

**Seminar User Mode Linux
Uni Koblenz-Landau
WS 05/06**

Domain Name System in VNUML

Rufus Linke
20.01.2006

Inhaltsverzeichnis

1 DNS Funktionsweise	3
1.1 Übersicht DNS	3
1.2 Die Baumstruktur des DNS-Namensraums	3
1.3 Domains	4
1.4 Nameserver	4
1.5 Unterschied: Zone – Domain	5
1.6 Die Zonendateien	6
1.7 Resolver und Namensauflösung	6
1.8 Caching	7
1.9 Die in-addr.arpa-Domain	8
2 Implementation: BIND	9
2.1 Übersicht BIND	9
2.2 Syntax der Zonendateien	9
2.2.1 SOA-Resource Record	10
2.2.2 NS-Resource Record	10
2.2.3 A-Resource Record	11
2.2.4 PTR-Resource Record	11
2.3 named.conf	11
2.4 Resolver: dig	12
3 Beispielnetz in VNUML	12
3.1 VNUML-Konfigurationsdateien	13
3.2 BIND-Konfigurationsdateien	13
3.3 Beispielauflösungen im simulierten Netz	14
4 Literatur	18

1 DNS Funktionsweise

1.1 Übersicht DNS

Im Internet werden die beteiligten Rechner normalerweise über ihre IP-Adressen angesprochen. Jedoch sind 32-bit lange Zahlenketten für den Menschen nicht unbedingt leicht zu merken, weshalb das Domain Name System entwickelt wurde. Es dient dazu die IP-Adresse eines Rechners in einen besser lesbaren Domainnamen zu übersetzen. Dabei gibt es einerseits den forward lookup, bei dem der Domainname in die zugehörige IP-Adresse übersetzt wird, sowie den reverse lookup, bei dem ein zu einer IP-Adresse entsprechender Domainname gesucht wird.

Beim DNS handelt es sich also um eine große Datenbank, in der IP-Adressen Domainnamen zugeordnet werden, bzw. umgekehrt. In den Anfängen des Internet verwaltete man diese Datenbank noch mit Hilfe von hosts-Dateien, welche immer sämtliche Rechner im Netzwerk kannten. Dies ist jedoch denkbar ungünstig, da so immer alle Rechner die Datei halten müssen und sich die Synchronisation der einzelnen hosts-Dateien in einem so stark wachsenden Netzwerk wie dem Internet fast unmöglich gestaltet. Daher nutzt das DNS eine auf viele Rechner verteilte, also dezentrale Datenbank.

DNS baut als Protokoll der Anwendungsschicht auf die Protokolle TCP und UDP auf und nutzt standardmäßig Port 53. Die meisten DNS-Anfragen werden über UDP abgewickelt, jedoch wird bei umfangreichen Antworten über 512 bytes Größe auf TCP umgestellt, um eine gesicherte Übertragung zu gewährleisten. Außerdem kommt das TCP immer beim Zonentransfer (siehe 1.6) zum Einsatz.

Im folgenden werde ich nun die 3 Hauptkomponenten des DNS vorstellen: Den DNS-Namensraum, die DNS-Server und die Resolver.

1.2 Die Baumstruktur des DNS-Namensraums

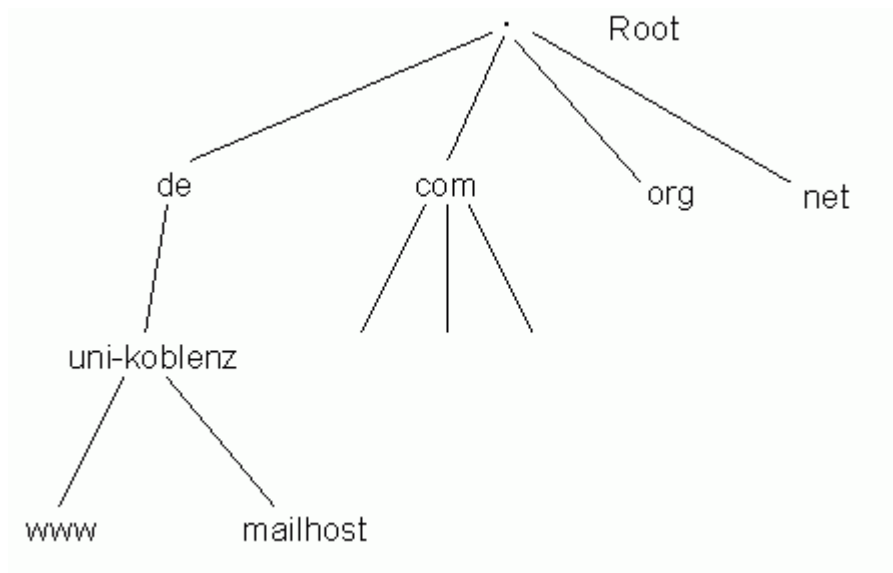


Abbildung 1: Beispiel: Domain-Baum

Der Namensraum des DNS ist baumartig aufgebaut. Als Wurzel steht dabei an der Spitze der Root-Knoten (.). Darunter geordnet sind dann weitere Unterteilungen des Baums in Subdomains. Die Struktur dieses Baumes ähnelt also der eines Unix-Dateisystems. Dort ist der / die oberste Wurzel und ihr untergeordnet sind mehrere Unterverzeichnisse, die wiederum eigene Unterverzeichnisse haben können. Gelesen wird ein Domainname immer von unten (vom Knoten) nach oben (zur Wurzel). Normalerweise gibt man Domainnamen ohne den abschließenden Punkt des Root-Knotens an, zu einem vollen Domainnamen gehört er jedoch eigentlich dazu. Gibt man den Root-Punkt mit an, spricht man von einem Fully Qualified Domain Name (FQDN). Z.B. ist der FQDN des Rechners mailhost im obigen Beispiel mailhost.uni-koblenz.de. .

