FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Diplomarbeit

Konzept und Implementierung eines automatisierten DNS Managements

Peter Kai Wimmer

Aufgabensteller:	Prof. Dr. Uwe Baumgarten
Abgabedatum:	15. Mai 1998

Hiermit versichere ich, daß ich die vorliegende Diplomarbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Mai 1998

....

(Peter Kai Wimmer)

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Aufgabenstellung

1.1 Motivation

Die schnell wachsende Anzahl von Rechnern, die mit dem Internet verbunden werden, stellt Netzbetreiber und -provider gleichermaßen vor die Problematik, die Übersicht über ihre Netzwerke und deren Konfiguration zu behalten.

Ein Teilgebiet dieser Konfiguration ist im Domain Name System oder kurz DNS abgebildet. Im weltumspannenden Internet werden Rechner durch eine viergeteilte Zahlenkombination, der sogenannten IP-Adresse, identifiziert. Für den Menschen stellen solche Zahlen eine Erschwernis im täglichen Umgang mit diesem Netzwerk dar. Das DNS erlaubt die Verwendung eines benutzerfreundlichen, weil aussagekräftigen, eineindeutigen Namens, den es diesen Adressen zuordnet. Die hierarchische, verteilte Struktur dieses Systems macht die Handhabung allerdings sehr komplex.

Gestaltet sich die normale Verwaltung eines einzelnen, nach außen sichtbaren Namensraumes noch relativ übersichtlich, so ist bei den hunderten von Zonen, für die Internet Service Provider (ISP) verantwortlich sind, eine manuelle Bearbeitung äußerst zeitraubend und fehleranfällig.

Da Erweiterungen und vor allem Fehlerbehebungen oft unter Zeitdruck geschehen, ist für einen technisch gesicherten Ablauf und zur Fehlervermeidung eine einheitliche, vorgegebene Vorgehensweise bei der DNS-Konfiguration wünschenswert.

Die wichtigsten Aufgaben sind dabei die Verwaltung der DNS-relevanten Daten, die Vermeidung und Behandlung von dabei auftretenden Fehlern sowie die Konfiguration der Nameserver. Zielsetzung ist ein effizienteres, sichereres und übersichtlicheres Arbeiten.

Besonders bei sicherheitsrelevanten Anwendungen ist eine lückenlose Dokumentation, vor allem auch der vorgenommenen Änderungen, beispielsweise auch den Kunden eines ISP gegenüber, unabdingbar. In Hinblick auf die immer höher eingeschätzte Zertifizierung der Vorgänge in Unternehmen nach DIN ISO 9000 ist eine solche Dokumentation ebenfalls vorteilhaft.

Diese Tätigkeiten sind bis zu einem gewissen Grad automatisierbar und somit dafür geeignet, von entsprechender Software übernommen zu werden. Das Netz- und Systemmanagement liefert dazu die entsprechenden Methodiken.

Das Konzept für ein solches Management sowie eine teilweise, beispielhafte Implementierung sind Thema der vorliegenden Arbeit.

1.2 Vorgehensweise

Am Anfang jedes Projektes steht das Vertrautwerden mit den wichtigsten Systematiken der zu bearbeitenden Bereiche.

Die Zusammenfassung der grundsätzlichen Aufgabenstellung dient dazu, ein passendes Konzept zu erstellen sowie das erwartete Ergebnis herauszuarbeiten.

Eine Sichtung vorhandener Arbeiten zu diesem Thema liefert Hinweise, welche Lösungsansätze bereits existieren und welche Erfahrungswerte diese in der Praxis geliefert haben.

Auf diesem Hintergrund werden die zu lösenden Aufgaben auf ihre Management-relevanten Aspekte zu untersuchen und die zu managenden Objekte zusamenzustellen sein. Dazu gehören das Fehlermanagement, die Konfiguration von Nameservern einschließlich der Generierung von Zonendateien, Überlegungen zur Verwaltung von IP-Adressen und zur entfernten Administration, sowie die Evaluierung einer automatisierten RIPE-Dokumentation.

Darauf aufbauend erfolgt die Implementierung der Ergebnisse und des Prototypen einer Verwaltungsoberfläche. Die Anwendung der erstellten Werkzeuge wird im Anschluß daran beschrieben.

Eine zusammenfassende Diskussion mit Ausblick auf künftige Entwicklungen schließt die Ausführungen ab.

Kapitel 2

Einführung

2.1 Eine kurze Geschichte des Internets

Am Ende der 60er Jahre begann das amerikanische Verteidigungsministerium, ein experimentelles Weitverkehrsnetzwerk, genannt ARPANET, zu entwickeln. Zum Standard-Netzwerkprotokoll wurde bald das TCP/IP (Transmission Control Protocol / InternetProtocol). Das rasante Wachstum vom Verbund einiger weniger Rechner, die hauptsächlich militärisch und dann wissenschaftlich genutzt wurden, zum heutigen Internet mit mehreren Millionen Teilnehmern und immer mehr kommerzieller Nutzung machte einige Weiterentwicklungen notwendig, um das reibungslose Zusammenspiel in der Zukunft sicherzustellen.

Die Namen der damals wenigen hundert Rechner des ARPANETs konnten in einer einzigen Datei, `HOSTS.TXT`, verwaltet werden. Aus dieser wurde die Datei `/etc/hosts` auf UNIX-Maschinen erstellt. Diese `HOSTS.TXT` wurde vom Network Information Center (NIC) verwaltet, welchem die einzelnen Administratoren die Änderungen zuschickten und sich von dort die jeweils neueste Version abholten. Daraus ergaben sich mehrere Probleme: einerseits eine untragbar hohe Netzbelastung, zum zweiten Namenskonflikte, da das NIC keine Authorität über die vergebenen Namen hatte. Darüberhinaus war es immer schwieriger, die Konsistenz der Daten in diesem sprunghaft wachsenden Netz sicherzustellen, denn ein solcher Mechanismus ist schlecht skalierbar.

Dezentralisierung durch lokale Administration der Domain Namen und eine hierarchische Struktur des Namensraumes versprachen der richtige Weg zu sein, dem rasch zunehmenden Netzverkehr Herr zu werden und die Daten effizient auf dem neuesten Stand zu halten.

2.2 Domain Name System (DNS)

Das Domain Name System ist die Grundlage der Namensauflösung im Internet. Weitere wichtige Funktionen, wie etwa das Mail Routing, gehören ebenfalls zu seinem Aufgabengebiet. Dieser Abschnitt dient der Erläuterung der Grundlagen des DNS.

Als Literatur zum Thema sei das Standardwerk *DNS and BIND*[1] empfohlen. Eine umfangreiche Zusammenstellung von Dokumenten findet man im Internet beim DNS Resource Directory (DNSRD)[1]. Dort sind auch die Internet Drafts und Request for Comments, die sich mit dem DNS beschäftigen, aufgeführt.

[1] `http://www.dns.net/dnsrd/docs/`

2.2.1 Funktionsweise

Das Domain Name System beruht auf einer Client/Server Architektur. Es ist eine verteilte Datenbank, bei der Zuverlässigkeit und Performance durch Replikation und Caching erreicht werden.

Auf der Serverseite finden sich *Nameserver*, die ihre Informationen den Clients, sogenannten *Resolvern*, zu Verfügung stellen.

Die hierarchische Struktur des DNS ist mit dem Dateisystem unter UNIX vergleichbar, wobei sich der sogenannte *Namensraum* als invertierter Baum darstellt, wie in Abbildung 2.1 zu sehen. Jeder Knoten in diesem Baum hat einen Namen und repräsentiert eine Partition des gesamten Datenbestandes, die *Subdomain* genannt wird.

Jede Domain kann von einer anderen organisatorischen Einheit verwaltet werden. Diese wiederum darf Subdomänen erstellen und die Verantwortung für diese delegieren.

2.2.2 Begriffe

Zur weiteren Erklärung und gleichzeitig übersichtlichen Referenz sind im Anschluß die wichtigsten Begriffe des DNS zusammengestellt.

Zone

Eine Zone ist ein Teil des DNS - Baumes, der als eine Einheit behandelt wird.

Forward Zone Die Daten einer Forward Zone werden genutzt, um Namen auf Adressen, Mail Exchanger, Aliase usw. abzubilden.

Reverse Zone Die zu IP-Adressen gehörenden Domain Namen sind in dieser Datei zusammengestellt. Weitere Informationen sind hier (im Gegensatz zu Forward Datei) nicht enthalten.

Nameserver

Nameserver halten Informationen über einen Teil des Namensraumes, welcher als Zone bezeichnet wird und für den sie damit *authoritativ* sind.

Der Unterschied zwischen einer Zone und einer Domain ist subtil: Eine Zone enthält Domänennamen und die zugehörigen Daten einer Domäne, bis auf jene solcher Subdomänen, die delegiert wurden. In einer Domäne wiederum können mehrere Zonen liegen, die aber alle Bestandteil dieser Domäne sind.

Zur Veranschaulichung sei die Domäne der TU München genannt: die Subdomäne `informatik.tu-muenchen.de` gehört zur Domäne `tu-muenchen.de`. Sie ist aber kein Bestandteil der Zone `tu-muenchen.de`, da sie an die Fakultät für Informatik delegiert wurde und somit eine eigene Zone darstellt.
Es gibt zwei Arten von Nameservern:

1. **Primary Nameserver**
   Ein authoritativer Server, für den die Zonen-Information lokal konfiguriert ist. Dies bedeutet, daß die Zonendateien auf diesem Host im Original abgelegt sind.

2. **Secondary Nameserver**
   Ebenfalls ein authoritativer Server, der Informationen über eine Zone von dessen Primary mittels *Zonentransfer* erhält. Der sekundäre Server spiegelt quasi die Daten des Primary, was sowohl der Ausfallsicherheit als auch der Lastverteilung zugute kommt.

Zonentransfer ist ein Mechanismus, bei dem sekundäre Nameserver nach Ablauf der Lebensdauer von Zonendaten beim Primary die aktuelle Zonedatei abholen.

Ein lokaler Nameserver kann Secondary für lokale und fremde Zonen sein, fremde Nameserver übernehmen diesen Dienst wiederum für lokale Zonen. Jede Zone soll ihre Daten zumindest von einem sekundären Nameserver spiegeln lassen.

Secondaries halten ihre Daten nach einem Zonentransfer meist lediglich im Hauptspeicher. Damit nach einem Neustart die Dateien nicht erneut übertragen werden müssen (auch könnte der Primary etwa gerade nicht erreichbar sein), legen manche sekundäre Nameserver Kopien der gespiegelten Zonendaten in ihrem lokalen Dateisystem ab. Auch diese Daten laufen natürlich am Ende der Time To Life ab.

Ein Nameserver kann für mehrere Zonen authoritativ sein, und zwar sowohl als primärer wie auch als sekundärer Server.

Root Nameserver Diesen Nameservern (an der „Wurzel“der Namenshierarchie), zur Zeit neun Stück an der Zahl, sind die Standorte der authoritativen Server für alle Top-Level Domains bekannt.

Die Nameserver für die TLDs verweisen dann auf jene in der zweiten Ebene der Namenshierarchie.

Auflösung

Den Prozeß der Abfrage von Informationen aus dem Domain Name System wird als *Auflösung* (engl. *Resolving*) bezeichnet.

Resolver *Resolver* wird der Client genannt, der über Anfragen an einen Nameserver die Domainnamen auflöst, d.h. die zugehörige IP-Adresse findet. Zu den Aufgaben eines Resolver gehören die Anfrage bei Nameservern und die Interpretation der Antwort, die entweder Resource Records oder einer Fehlermeldung besteht.

Je nachdem, ob die gewünschte Information lokal vorhanden ist, sei es, weil die lokalen Nameserver authoritativ dafür sind oder die Daten sich noch von vorherigen Abfragen im Cache befinden, werden verschiedene Arten der Auflösung angestoßen:

iterativ Bei der *iterativen* Auflösung antwortet der Nameserver anhand der Daten, die er selber lokal zur Verfügung hat. Fehlt ihm die gesuchte Informationen, gibt er einen Hinweis auf einen Nameserver zurück, der „näher“bei diesen Daten steht. Dieses *näher* dürfte in den allermeisten Fällen einen Verweis an die *Root*-Nameserver bedeuten.

rekursiv *Rekursive* Abfragen belasten hauptsächlich einen einzelnen Nameserver. Dieser sendet iterative Anfragen an andere Nameserver und hangelt sich anhand der Antworten durch die Hierarchie bis zu einem Nameserver, der ihm eine authoritative Auskunft geben kann.

Domänen

Eine Domäne, beispielsweise `informatik.tu-muenchen.de`, läßt sich am einfachsten als Unterbaum innerhalb des Domänennamensraumes (im Beispiel demnach als Unterbaum von `tu-muenchen.de`) ansehen, der Name des Vaterknotens entspricht also dem der Domäne.

Die Zusammenfassung von Hosts in Domänen geschieht nach logischen, z.B. organisatorischen, Gesichtspunkten und hat nicht notwendigerweise etwas mit den physikalischen Gegebenheiten (wie etwa der geographischen Verteilung) zu tun. Die Domänen an den Blättern des Baumes sind offensichtlich die einzelnen Hosts.

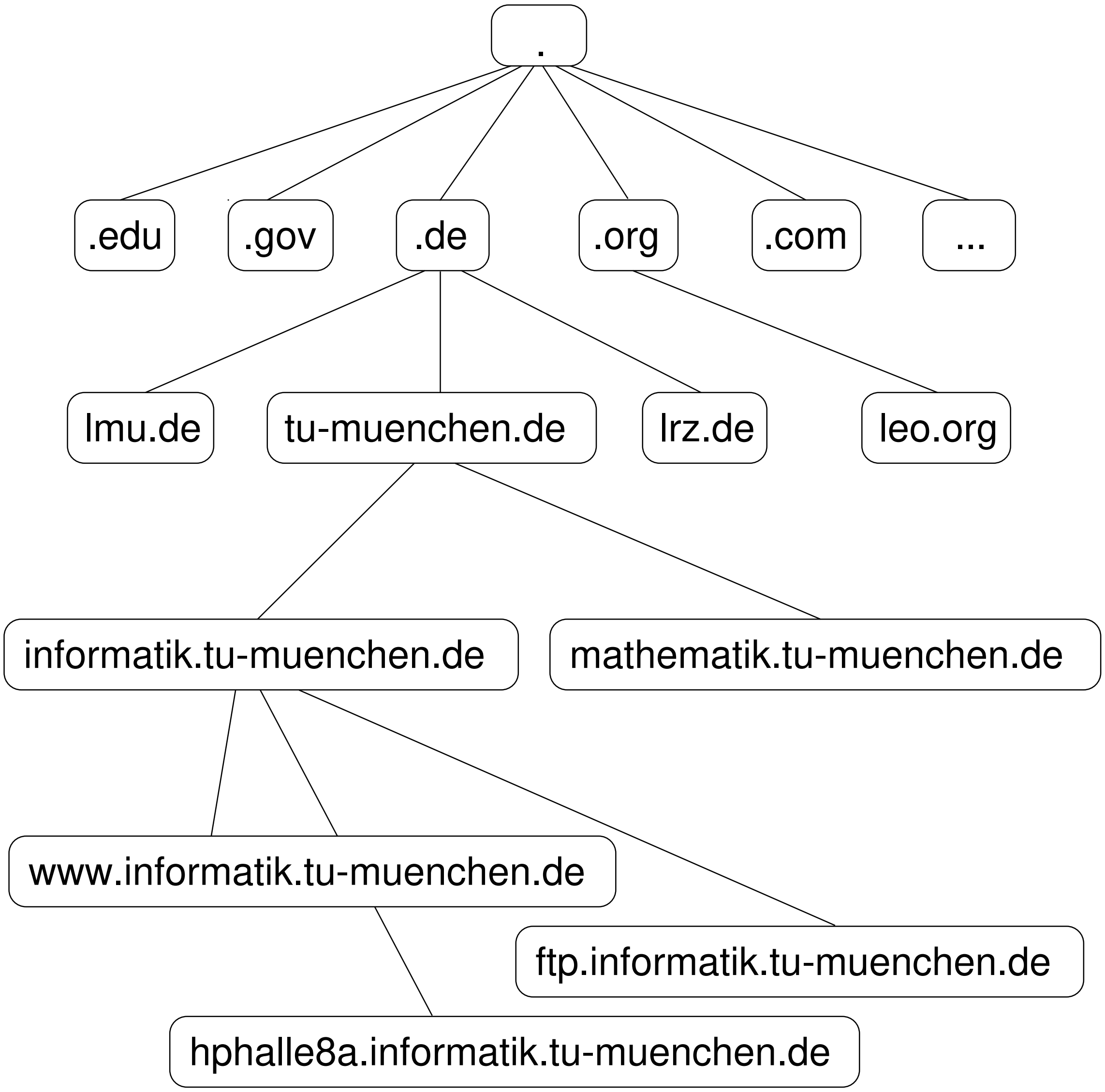

Abbildung 2.1: Die hierarchische Struktur des DNS

Domänennamen

Jede Domäne, also jeder Knoten im Baum, wird mit einem Namen bezeichnet. Die Kinder der selben Domäne müssen eineindeutig benannt sein.

Die (reservierte) Bezeichnung der Root-Domäne hat die Länge Null.

Der vollständige (engl. *fully qualified*) Domänenname[2] eines Knotens, der sich aus der Aneinanderreihung der Namen der jeweiligen Vaterdomänen bis hin zur Wurzel ergibt, bezeichnet gleichzeitig seine absolute Position in der Hierarchie.

In den Zonendateien werden solche Namen mit einem Punkt abgeschlossen (z.B. `www.leo.org.`), damit sie nicht für einen unvollständigen, etwa einen Hostnamen wie `www`, gehalten werden. Letztere werden nämlich sonst durch Anhängen der aktuellen Domäne (im Beispiel: `leo.org`) ergänzt.

Subdomains

Jeder Knoten in diesem Baum steht für eine Domäne im Domain Name System. Weitere Partitionierungen in Subdomänen, also die Unterteilung einer Domäne in weitere Bereiche, ergibt somit Kinder dieser Knoten. Die enthaltenen Hosts sind wiederum Domänen.

Es ist möglich, wenn auch relativ selten anzutreffen, daß der Name einer Vaterdomäne selbst auch Hostname ist, d.h. ein Rechner direkt unter diesem Namen (als Beispiel sei das ECRC mit `ecrc.de` genannt) erreichbar ist.

Subdomänen sollten immer in einer eigenen Zone verwaltet und dementsprechend auch eine eigene Zonendatei für sie erstellt werden. Es ist zwar möglich, Subdomänen auch in der Vater-Zonendatei einzutragen, aber dies wird nur aufgrund des geringeren Arbeitsaufwandes bei händischer Vorgehensweise gemacht. Sauberer und übersichtlicher ist dagegen die Erstellung einer eigenen Zone.

Delegierung, Parenting

Als *Delegierung* bezeichnet man die Übertragung der Verantwortung für eine Subdomäne an einen Nameserver (auch an den der Vaterdomäne, also denselben, von dem aus delegiert wird. Damit bleibt es sozusagen in der „Familie“).

Bewerkstelligt wird eine solche Delegierung durch Anlegen einer eigenständigen Zonendatei für die jeweilige Subdomäne und einem Eintrag in der Vaterzone, der einen Nameserver für diese Subdomäne bestimmt. Eine eigene Zonendatei allein bedeutet noch keine Delegierung - es muß immer auch die Verantwortung abgegeben werden.

Die Übernahme der Rolle als Primary für eine Domäne wird als *Parenting* bezeichnet.

DNS Zonendateien

Die Dateien, aus denen primäre Nameserver ihre Daten laden, werden *Zonendateien*[3] genannt. Die Einträge in dieser Datei heißen *Resource Records*; auf diese wird noch (in 2.2.3) eingegangen.

---

[2] *FQDN*, Fully Qualified Domain Name

[3] *db-Dateien* nach [1] (*db* steht dabei für Datenbank))

Glue Records

Glue Records können bei der Delegierung von Zonen erforderlich sein. In der Vaterzone wird bei Anfragen, die die delegierte Zone betreffen, auf deren Nameserver verwiesen. Allerdings wird nur der Hostname dieses Nameservers geliefert, seine IP-Adresse ist wiederum nur in der delegierten Zone bekannt. Um sie nun auch nach Außen zur Verfügung zu stellen, sind die IP-Adressen delegierter Nameserver über `A` Records in der Zonendatei der Vaterzone aufzunehmen.

TTL

Die Time To Live (TTL) ist die Zeit, die ein Nameserver Daten in seinem Cache halten darf. Danach sind diese zu verwerfen und die aktuellen Informationen vom authoritativen Nameserver zu holen.

Die Entscheidung für diese Zeitdauer ist ein Kompromiß zwischen Performanz und Konsistenz. Werden die Daten nach kurzer Zeit wiederaufgefrischt, sind Änderungen im Namensraum schneller bekannt. Der dadurch entstehende Netzverkehr und die längeren Auflösungszeiten verbieten allerdings ein zu kleines Intervall.

Grundsätzlich sind die TTLs[4] für jeden einzelnen Resource-Record einstellbar.

2.2.3 Resource Records

Resource Records enthalten die zu einer Domäne gehörenden Daten. Groß- und Kleinschreibung spielt bei ihnen keine Rolle, wird aber bei Abfragen erhalten. In der Zonendatei (s. 2.2.2) müssen die Einträge für Resource Records in der ersten Spalte beginnen.

Diese Records sind in **Klassen** unterteilt, je nach Netzwerk oder Software, für die sie Daten bereitstellen. Die wichtigste Klasse wiederum ist `IN` wie INternet, die wenigen anderen, teilweise experimentellen Klassen haben nur noch historische[5] oder lokale[6] Bedeutung.

SOA, Start Of Authority

Der jeweils erste Eintrag in den Zonendateien ist der `SOA` (Start Of Authority) Record, mit dem ein Nameserver als authoritativ für eine Zone gekennzeichnet wird. Anhand des nachfolgenden Beispieles seien die einzelnen Einträge erklärt:

```
haitec.de.      IN      SOA     haigate.haitec.de. root.haitec.de. (
                        1        ; Serial
                        10800    ; Refresh nach 3 Stunden
                        3600     ; Neuer Versuch nach 1 Stunde
                        604800   ; Verfallen nach 1 Woche
                        86400 )  ; Minimum TTL von 1 Tag
```

Der Name der Zone, `haitec.de`, muß in Spalte 1 der Datei beginnen. Da er *fully qualified* (s.o) ist, wird er mit einem Punkt beendet.

