.`
- Bei den meisten Installationen schon eingerichtet. Kann auch heruntergeladen werden von `ftp.rs.internic.net/domain/db.cache` und wird deshalb auch `db.cache` genannt

Root Hints Data (`db.root`)

- Der Name Server muss ja wissen, wo die Root Name Server zu finden sind
- Momentan existieren 13 Root-Server (A bis M).
- Ihre Namen haben die Form `buchstabe.root-servers.net.`
- Bei den meisten Installationen schon eingerichtet. Kann auch heruntergeladen werden von `ftp.rs.internic.net/domain/db.cache` und wird deshalb auch `db.cache` genannt

Root Hints Data (`db.root`)

- Der Name Server muss ja wissen, wo die Root Name Server zu finden sind
- Momentan existieren 13 Root-Server (A bis M).
- Ihre Namen haben die Form `buchstabe.root-servers.net.`
- Bei den meisten Installationen schon eingerichtet. Kann auch heruntergeladen werden von `ftp.rs.internic.net/domain/db.cache` und wird deshalb auch `db.cache` genannt

Root Hints Data (`db.root`)

- Der Name Server muss ja wissen, wo die Root Name Server zu finden sind
- Momentan existieren 13 Root-Server (A bis M).
- Ihre Namen haben die Form `buchstabe.root-servers.net.`
- Bei den meisten Installationen schon eingerichtet. Kann auch heruntergeladen werden von `ftp.rs.internic.net/domain/db.cache` und wird deshalb auch `db.cache` genannt

Konfigurationsfile (`named.conf`)

- Wird von BIND normalerweise unter `/etc/named.conf` erwartet. Bei Debian in `/etc/bind/`
- Kommentar wie bei C/C++ (`//` und `/* */`) und Shell (`#`)
- Zuerst die `options` und dann die Zonen. `db.root`, `db.local` usw. sind bereits eingetragen.
- Bei einem Caching-Only Server braucht es keine zusätzlichen Zonendefinionen.
- Autorative Zonen kommen am Anfang der Zonendefinition.
- Verwendung von `include` um z.B. `named.conf.options`, `named.conf.local` usw. zu laden (z.B. bei Debian/Knoppix)

Konfigurationsfile (`named.conf`)

- Wird von BIND normalerweise unter `/etc/named.conf` erwartet. Bei Debian in `/etc/bind/`
- Kommentar wie bei C/C++ (`//` und `/* */`) und Shell (`#`)
- Zuerst die `options` und dann die Zonen. `db.root`, `db.local` usw. sind bereits eingetragen.
- Bei einem Caching-Only Server braucht es keine zusätzlichen Zonendefinionen.
- Autorative Zonen kommen am Anfang der Zonendefinition.
- Verwendung von `include` um z.B. `named.conf.options`, `named.conf.local` usw. zu laden (z.B. bei Debian/Knoppix)

HOCHSCHULE FÜR TECHNIK ZÜRICH

Konfigurationsfile (`named.conf`)

- Wird von BIND normalerweise unter `/etc/named.conf` erwartet. Bei Debian in `/etc/bind/`
- Kommentar wie bei C/C++ (`//` und `/* */`) und Shell (`#`)
- Zuerst die `options` und dann die Zonen. `db.root`, `db.local` usw. sind bereits eingetragen.
- Bei einem Caching-Only Server braucht es keine zusätzlichen Zonendefinionen.
- Autorative Zonen kommen am Anfang der Zonendefinition.
- Verwendung von `include` um z.B. `named.conf.options`, `named.conf.local` usw. zu laden (z.B. bei Debian/Knoppix)

HOCHSCHULE FÜR TECHNIK ZÜRICH

Konfigurationsfile (`named.conf`)

- Wird von BIND normalerweise unter `/etc/named.conf` erwartet. Bei Debian in `/etc/bind/`
- Kommentar wie bei C/C++ (`//` und `/* */`) und Shell (`#`)
- Zuerst die `options` und dann die Zonen. `db.root`, `db.local` usw. sind bereits eingetragen.
- Bei einem Caching-Only Server braucht es keine zusätzlichen Zonendefinionen.
- Autorative Zonen kommen am Anfang der Zonendefinition.
- Verwendung von `include` um z.B. `named.conf.options`, `named.conf.local` usw. zu laden (z.B. bei Debian/Knoppix)

HOCHSCHULE FÜR TECHNIK ZÜRICH

Konfigurationsfile (`named.conf`)

- Wird von BIND normalerweise unter `/etc/named.conf` erwartet. Bei Debian in `/etc/bind/`
- Kommentar wie bei C/C++ (`//` und `/* */`) und Shell (`#`)
- Zuerst die `options` und dann die Zonen. `db.root`, `db.local` usw. sind bereits eingetragen.
- Bei einem Caching-Only Server braucht es keine zusätzlichen Zonendefinionen.
- Autorative Zonen kommen am Anfang der Zonendefinition.
- Verwendung von `include` um z.B. `named.conf.options`, `named.conf.local` usw. zu laden (z.B. bei Debian/Knoppix)