1.3 Domains

Eine Domain ist nun definiert als ein Unterbaum dieses Namensbaumes. Dabei erhält die Domain den Namen des obersten Knotens in diesem Unterbaum. Alle weiteren Knoten, die sich in diesem Unterbaum befinden, sind Teil dieser Domain. Nimmt man beispielweise den Unterbaum ab dem Knoten uni-koblenz in Abbildung 1, dann hat die Domain den Namen uni-koblenz.de. (Name des obersten Knotens im Unterbaum) und die Hosts www.uni-koblenz.de. und mailhost.uni-koblenz.de. gehören auch mit zur Domain.

Wie man hier schon erkennt repräsentieren die einzelnen Domainnamen des Baumes die Hosts des Netzes (sie zeigen also auf die Netzwerkadressen der Hosts). Dabei spiegeln die Blätter des Baumes meist nur einzelne Hosts wider, während die Knoten im Inneren des Baumes sowohl Informationen über den Host hinter dem Namen als auch weitere Details über die Domain beinhalten können. Die Art der zurückgegebenen Information ist dann abhängig von der gestellten Anfrage.

1.4 Nameserver

Bei einem Nameserver handelt es sich nun um die Software, die diese Datenbank verwaltet und Anfragen an sie beantwortet. Da es sich beim DNS wie anfangs bereits erwähnt um eine dezentrale Datenbank handelt, verwaltet jeder Server nur einen Teil der gesamten DNS-Datenbank, eine sogenannte Zone. Ein Server der eine Zone verwaltet heißt auch autoritativ für diese Zone (oder er besitzt die Autorität für diese Zone). Ein Nameserver kann auch für mehrere Zonen autoritativ sein. Für jede Zone, für die ein Server die Autorität besitzt, verwaltet er sämtliche Domainnamen in dieser Zone und kann Anfragen nach Hosts in dieser Zone beantworten.

Nun kommt das Prinzip der Delegation ins Spiel: Dadurch, dass Nameserver nur für einen Teil des Namensraumes, eine Zone, verantwortlich sind, können sie Anfragen nach Hosts in anderen Teilen des Netzes von sich aus nicht beantworten. Dann werden andere Nameserver gefragt, die für den Teil in dem sich der gesuchte Host befindet, autoritativ sind. Delegation bedeutet nun, dass ein Nameserver die Verwaltung einer Subdomain an einen anderen Server abgibt. Auf's Beispiel in Abbildung 1 bezogen würde hier der Nameserver der de.-Domain die Verwaltung der Subdomain uni-koblenz.de. an den Nameserver dieser Domain abgeben. Anfragen an die Domain uni-koblenz.de., die an den de.-Server gestellt werden, werden damit an den uni-koblenz.de.-Server weitergeleitet, der diese dann weiter beantworten kann, da er die

Autorität für diese Zone besitzt. Somit braucht sich der Nameserver der de.-Domain nicht mehr um weitere, im Baum tieferliegende Hosts zu kümmern. Er muss nur noch alle Server, die auf der ersten Ebene in der de.-Domain liegen kennen und kann dann Anfragen an Hosts aus diesen Netzen an die entsprechenden Server delegieren.

1.5 Unterschied: Zone - Domain

Da wir jetzt zwei Begriffe für Teile des Namensraums hatten, möchte ich kurz den Unterschied zwischen diesen beiden erläutern. Dazu folgende Abbildung:

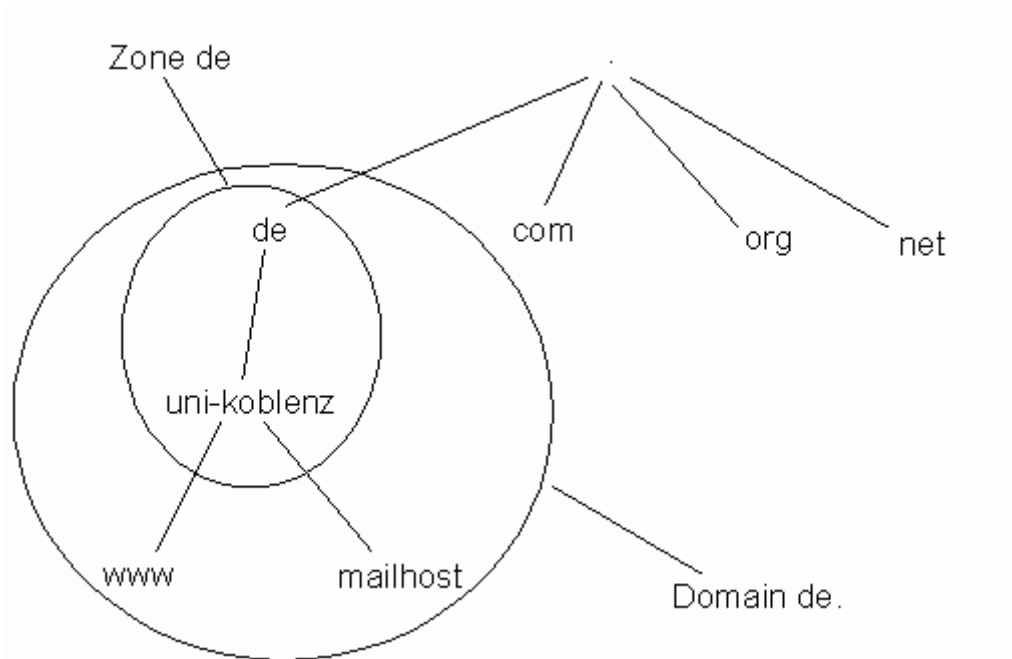


Abbildung 2: Zone und Domain

Wie schon erklärt beinhaltet in diesem Beispiel die Domain de. sämtliche Domains im Unterbaum des Knoten de. Die Zone de umfasst allerdings nur diejenigen Hosts, für die der Nameserver der Zone autoritativ ist, also die er verwaltet. Da er jeweils nur die Hosts kennt, die eine Ebene unter ihm liegen und alle tiefergehenden Anfragen an andere Nameserver delegiert, sind dies nicht alle Rechner in der Domain de., sondern eben nur ein Teil von diesen, die Zone de.