[4] BIND interpretiert diese übrigens etwas anders als in den RFCs vorgesehen ([1], S.167)
[5] Chaosnet Protokoll
[6] Hesiod am MIT

Hierauf folgt der primäre Nameserver, im Beispiel `haigate.haitec.de`, und die E-Mail-Adresse des Verantwortlichen für diese Zone. In E-Mail-Adressen wird das @-Zeichen (auch Klammeraffe genannt) durch einen Punkt ersetzt.

Refresh Der SOA-Record des primären Nameserver wird im Abstand der *Refresh*-Zeit von den sekundären Servern überprüft. Hat er sich geändert, wird ein Zonentransfer (s. 2.2.2) initiiert.

Retry Kann ein sekundärer Nameserver den primären Server nicht erreichen, wird er es nach der Zeitspanne, die in *Retry* angegeben ist, erneut versuchen.

Expire Ist der primäre Nameserver länger als durch die *Expire*-Zeit erlaubt nicht erreichbar, wird die gesamte Information über diese Zone bei den Secondaries ungültig, d.h. sie sind nicht mehr länger authoritativ für diese Zone.

Minimum TTL (Time To Live)

Der voreingestellte TTL-Wert für alle Einträge in der Zonendatei. Jedem Record kann darüberhinaus bei Bedarf ein eigener TTL-Wert zugewiesen werden.

Dieser Wert wird zwar als Minimum bezeichnet, und das ist nach STD 13 [RFC 1035] auch seine Funktion. Die meisten Implementierungen allerdings nehmen ihn als den default-Wert, der mit Records ausgeliefert wird, für die kein expliziter TTL-Wert angegeben ist.

Für die Top-Level Domain Server werden folgende Werte vorgeschlagen (nach [RFC 1537][7]):

```
  86400 ; Refresh     24 Stunden
   7200 ; Retry        2 Stunden
2592000 ; Expire      30 Tage
 345600 ; Minimum TTL  4 Tage
```

Alle anderen sollten diese Werte erhalten:

```
 28800 ; Refresh     8 Stunden
  7200 ; Retry       2 Stunden
604800 ; Expire      7 Tage
 86400 ; Minimum TTL 1 Tag
```

HINFO, Host Info

Hiermit wird die Hardware und das Betriebssystem eines Hosts angegeben, wie etwa in folgendem Beispiel:

```
www             IN      HINFO   "HP 700"        "HP-UX 10.20"
```

[7] [RFC 1537] ist zwar *obsoleted*, leider finden sich die Werte aber im neueren Dokument [RFC 1912] nicht mehr.

Die Bezeichnungen sollten sich nach den MACHINE NAMES und SYSTEM NAMES richten, die im RFC *Assigned Numbers* [RFC 1340] aufgeführt sind. Dies ist allerdings nicht verpflichtend, es dürfen auch eigene Abkürzungen verwendet werden.

Bei den meisten Hosts fehlt diese Information oder wird für andere Zwecke mißbraucht.

Die Angaben in `HINFO` sind darüberhinaus ein Sicherheitsrisiko: durch die Angabe von Maschinentyp und verwendetem Betriebssystem werden Einbruchsversuche erleichtert.

NS, Nameserver

Jeder Nameserver für eine Domain, der primäre und die sekundären, wird über `NS` angegeben. Eine gute Wahl für Nameserver sind Rechner, die *multi-homed* (also von mehr als einem Netzwerk zugreifbar) sind oder als Router eingesetzt werden, da solche Maschinen gut überwacht werden.

A

Die Zuordnung von Internet-Adresse(n) zu einem Hostnamen geschieht mittels des `A` Resource Records. Bei *multihomed* Hosts wird für jede Netzwerkkarte eine eigene IP-Adresse angegeben.

Router beispielsweise haben mehrere IP-Adressen. Demzufolge werden für sie entweder mehrere A-Records eingetragen oder alternativ jedem Interface des Routers ein Name, zusammen mit eigenem A-Record, gegeben.

PTR

Der umgekehrte Weg ist die Abbildung von Namen auf Adressen. Benötigt wird dies für das Reverse Lookup (s. 3.6.2 und 2.2.2). Diese Resource Records werden üblicherweise in einer eigenen Zonendatei abgelegt.

MX, Mail Exchanger

Ein *Mail Exchanger* ist ein Host, der elektronische Post entweder selbst verarbeitet oder sie an einen weiteren, dem Ziel „näheren“Exchanger weiterreicht. MX-Records werden vom Simple Mail Transport Protocol (SMTP) verwendet.

Übernimmt ein Mail Exchanger selbst die Behandlung der Mail, dann um sie entweder dem Empfänger, der sich in seiner Domain befindet, zuzustellen, oder sie einem anderen Mailprotokoll, etwa UUCP, zu übergeben.

Der zugehörige *Preference*-Wert bestimmt die Reihenfolge, falls zur Ausfallsicherheit mehrere Mail Exchanger angegeben werden. Dabei bedeutet ein kleinerer Wert höhere Priorität. Nur wenn ein solcher Host nicht erreicht werden kann, wird der nächste in der Liste herangezogen.

CNAME, Canonical Name (Alias)

Soll ein Rechner unter mehreren Namen bekannt sein, lassen sich Aliase für ihn definieren. Dies ist unter anderem nützlich, um einen Host, der Dienste wie WWW oder FTP anbietet, leicht auffindbar zu machen.

Zur Veranschaulichung ein konkretes Beispiel: `haigate.haitec.de` ist zugleich auch WWW- und FTP-Server, also trägt man Aliase ein, um diese Maschine ebenso unter den Namen `www.haitec.de` bzw. `ftp.haitec.de` ansprechen zu können.

WKS, Well Known Services

Mit `WKS`[8] werden häufig verwendete Dienste, die der Host anbietet, angegeben. Mailer könnten hier beispielsweise nachsehen, ob die betreffende Maschine SMTP spricht. Allerdings werden diese Einträge selten gepflegt, so daß man sich nicht darauf verlassen sollte.

TXT

`TXT`, also TeXT-Einträge, können zu beliebigen Zwecken eingesetzt werden. Eine denkbare Verwendung wären Angaben zu Standort (Gebäude, Raum) eines Hosts.

RP, Responsible Person

Zu jedem Host und jeder Domain läßt sich eine Kontaktperson angeben, für den Fall, daß ein Rechner oder Dienst Probleme bereitet. Dieser Resource Record kam erst in einer Erweiterung [RFC 1183] des DNS-Standards hinzu.

Es sind zwei Informationen erforderlich: die E-Mail Adresse der *Responsible Person* (im üblichen Format, bei dem das @-Zeichen durch einen Punkt ersetzt wird) und ein Domain-Name, um auf weitere Daten zu verweisen.

```
www         IN      RP      info.leo.org   hotline.leo.org
hotline     IN      TXT     "Hotline bei LEO Org, (089) 12 34 56 78"
```

Diese erweiterten Daten sind eine Möglichkeit, zusätzliche Informationen wie Telefonnummer, Standort und Ähnliches anzugeben. Sie sind jedoch nicht notwendig, d.h. der TXT Record darf entfallen.

2.2.4 Zusammenarbeit mit anderen Diensten

Eine Reihe von Diensten ist auf das Domain Name System angewiesen oder ihre Verwendung erfordert die Berücksichtigung der Funktionsweise des DNS.

Nachfolgend sei eine Auswahl solcher Dienste, die auch für ISPs relevant sind, zusammengestellt. Da einige davon im Zusammenspiel mit dem Domain Name System teilweise Kompromisse verlangen, geht dort eine kurze Bewertung einher.

E-Mail

Die bereits erwähnten Mail Exchange Records sind eine elementare Voraussetzung für das erfolgreiche Versenden von elektronischer Post im Internet.

[8] In [RFC 1912] wird dieser Resource Record als *deprecated* bezeichnet mit Verweis auf [RFC 1123]. Dort ist allerdings nur angegeben, daß Mailer sich nicht auf das Vorhandensein eines solchen Eintrages verlassen dürfen und deshalb WKS für sie ungeeignet sind.

DHCP

Das DHCP (Dynamic Host Configuration Protocol) ermöglicht die Konfiguration von Hosts. Der meistgenutzte Bestandteil ist dabei die dynamische Vergabe von IP-Adressen. Hier sind zwei Fälle zu unterscheiden:

1. **IP-Adressen von Hosts**
   Nach außen sichtbare Hosts (z.B. WWW-Server) benötigen eine feste Adresse, um ihre Dienste (WWW, FTP, etc.) anzubieten. Daher verbietet sich hier die Anwendung von DHCP.

2. **Einwahlknoten**
   Internet Service Provider bieten ihren Kunden, seien es Studenten an Universitäten oder Privatleute bei kommerziellen Unternehmen, Einwahlknoten an. Da die Anzahl der gleichzeitig aktiven Verbindungen durch die der vorhandenen Wählleitungen beschränkt ist und diese Zahl signifikant unter der *aller* Benutzer liegt, ist eine feste Zuweisung von IP-Adressen an diese Benutzer unökonomisch.

   Hier bietet sich als Alternative die dynamische Zuteilung aus einem Adreßpool für die jeweils aktuell bestehenden Verbindungen an. Die (fest vergebenen) IP-Adressen der Wählleitungen werden also für die Dauer einer Sitzung dem jeweiligen Benutzer zugewiesen.

Internet Firewalls

Firewalls dienen zur Abschottung des internen Netzes gegenüber dem Internet. Der Zugriff auf externe Dienste ist üblicherweise mehr oder weniger uneingeschränkt möglich, von Außen wird der meiste Netzverkehr jedoch abgeblockt oder über Gateways, etwa für E-Mail, abgehandelt.

Das Anbieten eigener Dienste geschieht meist über Rechner in der sogenannten *DeMilitarized Zone* (DMZ), also solche, die außerhalb des geschützten Netzes aufgestellt sind. So werden selbst bei großem internen Namensraum, der verborgen bleibt, nur wenige offizielle IP-Adressen verbraucht, die dann dem DNS bekannt sein müssen.

2.3 Netzwerk - Management

Die Zielsetzung dieser Arbeit besteht, wie oben (1.1) erläutert , in der Entwicklung eines Management-Konzeptes für die Verwaltung des Domain Name System. Die Bedeutung des Netzwerk-Management wird deshalb in diesem Abschnitt betrachtet.

2.3.1 Netz- und Systemmanagement

Netz- und Systemmanagement ist die „Summe aller Maßnahmen zur Planung, Konfiguration, Steuerung, Überwachung, Fehlerbehebung, Verwaltung von Rechnernetzen und verteilten Systemen mit benutzerfreundlicher und wirtschaftlicher Unterstützung der Betreiber und Nutzer der Systeme“. [Hege95]

Aufgrund der Heterogenität der zu managenden Resourcen befinden sich verschiedene Standards in der Entwicklung. Wunschziel ist das *integrierte* Management, bei dem einheitliche Konzepte für das Netz- und Systemmanagement, die dazugehörende Datenbank, die

Programmierschnittstellen und die Bedienoberflächen vorhanden sind. Eine solche standardisierte Management-Architektur existiert bis heute jedoch noch nicht.

Modelle

Zur Charaktierisierung des Managements wurden vier Modelle entwickelt:

1. **Informationsmodell**
   Beschreibung von Management-Information

2. **Kommunikationsmodell**
   Monitoring, Controlling

3. **Funktionsmodell**
   Funktionsbereiche, s. 2.3.2

4. **Organisationsmodell**
   Anpassung an Aufbau und Ablauforganisation der Betreiber

Aspekte

Die Anforderungen an ein Netzwerk-Management System lassen sich in verschiedene Aspekte untergliedern:

Planungsaspekte Durch Analyse der bestehenden Netze und Organisationsformen lassen sich *Betriebsziele* formulieren und dadurch eine Entscheidung für eine geeignete Lösung finden. Die Planungsanalysen betreffen den *Bedarfschwerpunkt*, die *Bedarfsgröße*, die *vorhandene Installation* sowie eventuelle *Randbedingungen*.

Betriebsaspekte Als größter Bereich umfassen die betrieblichen Aspekte die (automatisierbaren) *Informationsdienste*, die *Konfiguration* und *Generierung*, die *Überwachung*, *Steuerung* und *Fehlerbehandlung*, die *Benutzerverwaltung* sowie die *Abrechnung* und *Kontigentierung*.

organisatorische Aspekte Die Organisation umfaßt die Bildung von räumlichen und hierarchischen *Unterbereichen*, die Zusammenstellung der relevanten *Tätigkeiten* und eine passende *Zuständigkeitsverteilung*.

Diese Aspekte werden im konzeptionellen Teil (3.2.1) eine große Rolle spielen.

Objekte

Die Resourcen, die vom Management erfaßt werden, nennt man Management-Objekte (engl. *managed object*; *MO*). Beim Netzmanagement handelt es sich dabei unter anderem um Systeme, Komponenten, Verbindungen und Protokolle. Die Variablen werden in den Attributen dieser Objekte erfaßt.

Viele Objekte sind *zusammengesetzt* (engl. *composite*), damit ist gemeint, sie bestehen wiederum aus anderen Objekten. Im Domain Name System wäre etwa ein Nameserver ein solches zusammengesetztes Objekt, da er seinerseits auf einem Host läuft.

Management-Objekte können auch Relationen zwischen MOs sein, etwa *queries* oder *is-attached-to*. Es existieren *statische* und *dynamische* Objekte, und sie sind teilweise *funktionsbereichsspezifisch*. *Verfahren* wie Logs, Filtermechanismen oder Domänenbildung fallen als „Hilfsobjekte"ebenfalls unter den Begriff Managed Object.

2.3.2 Management-Architekturen

Im Laufe der Zeit entstanden verschiedene Management-Architekturen, von denen die zwei wichtigsten nachfolgend vorgestellt werden:

ISO-OSI

Eher theoretische Bedeutung erlangte das Netzwerk-Management-Modell der International Standards Organisation (ISO). Bezüglich der Einzelheiten sei auf die Literatur verwiesen. Interessant in diesem Modell ist die Aufgliederung der einzelnen Funktionsbereiche:

- **Accounting**
  Protokollierung der Verwendung der Betriebsmittel zur Abrechnung
- **Konfigurationsmanagement**
  1. Buchführung über Soll- und Ist-Zustand des Netzes
  2. Hinzufügen, Entfernen von Komponenten
  3. Fernbedienung (Operating, Installation, Diagnose, Reboot, ...)
- **Fehlermanagement**
  1. Registrierung von nicht-lokalen und Kommunikationsfehlern
  2. Entdecken, Melden, Diagnose, Eindämmen/Beheben, Rückmeldung
- **Sicherheitsmanagement** (Schutz gegen Mißbrauch)
  1. Geheimhaltung (Verschlüsselung) von Kommunikation
  2. Unversehrtheit von Nachrichten
  3. Verifikation des Absenders (digitale Unterschrift)
  4. Protokollierung
- **Leistungsmanagement**
  1. Messung von Dauer, Umfang, Auslastung, Performance etc.
  2. Statistiken zur Vorhersage von Trends
- **Resourcenmanagement**
  Ökonomische Verwaltung der Betriebsmittel

Internet (SNMP) Management

Ein weiteres Management-Modell ist das Internet- oder SNMP (Simple Network Management Protocol) - Management, wie es auch genannt wird. Dieses objektorientierte Modell beruht auf dem Manager-Agent - Prinzip.

In den Management-Objekten (s. 2.3.1) werden einfache Variablen und Tabellen abgelegt. Deren eindeutige Identifizierung geschieht über den Internet-Registrierungsbaum ([RFC 1155], [RFC 1442]). Definiert wird ein MO in der zugehörigen Management Information Base (MIB; s. [RFC 1611]).

Die Kommunikation zwischen Manager und Agent erfolgt über das bereits erwähnte Simple Network Management Protocol. Werte von Variablen in einer MIB, die auch in Tabellenform organisiert sein können, sind über dieses Protokoll mittels der Funktionen *Set*, *Get* und *Get Next* zugreifbar.

2.4 Réseaux IP Européens (RIPE)

Im weiteren Sinne ist auch die Dokumentation von IP-Adressen bei RIPE ein Teilbereich des DNS-Managements, auch wenn es nur Internet Service Provider betrifft.

Das RIPE[9] NCC (Network Coordination Center) in Amsterdam koordiniert das TCP/IP Networking für Forschungseinrichtungen in Europa.

2.4.1 RIPE-Datenbank

Als eine seiner Aufgaben unterhält das RIPE eine Datenbank der europäischen IP-Netzwerke, DNS Domänen, den zugehörigen Kontaktpersonen sowie des IP Routing. Diese Datenbank ist bekannt unter dem Namen *RIPE Network Management Database* oder auch kurz *RIPE Datenbank*.

Die enthaltenen Informationen sind frei zugänglich und sollen Netzwerkverwalter bei ihrer Arbeit unterstützen.

Als Beispiel sei hier kurz folgender Fall skizziert: In Netzwerk A kann ein Benutzer einen bestimmten Rechner im Netzwerk B nicht erreichen. Der Netzwerkverwalter von A wird dann anhand der Informationen in der RIPE-Datenbank den technischen Kontakt (die Responsible Person, s. 2.2.3) von B herausfinden und diesen auf das Problem hinweisen.

2.4.2 Beschreibung und Verwaltung der Datenbank-Objekte

Die RIPE Netzwerk Management Datenbank verwaltet Objekte, von denen jedes einer *Entity* in der Realität entspricht. Wie bei Datenbanken üblich, werden sie durch Attribute und deren Wert beschrieben.

Die relevanten Objekte für die Adreßdokumentation heißen:

inetnum (***in****ternet* ***net****work* ***num****ber assignment information*)
: Belegung und Zuteilung von IP-Adreßraum für die *IP Address Registry*

person (*contact* ***person*** *information*)
: Details über die Personen, die für die Operation von Netzwerken und Routern zuständig sind oder die Informationen in der RIPE-Datenbank verwalten.

[9] `http://www.ripe.net`

Die übrigen Datenbank-Objekte sind [ripe-157] zu entnehmen.

Für eine Erläuterung der Attribute sei auf das Konzept für die RIPE-Dokumentation (3.9) verwiesen. Eine detaillierte Beschreibung kann auch über den RIPE Whois Server[10] abgefragt werden, zum Beispiel durch Eingabe von `whois -v inetnum` für das gleichnamige Objekt.

In Kapitel 2.2 von [ripe-157] ist das Erstellen, Verändern und Löschen von Datenbank-Objekten beschrieben. Man geht hierfür folgendermaßen vor:

1. Template holen

2. Ausfüllen des Templates

3. vervollständigtes Template an die entsprechende E-Mail Adresse bei RIPE senden.

Beim Ändern oder Löschen von Objekten wird lediglich ein entsprechendes Attribut `changed` bzw. `delete` hinzugefügt.

[10] `http://www.ripe.net/db/whois.html`

Kapitel 3

Konzept

Aus der Aufgabenstellung ergeben sich die Anforderungen, zu denen konzeptionelle Überlegungen durchzuführen sind.

Die Sichtung bereits bestehender Arbeiten in diesem Umfeld kann weitere Hinweise und Anregungen liefern.

Anhand der Definition des Domain Name Systems und der Aufgabenstellung sollen zuerst die Management-relevanten Informationen zusammengetragen und in entsprechende Objektdefinitionen gegossen werden.

Für das Fehlermanagement ist anschließend eine Liste der zu beachtenden Regeln, häufiger Konfigurationsfehler und Abhängigkeiten der Daten untereinander zusammenzustellen.

Bei der Verwaltung der Daten über eine Benutzeroberfläche sind besonders die Erfordernisse Sicherheit, Verteiltheit und Mehrbenutzerfähigkeit im Auge zu behalten.

Ein wichtiger Gesichtspunkt ist die allgemeine Verwendbarkeit des zu entwickelnden Konzeptes, d.h. Orientierung nur an in den RFCs definierten Standards, nicht jedoch an proprietären Erweiterungen einzelner Hersteller.

3.1 Anforderungen

Die bereits in Kapitel 1 erwähnten Anforderungen an ein DNS Management Werkzeug seien hier nochmals ausführlicher zusammengestellt.

3.1.1 Import von bestehenden DNS-Daten

Üblicherweise existiert bereits eine große Anzahl von Zonen, ebenso wie die dazugehörenden Zonendateien. Da eine Neueingabe nicht nur sehr aufwendig, sondern dazu auch äußerst fehleranfällig wäre, ist ein Werkzeug zu erstellen, welches das Einlesen dieser Dateien automatisiert.

3.1.2 Sichere Datenpflege: Neueintrag, Ändern, Löschen

Die Pflege der Datenbank soll nur entsprechend den Standards in den relevanten RFCs erfolgen. Deshalb sind zumindest die umfangreicheren Änderungen in der Datenbank nicht von Hand, sondern über entsprechende Funktionen durchzuführen. Diese Funktionen überprüfen die Zulässigkeit der Daten und, soweit möglich, deren Plausibilität.

Beim Löschen sind die Abhängigkeiten der Daten untereindander zu berücksichtigen; entweder werden zugehörige Daten ebenfalls entfernt oder in besonderen Fällen die Löschanforderung abgewiesen.

3.1.3 Adreßverwaltung

Es ist praktikabel, jeder Zone einen bestimmten Adreßbereich zuzuteilen, aus dem dann bedarfsorientiert die IP-Adressen für neue Hosts vergeben werden. Dieser Bereich wird dem gesamten Adreßpool, der einem ISP zu Verfügung steht, entnommen.

Class-C Netzwerke werden bei Providern meist komplett einem seiner Point Of Presence (POP) zur Verfügung gestellt und daraus dann zusammenhängende Adreßblöcke für Zonen reserviert.

Die Bedienungsoberfläche soll dem Anwender eine IP-Adreßverwaltung ermöglichen und die korrekte Adreßvergabe erleichtern.

3.1.4 Fehlermanagement

Das Fehlermanagement betrifft hauptsächlich die syntaktische und die semantische Prüfung der DNS-Daten (also die Vermeidung von Fehlern), aber auch die Fehlersuche. Folgende Punkte sind demnach sicherzustellen:

1. Korrektheit des Datenbestandes bzgl. der Standards

2. Konsistenz der Daten untereinander

Grundvoraussetzung für das Auffinden dennoch auftretender Fehler ist nicht nur der bestehende Datenbestand, sondern besonders eine Protokollierung der Veränderungen (s. 3.5.3) daran.

3.1.5 Generieren von DNS Zonendateien

Aus dem Bestand in der Datenbank lassen sich konsistente Zonendateien und die Konfigurationsdatei für den jeweiligen Nameserver generieren.