Konfigurationsfile (`named.conf`)

- Wird von BIND normalerweise unter `/etc/named.conf` erwartet. Bei Debian in `/etc/bind/`
- Kommentar wie bei C/C++ (`//` und `/* */`) und Shell (`#`)
- Zuerst die `options` und dann die Zonen. `db.root`, `db.local` usw. sind bereits eingetragen.
- Bei einem Caching-Only Server braucht es keine zusätzlichen Zonendefinionen.
- Autorative Zonen kommen am Anfang der Zonendefinition.
- Verwendung von `include` um z.B. `named.conf.options`, `named.conf.local` usw. zu laden (z.B. bei Debian/Knoppix)

HOCHSCHULE FÜR TECHNIK ZÜRICH

Beispiel `named.conf` für Primary (Master)

```
options {
        directory "/var/named";
        allow-query { any; };
        allow-transfer { 193.5.54.102; };
};

zone "mydomain.ch" in {
        type master;
        file "db.mydomain";
};
```

Beispiel `named.conf` für Secondary (Slave)

```
options {
        directory "/var/named";
};

zone "mydomain.ch" in {
        type slave;
        file "bak.mydomain";
        masters  "193.5.54.101;
};
```

Diagnostic Tools

- `nslookup` Auf fast allen Systemen verfügbar (auch auf Windows). Benutzerschnittstelle nicht immer konsistent.
- `dig` Auf UNIX/Linux standardmässig dabei. Klare Benutzerschnittstelle.
- `host` Sehr einfaches und einfach anwendbares Tool, dafür etwas beschränkt in den Möglichkeiten.
- Auf Solaris sind die Tools in `/usr/sbin` zu finden

Diagnostic Tools

- `nslookup` Auf fast allen Systemen verfügbar (auch auf Windows). Benutzerschnittstelle nicht immer konsistent.
- `dig` Auf UNIX/Linux standardmässig dabei. Klare Benutzerschnittstelle.
- `host` Sehr einfaches und einfach anwendbares Tool, dafür etwas beschränkt in den Möglichkeiten.
- Auf Solaris sind die Tools in `/usr/sbin` zu finden

Diagnostic Tools

- `nslookup` Auf fast allen Systemen verfügbar (auch auf Windows). Benutzerschnittstelle nicht immer konsistent.
- `dig` Auf UNIX/Linux standardmässig dabei. Klare Benutzerschnittstelle.
- `host` Sehr einfaches und einfach anwendbares Tool, dafür etwas beschränkt in den Möglichkeiten.
- Auf Solaris sind die Tools in `/usr/sbin` zu finden

Diagnostic Tools

- `nslookup` Auf fast allen Systemen verfügbar (auch auf Windows). Benutzerschnittstelle nicht immer konsistent.
- `dig` Auf UNIX/Linux standardmässig dabei. Klare Benutzerschnittstelle.
- `host` Sehr einfaches und einfach anwendbares Tool, dafür etwas beschränkt in den Möglichkeiten.
- Auf Solaris sind die Tools in `/usr/sbin` zu finden

nslookup Beispiele

```
$ nslookup www.hsz-t.ch
Server:         193.5.54.121
Address:        193.5.54.121#53

www.hsz-t.ch    canonical name = patty.hsz-t.ch.
Name:   patty.hsz-t.ch
Address: 193.5.54.123

$ nslookup www.switch.ch
Server:         193.5.54.121
Address:        193.5.54.121#53

Non-authoritative answer:
www.switch.ch   canonical name = aslan.switch.ch.
Name:   aslan.switch.ch
Address: 130.59.108.36
```

`nslookup` Beispiel, explizite Angabe NS

```
$ nslookup www.switch.ch - ns.switch.ch
Server:         ns.switch.ch
Address:        130.59.10.30#53

www.switch.ch   canonical name = aslan.switch.ch.
Name:   aslan.switch.ch
Address: 130.59.108.36

$ nslookup aslan.switch.ch. - ns.switch.ch
Server:         ns.switch.ch
Address:        130.59.10.30#53

Name:   aslan.switch.ch
Address: 130.59.108.36
```

`nslookup` Beispiel, interaktiv

```
$ nslookup
> set type=MX
> ubs.com
Server:         193.5.54.121
Address:        193.5.54.121#53

Non-authoritative answer:
ubs.com mail exchanger = 100 ubs.com.s201a1.psmtp.com.
ubs.com mail exchanger = 200 ubs.com.s201a2.psmtp.com.
ubs.com mail exchanger = 300 ubs.com.s201b1.psmtp.com.
ubs.com mail exchanger = 400 ubs.com.s201b2.psmtp.com.

Authoritative answers can be found from:
ubs.com nameserver = jupiter.ubs.com.
ubs.com nameserver = mercury.ubs.com.
> exit
```