1.6 Die Zonendateien

Die Konfiguration einer Zone erfolgt nun mittels der Resource Record-Einträge in den Zonendateien des Nameservers. Ein Resource Record ist dabei ein Datenbankeintrag, beispielsweise die Zuweisung einer IP-Adresse zu einem Hostnamen oder die Festlegung eines Nameservers für diese Zone. Weitere Beispiele zum Aufbau der Zonendatei und den Resource Records folgen in Abschnitt 2.2.

Da es sich beim DNS um ein dezentrales System handelt, liegen diese Zonefiles demnach meist nicht nur auf einem Server, sondern werden auf Primary sowie Secondary Nameservern verwaltet, um eine höhere Ausfallsicherheit und bessere Lastverteilung zu erzielen. Der Primary Nameserver besitzt dabei die Ursprungsdateien und verteilt Kopien dieser an die Secondary Nameserver. Eine Zone muss logischerweise immer einen Primary Nameserver besitzen, da ansonsten kein Server für die Verwaltung der Zone zuständig wäre. Die Übertragung der Zonendateien auf einen Secondary Nameserver bezeichnet man als Zonentransfer. Es muss jedoch nicht immer der Primary Nameserver als Quelle für die Zonendateien dienen, die Secondary Nameserver können auch unter sich die Dateien austauschen. Dabei spricht man bei dem Rechner, der als Quelle dient, vom Master und bezeichnet den Server, der Daten von einem Master bezieht als Slave. Der Primary Nameserver ist demnach immer Master, denn auf ihm liegen die Ursprungsdaten und Secondary Nameserver können beim Zonentransfer sowohl Master als auch Slave sein. Zu beachten ist hier noch, dass die Secondary Nameserver für die entsprechende Zone ebenfalls autoritativ und damit in Bezug auf DNS-Anfragen gleichwertig mit dem Primary Nameserver sind.

1.7 Resolver und Namensauflösung

Auf der Clientseite steht beim DNS der Resolver. Der Resolver ist ein Programm, das Anfragen an die Nameserver stellt und die zurückbekommenen Antworten weiterverarbeitet oder an die anfragende Anwendung weiterleitet. Zur Namensauflösung im DNS gibt es dann verschiedene Verfahren:

Falls eine Anfrage nach einem Host, der in der Zone des befragten Nameservers liegt, gestellt wird, kann er diese direkt aus seiner Datenbank beantworten, da er für die Zone autoritativ, also verantwortlich ist. Kennt der befragte Nameserver die Antwort auf die Anfrage nicht, wird meist eine rekursive Auflösung durchgeführt. Dabei fragt der Nameserver den in der Hierarchie des Namensbaums eine Stufe höher stehenden Nameserver nach der Anfrage. Nun gibt es folgende Möglichkeiten:

1. Der befragte höher stehende Nameserver kennt die Antwort, da er für die Zone autoritativ ist
-> zurückgeben der Antwort an den ursprünglich befragten Nameserver, dieser kann dann dem Resolver die Antwort liefern.
2. Der Nameserver kennt die Antwort nicht, kennt jedoch einen Nameserver, der für die gesuchte Zone autoritativ ist
-> zurückgeben des autoritativen Nameservers, dieser wird dann vom ursprünglich befragten Nameserver befragt
3. Der Nameserver kennt die Antwort nicht und kennt auch keinen autoritativen Nameserver
-> Befragung des in der Hierarchie nächsthöherstehenden Nameservers; rekursive Wiederholung des Verfahrens

Falls das Verfahren zur Spitze des Baumes gelangt, kommen die Root-Nameserver ins Spiel. Von diesen gibt es derzeit im Internet 13 (Root-Server A bis M) und sie können über sämtliche Top-Level-Domains Auskunft geben.

Bei einer iterativen Anfrage befragt ein Nameserver keine weiteren Nameserver, sondern liefert als Antwort nur einen Verweis auf einen Nameserver, der als nächstes befragt werden muss.

Existieren für eine Zone mehrere autoritative Server, so wird derjenige mit der niedrigsten Round Trip Time befragt.

Ein Beispiel:

Der Nameserver der Zone uni-koblenz.de wird nach der IP des Hosts www.google.com befragt. Unter Voraussetzung, dass der Eintrag noch nicht gecachet ist (siehe 1.8), wird er dann zunächst den Nameserver der Zone de befragen, da ihm weder die Zone google.com, noch die Zone com bekannt sind. Der Nameserver der Zone de wird dann (sofern er keinen Cache besitzt) einen Root-Nameserver nach dem autoritativen Nameserver für die Zone com befragen. Diese Antwort liefert er an den Nameserver von uni-koblenz.de zurück. Als nächstes kann dieser jetzt den Nameserver der Zone com nach dem autoritativen Nameserver für google.com befragen. Schließlich kann der Nameserver in uni-koblenz.de den Nameserver von google.com nach der Adresse des Hosts www.google.com befragen und die Antwort an den Resolver zurückgeben.

1.8 Caching

Bei jeder Anfrage erneut den ganzen Baum rekursiv zu durchlaufen wäre jedoch viel zu aufwändig. Daher werden einmal erhaltene Antworten im Speicher des Nameservers gecachet. Somit kann ein Nameserver auch auf Anfragen direkt antworten, die nicht an seine Zone gerichtet sind. Diese Cache-Einträge werden allerdings nur im Speicher gehalten, sie werden nirgends in Zonefiles oder Ähnliches geschrieben. Der Nameserver von dem die gecachten Daten stammen vergibt dabei für jeden Eintrag eine Time To Live, nach der sie verfallen und neu angefragt werden müssen.