Dabei ist auf Konformität zu den Standards sowie auf Kompatibilität zur Nameserver-Software zu achten.

Der meistverbreitete Nameserver-Daemon BIND[1] ist zusätzlich mit nützlichen Funktionen wie dem *Secure Zone* Mechanismus und *Load Sharing* ausgestattet. Aufgrund der fehlenden Standardisierung solcher Erweiterungen werden sie hier jedoch nicht berücksichtigt.

3.1.6 Oberfläche zur Verwaltung

Eine einheitliche Vorgehensweise bei der Verwaltung der DNS-Daten ist am besten zu erreichen, wenn alle Benutzer eine einheitliche Oberfläche verwenden. Diese soll nicht nur eine Schnittstelle zu den erwähnten Funktionalitäten bieten, sondern darüberhinaus folgende Punkte erfüllen:

[1] `http://www.isc.org/bind.html`

- **entfernte Administration**
  Die Verwaltung ist sinnvollerweise standortunabhängig zu ermöglichen. Als Medium bietet sich das Internet an.

- **benutzungsfreundliche Oberfläche**
  Die Oberfläche selber soll den Anwendern ihre Arbeit vereinfachen und Fehlervermeidung unterstützen.

- **sicherer Zugang**
  Es ist ein Sicherheitskonzept erforderlich, um Ausspähen von Daten oder gar deren Mißbrauch zu unterbinden.

- **Benutzerverwaltung**
  Benutzer mit unterschiedlichen Berechtigungsstufen müssen sich beim Zugang zum System authentifizieren.

3.1.7 Dokumentation bei RIPE

Eine ebenso einfache wie zeitaufwendige Tätigkeit bei der Verwaltung vieler Zonen ist die Dokumentation gegenüber der RIPE-Datenbank (s. 2.4.1).

Die zu erwartende Zeitersparnis ist die primäre Motivation für eine Untersuchung, inwieweit dieser Prozeß zu automatisierbar ist.

3.2 Netzwerk-Management

In dieser Arbeit werden die Aspekte (s. 2.3.1) des Domain Name Systems für ihre Umsetzung in ein Management-Konzept betrachtet. Anhand der zum DNS gehörenden Resourcen sollen darüberhinaus die zu managenden Objekte erfaßt werden.

3.2.1 Aspekte

Wie in der Einleitung (2.3) bereits erwähnt, gibt es mehrere Aspekte des Netzwerk-Managements. Davon sind im Bezug auf das DNS besonders die betrieblichen und organisatorischen Aspekte interessant.

- Die **Planungsaspekte** werden durch die Erstellung des Konzeptes zum DNS-Management in diesem Kapitel abgedeckt.

- Den größten Anteil an diesem Konzept werden die **betrieblichen Aspekte** einnehmen. Sie umfassen die Konfiguration der Nameserver, die Generierung der Zonendateien, das Fehlermanagement sowie die Sicherheit.

- Die **organisatorischen Aspekte** betreffen unter anderem die Verwaltung des DNS durch Anwender.

3.2.2 Objekte

Die Resourcen des Domain Name System bestehen einerseits aus realen Objekten, wie Hosts und Nameservern, und deren Unterobjekten, z.B. den Resource Records, andererseits aus Relationen zwischen solchen Objekten.

Die zu erarbeitenden allgemeingültigen Ergebnisse können anschließend für Implementierungen, die auf den Standards aufbauen sollten, herangezogen werden. Anhand einer entsprechenden Funktionensammlung sowie einer beispielhaften Benutzerschnittstelle wird eine solche Implementierung in dieser Arbeit (s. Kapitel 4) vorgestellt.

Eine Einbindung in eine bestehende Management-Plattform, etwa HP-Openview oder Tivoli, hätte dagegen den Rahmen gesprengt. Für das am meisten verbreitete Internet-Management über SNMP ist auch aufgrund der nicht vorhandenen Realisierungen der entsprechenden MIBs (s. 2.3.2) eine solche Integration momentan nicht möglich, will man keine proprietären Nameserver schaffen.

3.3 Sichtung vorhandener Lösungen

Es empfiehlt sich zu Beginn größerer Projekte, Informationen über bereits existierende Lösungsansätze einzuholen. Die wichtigsten sind, in drei Kategorien aufgeteilt, hier zusammengestellt.

3.3.1 Werkzeuge

Werkzeuge[2] helfen dem Systemverwalter bei der Erstellung und Überprüfung von Teilen der DNS-Konfiguration. Sie sind allerdings isoliert, d.h. nur für bestimmte, eingegrenzte Aufgaben zu verwenden.

Aus der Vielzahl der vorhandenen Tools nachfolgend eine kleine Auswahl derer, die zumindest Teilbereiche der Aufgabenstellung dieser Diplomarbeit abdecken.

- **dnswalk**[3]
  dient zur Überprüfung des DNS-Baumes. dnswalk parst keine Zonendateien, sondern sendet DNS-Anfragen an einen Nameserver, dessen Zone(n) überprüft werden soll.

- **lamers**
  findet Lame Delegations (s. 3.5.2) und benachrichtigt die betroffenen Server.

- **nslint**
  spürt Inkonsistenzen in Zonendateien auf.

3.3.2 Halb-integrierte Lösungen

Als *integrierte Lösung* wird die Kombination mehrerer Werkzeuge unter einer gemeinsamen Benutzeroberfläche, zusammen mit einem Datenaustausch bzw. einem Zusammenwirken der einzelnen Tools verstanden.

Die folgenden Arbeiten gehen zwar über die oben vorgestellten Werkzeuge hinaus, ihnen fehlt aber entweder eine Oberfläche, oder eine Integration ist nur ansatzweise vorhanden.

[2]Zusammenstellungen von Werkzeugen unter, `http://www.dns.net/dnsrd/tools.html`

[3]`http://www.cis.ohio-state.edu/ barr/dnswalk/`

Diplomarbeit am LRZ

Eine Diplomarbeit [Hauck94] am Leibniz Rechenzentrum beschäftigt sich mit dem Management-Aspekt der DNS-Verwaltung.

Hauptaugenmerk liegt auf der Integration in die Management-Umgebung des LRZ. Es wird eine geeignete relationale Struktur für die in diesem speziellen Umfeld zu verwaltenden DNS-Daten und ein Algorithmus implementiert, der daraus die zugehörigen Zonen- und Konfigurationsdateien erstellt. Fehlermanagement und Verwaltungswerkzeuge sowie eine Oberfläche fehlen hingegen völlig.

WebDNS

WebDNS[4] ist ein frei verfügbares Perl Skript, mit dem sich bestehende DNS-Dateien in eine eigene, temporäre Datenbank einlesen lassen. Dort werden sie bearbeitet und auf Anforderung die Zonendateien neu erzeugt. Die weitere Entwicklung wurde allerdings eingestellt.

GASH

Die Group Administrator Shell (GASH[5]) wird an der University of Texas at Austin eingesetzt, um das dortige NIS und DNS zu verwalten. Die Bedienung von GASH ist kommandozeilenorientiert. Im Moment sind hauptsächlich Befehle zur NIS-Verwaltung vorhanden, die Implementierung der DNS-relevanten Funktionen ist nur rudimentär.

3.3.3 Kommerzielle integrierte Lösungen

Während die vorstehenden integrierten Lösungen frei erhältlich waren, sind die nachfolgend aufgeführten kommerzielle Produkte. Diese verwenden üblicherweise proprietäre Nameserver, wodurch sie nur bedingt zu den Standards kompatibel sind und eine Abkehr von der bisher genutzten Nameserver-Software verlangen.

DNS BOSS

DNS BOSS[6] ist eine graphische Benutzerinterschnittstelle für die Installation und Wartung eines DNS-Servers auf SUN-Plattformen. Es unterstützt den Anwender in der Verwaltung von DNS-Domänen und erzeugt Zonendateien.

NetID

NetID[7] von BayNetworks ist für Windows sowie Unix verfügbar. Bei NetID kommen eigenentwickelte DNS und DHCP Server zum Einsatz. Es wird eine Oracle bzw. Sybase - Lizenz benötigt, da NetID seine Daten in einer solchen Datenbank ablegt.

Interessant ist die Zusammenarbeit zwischen DNS und DHCP, wobei DHCP bei Internet Service Provider nicht benötigt wird (s. 2.2.4).

[4] `http://hem.passagen.se/hno/webdns/`
[5] `http://www.arlut.utexas.edu/csd/gash_docs/gash.html`
[6] `http://www.dnsboss.com`
[7] `http://www.baynetworks.com/Products/nav/f_netid_3_0.html`

Cisco DNS/DHCP Manager

Cisco ersetzt mit seinem DNS/DHCP Manager[8] den vorhandenen Nameserver durch eigene DNS und DHCP Server. Die mitgelieferte graphische Benutzeroberfläche greift direkt auf den Datenbestand dieser Server zu.

Namesurfer

Am weitesten fortgeschritten ist Namesurfer[9] der finnischen Firma Namesurfer Ltd. Es erfüllt viele der in der Aufgabenstellung beschriebenen Anforderungen, darüberhinaus ist eine speziell für ISPs angepaßte Version angekündigt.

Allerdings ist das zugrundeliegende Konzept doch in einigen Punkten unbefriedigend. Auf diese wird in der anschließenden Bewertung im Besonderen eingegangen.

3.3.4 Bewertung

Werkzeuge

Die teilweise recht ausgereiften Werkzeuge für Einzelaufgaben stellen sicherlich sehr nützliche und wichtige Funktionalitäten zur Verfügung. Aufgrund der völlig fehlenden Interoperabilität untereinander sind solche Einzellösungen nur als Anregung für die beim Management auftretenden Probleme sowie eine mögliche Implementierung zu bewerten.

(halb-)integrierte Lösungen

Bei den halbintegrierten Lösungen fehlt immer noch ein durchgängiges Konzept, sodaß für diese im Endeffekt das Gleiche wie für die Werkzeuge (s.o.) zutrifft.

Die integrierten Lösungen sind fast ausschließlich kommerzielle Entwicklungen. Neben dem meist proprietären Lösungsansatz als Hauptkritikpunkt treffen weiterhin folgende Punkte bei den meisten der betrachteten Lösungen zu:

- keine einheitliche Benutzeroberfläche (GASH) oder nur für proprietäre Systeme (DNS BOSS)
- fehlende IP-Adreßverwaltung (alle)
- keine Unterstützung bei der RIPE-Dokumentation
- Verwaltung nur der eigenen Domäne, aber nicht von entfernten Zonen - daher nicht geeignet für ISPs
- allenfalls rudimentäre Fehlerprüfungen beim Erzeugen der Zonendateien

NameSurfer

Wie bereits erwähnt, bietet NameSurfer viele der angestrebten Funktionalitäten.

Eine WWW-basierte, simple Benutzerschnittstelle dient zur Verwaltung der Daten. Deren Eingabemasken bieten jedoch keinerlei Auswahlmöglichkeiten aus bereits vorhandenen Daten, diese müssen vielmehr händisch aus dem Gedächtnis übertragen werden.

[8] `http://www.cisco.com/univercd/cc/td/doc/prod_cat/pcdns.htm`
[9] `http://www.namesurfer.com`

Es wird keine thematische Abgrenzung zwischen den Daten vorgenommen, weder bei der Auflistung noch bei ihrer Bearbeitung. Dies erschwert Einsteigern den Überblick und dem erfahrenen Benutzer verwehrt der willkürliche, dem üblichen Vorgehen widersprechende Aufbau ein intuitives Arbeiten.

Die Dokumentation der vergebenen Netze bei RIPE ist lediglich angekündigt. Eine Adreßverwaltung ist nicht vorgesehen.

Fazit

Die oben vorgestellten Lösungsansätze sowie etliche weitere, die hier nicht erwähnt werden konnten, bieten keine Lösung für die Verwaltung des Domain Name Service, die als befriedigend bewertet werden könnte. Um in der Lage zu sein, eine integrierte Lösung zu entwickeln, müssen zunächst die management-relevanten Bereiche des Domain Name Systems untersucht werden.

3.4 Management-Objekte

Elementarer Bestandteil des Netzwerk-Managements sind die zu managenden Objekte. Diese werden anhand der Anforderungen an die DNS-Verwaltung definiert.

3.4.1 Objektklassen

Die wichtigsten Objektklassen bzgl. des DNS sind die *Zonen*, *Hosts*, *Nameserver* und zur Verwaltung *Points Of Presence* sowie *Benutzer*.

Zone

Die Zonendaten wie Domain Name, SOA Record, Verzeichnisse und Dateinamen für die Zonendateien sind hier unterzubringen. Eine Zone wird von einem *Primary Nameserver* verwaltet.

Ihr zugeordnet können außerdem *Mail Exchanger*, *Vaterzone*, *Secondary Nameserver* und *Forwarder* sein. Im Rahmen der Verwaltung besitzt eine Zone einen oder mehrere *Blöcke mit IP-Adressen* und bzgl. der Anwenderschnittstelle sind möglicherweise *Zonenverwalter* für sie zuständig.

Host

Ein Host wird beschrieben durch einen Namen und eventuell durch seine Hard- und Softwareausstattung.

Er ist einer *Zone* zugeordnet, hat eine oder mehrere *IP-Adressen* sowie eventuell Aliase, fährt *wohlbekannte Dienste* und es kann ein *Benutzer* (als Responsible Person) für ihn zuständig sein.

Nameserver

Nameserver laufen auf einem *Host* und besitzen Informationen über *Adressen*, *Verzeichnisse* und *Dateinamen* der Zonen- und der Root-Cache Dateien.

Point Of Presence

Die Standorte eines Providers sind durch ihren *Namen* und den zugeordneten *Class-C Netze* definiert.

Benutzer

Bei Benutzern sind Informationen zur *Authentifizierung* und für *RIPE* zu finden.

3.4.2 Objekte und Relationen

Die weiteren MOs, die zum Teil zu den oben beschriebenen Objektklassen gehören oder diese verknüpfen[10], werden nachfolgend beschrieben.

zu Zonen

Die folgenden Objekte sind der Objektklasse *Zone* zugeordnet.

- **Mail Exchanger**
  Ein Mail Exchanger hat einen *Namen* sowie *Präferenzwert* und läuft auf einem *Host*.
- **Secondary Nameserver**
  Eine Anzahl von *Nameservern* spiegeln die *Zonen*daten.
- **Zonenverwalter**
  Ein *Benutzer* ist als Zonenverwalter nur für bestimmte *Zonen* zuständig.

zu Hosts

- **Alias**
  In einer bestimmten *Zone* wird ein *Alias* definiert.
- **WKS**
  Well Known Services.

IP-Adreßverwaltung (zu POP, Zone und Host)

- **IP-Adreßblock**
  Ein Adreßblock beginnt bei einer *Adresse* und hat eine bestimmte *Größe*
- **IP-Addresse**

Forwarder

Ein Forwarder leitet Anfragen zu einer Menge von *IP-Adressen* an bestimmte *Nameserver* weiter.

[10] Nach 2.3.1 sind Relationen ebenfalls Management-Objekte

3.4.3 Zentrale oder verteilte Datenhaltung

Das Für und Wider der verschiedenen Möglichkeiten der Datenhaltung sind abzuwägen, bevor die eigentlichen Strukturen definiert werden.

Simple Network Management Protocol (SNMP)

Aufgrund seiner Verbreitung soll hier zunächst die verteilte Datenhaltung über das sogenannte *Internet Management* betrachtet werden.

Das Internet-Management des DNS über SNMP wird in zwei Request For Comments vom Mai 1994 behandelt, die allerdings nur die entsprechenden Management Information Bases (MIB) definieren. Es sind dies DNS Server MIB Extensions [RFC 1611] und DNS Resolver MIB Extensions [RFC 1612].

Auf Anfrage bei einem der Autoren (s. Anhang E) wurde mitgeteilt, daß bis heute keine Implementierung, die ja im Nameserver selbst zu erfolgen hätte, existiert.

Abgesehen davon sind, wie bei SNMP üblich, nur grundlegendste Funktionalitäten abgedeckt, etwa eine minimale Konfiguration des Servers (die sich wiederum nur auf dessen Steuerung, also beispielsweise *Starten* und *Stoppen* beschränkt), oder die Abfrage von Statistik-Informationen (z.B. Counter). Es ist kein aber Zugriff auf Logdateien oder gar das Setzen einzelner Einträge in einer Zonendatei vorgesehen. Der Sicherheitsaspekt wird erst in der Version 2 des SNMP gebührend berücksichtigt.

Eine Realisierung aufbauend auf SNMP schloß sich damit sehr schnell aus.

Zentrale Datenhaltung

Auch wenn eine zentrale Datenhaltung gewisse Nachteile aufweist, eignet sie sich hervorragend für die Nameserver-orientierte Verwaltung eines Teilbereiches des DNS-Namensraumes.

Die Begründung hierfür liegt in der Aufgabenstellung: Die Konsistenz des *gesamten* Datenbestandes ist nur sicherzustellen, wenn zu jeder Zeit die Informationen zu *allen* verwalteten Zonen verfügbar sind. Dies wird sich ebenfalls bei der Administration zeigen: Es können aus den bestehenden Daten Auswahlmöglichkeiten angeboten werden, was gegenüber händischer Eingabe die Fehlerquellen reduziert.

Auf der anderen Seite ist die Frage der Verteilung der erzeugten Zonendateien zu beantworten ist (s. 3.8.1). Außerdem stellt eine zentrale Datenbank natürlich einen *Single Point of Failure* dar, wie etwa ein Firewall oder ein Webserver auch. Eben aus diesem Bereich sind Lösungen zur Ausfallsicherheit bekannt: Hochverfügbarkeit der Rechner durch ein Backup-System[11], sowie redundante Datenhaltung, um einen reibungslosen Betrieb zu gewährleisten.

3.4.4 Datenbank-Design

Die oben beschriebenen Objektklassen werden untereinander mittels des Entity-Relationship Modells relationaler Datenbanken verknüpft.

Entity-Relationship Modell

In diesem Modell werden Objekte, die sogenannten *Entities*, durch Zuordnungen (*Relationships*) miteinander verknüpft. Entities besitzen Attribute, das sind Variablen, über welche

[11] etwa HACMP von IBM

ihre Eigenschaften beschrieben werden.

Bei der Umsetzung eines Entity-Relationship-Modelles in eine relationale Datenbank werden die Objekte in Tabellen abgebildet. Die Relationen werden durch Verknüpfungen über *Schlüssel* realisiert.

Ein Schlüssel ist eine eindeutige Teilmenge der Attributwerte, durch welchen Objekt innerhalb der Datenbank gefunden werden kann.

Tabellen

Die Umsetzung der Objekte und Relationen in Tabellen wird im Rahmen der Implementierung (s. 4.1.1) angegeben.

Relationen

Sogenannte *Fremdschlüssel* (engl. *foreign keys*), die den *Primärschlüssel* (*primary key*) einer anderen Tabelle referenzieren, beschreiben die Relationen zwischen den Entities.

Diese Fremdschlüssel werden bei 1:1, 1:N und M:1 Beziehungen in die jeweilige Tabelle eingetragen, bei M:N Relationen ist eine eigene Tabelle nur mit den beiden Fremdschlüsseln zu erstellen.

Die Abhängigkeiten der Objekte untereinander sind in Tabelle 3.1 beschrieben.

Sonderfall

Eine zirkuläre Abhängigkeit besteht zwischen den Entities Zonen, Nameserver und Hosts, falls der Host mit dem Nameserver in der selben Zone steht, für die er primärer Nameserver ist[12]:

1. **Zonen**
   Jede Zone benötigt einen primären Nameserver, ohne den sie im Domain-Namensraum nicht bekannt sein kann.

2. **Nameserver**
   Ein Nameserver wiederum bedingt einen Host, auf welchem er läuft.

3. **Host**
   Hosts müssen immer einer Zone zugeordnet sein.

Dies ist demnach ein klassisches *Chicken-and-Egg* Problem. Die Frage, wie sich diese Verklemmung auflösen ließe (d.h. „wer zuerst da war"), ist bei relationalen Datenbanken nicht zu umgehen.

Soll der primäre Nameserver also in der erst zu erstellenden Zone liegen, würde eine dieser referentiellen Integritätsbedingungen immer verletzt. Aus der Tatsache, daß bei ISPs üblicherweise *ein* primärer Nameserver für viele hundert Zonen zuständig ist, ergibt sich allerdings, daß dieses Problem nur selten auftritt.

Der notwendige Einschnitt wird gemäß dem erwarteten Arbeitsablauf an der Stelle vorgenommen, die einerseits am wenigsten die Verwendung einschränkt und andererseits weiterhin eine hohes Maß an Integrität sicherstellt: es wird darauf verzichtet, daß einer Zone sofort ein primärer Nameserver zugewiesen wird.

[12] Entgegen [Hauck94], 5.5.1, der dies grundsätzlich annimmt.

	Tabelle 1	Tabelle 2	Relation	Beschreibung
1	Host	IP-Adresse	1:N	ein Host hat eine oder mehrere IP-Adressen (multihomed)
2	Zone	Host	1:N	ein Host kann genau einer Zone angehören
3	Zone	Zone	1:1	eine delegiertse Zone besitzt eine Vaterzone
4	Nameserver	Host	N:1	Host, auf dem ein Nameserver läuft
5	Mail Exchanger	Zone	N:1	Mail Exchanger innerhalb einer Zone
6	Mail Exchanger	Host	N:1	Host, auf dem der Mail Exchanger läuft
7	Alias	Zone	N:1	Alias innerhalb einer Zone
8	Alias	Host	N:1	Aliase für einen Host
9	WKS	Host	M:1	die wohlbekannten Dienste eines Hosts
10	IP-Adresse	Zone	1:1	IP-Adreßverwaltung: die Zuordnung einer (evtl. unbelegten) IP-Adresse zu einer Zone
11	IP-Adresse	Host	1:1	IP-Adreßverwaltung: die Belegung einer IP-Adresse in einer Zone
12	Secondary	Zone	1:N	ein Secondary Nameserver kann für mehrere Zonen zuständig sein
13	Secondary	Nameserver	N:1	Nameserver, auf dem der Secondary-Dienst läuft
14	Nameserver	Forwarder	1:N	Nameserver, für den die Forwarder-Adressen relevant sind
15	Class-C Netz	POP	N:1	die zu einem POP gehörenden Class-C Netzwerke
16	Adreßblock	Zone	N:1	Zuordnung von Adreßblöcken zu Zonen; nur zur übersichtlicheren Verwaltung
17	Host	Benutzer	N:1	Responsible Person für diesen Host
18	Zone	Benutzer	N:1	Responsible Person für diese Zone
19	Zone	Benutzer	M:N	Zonen, die diese Benutzer (mit Berechtigung Zonenverwalter) bearbeiten dürfen

Tabelle 3.1: Relationen zwischen den Objekten

3.5 Fehlermanagement

Die Korrektheit und Konsistenz der Daten, für die eben Strukturen entwickelt wurden, ist Thema des folgenden Abschnittes.