HOCHSCHULE FÜR TECHNIK ZÜRICH

dig Beispiel

```
$ dig +nocmd hsz-t.ch any +multiline +noall +answer
hsz-t.ch.    10800 IN A 193.5.54.123
hsz-t.ch.    10800 IN SOA lisa.isz.ch. hostmaster.hsz-t.c
                   2010090101 ; serial
                   10800      ; refresh (3 hours)
                   3600       ; retry (1 hour)
                   604800     ; expire (1 week)
                  86400       ; minimum (1 day)
                  )
hsz-t.ch.    10800 IN NS lisa.isz.ch.
hsz-t.ch.    10800 IN NS scsnms.switch.ch.
hsz-t.ch.    10800 IN MX 20 jay.hsz-t.ch.
```

Siehe auch `http://www.madboa.com/geek/dig/`

HOCHSCHULE FÜR TECHNIK ZÜRICH

Admin Tools

- `named-checkconf` überprüfen des Konfigurationsfiles typischerweise named.conf auf syntaktische Fehler. Bevor BIND neu gestratet wird, unbedingt Konfiguration überprüfen.
- `named-checkzone` überprüfen der Zonendatei auf syntaktische Fehler.

Weitere Infos wie immer über `man named-checkconf` bzw. `man named-checkzone`.

Admin Tools

- `named-checkconf` überprüfen des Konfigurationsfiles typischerweise named.conf auf syntaktische Fehler. Bevor BIND neu gestratet wird, unbedingt Konfiguration überprüfen.
- `named-checkzone` überprüfen der Zonendatei auf syntaktische Fehler.

Weitere Infos wie immer über `man named-checkconf` bzw. `man named-checkzone`.

Webbasierende Tools

- `http://www.intodns.com` Viele gute Infos v.a. Sicherheitsrelevantes.
- `https://my.ip-plus.net/tools/dns_check/index.en.mpl` Zonentransfer für Testserver muss freigeschaltet sein. Detaillierte Infos. Online dig: `https://my.ip-plus.net/tools/dig_dns/index.en.mpl`
- `http://www.allwhois.com` Zeigt Whois-Einträge an.

Webbasierende Tools

- `http://www.intodns.com` Viele gute Infos v.a. Sicherheitsrelevantes.
- `https://my.ip-plus.net/tools/dns_check/index.en.mpl`
  Zonentransfer für Testserver muss freigeschaltet sein.
  Detaillierte Infos. Online dig:
  `https://my.ip-plus.net/tools/dig_dns/index.en.mpl`
- `http://www.allwhois.com` Zeigt Whois-Einträge an.

Webbasierende Tools

- `http://www.intodns.com` Viele gute Infos v.a. Sicherheitsrelevantes.
- `https://my.ip-plus.net/tools/dns_check/index.en.mpl` Zonentransfer für Testserver muss freigeschaltet sein. Detaillierte Infos. Online dig: `https://my.ip-plus.net/tools/dig_dns/index.en.mpl`
- `http://www.allwhois.com` Zeigt Whois-Einträge an.

Sicherheit

- Sicherheit wurde bei der Konzeption des DNS nicht in das Design einbezogen.
- Einsatz von ACLs für Queries und Transfers
- Versteckte Master-Server
- Aufteilung in interne und externe Sicht
- Firewalls
- Eigene (interne) Root-Servers

Sicherheit

- Sicherheit wurde bei der Konzeption des DNS nicht in das Design einbezogen.
- Einsatz von ACLs für Queries und Transfers
- Versteckte Master-Server
- Aufteilung in interne und externe Sicht
- Firewalls
- Eigene (interne) Root-Servers

Sicherheit

- Sicherheit wurde bei der Konzeption des DNS nicht in das Design einbezogen.
- Einsatz von ACLs für Queries und Transfers
- Versteckte Master-Server
- Aufteilung in interne und externe Sicht
- Firewalls
- Eigene (interne) Root-Servers

Sicherheit

- Sicherheit wurde bei der Konzeption des DNS nicht in das Design einbezogen.
- Einsatz von ACLs für Queries und Transfers
- Versteckte Master-Server
- Aufteilung in interne und externe Sicht
- Firewalls
- Eigene (interne) Root-Servers

Sicherheit

- Sicherheit wurde bei der Konzeption des DNS nicht in das Design einbezogen.
- Einsatz von ACLs für Queries und Transfers
- Versteckte Master-Server
- Aufteilung in interne und externe Sicht
- Firewalls
- Eigene (interne) Root-Servers

Sicherheit

- Sicherheit wurde bei der Konzeption des DNS nicht in das Design einbezogen.
- Einsatz von ACLs für Queries und Transfers
- Versteckte Master-Server
- Aufteilung in interne und externe Sicht
- Firewalls
- Eigene (interne) Root-Servers