Zu erwähnen bleibt hier noch, dass es auch das sogenannte negative Caching gibt. Kann eine Anfrage nicht beantwortet werden, weil der Eintrag nirgends existiert, erstellt der Nameserver einen negativen Cacheeintrag, so dass bei der nächsten Anfrage nach diesem nicht existierenden Eintrag direkt entsprechend geantwortet werden kann, ohne vorher wieder eine lange Rekursion zu starten.

1.9 Die in-addr.arpa-Domain

Wie bereits anfangs erwähnt gibt es auch die Möglichkeit der inversen Auflösung, also der Suche eines Domainnamens zu gegebener IP-Adresse. Dazu wurde die in-addr.arpa-Domain eingerichtet. Es handelt sich hierbei um einen eigenen Namensbaum, der genauso wie der normale Domainbaum aufgebaut ist, jedoch statt Domain bzw. Hostnamen IP-Adressen als Labels besitzt.

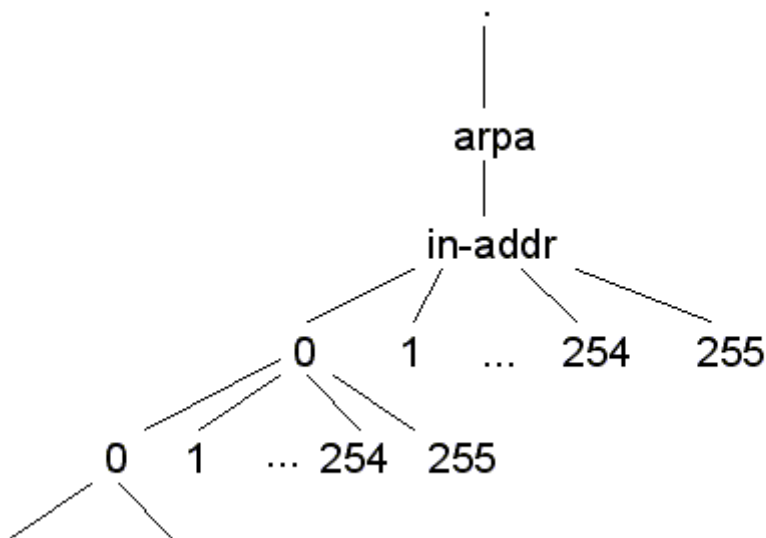


Abbildung 3: in-addr.arpa-Domain

Auch hier werden die Adressen von unten nach oben gelesen, so dass sich in der Schreibweise der Adressen eine umgekehrte Leserichtung ergibt. Beispielsweise würde die IP-Adresse 10.1.1.1 folgendermaßen notiert: 1.1.1.10.in-addr.arpa.

Diese Umkehrung ist auch notwendig, damit die Delegation funktionieren kann. Wie im normalen Namensbaum durchläuft ein Nameserver die Adresse von links nach rechts und befragt immer den in der Hierarchie nächsthöherliegenden Server, falls keine Antwort geliefert werden kann.

2. Implementation: BIND

2.1 Übersicht BIND

Als meist verbreitete Implementation des DNS kommt der Nameserver BIND (Berkeley Internet Name Domain) vor. Er wurde an der Universität Berkeley entwickelt und ist als Open Source verfügbar. Der Server läuft als Prozess `named` und wird folgendermaßen konfiguriert:

1. Die Zonefiles: pro Zone existiert eine Zonendatei
Format: `db.Zonename`
2. `named.conf`: Einstellungen für den Server und Zuordnung der Zonefiles zu den entsprechenden Zonen

2.2 Syntax der Zonendateien

Die Zonendateien besitzen wie schon erklärt die eigentlichen Einträge der DNS-Datenbank in Form von Resource Records. Dabei gibt es verschiedene Typen von Resource Records. Ich werde hier nur die vier wichtigsten Typen vorstellen, es existieren aber noch etliche weitere, die in der Dokumentation zu BIND näher erklärt werden.

- SOA-Record (Start Of Authority): Gibt den Primary Nameserver an
- NS-Record (NameServer): Gibt weitere autoritative (Secondary) Nameserver an
- A-Record (Address): Adresseintrag für einen Hostnamen
- PTR-Record (Pointer Resource Record): Eintrag für inverse lookup

Im Folgenden einige Beispiele aus dem Zonefile `db.e6` im verwendeten VNUML-Beispiel zu den genannten Resource Records.

2.2.1 SOA-Resource Record

```
$TTL 300
@           IN      SOA   ns1.e6.      root.ns1.e6. (
                1          ; serial
                3600       ; refresh (1hour)
                1800      ; retry (30 min)
                604800    ; expire (7 days)
                3600      ; minimum (1 hour)
                )
```

Die Direktive \$TTL 300 gibt an, dass für alle in diesem Zonefile verwendeten Einträge eine standard-TTL von 300 Sekunden gelten soll. An Stelle des @ müsste hier normalerweise der Zonennamen angegeben werden, jedoch wird hier da keine weitere Angabe gemacht wurde der in der named.conf für diese Zone definierte Name eingesetzt („origin“). Der Origin kann mittels der \$ORIGIN-Direktive auch manuell auf einen beliebigen Wert gesetzt werden. Hier wird also statt des @ der Zonename e6 eingesetzt, da dieser für die Datei db.e6 in der named.conf gesetzt wurde.