Aufgabe des Fehlermanagement ist (nach [Hege95]) das „Erkennen, Lokalisieren und Beheben von Fehlern“.

Die *Erkennung* basiert auf der Überwachung von Schwellwerten, beispielsweise bei Zählern oder Prüfsummen. Für die *Lokalisierung* eines Fehlers sind spezielle Kenntnisse des überwachten Systems sowie entsprechende Werkzeuge unabdingbar. Die anschließende *Behebung* wiederum sollte sich in den meisten Fällen weniger kompliziert gestalten.

Der Fehlererkennung muß eine Zusammenstellung der möglichen Arten und Symptome von Fehlern vorausgehen. Gleichzeitig sind Strategien zur Vermeidung solcher Fehler bzw. ihrer Behandlung, sollten sie doch auftreten, zu entwickeln.

Dabei sind nicht nur die einfacheren Verstöße gegen die Syntax, sondern besonders das Zusammenspiel der Daten als Ganzes, das ja mehr als die Summe seiner Teile darstellt, zu betrachten.

Im Rahmen der Implementierung ist bereits auf der Ebene des Datenbankdesigns eine erste Kontrolle wünschenswert und auch realisierbar. Bei relationalen Datenbanken lassen sich über sogenannte *Constraints* (unter anderem `NOT NULL`, `UNIQUE`) sowie durch die verschiedenen Datentypen Einschränkungen für die Daten definieren. Die Relationen selber bieten sich für die Modellierung der Abhängigkeiten bestimmter Daten untereinander an. Vorteil dieser Vorgehensweise ist die zusätzliche, wenn natürlich auf dieser Ebene bei weitem nicht vollständige, Überprüfung der Daten: selbst wenn in den Algorithmen des Verwaltungswerkzeuges etwas übersehen wird, meldet spätestens das DBMS einen Fehler.

Auch die Verwaltungsoberfläche kann zur Fehlervermeidung herangezogen werden, indem Eingabefelder auf eine bestimmte Länge beschränkt werden, nur eine Auswahl aus Listen ermöglicht oder während der Eingabe bereits eine syntaktische Prüfung vorgenommen wird.

Prüfungen können bereits bei der Eintragung in die Datenbank geschehen, so beispielsweise auf korrekte Syntax. Andere Fehler sind erst im Zusammenspiel der gesamten Daten beim Erzeugen einer Zonendatei ersichtlich. Die zwei folgenden Abschnitte behandeln diese beiden Konstellationen.

3.5.1 Prüfungen

Die syntaktischen Beschränkungen, von denen nur die wichtigsten aufgeführt werden, orientieren sich hauptsächlich an den entsprechenden Vorschriften [RFC 1035] zu den jeweiligen Resource Records.

Allgemeine Regeln

Als *Argument* (also dem Eintrag in den Datenbereich des Resource Records) sind keine Aliase zugelassen und auch nicht sinnvoll, da sonst erst der Alias aufgelöst werden muß und somit die Performanz leiden würde.

Bestimmte Resource Records erwarten zwei Argumente. Fehlt eines, wird für dieses von einigen Servern ein undefiniertes Ergebnis geliefert.

Enthält ein Argument Leerzeichen, so ist der gesamte String in Anführungszeichen zu fassen.

Domänen-, Host- und Mailer-Namen

Obwohl das DNS wenig restriktiv im Bezug auf *Domain*namen ist, sollten bei *Host*- bzw. *Mailer*-Namen die entsprechenden Regeln beachtet werden.

Ein **Domain Name** darf insgesamt 255 Zeichen umfassen, wobei der Standard pro *Label*, also dem Abschnitt zwischen zwei Punkten, eine Länge von 63 erlaubt.

Hostnamen können aus Buchstaben und Ziffern (allerdings nicht ausschließlich) bestehen ([RFC 1035], [RFC 1123]). Auch das Minuszeichen darf vorkommen, jedoch nicht als erstes oder letztes Zeichen des Namens. Unterstriche sind nicht ausdrücklich verboten, können aber Probleme mit einigen Nameserver-Implementierungen verursachen.

Diese Regeln gelten ebenfalls für SMTP-basierte **Mail** [RFC 822]. Die Sonderzeichen `()≪≫[]@;:.` sind hingegen tabu. Das Ausrufezeichen '!' sollte aufgrund des Konfliktes mit UUCP-Mail[13] ebenfalls gemieden werden.

SOA, Start Of Authority

In der **Mailadresse** für die Kontaktperson ist der Klammeraffe (`@`) durch einen Punkt (.) zu ersetzen.

HINFO, Host Information

Wenn Host Informationen weitergegeben werden sollen, so ist es unerläßlich, daß beide Einträge (Hardware *und* Betriebssystem) vorhanden sind.

MX, Mail Exchanger

Falls kein Name angegeben wird, bezieht sich der MX-Record auf die aktuelle Zone.

Es sollten keine Wildcard-MX Records verwendet werden. Sie werden sowieso nur für Hosts benötigt, die nicht mit dem Internet verbunden sind; dieser Fall dürfte bei ISPs nicht vorkommen.

CNAME, Canonical Name (Alias)

Aliase können nur auf Blätter des Namensbaumes, also auf Hosts, zeigen. Will man beispielsweise alle Hosts in eine neue Domain umziehen, müßte für jeden Einzelnen ein `CNAME`-Record eingetragen werden, nicht einfach nur für die Zone selbst.

Obwohl ein Alias auf einen anderen Alias nicht ausdrücklich verboten ist, sollte dies tunlichst vermieden werden, um unübersichtliche Konfigurationen zu vermeiden.

Ein Alias ist auch auf fremde Hosts außerhalb der eigenen Zone möglich.

NS, Name Server

Mindestens zwei Nameserver sollen pro Zone vorhanden sein. Secondaries außerhalb der eigenen Domain sind ebenfalls sinnvoll (oft stehen diese beim darüberliegenden Provider).

3.5.2 Häufige Fehler

Einen Nameserver aufzusetzen ist keine triviale Aufgabe, nicht nur aufgrund der verteilten Struktur des Domain Name System. Verschiedenartigste Fehler können sich einschleichen und viele Entscheidungen müssen getroffen werden, welche Daten in das DNS einzutragen sind und wie die Nameserver konfiguriert werden soll.

Einige der zusammengetragenen Fehlerquellen sind aus *Common DNS Operational and Configuration Errors* [RFC 1912] entnommen. Ähnlichen Inhaltes sind eine weitere Anzahl von Dokumenten, wie etwa die CIX Tech Tips[14].

---

[13] UUCP trennt die einzelnen *Hops*, die die Mail durchlaufen soll, durch Ausrufezeichen; Beispiel: `uunet!nicnet!asterix!ram`

[14] `http://www.cix.org/Archive/1995/TechTips/dec.html`

Daraus sind nachfolgend die häufigsten Fehler sowie deren Auswirkungen zusammengestellt, verbunden mit einer Analyse, ob und wie sie durch Einsatz eines Werkzeuges vermieden werden können.

Für die Einzelheiten zu den Begriffen sei auf den Abschnitt über das Domain Name System in der Einleitung (2.2.2) verwiesen.

Seriennummer

Als **Serial Number** sind zwar Dezimalzahlen zulässig [RFC 1982], ihre Verwendung verursacht jedoch Schwierigkeiten. Diese Zahl wird intern in einen vorzeichenlosen 32-Bit Integerwert umgewandelt, wodurch bei Vergleichen unerwartete Ergebnisse die Folge sind.

Die selben Auswirkungen hat das Belassen der Seriennummer auf ihrem ursprünglichen Wert[15] nach einer Änderung der Zonendaten: Secondary Nameserver halten ihre (gespiegelten) Daten noch für aktuell und liefern somit veraltete Daten aus.

Eine Lösung, die bis ins Jahr 4294 vorhalten wird, ist diese Formatierung der Seriennummer: `JJJJMMTTnn`, wobei `JJJJ` das Jahr, `MM` den Monat und `TT` den Tag bezeichnet; `nn` ist eine zweistellige Revisionsnummer, um mehrere Änderungen pro Tag zu erlauben.

Solchermaßen geartete Seriennummern können automatisch bei der Generierung neuer Zonendateien eingesetzt werden.

Domänennamen

Werden vollständige Domänennamen (FQDNs) in der Zonendatei nicht mit einem Punkt beendet, ergänzt sie der Nameserver mit dem Namen der aktuellen Zone.

Dieses simple, aber oft anzutreffende Versäumnis verhindert die korrekte Registrierung eines Hosts im Namensraum; dieser bleibt somit unbekannt.

Reverse DNS

Bei Neueintragungen oder Änderungen der IP-Adresse(n) von Hosts werden gerne die entsprechendenden `PTR` Records für den Reverse Lookup vergessen. Dies wird durch die Aufteilung der `A` und `PTR` Records in zwei Dateien begünstigt.

Ist ein Host multi-homed, hat er also mehr als eine IP-Adresse, so müssen alle Adressen einen entsprechenden `PTR`-Record haben, nicht nur die erste.

Bemerkbar macht sich dieser Fehler durch langsame Start- bzw. Ausführungszeiten von Programmen, die ein Reverse Lookup durchführen. Zu diesen gehören unter anderem Telnet und Traceroute. Noch einschneidender ist ein kompletter *Denial of Service* von einem Host ohne `PTR` Eintrag.

Mangelnde Koordination zwischen Zonenverwaltern

Die hierarchische Struktur des DNS ist nicht nur von Vorteil: wird der jeweilige Partner in der höheren bzw. niedrigeren Ebene nicht von Änderungen informiert, die ihn ebenso betreffen (das ist üblicherweise bei Delegierungen der Fall), können Konflikte auftreten. *Lame Delegation* und fehlerhafte *Glue Records* treten dabei am häufigsten auf.

[15] dies ist der häufigste Grund für fehlerhaftes Verhalten des DNS

Liegen sowohl der Vaterknoten als auch sein Sohn und somit die zugehörige Delegierung innerhalb des Namensraumes, welcher mittels der Datenbank verwaltet wird, können solche Fehler vom DNS-Management abgefangen werden.

Glue Records

Oftmals werden unnötige Glue Records (s. 2.2.2) in Zonendateien eingetragen. Schlimmer noch sind *falsche* Glue Records für einen externen Host in der primären Zonendatei.

Ist ein Host zwar über seine IP-Adresse erreichbar, nicht aber über seinen Namen, liegt höchstwahrscheinlich ein fehlerhafter Glue Record vor.

Lame Delegation

Als *Lame Delegation*[16] wird der Umstand bezeichnet, daß ein Nameserver als authoritativ für eine Zone eingetragen ist, in Wirklichkeit aber keinen Dienst für diese Domain leistet.
Mögliche Ursachen für Lame Delegations:

- Ein primary oder secondary Nameserver wird umgezogen, ohne die zuständigen Administratoren der anderen betroffenen Nameserver zu informieren. (s. 3.5.2)

  Zieht ein Primary um, müssen alle Secondaries ihre Konfiguration anpassen und ihre Nameserver neu starten. Beim Umzug eines Secondaries müssen die Adressen beim Primary und in der Vaterzone geändert werden.

- Nameserver Einträge werden gelöscht, ohne den Parent zu informieren, welcher einem Authorität über diese Zone verliehen hat.

- Eine Vaterdomäne delegiert eine Subdomain an einen Nameserver, der für diese Zone gar nicht authoritativ ist.

Die Folgen einer solchen Lame Delegation reichen von zusätzlichen Netzwerkverkehr bis zu nichtauflösbaren Hosts und unzustellbarer E-Mail. Auch die Effizienz des Domain Name Systems leidet darunter.

Root Cache

In der Root Cache Datei sind die Domainnamen und IP-Adressen der Rootserver für jede der Top-Level Domains zu finden. Änderungen an diesen Informationen werden nicht automatisch verteilt, sondern die Datei muß erneut geholt werden[17].

3.5.3 Protokollierung

Für das Anlegen von Logdateien sprechen vor allem zwei Gründe:

- **Nachvollziehbarkeit der erfolgten Veränderungen**
  Dies ist vor allem zur Fehlerbehebung unerläßlich, um beispielsweise zum ursprünglichen Stand zurückzukehren.

[16] unbeschadet der Tatsache, daß ISPs üblicherweise keine Delegierungen innerhalb ihres Namensraumes vornehmen

[17] per FTP von `rs.internic.net/domain/named.root`; ein Update von Hand ist aus naheliegenden Gründen nicht zu empfehlem

- **Aufspüren von Manipulationen**
  Unter *Manipulation* versteht man das unberechtigte Verändern von Daten[18]. Als Reaktion darauf darf nicht nur die Korrektur der manipulierten Daten stehen, sondern auch das Aufspüren des Verursachers. Dies ist wiederum nur anhand von Aufzeichnungen möglich.

Die für eine solche Archivierung interessanten Daten sind auf alle Fälle *Datum* und *Uhrzeit*, der *Benutzer*(über Login) und die ausgeführte Aktion.

Log-Level

Um nicht an der Datenflut zu ersticken, sind je nach Bedarf mehrere Stufen für die Art der Protokollierung vorzusehen. Sinnvoll erscheint eine Aufteilung, wie sie in Tabelle 3.2 vorgeschlagen wird.

Level	Art der Protokollierung
0	kein Logging
1	nur Fehler; Login/Logout
2	insert, update, delete
3	alles

Tabelle 3.2: Verschiedene Stufen der Protokollierung

Log-Dateien

Eine Kategorisierung der verschiedenen Logmeldungen, die dann in unterschiedliche Dateien geschrieben werden, erhöht die Übersichtlichkeit. Die einzelnen Dateien sind in Tabelle 3.3 zusammengestellt.

Datei	Inhalt
`access.log`	Datenbank-Zugriffe, je nach Log-Level
`auth.log`	erfolgreiche und zurückgewiesene Anmeldungen
`error.log`	Fehlermeldungen

Tabelle 3.3: Aufteilung der Log-Dateien nach Meldungsart

3.6 Konfiguration und Generierung

Dreh- und Angelpunkt der DNS-Verwaltung ist die korrekte Generierung der Zonen- und Konfigurationsdateien für den Nameserver. Da oftmals bereits eine nicht zu unterschätzende Anzahl an Zonendateien vorhanden ist, ist ein (einmaliger) Import in die Datenbank wünschenswert. Bei der Generierung ist die ganze Palette der Möglichkeiten des Domain Name Systems zu beachten sowie die in 3.5 erstellten Mechanismen zur Fehlervermeidung einzubeziehen.

[18]Somit spielt dieser Punkt auch in das Sicherheitsmanagment mit hinein.

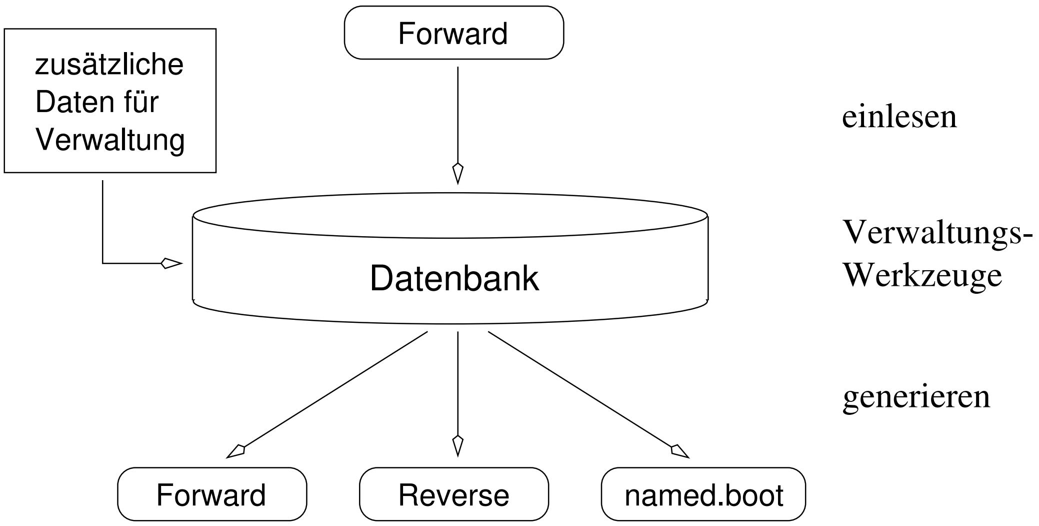

Abbildung 3.1: Quellen und Senken der Zonendaten

Das Einlesen der Zonendaten in die Datenbank sowie die Generierung aktueller Konfigurationen ist in Abbildung 3.1 dargestellt.

3.6.1 Vorhandene Zonendateien einlesen

Die vorhandenen Datenbestände sollen bei der Umstellung auf dieses Verwaltungswerkzeug in die Datenbank eingelesen werden, um eine zeitaufwendige und fehleranfällige Neueingabe zu umgehen.

Zwei Durchläufe

Die in den Zonendateien enthaltenen Informationen beziehen sich zum großen Teil direkt aufeinander (so etwa bei Aliasen) oder auf Informationen aus anderen Zonendateien (zum Beispiel auf Secondary Nameserver).

Aus diesem Grund werden die Zonendaten in zwei Durchläufen in die Datenbank übertragen:

1. Zonen (`SOA`), Hostnamen und IP-Adressen (`A`)
2. Nameserver (`NS`), Mail Exchanger (`MX`), Alias (`CNAME`);
   Primary Nameserver der Zonen

Zuerst müssen nämlich einige Basisinformationen, das sind die Zonen-`SOA`s sowie die Hosts zusammen mit ihren IP-Adressen (`A`-Records), von *allen* zu importierenden Zonendateien in der Datenbank vorhanden sein.

Erst im zweiten Durchgang, bei dem Nameserver (`NS`), Mail Exchanger (`MX`), Alias (`CNAME`) eingelesen werden, können dann die entsprechenden Relationen aufgebaut werden. Andernfalls würden die meisten Ziele der zu erstellenden Referenzen im Datenbestand noch fehlen. Die Festlegung des primären Nameservers einer Zone ist jeweils erst ganz am Ende möglich, da das Vorhandensein eines normalen `NS`-Records für diesen sicherzustellen ist.

In den Zonendateien enthaltene Kommentare werden, da sie nicht zuordenbar sind, verworfen.

`/etc/hosts` importieren

Die `/etc/hosts` - Dateien werden nur für lokale Rechner in abgeschlossenen Netzen eingesetzt, bei der Verwaltung von externen Netzen muß das Domain Name System verwendet werden. Schon allein aus diesem Grund lohnt sich die Implementierung eines Werkzeuges, das auch das Format der `/etc/hosts` - Dateien versteht, nicht.

Abgesehen davon ist über das weitverbreitete Tool `h2n` (aus [1]) die Umsetzung von `/etc/hosts` in Zonendateien möglich. Die so erhaltenen Dateien lassen sich dann ohne weiteres mit obiger Importfunktion einlesen.

3.6.2 Zonendateien erzeugen

Bei der Generierung der Dateien für einen Nameserver aus dem Bestand der Datenbank werden mehrere Dateien erzeugt.

Jede einzelne Zone erhält ihre Forward-Zonendatei, die für die Rückwärtsauflösung werden Class-C - weise zusammengefaßt. Ausnahme ist dabei das sogenannte Loopback-Interface.

Jede Zonendatei erhält am Anfang einen `SOA`-Record. Dessen Seriennummer wird bei jedem Durchlauf neu erstellt (zum Format siehe 3.5.2). Daran schließen sich die Einträge für die primären und sekundären Nameserver an.

Die einzelnen zu generierenden Dateien sind nachfolgend zusammengestellt.

Nameserver-Konfiguration

Pro Nameserver wird eine Konfigurationsdatei benötigt, `named.boot` genannt. In ihr sind die Namen der Dateien für den Root-Cache und die der einzelnen Zonen eingetragen, für welche ein Nameserver Primary oder Secondary ist. Außerdem werden bei Bedarf *Forwarders* definiert.

Forward-Zonendateien

Die meisten Informationen aus der Datenbank gehen in diese Zonendateien in Form von Resource Records ein. Besondere Beachtung ist dabei den Nameserver-Einträgen bzgl. vorhandener Delegierungen sowie aufgrund dieser benötigten Glue Records zu schenken.

Reverse Mapping

Für jede Zone wird normalerweise eine eigene Datei generiert, die das Reverse Mapping enthält.

Um die Anzahl der Dateien niedrig zu halten[19], werden hier jedoch ganze Class-C Netze jeweils in einer Reverse Mapping Zonendatei zusammengefaßt. Dies ist möglich, da Class-C Netze immer einem bestimmten POP zugeordnet werden (s. 3.7.2) und somit denselben Primary Nameserver verwenden.

Als Secondary Nameserver werden alle Secondaries eingetragen, die in den jeweiligen Forward Zonendateien verwendet werden.

[19] ISPs verwalten meist einige hundert Zonen

Loopback Interface

Für den sogenannten *Localhost* wird pro Nameserver eine Zonendatei erstellt, die das Reverse Mapping bewerkstelligt.

Das Forward Mapping für das Loopback Interface wird standardmäßig in der normalen Zonendatei einer Zone eingetragen[20].

Root Server (`named.cache`)

Die IP-Adressen der Root Server sind in der Datei `named.cache` enthalten, die von einem dieser Server geholt und dann an die verwalteten Nameserver verteilt wird.

3.7 Verwaltung der IP-Adressen

Eine Verwaltung der Vielzahl von IP-Adressen ist für ein effizientes und fehlerfreies Arbeiten unabdingbar. Bei ISPs trifft dies aufgrund des sich andauernd und rasant verändernden Datenbestandes (ständig werden neue Teilnetze belegt, andere wieder zurückgegeben) um so mehr zu.

3.7.1 Adreßblöcke

Adreßblöcke können nur in bestimmten Größen vergeben werden, die den Potenzen der Zahl 2 entsprechen, also 2, 4, 8, 16, usw. Da zwei IP-Adressen davon verloren gehen, nämlich die niedrigste für das Routing und die höchste als Broadcast-Adresse, ist erst ein Block ab der Größe 4 (also 2^2) sinnvoll einsetzbar.