Weiterhin folgt IN für einen Internet-Eintrag, SOA um anzuzeigen, dass der folgende Server Primary Nameserver für diese Zone ist und dann der Name des Servers (ns1.e6). Der abschließende Punkt hinter dem Servernamen ist hier sehr wichtig, da ohne ihn der Zonename (origin) noch an den Server angehängt werden würde. root.ns1.e6. gibt den Ansprechpartner für diese Zone in Form einer eMail-Adresse an. Der erste Punkt wird durch ein @ ersetzt, die Adresse des Admins lautet dann also root@ns1.e6.

Dann folgt die Versionsnummer des Eintrags (serial), welche bei jeder Änderung inkrementiert wird, so dass Secondaries stets vergleichen können, ob sie die aktuelle Version besitzen. Schließlich werden noch verschiedene Times To Live angegeben. Die erste (refresh) gibt an, nach welcher Zeit die Slaves den Eintrag aktualisieren müssen. Der Zweite (retry) gibt die Zeit an, nach der Slaves bei nicht erreichbarem Master erneut versuchen sich zu aktualisieren. Expire ist die Zeit, nach der die Zone bei einem Slave abgeschaltet wird, falls kein Master mehr verfügbar ist und der letzte Eintrag (minimum) gibt die negativ caching-TTL an.

2.2.2 NS-Resource Record

```
e6.          IN      NS     ns1.e6.
upm.e6.     IN      NS     ns1.upm.e6.
umu.e6.     IN      NS     ns1.umu.e6.
```

e6. bezeichnet den Namen der Zone, IN steht wiederum für Internet, NS bedeutet ein Nameserver für die Zone wird hier definiert und ns1.e6. ist schließlich der Name des Servers.

Außerdem werden hier Anfragen an die Zonen upm.e6 und umu.e6 an die Nameserver ns1.upm.e6 und ns1.umu.e6 delegiert. Bekommt also der Nameserver dieser Zone (ns1.e6) eine Anfrage an eine dieser beiden Zonen, weiß er gleich, dass ns1.upm.e6 bzw. ns1.umu.e6 dafür verantwortlich sind und kann die Anfrage an den entsprechenden Server delegieren (weiterleiten).

2.2.3 A-Resource Record

```
ns1.e6.          IN   A           10.1.1.1
ns1.e6.          IN   AAAA        2001:db8::1
```

Hier wird dem Hostnamen ns1.e6 die IPv4-Adresse 10.1.1.1 zugewiesen. AAAA ist der entsprechende Resource Record für IPv6.

2.2.4 PTR-Resource Record

```
$ORIGIN 1.1.10.in-addr.arpa.
1      IN   PTR    ns1.e6.
2      IN   PTR    ns1.upm.e6.
```

Diese Einträge stammen aus der Datei db.10 und dienen der inversen Auflösung der Adressen 10.1.1.1 und 10.1.1.2. \$ORIGIN wird hier auf 1.1.10.in-addr.arpa gesetzt. Dadurch wird dies an allen Stellen eingesetzt, wo nur @ steht oder bei Einträgen, die nicht auf . enden. Alternativ hätte man auch schreiben können:

```
1.1.1.10.in-addr.arpa.  IN   PTR    ns1.e6.
2.1.1.10.in-addr.arpa.  IN   PTR    ns1.upm.e6.
```

Auf eine Anfrage nach dem Hostnamen zur IP-Adresse 10.1.1.2 würde dieser Nameserver also ns1.upm.e6 antworten können.

2.3 named.conf

Auszug aus der named.conf des Nameservers ns1.e6 im Beispiel:

```
options {
    directory "/etc/bind";
    port 53;
    pid-file "named.pid";
    listen-on { any; };
};

zone "e6" {
    type master;
    file "db.e6";
};

zone "10.in-addr.arpa" {
    type master;
    file "db.10";
};
```

Directory gibt den Speicherort der Zonefiles an. *Port* ist der Port für Anfragen an der Server, sollte standardmäßig immer auf 53 stehen. Im *pid-file* wird die Prozess-ID des named gespeichert und *listen-on {any}* besagt, dass der Nameserver Anfragen an allen Interfaces bearbeiten soll.

Danach folgt die Zuordnung der Zonefiles zu den Zonen. Für die Zone *e6* ist dieser Server master und die Resource Record-Einträge liegen in der Datei *db.e6*. Das gleiche darunter für die Zone *10.in-addr.arpa*.

2.4 Resolver: dig

Auch hier nur eine kleine Übersicht über die Möglichkeiten von *dig*, welcher viele weitere umfangreiche Optionen bietet, die beispielsweise auf der Manpage weiter erläutert werden. Die vereinfachte Syntax lautet:

```
dig [@server] [name] [type]
```

@server gibt den Server an, der befragt werden soll, ist keiner angegeben, wird der default-Server aus der *resolv.conf* befragt. *Name* ist der eigentliche Teil nach dem gefragt wird, beispielsweise ein Domainname oder eine IP-Adresse und mit *type* kann der Typ des Resource Records spezifiziert werden. Ist kein *type* angegeben, wird eine Anfrage nach einem A-Record ausgeführt. Ganz ohne Argumente befragt *dig* den standard-Nameserver nach dem Root-Knoten (.).

Um einen reverse lookup durchzuführen gibt man statt *name* die Option *-x address* an. Eine weitere Option, die ich später nutzen werde ist *+norecurse* um die Rekursion zu deaktivieren.