Abhängig von dieser Größe ist auch der Beginn eines Adreßblocks innerhalb eines Netzes, denn er muß auf ein Vielfaches der Blockgröße fallen. Anhand eines Beispiels sei dies verdeutlicht: Ein Block mit 16 Adressen soll belegt werden im Class-C Netz 194.55.3. Die in Frage kommenden Anfangsadressen sind demnach 0, 16, 32, 48, usw., je nachdem, wo noch ein solcher Bereich frei ist.

3.7.2 Verteilung der Adressen

Im Grunde genommen ist die Verwaltung von IP-Adressen dreigeteilt. Folgende Objekte erhalten IP-Adressen bzw. ganze Blöcke davon:

1. **POPs**
   Den verschiedenen Points of Presence eines Providers werden ganze Class-C Netze zugewiesen, über die dieser dann verfügen kann. Die jeweils 256 IP-Adressen werden sofort in der Datenbank angelegt und ihr Status auf *frei* und *verfügbar* (s. 3.7.3) gesetzt.

2. **Zonen**
   Aus dem Adreßpool eines POPs werden den Zonen Blöcke von IP-Adressen zugeteilt, die diese nach und nach belegen.

[20] entgegen [Hauck94], 6.4.7, denn das Forward Mapping für Hosts darf nur in der Zonendatei für genau diese Zone erfolgen ([1], S. 138). Beim Reverse Mapping dagegen „heißt“die Zone ja *127.0.0*

3. **Hosts**
   Ein Host kann eine oder in Ausnahmefällen (etwa als Gateway) mehrere IP-Adressen belegen.

3.7.3 Umsetzung

Anhand dieser Überlegungen können nun die entsprechende Objekte und die Zugriffsfunktionen definiert werden.

Datenstrukturen

Die Dreiteilung der Adreßverwaltung ergibt sinnvollerweise drei Objekte in der Datenbank sowie Relationen zu bereits bestehenden Objekten.

1. **Point Of Presence**
   Ein POP ist durch seinen *Standort* definiert.

2. **Class-C Netz**
   Die Class-C Netze werden anhand ihrer *Adresse* identifiziert. Sie erhalten einen *Verweis* auf den POP, dem sie zugeordnet sind.

3. **Adreßblöcke einer Zone**
   Ein Block von Adressen (oder mehrere, falls die Zone später weitere IP-Adressen benötigte) ist über einen *Verweis* einer Zone zugeteilt. Er beginnt bei einer bestimmten *IP-Adresse* und hat eine vorgegebene *Größe*.

Funktionen

Die wichtigsten Funktionaliäten auf diesen Datenstrukturen ergeben sich wie folgt:

- **Class-C Netz**
  - freien Bereich der Größe N suchen (s. 3.7.1)
  - Bereich für Zone reservieren
- **Adreßpool einer Zone**
  - freie Adresse suchen

Status einer IP-Adresse

IP-Adressen existieren im System, sobald ein Class-C Netz bei einem POP angelegt wird. Zu diesem Zeitpunkt sind sie allerdings noch keiner Zone und somit auch keinem Host zugeordnet, also *verfügbar* und *frei*. Erhält eine Zone einen Adreßblock, in welchem eine bestimmte Adresse liegt, so ist diese nunmehr zwar an diese Zone *vergeben*, aber immer noch *frei*. Erst wenn ein Host diese IP-Adresse belegt, ist sie *vergeben* und *belegt*. Diese Zustände sind in Tabelle 3.4 anhand eines Beispieles zusammengefaßt.

	verfügbar, frei	vergeben, frei	vergeben, belegt
Host	-	-	`haigate`
Zone	-	`haitec.de`	`haitec.de`

Tabelle 3.4: Belegungszustände von IP-Adressen

3.7.4 Einlesen bestehender Dateien

Zu berücksichtigen ist, daß beim Einlesen von bestehenden Zonendateien (s. 3.6.1) keinerlei Information verfügbar ist, welche unbelegten Adressen eventuell einer Zone zugedacht sind. Demzufolge lassen sich die Adreßblöcke einer Zone nicht völlig automatisch anlegen.

Allerdings können zumindest die IP-Adressen selber gemäß obigen Schemas mit einem entsprechenden Status belegt werden. Sollten Lücken zwischen zwei IP-Adressen, die dem selben Class-C Netz angehören, liegen, so darf vermutet werden, daß alle dazwischenliegenden Adressen auch zu dieser Zone gehören. Sie sind entsprechend zu markieren.

Anhand der kleinsten Blockgröße, in die die so gekennzeichneten IP-Adressen noch passen, können auch die davor und danach liegenden Adressen einer Zone zugeteilt werden. Eventuell wird dabei aber ein kleinerer (nicht jedoch ein zu großer) Umfang des Adreßblocks angenommen.

3.8 Administration

Eine flexible Verwaltung der DNS-Daten hat zwei Hauptkriterien zu erfüllen: sie muß auf einfache Weise von überall erreichbar sein, andererseits aber dem Sicherheitaspekt für diese kritischen Daten Rechnung zu tragen.

Dieser scheinbare Widerspruch läßt sich auf elegante Art lösen: die Erreichbarkeit wird durch eine entfernte Verwaltung über das Netzwerk ermöglicht, für die Sicherheit sind bereits existente Vorgehensweisen anzuwenden.

In der folgenden Erörterung werden die Bedürfnisse der Verwaltung bei Internet Service Providern mit einbezogen.

3.8.1 Entfernte Verwaltung

Bei einer entfernten Administration greifen Benutzer über ein Netzwerk auf die Daten zu. Meist wird hierfür eine eigene Oberfläche erstellt, um den Sicherheitsbedürfnissen und der Benutzungsfreundlichkeit gerecht zu werden. Ein beispielhafter Prototyp einer solchen Oberfläche wird im Rahmen dieser Arbeit vorgestellt (s. 4.2.3 sowie Kapitel 5).

Zugriff auf die Datenbank

Als einzige Voraussetzung für den Zugriff auf die DNS-Verwaltung soll ein Internet-Anschluß genügen. Der dazugehörende HTTP-Server ist somit direkt mit dem Internet zu verbinden, also in der DMZ (*DeMilitarized Zone*) aufzustellen, wo auch andere öffentliche Server, beispielsweise für FTP, laufen.

Die eigentliche Datenbank wird hinter dem Firewall gehalten. Um nun aus der DMZ auf die Daten zugreifen zu können, ist der entsprechende Port, über den die Kommunikation zwischen

Datenbank und der eigentlichen Verwaltungssoftware abgewickelt wird, in den Filterregeln des Firewalls freizuschalten.

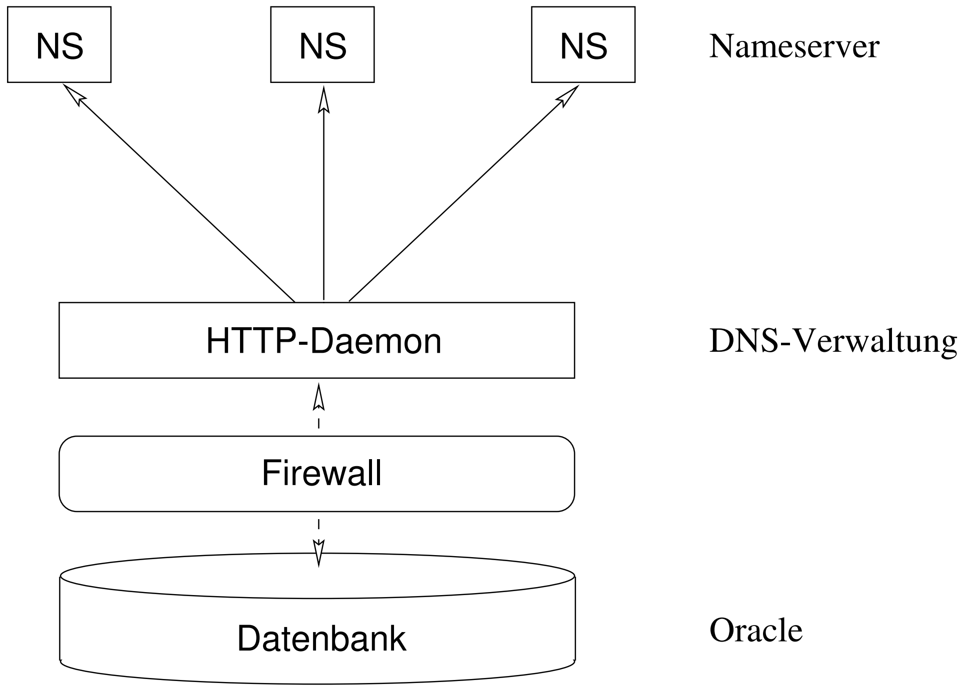

Abbildung 3.2: Datenfluß durch einen Firewall und Verteilung an die Nameserver

Verteilung der Zonendateien

Nach der Erzeugung der neuen Zonendateien müssen diese noch an die entsprechenden Nameserver übergeben und diese wiederum veranlaßt werden, die neue Konfiguration einzulesen.

Die nachfolgend angestellten Überlegungen beziehen sich dabei auf Unix-Systeme, da nur diese entsprechende Funktionalitäten zur Verfügung stellen. Dies ist jedoch nicht als Einschränkung zu sehen, da die wirklich leistungsfähigen und stabilen Systeme, wie sie ISPs zum reibungslosen Betrieb ihres Domain Name Service verwenden, nur im Unix-Umfeld zu finden sind.

Zugriff auf entfernte Rechner, zumindest aber deren Dateisysteme ist grundsätzlich über folgende Verfahren (mit Bewertung ihrer Einsetzbarkeit) möglich:

1. Dateiübertragung:

   (a) **FTP** (File Transfer Protocol)
   Einschränkung auf Teilbereiche des Dateisystems möglich

   (b) **RCP** (Remote Copy)
   unsicherer als FTP

   (c) **NFS / AFS** (Network / Andrew File System)
   nur innerhalb eines LANs vorhanden

2. Ausführung von Prozessen:

   (a) **REMSH** (bzw. **RSH** unter BSD) (Remote Shell)
   Aufruf von Programmen auf entfernten Maschinen

   (b) **RPC** (Remote Procedure Call)
   erfordert entsprechende Änderung der Nameserver-Software

Auf den Zielrechnern muß, unabhängig von der gewählten Vorgehensweise, ein entsprechender Benutzer mit ausreichenden Rechten angelegt werden.

Der Nameserver (named) läuft unter Root-Kennung, da er TCP/UDP-Sockets mit der Portnummer 53 verwendet und Portnummern unter 1024 bekanntlich nur diesen Kennungen zugänglich sind. Da der anzulegende Benutzer aus Sicherheitsgründen aber keinesfalls Superuser-Rechte haben darf, ist ein entsprechendes Skript für den Neustart des Nameservers zu schreiben. Dieses Skript schickt dem Nameserver-Prozess das entsprechende Signal SIGHUP, welches ein erneutes Einlesen der Konfigurationsdateien anstößt. Das Setzen des SUID (Set User ID) - Bits für die Skript-Datei bewirkt die Ausführung desselben mit root-Rechten.

Aufgrund dieser Anforderungen ergibt sich, daß für die Datenübertragung FTP und für die entfernte Ausführung des Nameserver-Neustarts remsh am besten geeignet sind.

3.8.2 Sicherheit

Die mitunter wichtigsten Aspekte beim Schutz von Daten sind deren Unversehrtheit und die Authentizität des Verfassers. Aus diesem Grund ist eine Benutzerverwaltung zu integrieren, die den Zugang und die Berechtigungen der Anwender regelt.

Zugangskontrolle

Die Benutzung der Verwaltungsoberfläche wird erst nach erfolgreicher Authentifizierung über Login und Paßwort gestattet. Darüberhinaus läßt sich bei den heute üblichen Webservern der Zugang auf bestimmte Netzwerke oder gar einzelne Hosts einschränken. Die Verwendung eines unüblichen Ports für den HTTP-Daemon verbirgt die DNS-Verwaltung vor neugierigen Augen.

Zusätzliche Sicherheit bietet die Protokollierung (s. 3.5.3) von akzeptierten und vor allem zurückgewiesenen Verbindungen, die unerlaubte Zugriffsversuche offenlegt.

Rollen

Unterschiedliche Aufgabenbereiche legen verschiedene Stufen von Berechtigungen nahe. Dies ist von anderen Systemen bereits hinreichend bekannt, etwa Unix, wo die Aufteilung in Benutzer, Gruppen, und Superuser verschiedene Privilegierungsstufen wiederspiegelt.

Für die DNS-Verwaltung erscheinen drei Rollen sinnvoll: *Administrator*, *Zonenverwalter* und *Responsible Person*.

Administrator Ein Administrator hat Zugriffsrechte auf alle Bereiche der Datenbank, daher sollten nur wenige Anwender dieser Stufe angehören. Ihm ist insbesondere die Benutzer- und IP-Adreßverwaltung vorbehalten.

Zonenverwalter Wie der Name bereits andeutet, ist sein Aufgabenbereich die Pflege von Zonen sowie der zugehörigen Hosts und Nameservereinträge. Durch die Beschränkung auf bestimmte Zonen kann hier auch einem Kunden das Recht eingeräumt werden, seine eigene Zone zu verwalten.

Responsible Person Diese an den korrespondierenden Resource Record (s. 2.2.3) angelehnte Bezeichnung weist auf die Verwendung hin: Eine Responsible Person erhält keine Rechte, nicht einmal für Login. Sie dient lediglich zur Zuweisung an Hosts und Zonen als Verantwortlicher und zur Verwaltung der Kontaktpersonen für die RIPE-Datenbank.

3.9 RIPE

Die Bewertung der Automatisierungsmöglichkeiten für die IP-Adreßdokumentation beim RIPE ist Thema des folgenden Abschnittes.

Zunächst wird die allgemeine Vorgehensweise erläutert, danach anhand eines konkreten Beispiels ein Ablaufplan erstellt. Diese Schematisierung dient anschließend zur Erörterung des Automatisierungspotentials.

3.9.1 Beantragung und Dokumentation

Die Adreßdokumentation ist eigentlich zweigeteilt: Man beantragt einen Bereich von IP-Adressen (die innerhalb des dem Provider zugewiesenen Adreßpools liegen müssen) für einen Kunden. Nach Erhalt der Genehmigung durch das RIPE erfolgt erst die eigentliche Dokumentation.

Beantragung Zur Beantragung eines Adreßbereiches wird ein *Address Space Request Form* [ripe-141] ausgefüllt und per E-Mail an die RIPE Hostmaster gesandt. Diese überprüfen den Inhalt innerhalb von ein bis drei Tagen. In seltenen Fällen, etwa bei Beantragung zu großer Netze, wird die Zuteilung, zusammen mit einer Begründung, abgelehnt.

Dokumentation Anschließend wird das Network Template um die richtige IP-Adresse samt Netzwerkmaske, sofern sie noch nicht bekannt war, ergänzt und an die RIPE Datenbank zur Dokumentation gesandt.

Assignment Window

Daneben erlaubt das RIPE erfahreneren Providern, selber Bereiche innerhalb des erhaltenen IP-Adressenpools zuzuweisen (anstatt sie zu beantragen), solange eine vorgegebene Größe nicht überschritten wird.

Diese Größe wächst im Laufe der Zeit. Sollte seitens des RIPE entschieden werden, dieses sogenannte *Assignment Window* eines Providers zu erhöhen, vereinfacht sich der Vorgang etwas.

Nach wie vor wird ein komplettes RIPE-141 Dokument erstellt, dieses jedoch nicht an RIPE Hostmaster gesandt, sondern nur beim Provider vorgehalten. Einzig das Network Template wird zur eigentlichen Dokumentation an die RIPE Datenbank weitergeleitet.

Bevor ein Provider *neue* IP-Adressen bekommt, pickt das RIPE drei zufällige Netze aus dem alten Bereich und fordert deren RIPE-141 Dokumentation zur Überprüfung an.

Wird das Assignment Window beispielsweise von anfänglichen 0 auf 27 gesetzt, können Netze bis 32 IP-Adressen selber dokumentiert werden.

Netzwerkmaske

Die Größe des Assignment Window wird entsprechend der zugehörigen Netzwerkmaske (auch *Subnetzmaske* genannt) angegeben. Die Netzwerkmaske ergibt die Anzahl von IP-Adressen, die in einem (Sub-)Netz vergeben sind.

Es existiert eine Kurzschreibweise (s. Tabelle 3.5) für diese Maske, die mit einem Schrägstrich an die eigentliche IP-Adresse angehängt wird. Sie entspricht der Anzahl der (von den 32 möglichen) auf 1 gesetzten Bits einer Netzwerkmaske.

Lange Form	Kurze Form	Anzahl IP-Adressen
255.255.255.255	32	1
255.255.255.254	31	2
255.255.255.252	30	4
255.255.255.248	29	8
255.255.255.240	28	16
255.255.255.224	27	32
255.255.255.192	26	64
255.255.255.128	25	128
255.255.254.0	24	256
255.255.252.0	23	512
255.255.248.0	22	1024
...		

Tabelle 3.5: Schreibweisen für die Netzwerkmaske

`195.88.98.96/29` bedeutet beispielsweise, daß acht IP-Adressen vergeben sind, das Subnetz also den Bereich von 96 bis 103 umfaßt.

3.9.2 Ablauf anhand eines Beispiels

Anhand eines Beispiel-Szenarios[21] wird nachfolgend der Aufbau eines RIPE-141 Dokumentes im Einzelnen erörtert. Dabei sei angenommen, Kunde *Irgendeine Firma GmbH* möchte ein Netz der Größe 32 belegen.

Das RIPE-141 Dokument enthält einige Abschnitte, deren Daten teils spezifisch für diese Anfrage sind (neue Daten), teils aus alten Anfragen und deren Bearbeitung entstehen (alte Daten) sowie quasi statische, wie dem Requester (Anfragenden), die sich nur selten ändern.

Firma

```
#[OVERVIEW OF ORGANIZATION TEMPLATE]#

The company Irgendeine Firma GmbH is located in Posemuckel, Germany and is
specialized in manufacturing toys. ...
```

Dieser Teil wird mit Informationen über die Firma gefüttert. Dabei handelt es sich um allgemeine Angaben zum Firmenzweck. Bei einer etwaigen geographischen Aufteilung ist die

[21] weitere Beispiele sind [ripe-142] zu entnehmen

Art der Verbindung zwischen diesen Standorten zu erwähnen, also beispielsweise durch ein flaches Netz oder viele Teilnetze.

Antragsteller

```
#[REQUESTER TEMPLATE]#

name:         Martin Rastetter
organisation: Internet Services Deutschland GmbH
country:      DE
phone:        +49 7152 9733 0
fax-no:       +49 7152 9733 13
e-mail:       m.rastetter@idnet.de
```

Der *Requester* ist der offiziell Anfragende. Es handelt sich dabei meist über einen längeren Zeitraum um diesselbe Person.

Kontaktperson

```
#[USER TEMPLATE]#

name:
organisation:
country:      DE
phone:
fax-no:
e-mail:
```

Bei diesem Teil handelt es sich um den Benutzer, bzw. dessen administrativen Kontakt. Hat der Kunde bereits eine .de Domain (bzw. eine europäische Top Level Domain (TLD) wie z.B. `.be`, `.es`, `.ch` oder eine der künftigen anderen generischen TLDs, beispielsweise `.firm`, `.web`), so ist er schon beim RIPE registriert. Andernfalls muß er dem RIPE bekannt gemacht werden.
Somit ergeben sich mehrere Möglichkeiten:

1. Der Benutzer ist bereits beim RIPE bekannt.

   Beispiel: `idnet.de` - eine Anfrage unter `http://www.ripe.net/db/whois.html?idnet.de` ergibt neben Informationen zur Domäne auch ein Personendokument:

   ```
   person:     Ralf Steiner
   address:    ID.net Kirchheim
   address:    Neue Weilheimer Strasse 14
   address:    D-73230 Kircheim
   address:    Germany
   phone:      +49 7021 94391 00
   fax-no:     +49 7021 94391 50
   e-mail:     rsn@idnet.de
   nic-hdl:    RS233-RIPE
   ```

```
changed:    foxx@idnet.de 971021
source:     RIPE
```

Darin sind alle benötigten Informationen enthalten. Tabelle 3.6 beschreibt die Inhalte des Personenobjekts.

Attribut	Inhalt
`name`	`person`
`organisation`	1.Zeile `address`
`country`	letzte Zeile `address`, aber als ISO3166 zwei-Buchstaben-Kürzel, optional als vollständiger Name
`phone`	wie
`fax-no`	Requester-
`e-mail`	Template

Tabelle 3.6: Attribute des Personen-Templates

Ein Problem kann mit der E-Mail entstehen - sie ist nicht zwingender Bestandteil eines Personendokuments, jedoch im RIPE-141. Die Fax-Nummer ist in beiden Templates optional.

2. Die Möglichkeit, daß eine Person dem RIPE ohne zugehörige Domäne bekannt ist, besteht zwar, erfordert aber zusätzlichen Aufwand. Zumindest kann ein Versuch unternommen werden, unter `http://www.ripe.net/db/whois.html` zu suchen. Wenn man zum Beispiel nach `H Hippenstiel` sucht, findet man ein volles Personendokument, obwohl dieser Name nicht vollständig ist. Ansonsten gilt 1).

3. Wenn ein Personendokument nicht existiert, muß es angelegt werden. Dazu wird ein Template an RIPE geschickt, welches automatisch bearbeitet wird. Empfänger ist die Adresse `auto-dbm@ripe.net`. Das Subjekt ist nicht zwingend notwendig, man kann aber zur übersichtlicheren Archivierung die Person und anschließend `X-Ncc-Regid: de.id-net`[22] verwenden. Ein Beispiel hierzu: `Subjekt: Johnny Benutzer; X-Ncc-Regid: de.id-net`

   Der *Body* sieht so aus:

```
password:   PassWort

person:     Johnny Benutzer
address:    Irgendeine Firma GmbH
address:    Feldbergweg 1
address:    D-82828 Posemuckel
address:    Germany
phone:      +49-8282-288
fax-no:     +49-8282-289
```

[22] `X-Ncc-Regid:` dient dazu, den Absender zu kennzeichnen.

```
email:      j.benutzer@domain.de
nic-hdl:    AUTO-1
mnt-by:     IDNET-MNT
changed:    h.hippenstiel@system-design.com
source:     RIPE
```

`IDNET-MNT` heißt, daß ID.net der Maintainer ist, `source` ist eine feste Vorgabe. Alles andere sollte anhand des Namens klar sein, bis auf `AUTO-1`: `AUTO` ist ein Platzhalter und wird aus den Initialen der Person sowie einem Zähler, wie häufig diese Initialien schon auftraten, vom RIPE gebildet. Beispielsweise hat Ralf Steiner `RS233`, weil es schon 232 vor ihm mit der Kombination RS gab.