3. Beispielnetz in VNUML

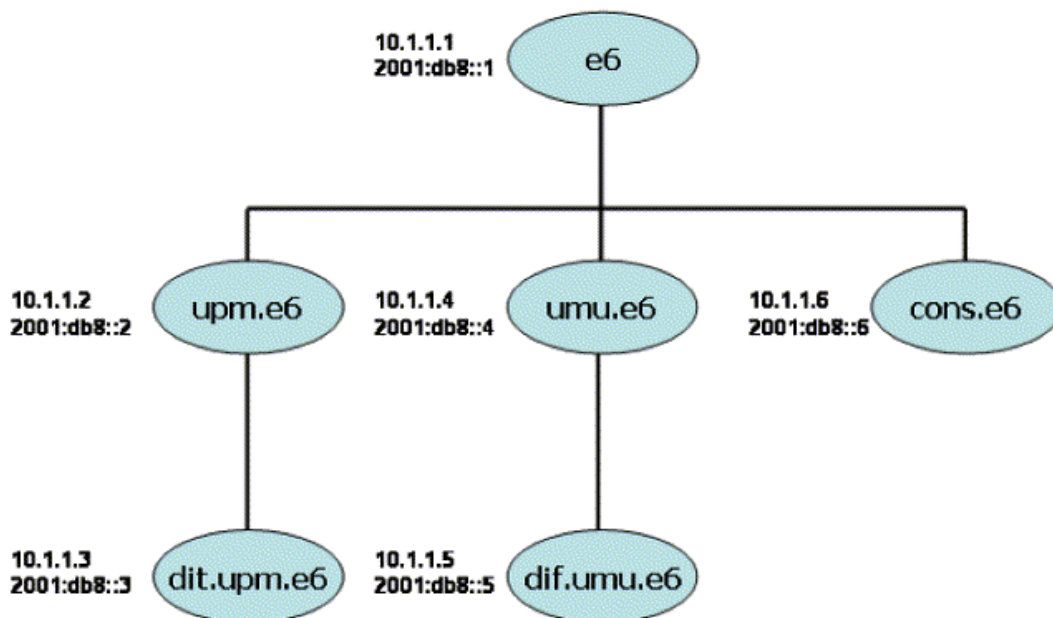


Abbildung 4: Beispielnetz

Das Netz aus Abbildung 4 wird in VNUML simuliert. In den Ovalen steht jeweils der Name der Zone, links daneben die IPv4 bzw IPv6-Adresse des Nameservers der Zone (jede Zone besitzt nur einen Nameserver).

3.1 VNUML-Konfigurationsdateien

Die VNUML-Szenariodatei dns.xml besitzt nicht viele Besonderheiten. Es werden 6 miteinander direkt vernetzte Rechner simuliert, die nachher als Nameserver dienen. Über start/stop-Skripte werden die named-Daemons gestartet und gestoppt.

```
<filetree when="start" root="/etc/bind">/usr/local/share/vnuml/examples/dns/ns1</filetree>  
<exec seq="start" type="verbatim">rm -f /etc/bind/*</exec>  
<exec seq="start" type="verbatim">mv /etc/bind/ns1/* /etc/bind</exec>  
<exec seq="start" type="verbatim">killall named</exec>  
<exec seq="start" type="verbatim">/usr/sbin/named -c /etc/bind/named.conf</exec>  
<exec seq="stop" type="verbatim">killall named</exec>
```

Über `<filetree>` wird die Konfiguration des jeweiligen named zunächst ins Unterverzeichnis `/etc/bind/nsx` der virtuellen Maschine eingebunden. Dann wird eine eventuell bestehende named-Konfiguration aus `/etc/bind` gelöscht und die neue hinein kopiert. Falls bereits ein named-läuft wird zur Sicherheit ein `killall named` durchgeführt um dann schließlich den Nameserver mit der gewünschten Konfiguration zu starten. Das stop-Kommando beendet lediglich den named.

3.2 BIND-Konfigurationsdateien

Die Konfiguration des BIND werde ich kurz am Beispiel des VNUML-Rechners e6 vorstellen, alle anderen Rechner sind ähnlich konfiguriert. Hier ist zu beachten, dass dieser virtuelle Rechner in der VNUML-Simulation e6 heißt, jedoch im simulierten DNS den Hostnamen ns1.e6 hat!

Die named.conf wurde bereits in 2.3 vorgestellt. Sie besitzt außer den dort stehenden Einträgen noch weitere Zoneneinträge für inverse Loopback-Auflösung in IPv4 und v6, eine inverse IPv6 Zone für das Netz 2001:db8::/32 und einen Eintrag für die Zone ., die auf das Zonefile root.hint hinweist. In dieser Datei stehen Einträge für die Root-Server. In diesem Beispiel gibt es nur einen Root-Server, ns1.e6.

Als Zonefiles liegen im Konfigurationsverzeichnis die Dateien db.10 für die Zone 10.0.0.0/8, db.8.b.d.0.1.0.0.2.ip6.arpa für die Zone 2001:db8::/32, db.e6 für die Zone e6 und localhost für die Loopback-Zone. Die Zonendateien sind entsprechend der Grafik konfiguriert, so dass e6 beispielsweise nur die Nameserver von upm.e6, umu.e6 und cons.e6 kennt. Außerdem sind in jeder Zone 3 Beispielhosts konfiguriert. Zum Beispiel in e6: h1.e6, h2.e6, h3.e6 mit den IP-Adressen 10.11.1.1, 10.11.1.2 und 10.11.1.3.

Zone	e6	upm.e6	dit.upm.e6	umu.e6	dif.umu.e6	cons.e6
Server	ns1.e6 10.1.1.1 2001:db8::1	ns1.upm.e6 10.1.1.2 2001:db8::2	ns1.dit.upm.e6 10.1.1.3 2001:db8::3	ns1.umu.e6 10.1.1.4 2001:db8::4	ns1.dif.umu.e6 10.1.1.5 2001:db8::5	ns1.cons.e6 10.1.1.6 2001:db8::6
Autorität für Zone	e6	upm.e6	dit.upm.e6	umu.e6	dif.umu.e6	cons.e6
Autorität für inverse Anfrage	10.0.0.0/8 2001:db8::/32	10.12.0.0/16 2001:db8:12::/48	10.12.10.0/24 2001:db8:12:1000::/56	10.13.0.0/16 2001:db8:13::/48	10.13.10.0/24 2001:db8:13:1000::/56	10.14.0.0/16 2001:db8:14::/48
Prefixes der lokalen Hosts	10.11.0.0/16 2001:db8:11::/48	10.12.1.0/24 2001:db8:12:100/56	10.12.10.0/24 2001:db8:12:1000::/56	10.13.1.0/24 2001:db8:13:100/56	10.13.10.0/24 2001:db8:13:1000::/56	10.14.0.0/16 2001:db8:14::/48
Beispiel-Host	h1.e6 10.11.1.1 2001:db8:11::1	h1.upm.e6 10.12.1.1 2001:db8:12:100::1	h1.dit.upm.e6 10.12.10.1 2001:db8:12:1000::1	h1.umu.e6 10.13.1.1 2001:db8:13:100::1	h1.dif.umu.e6 10.13.10.1 2001:db8:13:1000::1	h1.cons.e6 10.11.1.1 2001:db8:14:100::1

Die komplette Übersicht über alle Adressen und Nameserver hier noch einmal tabellarisch.

3.3 Beispielauflösungen im simulierten Netz

Zunächst wird das VNUML-Szenario gestartet

```
vnumlparser.pl -t dns.xml
```

Anschließend werden über das start-Skript die Nameserver-Daemons hochgefahren.

```
vnumlparser.pl -x start@dns.xml
```

Nun ist die Simulation einsatzbereit und wir können zu Beginn eine einfache DNS-Anfrage stellen. Beispielsweise wird der Nameserver 10.1.1.1 (Nameserver der Zone e6) nach der IP-Adresse des Hosts h1.dit.upm.e6 gefragt:

```
dig @10.1.1.1 h1.dit.upm.e6
```

Als Antwort erhalten wir:

```
; <<<>> DiG 9.2.3 <<<>> @10.1.1.1 h1.dit.upm.e6
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11975
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;h1.dit.upm.e6.          IN      A

;; ANSWER SECTION:
h1.dit.upm.e6.         300    IN      A      10.12.10.1
```

```
;; AUTHORITY SECTION:
dit.upm.e6.      300  IN   NS    ns1.dit.upm.e6.
```

```
;; Query time: 27 msec
;; SERVER: 10.1.1.1#53(10.1.1.1)
;; WHEN: Thu Mar 16 13:51:02 2006
;; MSG SIZE rcvd: 65
```

Erklärung der einzelnen Antwortfelder:

```
; <<>> DiG 9.2.3 <<>> @10.1.1.1 h1.dit.upm.e6
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 11975
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0
```

Dig gibt hier seine Versionsnummer aus und wiederholt nochmal die gestellte Anfrage. Dann folgt der Header der Antwort. Der opcode ist fast immer QUERY für Anfrage, es gab keinen Fehler und jeder Antwort wird eine id zugeordnet. Danach folgen die Flags. qr bedeutet, dass es sich bei dieser Nachricht um eine Antwort handelt (query response), rd steht für rekursive Auflösung (recursion desired) und ra besagt, dass Rekursion möglich ist (recursion available). Schließlich folgt noch eine Zusammenfassung. Es wurde eine Anfrage gestellt, ein Antwortfeld und ein Authorityfeld wurde zurückgeschickt.

```
;; QUESTION SECTION:
h1.dit.upm.e6.      IN   A

;; ANSWER SECTION:
h1.dit.upm.e6.      300  IN   A    10.12.10.1
```

Wiederholung der Anfrage in Zonefileschreibweise und die erste Antwort. Die IP-Adresse des Hosts h1.dit.upm.e6 ist 10.12.10.1 und die Time To Live für diesen Eintrag beträgt 300 Sekunden.

```
;; AUTHORITY SECTION:
dit.upm.e6.      300  IN   NS    ns1.dit.upm.e6.
```

Zusätzlich gibt uns der Nameserver in der Antwort noch Informationen über für diese Zone autoritative Server mit, damit bei der nächsten Anfrage in diese Zone direkt der Nameserver ns1.dit.upm.e6 befragt werden kann.

```
;; Query time: 27 msec
;; SERVER: 10.1.1.1#53(10.1.1.1)
;; WHEN: Thu Mar 16 13:51:02 2006
;; MSG SIZE rcvd: 65
```

Schließlich noch eine Übersicht über Bearbeitungszeit, Server, Zeitpunkt und Nachrichtenlänge der Antwort.

Als nächstes wollen wir eine inverse Anfrage stellen. Der Nameserver 10.1.1.1 wird nach dem zur IP-Adresse 10.1.1.5 gehörenden Hostnamen gefragt.

```
dig @10.1.1.1 -x 10.1.1.5
```

Ich beschränke mich hier darauf die Antwortfelder anzugeben, da die übrigen Angaben der Antwort kaum neue Informationen bieten.

```
;; ANSWER SECTION:
5.1.1.10.in-addr.arpa. 3600 IN PTR ns1.dif.umu.e6.

;; AUTHORITY SECTION:
10.in-addr.arpa. 3600 IN NS ns1.e6.

;; ADDITIONAL SECTION:
ns1.e6. 300 IN A 10.1.1.1
ns1.e6. 300 IN AAAA 2001:db8::1
```

In der Answer-Section bekommen wir hier die Antwort: zur IP-Adresse 10.1.1.5 gehört der Hostname ns1.dif.umu.e6. In der Authority-Section wird uns wiederum der zuständige Nameserver angegeben und in der Additional-Section werden uns noch die IP-Adressen dieses Nameservers mitgegeben, so dass diese bei einer weiteren Anfrage nicht erst erneut gesucht werden müssen.