Ein Personen-Template kann mehrere Personen umfassen, deshalb die Ziffer hinter `AUTO`, die pro Person um eins hochzuzählen ist.

Hinter `AUTO-1` darf man Initialen angeben, andernfalls werden sie automatisch aus dem Namen gebildet.

Das Feld `changed` kann mehrere Zeilen umfassen. Es enthält einen Hinweis auf die ändernde Person.

Das *Datum* wird automatisch vom RIPE ergänzt. Es stellt eine Art Versionskontrolle dar und findet auch Verwendung im Netzwerktemplate weiter unten. Das Format ist immer `JJMMTT`.

Das *Paßwort* ist zwingend in die erste Zeile zu setzen, da sonst die automatische Bearbeitung nicht funktioniert.

Damit sind alle Daten für das User-Template vorhanden:

```
name:         Johnny Benutzer
organisation: Irgendeine Firma GmbH
country:      DE
phone:        +49-8282-288
fax-no:       +49-8282-288
e-mail:       j.benutzer@domain.de
```

Allerdings fehlt noch das im Network-Template benötigte `nic-hdl`, welches vom RIPE erst vergeben wird. Für dieses Beispiel bekommt Johnny Benutzer das NIC-Handle `JB123`.

Aktuelle Verwendung des Adreßbereiches

```
#[CURRENT ADDRESS SPACE USAGE TEMPLATE]#

                                        Addresses Used
Prefix         Subnet Mask       Size    Current 1-yr    2-yr    Description
193.1.193.0    255.255.255.192    64        28    34      60     Candy Cane Dept.
193.1.193.64   255.255.255.224    32        10    21      28     Toy Design
                                  96        38    55      88     Totals
```

Das RIPE möchte wissen, wie die momentane Belegung (current) und künftige Planung des Kunden in einem (1-yr) und in zwei Jahren (2-yr) aussieht.

Waren noch keine IP-Adressen vergeben, ist nur die Totals Zeile mit vier Nullen einzutragen (0 0 0 0 Totals).

Übersicht zur Anfrage

```
#[REQUEST OVERVIEW TEMPLATE]#

request-size:        32
addresses-immediate: 32
addresses-year-1:    32
addresses-year-2:    32
subnets-immediate:   1
subnets-year-1:      1
subnets-year-2:      1
inet-connect:        Plan to connect DATUM through Internet Services Deutschland
country-net:         DE
private-considered:  yes
request-refused:     no
PI-requested:        no
addresses-returned:
```

Hier wird deklariert, was der Kunde mit dem Netz plant.

private-considered setzt man auf yes. Hierbei geht es darum, öffentliche IP-Adressen nicht für ein internes Netz zu verschwenden, da für die übliche Installation mit Router und Firewall zwei Adressen genügen.

Die Planung bis in zwei Jahren wird mit den endgültigen Grßen belegt, da meist alle Adressen auf einmal verbraucht werden. Die Anzahl der Subnetze läßt sich hier ebenfalls angeben.
Der inet-connect - Status kann folgende Werte haben:

- Will never connect
- Already connected through <Provider Name>
- Plan to connect <DATUM> <Provider Name>

Für den Fall, daß Adressen zurückgegeben werden, dokumentiert man dies mittels addresses-returned: <IP Bereich Anfang> - <IP Bereich Ende> to <Provider Name> on <DATUM>.

Adressierungsplan

```
#[ADDRESSING PLAN]#

Relative                              Addresses Used
Prefix#  Subnet Mask       Size   Immediate 1yr 2yr Description
0.0.0.0  255.255.255.224    32       32      32  32  Customer flat network,
```

```
                                                router, firewall, servers
                         32        32      32   32  Totals
```

Der Addressierungplan entspricht im Format dem *Current Address Space Usage* - Template, beschreibt aber die abzusehende künftige Entwicklung.

Eigentliches Netzwerk

```
#[NETWORK TEMPLATE]#

inetnum: 195.177.100.64/224
netname: DOMAIN-NET
descr:   Irgendeine Firma GmbH
country: DE
admin-c: JB123
tech-c:  RS233-RIPE
tech-c:  MR124-RIPE
rev-srv: ns1.idnet.de
rev-srv: ns2.idnet.de
status:  ASSIGNED PA
notify:  registry@idinter.net
mnt-by:  IDNET-MNT
changed: h.hippenstiel@system-design.com DATUM
source:  RIPE
```

Mit diesem Template wird das Netz in der RIPE-Datenbank dokumentiert. Die Daten ergeben sich aus den vorangegangenen Templates.

Der `netname` kann beliebig gebildet werden. Üblicherweise wird bei vorhandenem Domänennamen dieser verwendet, ansonsten der Firmenname des Kunden, beides ergänzt mit `-NET`. Dieser Name darf bereits existieren, da die eindeutige Zuordnung letzendlich über die IP-Adressen erfolgt.

Die `descr` wiederholt lediglich die Firmenbeschreibung aus dem User Template.

Für `admin-c` wird das bereits erwähnte NIC-Handle (`nic-hdl`) verwendet, in diesem Beispiel also `JB123`.

Beachtenswert ist der `status`: Providerabhängige IP-Adressen (PA) werden providerunabhängigen (PI) bevorzugt, denn das Routing eines (großen) Adreßbereiches wird bei PA dem Provider überlassen, wodurch der Aufwand für RIPE geringer ist.

Das DATUM-Feld wird automatisch ergänzt, muß aber anschließend aus der Bestätigung von RIPE extrahiert werden, da diese Information sonst verlorengeht.

Zum **Löschen** wird ähnlich wie bei einem Personen-Template vorgegangen. In der entsprechenden Mail ist das Paßwort in die erste Zeile zu setzen, in die zweite Zeile das Schlüsselwort `delete:` mit einer Begründung als Kommentar dahinter. Dahinter schließt sich das vollständige Network Template an.

Ende der Templates

```
#[TEMPLATES END]#
```

kennzeichnet das Ende der Templates.

Die vervollständigten Templates sind als Beispiel nochmal in Anhang B abgedruckt.

3.9.3 Evaluierung einer Automatisierung

Im Folgenden soll untersucht werden, inwieweit sich diese Dokumentation bei RIPE durch geeignete Erweiterung der zu entwickelnden DNS-Verwaltung automatisieren läßt.

Aus dem vorherigen Abschnitt ist ersichtlich, daß die Dokumentation zwei RIPE-Datenbank-Einträge zum Ergebnis hat. Diese werden durch die Daten aus mehreren Templates gebildet.

Schematisierter Ablaufplan

Zur besseren Übersicht werden die oben beschriebenen Schritte in einem Ablaufplan mit den Sonderfällen noch einmal zusammengefaßt:

1. Bekannte Daten eintragen
2. Ist der Kunde dem RIPE bekannt?
   (a) Er hat eine dem RIPE bekannte Domäne.
   Somit kann man die Daten daraus erfahren, höchstens die E-Mail Adresse muß ergänzt werden.
   (b) Man versucht, den Namen bei RIPE abzufragen.
   Auch dann muß man evtl. die E-Mail Adresse ergänzen.
   (c) Es muß eine Person angelegt werden.
3. Das RIPE-141 Dokument ist vollständig und wird an die Hostmaster gesandt. Ausnahme: Der IP-Bereich darf selber dokumentiert werden.
4. Der Network Template Teil (inetnum-Objekt) wird vervollständigt und dann gegenüber der RIPE-Datenbank dokumentiert.

Aufgaben

Die einzelnen Aufgaben für eine Automatisierung ergeben sich wie folgt:

1. **Erfassung der Daten**
   Eine Anzahl der benötigten Daten sind der DNS-Verwaltung bereits bekannt. Bei anderen sind die entsprechenden Tabellen um geeignete Attribute zu erweitern. Bzgl. des Personenobjektes gestaltet sich dies recht trivial.
2. **Generierung der Templates**
   Dies ist der eigentlich interessante Teil der Automatisierung, dem aufgrund seiner Komplexität weiter unten eine eigene, ausführliche Besprechung gewidmet ist.
3. **Kommunikation mit RIPE**
   Da zum Versenden der Templates (und auch für die Antwort von RIPE) E-Mail verwendet wird, sind lediglich die ausgefüllten Templates in der richtigen Reihenfolge zu einem Dokument zusammenzufassen. Dieses wird dann dem lokalen Mailer zur Auslieferung übergeben.

Zusätzliche Datenstrukturen

Es wird eine zusätzliche Tabelle für die Firmendaten benötigt. Eine solche Tabelle sollte die Beschreibung des Firmenzwecks, das Land, Verweise auf die administrativen sowie technischen Kontakte (diese dürften für alle Netze und Zonen einer Firma gleich sein), und ähnliche Daten enthalten.

Außerdem ist die Adreßverwaltung (s. 3.7) um einen Verweis (Fremdschlüssel) auf diese Firmentabelle sowie ein Beschreibungsfeld zu erweitern.

Generierung der Templates

Der eigentliche Ablauf der Dokumentation wurde ja bereits ausführlich dargestellt. Nur die Gewinnung der einzelnen Belegungen der Template-Attribute ist noch ungeklärt. Sie soll hier anhand des oben erstellten Ablaufplanes erörtert werden.

1. **Overview of Organization**-Template Dieser Text wird der neu zu erstellenden Datenstruktur entnommen.

2. **Requester**- und **User**-Template
   Für die zusätzlich vom RIPE benötigten Daten wird die Benutzerverwaltung mit den Einträgen für NIC-Handle, Adresse, etc., ergänzt. Bei Neueintragung eines Benutzer kann dann entweder gleich sein NIC-Handle angegeben (falls er schon in der RIPE-Datenbank eingetragen ist) oder ein entsprechendes Template anhand der einzugebenden Daten generiert und an RIPE versandt werden, um ein solches Handle zu beantragen.

3. **Current Address Space Usage**-Template
   Die Netzwerkmaske ist (beispielsweise anhand der Tabelle 3.5) zu berechnen.

4. **Request Overview**-Template
   Die meisten Einträge sind fest vorgegeben, nur die beantragte Netzgröße sowie evtl. Angaben zu zurückzugebenden Netzen sind einzutragen.

5. **Addressing Plan**-Template
   Vom Aufbau her ist dieses Template dem *Current Address Space Usage*-Template gleich. Aufgrund der üblichen Vorgehensweise, die Größe auch gleich bei den Feldern für *Immediate*, *1-yr* und *2-yr* anzugeben, wird dieser Wert einfach aus dem *Request Overview*-Template übernommen.

6. **Network**-Template
   Anhand der Daten der bereits existierenden sowie der neu zu schaffenden Tabelle ist das letzte Template auszufüllen.

Als kleiner Hinweis für eine Implementierung sei noch dies angemerkt: Bei RIPE existiert eine spezielle E-Mail Adresse (`test-dbm@ripe.net`), an die testweise Einträge verschickt werden und das Ergebnis in einer speziell dafür angelegten Testdatenbank geprüft werden können.

Erwartetes Ergebnis

Neben der Zeitersparnis durch die automatisierte Erstellung der Templates werden sich auch die anderen Vorteile der rechnergestützten Verwaltung des DNS bemerkbar machen. Als wichtigste sind dabei die zu erwartende Verminderung der Fehler und der vereinfachte, weil in das Gesamtumfeld integrierte Arbeitsablauf zu nennen.

Auch müssen die Templates für die Beantragung, die das RIPE bei existierendem Assignment Window nur stichprobenartig verlangt, nicht mehr für alle Teilnetze angelegt werden. Vielmehr lassen sich die wenigen, die tatsächlich angefordert werden, direkt aus der Datenbank erzeugen.

Kapitel 4

Implementierung

In diesen Anmerkungen wird auf die Aufgaben und Besonderheiten, die während Implementierung zu lösen waren, eingegangen.

Aufgrund seiner Verbreitung wurde als relationales Datenbanksystem *Oracle*[1] gewählt, für die Realisierung der Verwaltungswerkzeuge wird *Perl* bzw. *Oraperl* verwendet.

In Oracle wird für den Zugriff auf die Datenbank SQL, die *Simple Query Language*, eingesetzt.

Perl[2] ist eine interpretierte Sprache mit mächtigen Befehlen zur Verarbeitung von Strings (u.a. über reguläre Ausdrücke), Skalaren und Feldern. Sie wurde von Larry Wall [2] entwickelt, um die Vorzüge der Shell- und C-Programmierung zu vereinen. Darüberhinaus sind die Funktionalitäten von `sed` und `awk` in Perl vorhanden.

Oraperl[3] von Kevin Stock ist eine Anzahl von Patches für die Version 4 von Perl, welche Zugriff auf Oracle-Datenbanken erlauben.

Die Kurzbeschreibung der einzelnen Programmodule ist thematisch gegliedert: 4.1 definiert die Tabellen und Relationen, 4.2 faßt die wichtigsten Routinen zur DNS-Verwaltung zusammen 4.2.2 erläutert das Einlesen und Generieren von Zonendateien und 4.2.3 beschreibt einige Funktionen für die Verwaltungsoberfläche.

4.1 Datenbank

4.1.1 Tabellen und Relationen

Die Attribute einzelnen Tabellen sind, zusammen mit einer kurzen Beschreibung, im Anhang C zu finden. Ein kurzer Überblick findet sich in Tabelle 4.1.

Die Tabellen für die Datenbank werden mit dem SQL-Skript `dns.sql`, die zugehörigen Indexe durch `idx.sql` erzeugt; das Löschen erfolgt mittels `drop.sql`.

Andere Resource-Record Klassen außer `IN` sind (nach 2.2.3) nicht relevant, deshalb wird diese standardmäßig verwendet. In der vorliegenden Arbeit werden TTLs nur für die Zone selbst verwaltet.

Die verwendeten *Constraints* sind `NOT NULL`, `UNIQUE` sowie `PRIMARY KEY` und `FOREIGN KEY`. Wird ein Constraint verletzt, meldet das DBMS einen Fehler. `NOT NULL` bedeutet, daß

[1] `http://www.oracle.com`
[2] `http://perl.com`
[3] eine Einführung findet sich unter `http://osprey8.npac.syr.edu:1963/oraperl.html`

	Name der Tabelle	enthaltene Daten
1	dnszone	Zonen
2	dnsnameserver	Nameserver
3	dnshost	Hosts
4	dnsipaddr	IP-Adressen
5	dnsalias	Alias
6	dnswks	Well Known Services
7	dnsmailx	Mail Exchanger
8	dnsforwarder	Forwarder
9	dnssecondary	Secondary Nameserver
10	dnspop	Point of Presence
11	dnsnetwork	Class-C Netze eines POPs
12	dnszoneblock	IP-Adressblöcke einer Zone
13	dnsuser	Benutzer
14	dnsuserzone	einem Benutzer zugeordnete Zonen

Tabelle 4.1: Tabellen in der Datenbank

ein Attribut auf jeden Fall einen Wert erhalten muß. `UNIQUE` erfordert die Eineindeutigkeit des Wertes innerhalb einer Spalte der Tabelle. Die Festlegung eines Attributes als Primärschlüssel durch `PRIMARY KEY` impliziert `NOT NULL` und `UNIQUE`.

4.1.2 Optimierungen

automatische Wertzuweisung an Primärschlüssel

Primärschlüssel müssen eindeutige, möglichst numerische Werte haben. In DBMS existieren Funktionen, mit denen die Anwendung von der Verwaltung solcher Zähler befreit wird: *Sequenzen* und *Trigger*:

Eine **Sequenz** ist ein Automatismus, der eine Zahl ab einem beliebigen Startwert um einen frei wählbaren Wert inkrementiert.

```
CREATE SEQUENCE [sequ_name] INCREMENT BY [value] START WITH [value]
```

Ein **Trigger** wird entweder vor (`BEFORE`) oder nach (`AFTER`) dem Zugriff (`INSERT`, `UPDATE` oder `DELETE`) auf eine Tabelle aufgerufen.

Hiermit läßt sich beispielsweise der Sequenzwert in ein Feld schreiben:

```
CREATE TRIGGER [trig_name] BEFORE INSERT ON [table_name] FOR EACH ROW
DECLARE
 xSEQ NUMBER;
BEGIN
 SELECT [sequ_name].NEXTVAL INTO xSEQ FROM DUAL;
 :NEW.[field_name] := xSEQ;
END;
/
```

Jedes Mal, wenn in der Tabelle `[table_name]` ein Record eingefügt wird, wird der Trigger `[trig_name]` aufgerufen und das Feld `[field_name]` aus Tabelle `[table_name]` mit der nächsten Nummer aus der Sequence `[sequ_name]` gefüllt.

Views

Eine View ist eine Tabelle, die bei Zugriff darauf dynamisch durch Ausführung der zugeordneten SQL-Abfrage erzeugt wird. Vorteil dieser Vorgehensweise ist die Vereinfachung von komplexen Verknüpfungen über mehrere Tabellen hinweg. Ein anderes Anwendungsgebiet liegt in der Beschränkung von Zugriffen auf bestimmte Teilbereiche von Tabellen, indem nur eine Untermenge der Attribute in der View zusammengefaßt werden. Die gesamte Tabelle dagegen wird nur einigen wenigen Benutzern zugänglich gemacht.

Indexe

Indexe erlauben einem DBMS auf Daten schneller zuzugreifen, indem die Inhalte bestimmter Spalten in einer optimierten internen Datenstruktur gehalten werden.

4.2 Verwaltung der DNS-Daten

Die programmtechnische Verwaltung der Datenbank-Inhalte ist auf drei Ebenen aufgeteilt: allgemeine Zugriffsfunktionen, Verarbeitung von Zonendateien sowie Benutzeroberfläche. Gemäß dieser Einteilung werden die verschiedenen Skripte nachfolgend erläutert.

4.2.1 Bibliotheken

Der Zugriff auf die Datenbank soll hauptsächlich über Funktionen erfolgen, die den Anforderungen des konzeptionellen Teils genügen. Werden Hosts, Zonen, Nameserver etc. als Parameter übergeben, ist ihr Primärschlüssel zu verwenden, um ihre Eindeutigkeit zu gewährleisten.

Utility - `util.pl`

Hier sind einige allgemeinere Funktionen, die von vielen Modulen benötigt werden, zusammengestellt. Darunter befinden sich die Erstellung der Seriennummer und des aktuellen Datums, Überprüfungen von Adressen auf Gültigkeit und die Verwaltung der Cookies.

Datenbank-Schnittstelle - `ora-lib.pl`

`ora-lib.pl` dient dem allgemeinen Zugriff auf eine Oracle-Datenbank über die Funktionen, die `oraperl` zur Verfügung stellt. Datenbankspezifische Einstellungen wie das Datenformat, Zugangscodes und Umgebungs-Variablen werden hier vorgenommen. Auch sind grundlegende Datenbankfunktionen wie An- und Abmelden sowie Öffnen bzw. Schließen der Datenbank, Cursorverwaltung und das Absetzen von SQL-Befehlen enthalten. Diese SQL-Befehle werden, je nach gewähltem Log-Level, protokolliert (s. 3.5.3). Die Routine für die Fehlerbehandlung schreibt ebenfalls eine Meldung in eine eigene Datei.

Datenbank-Funktionen für DNS-Verwaltung - `dnsora-lib.pl`

Diese Bibliothek ist der eigentliche Mittelpunkt der DNS-Verwaltung. Sie liefert die Funktionen *Einfügen*, *Ändern*, *Löschen* und *Prüfen* für die Datenbank-Objekte (s. 3.4.1) Zone, Nameserver, Host, Benutzer, POP und Netzwerk. Außerdem sind Funktionen zum Finden von Name bzw. ID eines Hosts sowie für die Verarbeitung von Host- und Domain-Namen (Hostnamen extrahieren, FQDN erstellen, etc.) vorhanden. Die Suche nach freien Adreßbereichen (s. 3.7) ist ebenfalls hier zu finden.

4.2.2 Zonendateien

Die grundsätzliche Vorgehensweise beim Einlesen und Generieren von Zonendateien wurde bereits in 3.6 ausführlich behandelt. Die Skripte dazu sind recht kurz, da sie auf den Funktionen der oben aufgeführten Bibliotheken stützen können.

Import in Datenbank - `dns2ora.pl`

Im ersten Durchlauf werden der SOA-Record[4] und die Hosts mit ihren IP-Adressen in die Datenbank geschrieben. Neben den Prüfungen der Daten durch die Bibliotheksroutinen wird beispielsweise hier der Punkt in der Mailadresse durch einen Klammeraffen ersetzt.

Kommentare und der Eintrag für den Localhost werden ignoriert.

Import, 2. Durchgang - `dns2ora2.pl`

Zu Beginn wird der SOA-Record erneut gelesen, um anhand des Zonen-Namens den zugehörigen Primärschlüssel[5] zu erfahren. Daneben wird der Primärschlüssel des Hosts, auf dem der primäre Nameserver der jeweiligen Zone läuft, festgestellt, um den Primary von den sekundären Nameservern unterscheiden zu können.

Resource Records wie Nameserver, Mail Exchanger und Aliase werden anschließend importiert.

Zum Schluß erfolgt in der Zonentabelle ein Verweis auf den oben festgestellten primären Nameserver, weil für diesen erst jetzt ein Nameserver-Objekt in der Datenbank besteht, welches zur Verknüpfung herangezogen werden kann.

IP-Adressblöcke zuordnen - `ipfill.pl`

Die IP-Adressen aller vorhandenen Class-C Netze werden auf Zwischenräume zwischen zwei Adressen der gleichen Zone untersucht, um diese ebenfalls der Zone zuzuordnen.

In einer zweiten Schleife wird die minimale Größe der jeweiligen Adreßblöcke determiniert und diese Blöcke den entsprechenden Zonen zugeteilt.

Generierung - `ora2dns.pl`

In einer Schleife werden für alle beim Aufruf von `ora2dns.pl` angegebenen Nameserver die Konfigurations- und Zonendateien generiert.

Die Namenszuordnung der für die einzelnen Dateien ist Tabelle 4.2 zu entnehmen.

[4] bis auf den Verweis auf den primären Nameserver, der erst im zweiten Durchlauf gesetzt wird

[5] Dieser wird für die weitere Bearbeitung verwendet, siehe einführende Bemerkungen (4.2.1).

Dateinamen	Inhalt	Beispiel
db.DOMAIN	Hostnamen - IP-Adressen	`db.haitec`
db.ADDR	IP-Adressen - Hostnamen	`db.194.55.3`
db.cache	Root Server	*konstant*
db.127.0.0	Loopback für localhost	*konstant*

Tabelle 4.2: Namen der Zonendateien

Pro Nameserver werden in einer weiteren Schleife die einzelnen Zonen, die dieser Nameserver als Primary bedient, verarbeitet. Für jede dieser Zone erfolgen als erstes die Einträge der Verweise in der Datei (`named.boot`). Anschließend werden die Dateien für das Forward und Reverse Mapping erstellt.

4.2.3 WWW-Benutzeroberfläche

Aufteilung der Perl-Skripten

Es existieren entsprechend den Objektklassen (s. 3.4.1) je zwei Skripten, eines für das Neuanlegen, Bearbeiten und Löschen, das andere zur Auswahl eines Objektes aus dem vorhanden Datenbestand. Die Namen sind entsprechend gewählt, so zum Beispiel `nameserver.pl` bzw. `nameserver-select.pl`

Zugangskontrolle

Nach erfolgreicher Authentifizierung wird ein *Cookie* [draft-ietf-http-state-mgmt-05] zum Client des Benutzers geschickt.

Ein Cookie (Kurzform für *Persistent Client State HTTP Cookie*) wird im Header der Client-Server-Kommunikation übertragen. Er enthält beliebige Informationen, die vom Server gesetzt werden. Die Information wird wieder zu diesem Server zurückgeschickt, wenn der Client auf ein Verzeichnis seiner Domäne zugreift. Dies geschieht solange, bis der Cookie *abgelaufen* (engl. *expired*) ist.

Zum Verwaltung der Cookies wird die Bibliothek *cookielib*[6] verwendet.

Da HTTP kein verbindungsorientiertes Protokoll (also *stateless*) ist, muß bei jeder Aktion erneut festgestellt werden, ob der Anfragende bereits authentifiziert ist. Dazu wird von jedem Skript als erstes der Authentifizierungs-Cookie abgefragt. Sollte er nicht vorhanden sein, wird die Ausführung abgebrochen.

Beim Abmelden von der DNS-Verwaltung (siehe *Logout*, 5.2.7) oder spätestens beim Beenden des Browsers wird der Cookie wieder entfernt.

Parameterübergabe zwischen Skripten

Wie bereits erwähnt, ist wegen der Funktionsweise des HyperText Transfer Protocols beim Aufruf eines Skriptes keine Zustandsinformation verfügbar. Zumindest die aktuell gewählte Zone (s. 5.2) wird aber benötigt.

[6] `http://www.worldwidemart.com/scripts/cookielib.shtml`

Hier bietet sich die gleiche Vorgehensweise wie für die Authentifizierung an: In einem weiteren Cookie ist die momentan bearbeitete Zone gespeichert, das die Skripten `zone.pl` bzw. `zone-select.pl` setzen, wenn eine Zone ausgewählt, geändert oder neu erstellt wird.

Kapitel 5

Anwendung

Die Anwendung der beispielhaft entwickelten Schnittstelle zum Anwender wird in diesem Kapitel erläutert.

Sie ermöglicht eine einheitliche Vorgehensweise durch vorgegebene Abläufe, achtet aber gleichzeitig darauf, den Benutzer nicht unnötig einzuschränken.

Die Verwendung eines relationalen Modells erlaubt, bei der Erstellung von Verknüpfungen zwischen Objekten wie Zonen, Hosts oder Nameservern Auswahlmöglichkeiten aus dem vorhandenem Datenbestand anzubieten.

Aus dieser Erleichterung der Bewältigung von Standardaufgaben eines DNS-Verwalters, hier speziell bei Internet Service Providern, resultiert nicht nur eine nicht zu unterschätzende Zeitersparnis, sondern auch die Vermeidung vieler unnötiger, weil eigentlich trivialer, in ihrer Wirkung aber teils verheerender Fehler.

5.1 Konfiguration

Bevor die Arbeit mit der DNS-Verwaltung begonnen werden kann, sind einige Vorbereitungen zu treffen. Die Datenbank benötigt Oracle ab Version 7, für das WWW-Interface läßt sich ein beliebiger WWW-Server einsetzen. Auch ist ein gewisser administrativer Aufwand für die Verteilung der Zonendateien erforderlich.

5.1.1 WWW-Server

Für die Installation des Verwaltungswerkzeuges werden einige Unterverzeichnisse benötigt, die aus Tabelle 5.1 ersichtlich sind:

Verzeichnis unter -serverroot	Inhalt
`cgi-bin/dns/`	Perl-Skripten
`cgi-bin/dns/log/`	Logdateien
`html-data/dns/`	HTML-Seiten

Tabelle 5.1: Verzeichnisstruktur

5.1.2 Oracle-Datenbank

Es empfiehlt sich, einen eigenen Oracle-Benutzer sowie einen eigenen Tablespace für die Tabellen (siehe 4.1.1) anzulegen.

Dieser Benutzer und sein Paßwort ist in der Datei `ora-lib.pl` einzutragen und bei Änderungen derselben entsprechend anzupassen.

Sollte direkt, also über `sqlplus`, auf die Datenbank zugegriffen werden, sind erst zwei Oracle-Umgebungsvariablen gemäß Tabelle 5.2 zu belegen.

Umgebungsvariable	Belegung
`ORACLE_SID`	Name der Oracle-Instanz
`ORACLE_HOME`	Pfad zum Homeverzeichnis der Oracle-Installation

Tabelle 5.2: Oracle-Umgebungsvariablen

5.1.3 Verteilung der Zonendateien

Die Verteilung der neu erzeugten Dateien an die Rechner, auf denen die zu beschickenden Nameserver laufen, geschieht über FTP, die Steuerung des Nameserver über remsh (bzgl. der Einzelheiten sei auf 3.8.1 verwiesen).

Dazu ist auf diesen Rechnern eine Benutzerkennung `dnsadmin` einzurichten. In der zugehörigen `.rhosts` - Datei müssen der Rechnername bzw. die IP-Adresse der Maschine, auf welcher die DNS-Verwaltung eingerichtet ist, sowie die Kennung eingetragen werden, unter der sie läuft.

5.2 WWW - Interface

Dieses Interface soll als Prototyp zeigen, wie eine sichere Verwaltung des DNS über Internet aussehen kann. Es erhebt keinesfalls Anspruch auf Vollständigkeit.

Verschiedene Aspekte, wie beispielsweise die Verwaltung von `CNAME`s, sind nicht implementiert, aber ohne größeren Aufwand ebenso realisierbar wie die bereits vorhandenen Teile der Benutzeroberfläche.

Kursiv beschriftete Felder in den HTML-Formularen sind optional auszufüllen.

Bei der Verwaltung von DNS-Daten ist, bis auf wenige Ausnahmen (beispielsweise *Nameserver*, s. 5.2.4), vorher die entsprechende Zone zu wählen, um die Menge der Daten, die man bearbeitet, zu reduzieren. Somit werden etwa beim Punkt *Hosts bearbeiten* (s. 5.2.3) nur die Hosts der aktuellen Zone aufgelistet.

Entsprechend wird dann entweder der vollständige Domain-Name oder, bei entsprechendem Kontext, nur der Hostname angezeigt.

5.2.1 Login

Meldet man sich zum ersten Mal bei der DNS-Verwaltung an, geschieht das über den Benutzer `admin` mit dem Paßwort `admin`. Dieses Paßwort sollte umgehend geändert werden, um nicht eine Sicherheitslücke zu schaffen.

Es wird ein Cookie[1] gesetzt, welcher der DNS-Verwaltung ermöglicht, den entsprechenden Benutzer wiederzuerkennen. Aus diesem Grund muß der verwendete Browser so eingestellt sein, daß er Cookies akzeptiert.

Nun sind die Menüpunkte im linken Bereich verwendbar, die sonst vor einer erfolgreichen Authentifizierung den Zugriff verweigern und zum Login verweisen. Bei einigen Funktionen ist erst die entsprechende Zone, auf die sich eine Aktion beziehen soll, auszuwählen.

5.2.2 Zonen

Die Belegungen für den `SOA`-Record, das sind Zonenname, E-Mail des Administrators, Name der Zonendatei sowie die TTL-Werte, sind hier anzugeben.

Extern bezeichnet Zonen, die man nicht selbst verwaltet, die aber Dienste wie etwa Secondary Nameservice zur Verfügung stellen und deshalb bekannt sein müssen.

Um eine Zone zu delegieren, wählt man aus den vorhandenen Zonen die gewünschte Vaterzone aus. Hierbei muß ein Zonenverwalter (s. 3.8.2) für beide Zonen berechtigt sein, da ja bei beiden entsprechende Einstellungen (Nameserver, Secondaries) vorzunehmen sind.

Neu

Ein primärer Nameserver innerhalb dieser Zone kann erst eingetragen werden, nachdem die Zone erstellt und der entsprechende Host angelegt wurde, ansonsten darf natürlich einer der bereits existierenden ausgewählt werden.

Auswählen, bearbeiten

Zuerst wird eine Auswahlliste der bereits existierenden Zonen angeboten, wobei delegierte Zonen sich hinter ihren Vaterzonen befinden. Darunter sind die externen Zonen aufgelistet.

Durch Auswahl einer der Nameserver gelangt man in die bereits bekannte Eingabemaske, diesmal mit den Optionen *Ändern* und *Löschen*.

Löschen

Eine Vaterzone läßt sich erst löschen, wenn sämtliche Delegierungen, die sie betreffen, aufgehoben wurden. Es darf also keine Zone mehr auf die zu Löschende verweisen.

Beim Löschen einer Zone werden auch weitere Informationen, vor allem die zugehörigen Hosts, aus der Datenbank entfernt!

5.2.3 Hosts

Die obligatorischen Einträge für einen Host umfassen seinen Namen sowie die IP-Adresse. Zusätzlich lassen sich Informationen über die Responsible Person, die CPU und Betriebssystem sowie ein beliebiger Kommentar angeben.

Bei den IP-Adressen erscheint eine Liste der noch für diese Zone verfügbaren Adressen. Diese sind nicht bindend, es kann durch Auswahl von *direkt* eine andere Adresse eingetragen werden.

[1] Die aktuell ausgewählte Zone wird ebenfalls über einen Cookie zwischengespeichert; s. 4.2.3

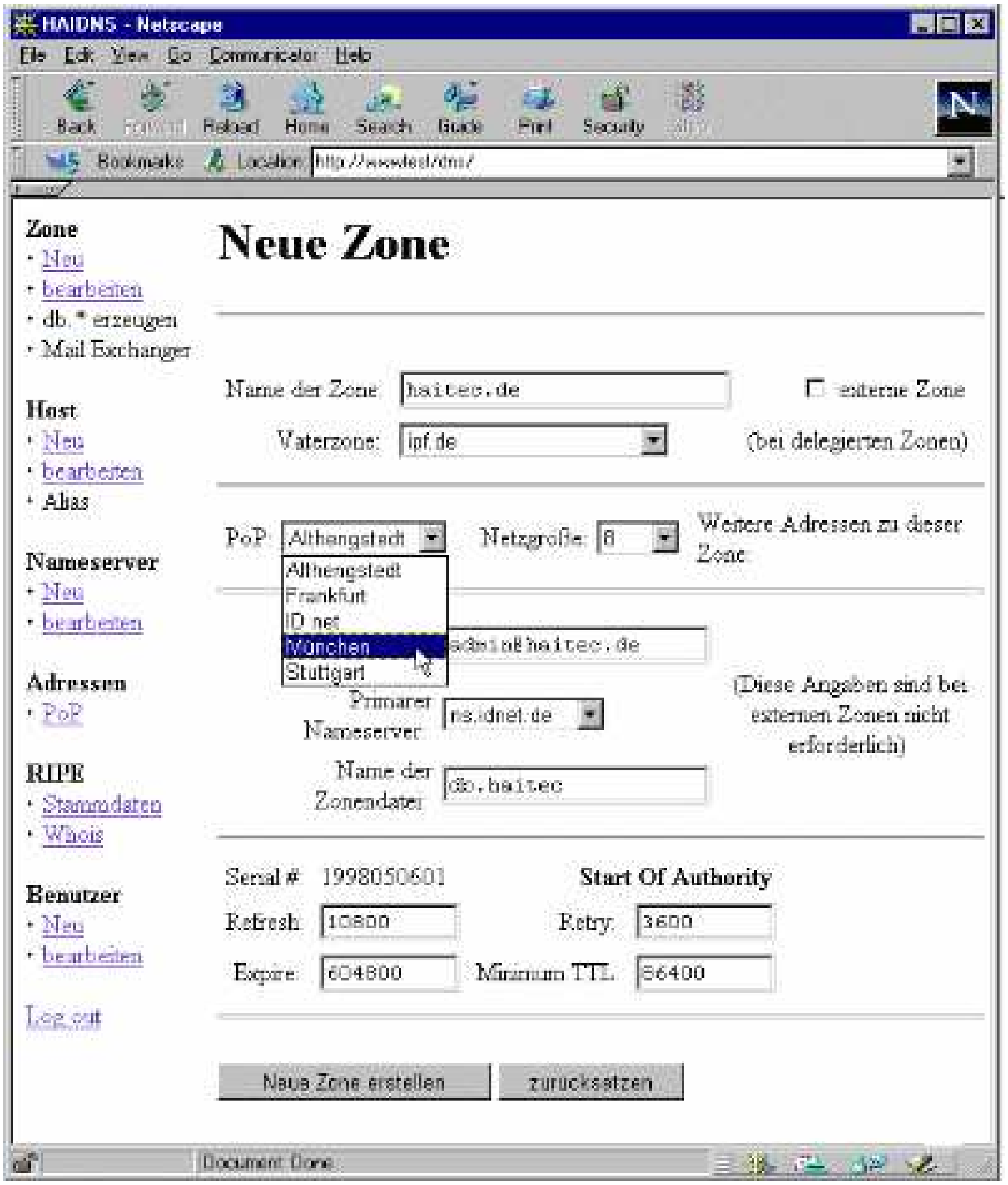

Abbildung 5.1: WWW-Benutzerschnittstelle mit dem Formular für neue Zonen

Neu

Zur aktuell gewählten Zone soll ein Host hinzugefügt werden.

Auswählen, bearbeiten

Aus einer Liste mit den Hostnamen sowie IP-Adresse und eventuellen Aliasen wird hier ein Host zur Bearbeitung ausgewählt. Es werden nur die Hosts der aktuellen Zone angezeigt, um ihre Zahl in Grenzen zu halten.

Löschen

Hosts, auf denen noch ein Nameserver eingetragen ist, können nicht gelöscht werden. Ein solcher Nameserver ist vorher auf auf einen anderen Host umzuziehen.

Beim Löschen eines Hosts werden ebenso Aliase, die auf ihn verweisen, entfernt, die IP-Adresse wird als für diese Zone wieder verfügbar gekennzeichnet.

5.2.4 Nameserver

Wichtigste Bestandteile sind hier die Festlegung des Hosts, auf dem der Nameserver läuft, und der Zonen, die dieser als Secondary spiegeln soll. Außerdem sind Angaben über die DNS-Zonendateien vorgesehen, nämlich das Verzeichnis auf dem Zielhost, in dem sie abzulegen sind, sowie die Dateinamen für das Loopback Interface und den Root-Cache. Letztere wird unter dem bei *Datei* angegebenen Namen von dem bei *Host* eingetragenen FTP-Server aus dem entsprechenden *Verzeichnis* geholt.

Neu

Es werden Standardwerte für die Einträge zu den DNS-Zonendateien und den Root-Cache Servern vorgeschlagen, die normalerweise übernommen werden sollten.

Je Host ist nur ein Nameserver sinnvoll, da dieser ja Dienste für eine beliebige Anzahl von Zonen leisten kann. Deshalb werden bei der Auswahl des Hosts, auf dem der Nameserver laufen soll, nur noch jene angezeigt, die nicht schon durch einen Nameserver belegt sind.

Auswählen, bearbeiten

In der Auswahl sind neben den Nameservern auch die Zonen aufgelistet, für die der jeweilige Server Primary oder Secondary ist.

Löschen

Das Löschen eines Nameservers ist nur erlaubt, wenn er für keine Zone mehr als Primary eingetragen ist. Sämtliche Verweise auf diesen Nameserver als Secondary werden dagegen ohne weitere Rückfrage gelöscht, demnach ist diese Funktion mit Vorsicht anzuwenden.

5.2.5 Adressen

Die Class-C Netzwerke, welche ein Provider vom RIPE erhält, werden hier verwaltet.

POP

Für die vereinfachte Adreßverwaltung müssen dem System zuerst die verfügbaren Class-C Netze bekannt gemacht werden, indem man sie den entsprechenden POPs des Providers zuordnet.

Dazu existiert ein Formular, das die bestehenden POPs mit den zugeordneten Class-C Netzen auflistet und die Möglichkeit bietet, neue POPs einzutragen oder bestehenden neue Class-C Netze zuzuweisen.

Ein Löschen ist ebenso möglich, wobei ein Standort erst entfernt werden darf, wenn ihm keine Class-C Netze mehr gehören.

5.2.6 Benutzer

Die einzelnen Benutzer, die hier mit vollständigen Namen einzutragen sind, müssen sich anhand ihres Logins unterscheiden. Zur Authentifizierung bei der Verwaltungsoberfläche ist hier ein Paßwort[2] einzutragen sowie die Rolle des Benutzers festzulegen. Handelt es sich um einen Zonenverwalter, müssen ihm die Zonen, die er bearbeiten darf, zugewiesen werden.

Neu

Beim Neuanlegen eines Benutzers wird aus Sicherheitsgründen *Responsible Person* als Berechtigungsstufe voreingestellt. In der Auswahlliste der Zonen, die ein Zonenverwalter bearbeiten darf, sind keine Einträge selektiert.

Auswählen, bearbeiten

Die Liste, aus welcher man den zu bearbeitenden Benutzer auswählt, enthält das Login, den vollständigen Namen, seine Berechtigungsstufe sowie das NIC-Handle, soweit vorhanden.

Löschen

Der vordefinierte Benutzer `admin` kann nicht entfernt werden, da sonst der Zugriff auf die Verwaltungsoberfläche nicht mehr möglich wäre.

5.2.7 Logout

Nach Erledigung der Arbeiten mit der DNS-Verwaltung sollte man sich aus Sicherheitsgründen beim System abmelden. Dies wird durch den *Logout* erledigt.

5.2.8 Hinweis

Sperren von Bereichen

Die (vorläufige) Belegung einer Zone und des von ihr belegten IP-Bereiches läßt sich dazu verwenden, einen bestimmten Bereich zu sperren. Dies ist sehr einfach durch Anlegen einer entsprechenden *externen* Zone umsetzen. Dies ist beispielsweise nützlich, wenn Transfernetze Verwendung finden.

5.3 Migration eines vorhandenen Datenbestandes

Will ein Internet Service Provider seine Zonenverwaltung künftig mit diesem Werkzeug durchführen und soll deswegen der vorhandene Datenbestand in die Datenbank importiert werden, so sind einige Vorarbeiten zu leisten.

[2]Das Paßwort wird nicht im Klartext dargestellt, sondern jeder Buchstabe durch '`*`' repräsentiert.

5.3.1 Häufige Fehler in bestehenden Zonenateien

An erster Stelle steht eine Bereinigung der bereits existierenden Zonendaten, damit auf einem korrekten, funktionierenden Datenbestand aufgebaut werden kann. Dazu werden die Zonendateien probehalber eingelesen. Dies kann entweder in einen eigenen Tablespace für solche Testzwecke geschehen, oder man löscht die Tabellen, um sie vor dem endgültigen Einlesen nochmals neu zu generieren.

Im Folgenden werden die Fehler beschrieben, die bei der Anwendung des Übertragungswerkzeuges `dns2ora.pl` auf den realen Datenbestand eines Providers zu finden waren.

`CNAME` auf `CNAME`

Eine Unsitte ist die Verkettung von Aliasen. Aufgrund der zweifelhaften Nützlichkeit eines solchen Vorgehens[3] sind Aliase direkt auf „echte"Namen zu setzen.

`NS`-Eintrag auch für Primary

Häufig wird der `NS` Resource Record für den Primary Nameserver vergessen, da dieser bereits im `SOA`-Record eingetragen ist. Falls dies der Fall ist, muß zumindest in einer der Dateien der Primary Nameserver über einen `NS` Resource Record nachgetragen werden, damit er korrekt erkannt werden kann.

5.3.2 Eingabe von externen Hosts

Oft verweisen Resource Records auf externe Hosts, also solche, die nicht im eigenverwalteten Namensraum liegen. Zumindest ein externer Nameserver findet in den allermeisten Fällen Verwendung, aber auch fremde FTP- und Newsserver oder Mail Exchanger werden referenziert.

Für solche fremden Resourcen sind *externe* Zonen (s. 5.2.2) mit Hosts und eventuell Nameserver anzulegen.

5.3.3 Anlegen der POPs und zugehöriger Class-C Netze

Die Standorte des Providers sowie die zugehörenden Class-C Netze müssen der DNS-Verwaltung bereits vor dem Import mitgeteilt werden.

5.3.4 Einlesen von vorhandenen Zonendateien

Der bereinigten Zonendateien (s. 5.3.1) sind in zwei Schritten zu importieren.

1. Sämtliche einzulesenden Dateien sind *zuerst* über den Befehl `dns2ora db.*` zu bearbeiten. Hierbei werden die grundlegenden Informationen gesammelt (s. 3.6.1)

2. Die weiteren Resource Records erhält man im zweiten Teil durch `dns2ora2 db.*`.

Die Reihenfolge, in der die Zonendateien eingelesen werden, ist wichtig: Meist gibt es bei ISPs eine Master-Datei[4], die allgemeine Hosts wie etwa den primären Nameserver der

[3] siehe dazu die Erörterung in 3.5.1

[4] üblicherweise die ihrer eigenen Zone

Kundenzonen definiert. Da diese Informationen von den Resource Records in den anderen Zonen referenziert werden, werden sie gleich zu Anfang benötigt und müssen somit als erstes importiert werden.

5.3.5 Nachbehandlung der eingelesenen Daten

Nach dem Einlesen der Daten fehlen oft noch Zusatzinformationen, etwa weil diese nicht den Zonendateien enthalten sind.

Delegierungen

Delegierungen können von den Import-Werkzeugen nicht erkannt werden. Sie müssen daher nachgetragen werden.

Zuordnung von IP-Adressen und Adreßblöcken zu Zonen

Das Werkzeug `ipfind.pl` durchforstet den importierten Datenbestand nach möglichen Adreßblöcken und teilt diese sowie die enthaltenen IP-Adressen der entsprechenden Zone zu.