Wie wir hier sehen werden ohne Angabe von *type* immer nur A-Queries gestartet. Durch Angabe des gewünschten Record-Typen kann aber beispielsweise auch nur der für einen Host oder eine Zone zuständige Nameserver abgefragt werden.

```
dig @10.1.1.1 h1.cons.e6 NS
```

Die Antwort lautet dann:

```
;; AUTHORITY SECTION:
cons.e6. 300 IN SOA ns1.cons.e6. root.ns1.cons.e6. 1 3600
1800 604800 3600
```

Der autoritative Nameserver für den Host h1.cons.e6 ist ns1.cons.e6, welcher Nameserver für die Zone cons.e6 ist. Administrator dieser Zone ist root@ns1.cons.e6. Abschließend sind noch die Times To Live für diesen Eintrag mitgegeben, wie sie auch im Zonefile angegeben werden.

Bisher haben wir nur Anfragen mit rekursiver Auflösung gestellt. Ich werde nun bei der nächsten Anfrage mit Hilfe der Option *+norecurse* diese rekursive Auflösung deaktivieren. Der Nameserver 10.1.1.1 (ns1.e6) wird nach der IP des Hosts h1.dif.umu.e6 gefragt

```
dig @10.1.1.1 h1.dif.umu.e6 +norecurse
```

Antwort:

```
;; AUTHORITY SECTION:  
umu.e6.      300  IN   NS    ns1.umu.e6.
```

```
;; ADDITIONAL SECTION:  
ns1.umu.e6.  300  IN   A     10.1.1.4  
ns1.umu.e6.  300  IN   AAAA  2001:db8::4
```

Wie man sieht gibt es keine Answer-Section mit der gewünschten Antwort. Der Server ns1.e6 liefert hier lediglich den nächsten Nameserver, der befragt werden muss. Möchte man jetzt an die gesuchte Adresse kommen, wird diesem als nächstes die Anfrage gestellt:

```
dig @10.1.1.4 h1.dif.umu.e6 +norecurse
```

Antwort:

```
;; AUTHORITY SECTION:  
dif.umu.e6.  300  IN   NS    ns1.dif.umu.e6.
```

```
;; ADDITIONAL SECTION:  
ns1.dif.umu.e6.  300  IN   A     10.1.1.5  
ns1.dif.umu.e6.  300  IN   AAAA  2001:db8::5
```

Wiederum erhalten wir nicht die gesuchte Adresse, sondern einen weiteren Nameserver, an den die Anfrage delegiert wird. Also befragen wir jetzt 10.1.1.5:

```
dig @10.1.1.5 h1.dif.umu.e6 +norecurse
```

Antwort:

```
;; ANSWER SECTION:  
h1.dif.umu.e6.  300  IN   A     10.13.10.1
```

```
;; AUTHORITY SECTION:  
dif.umu.e6.  300  IN   NS    ns1.dif.umu.e6.
```

```
;; ADDITIONAL SECTION:  
ns1.dif.umu.e6.  300  IN   A     10.1.1.5  
ns1.dif.umu.e6.  300  IN   AAAA  2001:db8::5
```

Schließlich kann uns der Nameserver ns1.dif.umu.e6 (10.1.1.5) die Antwort liefern, weil er für die Zone in der der Host h1.dif.umu.e6 steht autoritativ ist.

Hätten wir die Anfrage mit Rekursion gestartet, hätte ein dig gereicht, da der Nameserver dann diese ganzen Schritte für uns durchgeführt hätte.

Abschließend werde ich noch kurz das Caching der Nameserver demonstrieren. Dazu stelle ich zunächst eine beliebige Anfrage nach einem Host, der nicht in der Zone des befragten Nameservers liegt. 10.1.1.2 ist der Server ns1.upm.e6.

dig @10.1.1.2 h1.dif.umu.e6

Antwort:

```
;; ANSWER SECTION:  
h1.dif.umu.e6.      300  IN  A    10.13.10.1
```

```
;; AUTHORITY SECTION:  
dif.umu.e6.        300  IN  NS   ns1.dif.umu.e6.
```

Wie erwartet bekommen wir ganz normal die Antwort geliefert. Stelle ich nun die gleiche Anfrage erneut und deaktiviere die rekursive Auflösung, so müsste eigentlich nur der nächste zuständige Nameserver zurückgeliefert werden.

dig @10.1.1.2 h1.dif.umu.e6 +norecurse

Die Antwort lautet jedoch:

```
;; ANSWER SECTION:  
h1.dif.umu.e6.      226  IN  A    10.13.10.1
```

```
;; AUTHORITY SECTION:  
dif.umu.e6.        226  IN  NS   ns1.dif.umu.e6.
```

Da diese Anfrage bereits vorher gestellt wurde, hat sich der Nameserver ns1.upm.e6 die Antwort in seinem Cache gemerkt und kann jetzt direkt darauf antworten, ohne erneut rekursiv mehrere Nameserver befragen zu müssen. Man sieht hier auch, dass die Time To Live bereits auf 226 Sekunden gesunken ist. Stellt man nach Ablauf dieser Zeit die Anfrage nochmals (ohne Rekursion), wurde der Eintrag aus dem Cache verworfen und man erhält als Antwort:

```
;; AUTHORITY SECTION:  
.                999999  IN  NS   ns1.e6.
```

Der Nameserver weiß nun also nicht mehr über den Host h1.dif.umu.e6 oder einen zuständigen autoritativen Nameserver bescheid und müsste zunächst wiederum den Root-Nameserver ns1.e6 nach weiteren Informationen befragen.

14 Literatur

- [1] VNUML DNS Example
<http://jungla.dit.upm.es/~vnuml/doc/examples/dns/dns.html>
- [2] DNS and Bind, 4th Edition; Paul Albitz, Cricket Liu; O'Reilly 2001
- [3] BIND 9 Administrator Reference Manual
<http://www.isc.org/index.pl?sw/bind/arm93/>
- [4] dig-manpages