Dabei wird möglicherweise die Größe des Adreßblocks nicht korrekt erkannt und muß daher händisch nachgebessert werden.

Kapitel 6

Ausblick

6.1 Zusammenfassung

In dieser Arbeit wurden die managementrelevanten Aspekte des Domain Name Systems betrachtet. Darauf aufbauend wurden die Management-Objekte definiert und ein Konzept entwickelt, das sich besonders den Themen Fehlermanagement und Administration widmet. Facetten davon sind die Verwaltung der IP-Adressen, die Generierung von Zonendateien sowie die Netzdokumentation gegenüber RIPE.

Eine beispielhafte Implementierung einschließlich einer Benutzerschnittstelle zeigte schließlich die Anwendung der vorgestellten Konzepte, mit speziellem Augenmerk auf ihre Verwendung durch Internet Service Provider.

Aus der Beachtung der relevanten Standards bei der Erarbeitung resultiert eine hohe Plattformunabhängigkeit. Diese wiederum ermöglicht eine Übertragung der Ergebnisse auf Management-Umgebungen, wie beispielsweise Tivoli, sowie den Einsatz mit jeglicher standardkonformer Nameserver-Software.

6.2 Erweiterungen

Verschiedene denkbare Weiterentwicklungen, welche die DNS-Verwaltung sinnvoll ergänzen, sind nachfolgend zusammengestellt.

6.2.1 Funktionalität

IPv6 Kompatibilität

Die Evolution des IP-Protokolls, das sich momentan in seiner Version 4 im Einsatz befindet, wird auch Änderungen im DNS mit sich bringen[4].

Allerdings sind die entsprechenden RFCs zum jetzigen Stand (Mai 1998) eben erst erneut überarbeitet worden, somit wird sich sowohl die Implementierung der IP-Stacks als auch die Anpassung der Nameserver-Software noch hinziehen.

Zuordnung IP-Adresse - Host*name*

Es ist prinzipiell möglich, für die Zuordnung von mehreren Hostnamen zu einer IP-Adresse anstelle von Aliasen auch `A`-Records zu verwenden[1]. Dies ist über die bestehenden Datenstrukturen nicht abbildbar.

Adreßsuche

Die Auswahl eines freien, genügend großen Adreßbereiches beschränkt sich im Moment auf die Suche nach dem ersten passenden Block. Zur besseren Nutzung des zur Verfügung stehenden Adressenpools ist ein optimiertes Verfahren zu entwickeln, das bei der Vergabe die bereits entstandene Fragmentierung mit einbezieht.

Logging

Protokolliert werden zum jetzigen Stand lediglich die abgesetzen SQL-Befehle (s. 3.5.3). Eine übersichtlichere Lösung besteht in der Zusammenfassung einer ganzen Benutzeraktion, etwa *Zone ändern*, zu einem einzigen Eintrag, der natürlich die entsprechenden Parameter mit enthält. Desweiteren wäre abzuklären, inwieweit sich eine Übernahme der Protokolldaten in die Datenbank sinnvoll einsetzen läßt.

Resource Records

Die Anzahl der Resource Records verändert sich im Lauf der Zeit, da mit der technologischen Weiterentwicklung eine entsprechende Anpassung des Domain Name Systems einhergeht. Als stellvertretendes Beispiel sei hier der neu hinzugekommene `ISDN` Resource Record [RFC 1183] genannt, welcher für die Verwendung mit dem *Integrated Services Digital Network* eingeführt wurde. Gleichzeitig werden einige wenige als *deprecated*, also veraltet, gekennzeichnet.

Eine andere Erweiterung, deren sinnvolle Einsetzbarkeit zu prüfen wäre, ist folgende: Einträge wie die Well Known Services (`WKS`), beruhen auf Informationen, die sich eventuell aus entsprechenden Dateien (hier: `/etc/services`) oder anderen Quellen automatisch extrahieren lassen. Dazu sind allerdings Zugriffsmöglichkeiten auf die entsprechenden Hosts erforderlich.

6.2.2 Benutzeroberfläche

Die Implementierung des Prototypen für die benutzerfreundliche Verwaltung läßt natürlich Raum für Erweiterungen offen. Einige davon sind nachfolgend aufgeführt, wobei der Vervollständigung der Masken für die Datenpflege sicherlich Vorrang einzuräumen ist. Weitere Ergänzungsanforderungen dürften sich im Laufe der an die Diplomarbeit anschließenden Pilotphase ergeben.

Verwaltung

Im bestehenden Prototypen sind noch Editiermöglichkeiten für Aliase (`CNAME`), Host Information (`HINFO`) sowie Well Known Services (`WKS`) zu implementieren.

Je nach Bedarf könnte sich eine Suchfunktion, zum Beispiel nach Hostnamen, als nützlich erweisen.

---

[1] beim Reverse Lookup, etwa über `nslookup`, wird allerdings nur eine dieser Adressen zurückgeliefert

Die Protokollierung wird momentan im klassischen Unix-Stil in ASCII-Dateien abgelegt. Ein Auswertungswerkzeug sowie die Darstellung der Logdateien über die Benutzeroberfläche (mit Abfragemöglichkeit zur sortierten Ausgabe) stehen als nächster Schritt an.

Hilfetexte

In den Eingabemasken ist eine Verknüpfung der wichtigsten Begriffe (beispielsweise *Nameserver*) über einen Hyperlink zu einer Seite mit Hilfetext denkbar. Dort könnte eine Zusammenfassung der wichtigsten Konzepte zum jeweiligen Stichwort dem Anwender ein Blättern in der Literatur ersparen und somit seine Effizienz steigern.

Java-Oberfläche

Die Bedienungsfreundlichkeit eines Werkzeuges und somit die Akzeptanz beim Anwender ist zu einem großen Teil von seiner Komfortabilität abhängig. Geschwindigkeit, Übersichtlichkeit und Funktionalität werden auch von der verwendeten Schnittstelle beeinflußt. Das hier verwendete Common Gateway Interface (CGI) ist üblicherweise beschränkt auf die Ausgabe von HTML-Seiten, die eventuell noch JavaScript[2] verwenden.

Umfangreiche Möglichkeiten bietet hier eine (wesentlich aufwendigere) Realisierung in JAVA. Neben den gewohnten GUI-Elementen hat Java den Vorteil, das Programm auf dem Client, statt wie bei CGI auf dem Server, auszuführen und damit den Server zu entlasten. Als konkretes Beispiel sei die Auswahl von Hosts über eine ansprechende Visualisierung der Baumstruktur genannt.

6.2.3 RIPE und NIC

Die Machbarkeit einer IP-Adreßdokumentation beim RIPE wurde in dieser Arbeit bereits abgehandelt (3.9). Natürlich sind einige Erweiterungen möglich, so zum Beispiel die rechnergestützte Auswertung der Reply-Mails des RIPE oder eine Automatisierung der Personenverwaltung und der zugehörigen NIC Handles.

Eine ähnlich gelagerte Aufgabe ist die Beantragung und Verwaltung von Domain Namen beim deutschen DE-NIC[3] und bei internationalen NICs für Top Level Domains wie beispielsweise `.com` oder `.org`, die weltweit Verwendung finden. Da hier die grundsätzliche Vorgehensweise der beim RIPE ähnelt, sollte eine Implementierung auf den gewonnenen Erfahrungen aufbauen.

6.2.4 Sicherheit

Bei sensiblen Daten ist ein umfassendes Sicherheitskonzept unabdingbar. Die bereits vorhandenen Mechanismen wie paßwortgestützte Authentifizierung und die Vergabe von Berechtigungen über Rollen ist demzufolge auszubauen.

Ein wichtiger Gesichtspunkt ist die Vertraulichkeit der Daten, die sich über entsprechende Verschlüsselung der Datenübertragung gewährleisten läßt. Die gegenseitige Authentifizierung der Partner (also auch des Verwaltungsservers) kann durch Verwendung von Zertifikaten sicherer und einfacher gestaltet werden.

[2] Code für JavaScript wird in HTML eingebettet

[3] `http://www.nic.de`

Auch mit geringerem Aufwand ist der Schutz vor Mißbrauch ausbaubar: So stellt ein Verfallen von Paßwörtern nach einer gewissen Zeitspanne oder ein automatisches Abmelden von Benutzern, die länger inaktiv sind, einen weiteren Sicherheitsgewinn dar.

Anhang A

Abkürzungen

Abkürzungen in `Schreibmaschinenschrift` bezeichnen Resource Records.

`A` - Address

BIND - Berkeley Internet Name Daemon

CGI - Common Gateway Interface

`CNAME` - Alias

DBMS - Data Base Management System

DHCP - Dynamic Host Configuration Protocol

DMZ - DeMilitarized Zone

DNS - Domain Name System

GUI - Graphical User Interface

HACMP - High Availability Clustered Multi Processing

MIB - Management Information Base

MO - Managed Object

`MX` - Mail Exchanger

`NS` - Name Server

PERL - Practical Extraction and Report Language

`PTR` - Pointer

`RP` - Responsible Person

RIPE - Réseaux IP Européens

RFC - Request for Comments

RR - Resource Record

SNMP - Simple Network Management Protocol

`SOA` - Start Of Authority

TLD - Top Level Domain

`TXT` - Text

`WKS` - Well Known Services

WWW - World Wide Web

Anhang B

RIPE-Template

Ein ausgefülltes RIPE-141 Template könnte so aussehen:

```
#[OVERVIEW OF ORGANIZATION TEMPLATE]#

The company Irgendeine Firma GmbH is located in Posemuckel, Germany
and is specialized in manufacturing toys. Blafasel ...

#[REQUESTER TEMPLATE]#

name:         Martin Rastetter
organisation: Internet Services Deutschland GmbH
country:      DE
phone:        +49 7152 9733 0
fax-no:       +49 7152 9733 13
e-mail:       m.rastetter@idnet.de

#[USER TEMPLATE]#

name:         Johnny Benutzer
organisation: Irgendeine Firma GmbH
country:      DE
phone:        +49-8282-288
fax-no:       +49-8282-288
e-mail:       j.benutzer@domain.de

#[CURRENT ADDRESS SPACE USAGE TEMPLATE]#

                                         Addresses Used
Prefix         Subnet Mask       Size  Current  1-yr  2-yr   Description
193.1.193.0    255.255.255.192    64      28     34    60    Candy Cane Dept.
193.1.193.64   255.255.255.224    32      10     21    28    Toy Design
                                  96      38     55    88    Totals

#[REQUEST OVERVIEW TEMPLATE]#

request-size:        32
addresses-immediate: 32
addresses-year-1:    32
```

```
addresses-year-2:    32
subnets-immediate:    1
subnets-year-1:       1
subnets-year-2:       1
inet-connect:        Plan to connect DATUM through Internet Services Deutschland
country-net:         DE
private-considered:  yes
request-refused:     no
PI-requested:        no
addresses-returned:

 #[ADDRESSING PLAN]#

Relative                              Addresses Used
Prefix#   Subnet Mask       Size  Immediate  1yr  2yr  Description
0.0.0.0   255.255.255.224    32       32      32   32  Customer flat network,
                                                       router, firewall, servers
                             32       32      32   32  Totals

#[NETWORK TEMPLATE]#

inetnum:  195.177.100.64/224
netname:  DOMAIN-NET
descr:    Irgendeine Firma GmbH
country:  DE
admin-c:  JB123
tech-c:   RS233-RIPE
tech-c:   MR124-RIPE
rev-srv:  ns1.idnet.de
rev-srv:  ns2.idnet.de
status:   ASSIGNED PA
notify:   registry@idinter.net
mnt-by:   IDNET-MNT
changed:  h.hippenstiel@system-design.com DATUM
source:   RIPE

#[TEMPLATES END]#
```

Anhang C

Datenbank-Tabellen

C.1 dnshost

Attribut	Wertebereich	not null	unique	Beschreibung
id	number(5)	x	x	Primärschlüssel
zone	number(5)	x		Zone (Fremdschlüssel auf dnszone.id)
name	char(63)	x		Hostname
rp	number(5)			Responsible Person (Fremdschlüssel auf dnsuser.id)
cpu	char(20)			CPU für HINFO
os	char(20)			Betriebssystem für HINFO
comments	char(60)			Kommentare zu diesem Host (für TXT-Records)

C.2 dnszone

Attribut	Wertebereich	not null	unique	Beschreibung
id	number(5)	x	x	Primärschlüssel
parent	number(5)			Vaterzone (Fremdschlüssel auf dnszone.id)
primary_ns	number(5)			Primärer Nameserver (Fremdschlüssel auf dnsnameserver.id)
name	char(240)	x	x	Name der Zone
admin_email	char(64)			E-Mail Adresse des Administrators
soa_serial	number(10)			SOA-Seriennummer
soa_refresh	number(10)			SOA-Refresh
soa_retry	number(10)			SOA-Retry
soa_expire	number(10)			SOA-Expire
soa_minttl	number(10)			SOA-Minimum Time To Live
filename	char(64)			Name der Zonendatei
rp	number(5)			Responsible Person für diese Zone (Fremdschlüssel auf dnsuser.id)
extern	number(1)			Flag für externe Zone

C.3 dnsnameserver

Attribut	Wertebereich	not null	unique	Beschreibung
id	number(5)	x	x	Primärschlüssel
host	number(5)	x	x	Host, auf dem der Nameserver läuft (Fremdschlüssel auf dnshost.id)
file_dir	char(63)			Verzeichnis zur Ablage der Zonendatei-en
file_loopback	char(63)			Name der Loopback-Zonendatei
file_cache	char(63)			Name der Root-Server Cache Datei
ftp_host	char(240)			Host, von dem Cache-Datei bezogen wird
ftp_dir	char(240)			Verzeichnis, in der sich die Cache-Datei befindet
ftp_file	char(240)			entfernter Name der Cache Datei

C.4 dnsmailx

Attribut	Wertebereich	not null	unique	Beschreibung
zone	number(5)	x		Zone, für die der Mail Exchanger zuständig ist Fremdschlüssel auf zone.id
host	number(5)	x		Host, auf dem der Mail Exchanger läuft Fremdschlüssel auf host.id
name	char(240)			Name des Mail Exchanger
pref	number(5)	x		Präferenz-Wert
ttl	number(10)			Time To Live

C.5 dnsalias

Attribut	Wertebereich	not null	unique	Beschreibung
zone	number(5)	x		Zone, in der dieser Alias definiert ist Fremdschlüssel auf zone.id
host	number(5)	x		Host, auf den der Alias zeigt Fremdschlüssel auf host.id
can_name	char(63)	x		Alias

C.6 dnswks

Attribut	Wertebereich	not null	unique	Beschreibung
host	number(5)	x		Host, auf dem die WKS laufen Fremdschlüssel auf host.id
service	char(240)			Beschreibung der Dienste

C.7 dnsipaddr

Attribut	Wertebereich	not null	unique	Beschreibung
zone	number(5)			Zone, der IP-Adresse zugeordnet ist Fremdschlüssel auf zone.id
host	number(5)			Host, der IP-Adresse belegt Fremdschlüssel auf host.id
ip_addr	char(15)	x		IP-Adresse

C.8 dnssecondary

Attribut	Wertebereich	not null	unique	Beschreibung
zone	number(5)	x		Zone, für die der Nameserver Secondary ist Fremdschlüssel auf zone.id
nameserver	number(5)	x		sekundärer Nameserver Fremdschlüssel auf nameserver.id

C.9 dnsforwarder

Attribut	Wertebereich	not null	unique	Beschreibung
nameserver	number(5)	x		Nameserver, der als Forwarder arbeitet Fremdschlüssel auf nameserver.id
ip_addr	char(15)	x		IP-Adressen, die geforwardet werden sollen

C.10 dnspop

Attribut	Wertebereich	not null	unique	Beschreibung
id	number(5)	x	x	Primärschlußel
standort	char(63)	x	x	Beschreibung des Standortes

C.11 dnsnetwork

Attribut	Wertebereich	not null	unique	Beschreibung
pop	number(5)	x		POP, dem ein Class-C zugewiesen wird Fremdschlüssel auf pop.id
address	char(12)	x	x	Class-C als IP-Adresse

C.12 dnszoneblock

Attribut	Wertebereich	not null	unique	Beschreibung
zone	number(5)	x		Zone, der ein Block zugewiesen wurde Fremdschlüssel auf zone.id
ip_addr	char(15)	x	x	Anfang des zugewiesenen Blocks als IP-Adresse
block	number(3)	x		Größe des Blocks

C.13 dnsuser

Attribut	Wertebereich	not null	unique	Beschreibung
id	number(5)	x	x	Primärschlüssel
fullname	char(240)			Vollständiger Name des Benutzers
rights	number(1)	x		Rechte (s. 3.8.2)
phone	char(63)			Telefonnummer
email	char(63)			E-Mail Adresse
login	char(24)	x	x	Login
password	char(24)	x		zugehöriges Paßwort
comments	char(240)			Kommentare
company	char(50)			Firma
street	char(40)			Straße
city	char(40)			Ort
country	char(20)			Land
fax_no	char(20)			Fax-Nummer
nic_hdl	char(10)			NIC-Handle
notify	char(63)			RIPE-*Notify* E-Mail Adresse
mnt_by	char(63)			RIPE-*maintained by* Handle

C.14 dnsuserzone

Attribut	Wertebereich	not null	unique	Beschreibung
dnsuser	number(5)	x		Zonenverwalter Fremdschlüssel auf dnsuser.id
zone	number(5)	x		Zone, für die Berechtigung existiert Fremdschlüssel auf zone.id

Anhang D

Rückgabe- und Defaultwerte

In der Datei `defaults.pl` sind Standardwerte sowie die Klartexte für Rückgabewerte enthalten. Die relevanten Ausschnitte sind hier zur Referenz abgedruckt.

D.1 Defaultwerte

Diese Standardwerte dienen der Vorbelegung von Attributen der Datenbankobjekte.

```
%NAMESERVER = (
  'FILE_DIR',      '/etc/namedb',
  'FILE_LOOPBACK', 'db.127.0.0',
  'FILE_CACHE',    'named.cache',
  'FTP_HOST',      'rs.internic.net',
  'FTP_DIR',       '/domain',
  'FTP_FILE',      'named.cache'
);

%ZONE = (
  'PREFIX',      'db.',
  'CONTACT',     'hostmaster',
  'SOA_REFRESH', '10800',
  'SOA_RETRY',   '3600',
  'SOA_EXPIRE',  '604800',
  'SOA_MINIMUM', '86400'
);

%USERRIGHTS = (
  '1', 'Administrator',
  '2', 'Zonenverwalter',
  '3', 'Responsible Person'
);

%RIPE = (
  'EMAIL',      'auto-dbm@ripe.net'
```

```
);

%INETNUM = (
  'NETNAME',     'IPF-IDNET',
  'DESCR1',      'ID.net',
  'DESCR2',      'Internet Services Deutschland GmbH',
  'COUNTRY',     'DE',
  'SOURCE',      'RIPE'
);

%PERSON = (
  'COUNTRY',     'Germany'
);
```

D.2 Rückgabewerte

Negative Werte bezeichnen Fehlerkonditionen, die anderen liefern Statusmeldungen bei erfolgreicher Ausführung.

```
%ERROR = (
  # Allgemein 1-99

  # 1-9
  -1, 'Zuerst Zone waehlen',
  -5, 'Nicht authorisiert (Administrator-Rechte benoetigt)',
  -6, 'Nicht authorisiert (Rechte fr diese Zone benoetigt)',

  # Email 10-19
   -10, 'fehlerhafte Email-Adresse',

  # Host 20-29
   -20,  'fehlerhafter Host-Name',
   -21,  'Host existiert nicht',

  # IP Adresse
   -40, 'fehlerhafte IP-Adresse',
   -41, 'IP-Adresse existiert bereits',
   -42, 'IP-Adresse existiert nicht',

  # Zonen 100
  -100, 'Zonenname existiert bereits',
  -110, 'Primary NS exisitert nicht (->Programmfehler)',
  -111, 'Vaterzone existiert nicht (->Programmfehler)',
  -112, 'RP existiert nicht (->Programmfehler)',
  -120, 'Loeschen: Diese Zone delegiert an andere Zonen',
   120, 'Zone erfolgreich geloescht',
```

```
  # Host 200
  -200, 'zugehoerige Zone exisitert nicht (->Programmfehler)',
  -201, 'HINFO: CPU/OS Paar fehlerhaft',
  -203, 'IP Adresse nicht in dieser Zone verfuegbar',
   201, 'Info: IP Adresse bereits belegt',
   202, 'Host erfolgreich geaendert',
   203, 'Host-Check in Ordnung',
  -220, 'Loeschen: Nameserver existiert noch auf diesem Host',
   220, 'Host erfolgreich geloescht',

  # Nameserver 300
  -320, 'Loeschen: Zone wuerde ihren Primary verlieren',
   320, 'Nameserver erfolgreich geloescht',

  # Adressen 400
  -400, 'POP Standort existiert bereits',
  -401, 'POP Standort existiert nicht',
  -402, 'zugehoeriger POP Standort fehlt',
  -403, 'Name fr POP Standort fehlerhaft',
  -420, 'Loeschen: POP Standort hat noch Class C Netz(e) zugewiesen',
  -421, 'Loeschen: Class C Netz ist noch in Verwendung',
  -490, 'Kein Block dieser Groesse bei diesem POP verfuegbar',
   400, 'POP Standort erfolgreich hinzugefuegt',
   420, 'POP Standort erfolgreich geloescht',
   401, 'Class C erfolgreich hinzugefuegt',
   421, 'Class C erfolgreich geloescht',
);
```

Anhang E

E-Mail von Rob Austin

```
Date: Fri, 05 Dec 1997 13:14:15 -0500
From: Rob Austein <sra@epilogue.com>
To: Stephen Heilbronner <heilbron@nm.informatik.uni-muenchen.de>
Cc: Jon Saperia <saperia@zko.dec.com>,
    Peter Kai Wimmer <wimmer@informatik.tu-muenchen.de>,
    Alexander Keller <keller@nm.informatik.uni-muenchen.de>
Subject: Re: Implementations of RFC 1611 ?

  Date: Fri, 5 Dec 1997 11:10:42 +0100 (MET)
  From: Stephen Heilbronner <heilbron@nm.informatik.uni-muenchen.de>

> We are currently looking for DNS server software providing an SNMP
> management interface. Since you wrote RFC 1611, we thought, you could
> provide us with some pointers to implementations. We tried to search the
> Web and your company sites but to no avail.

Er, well, this is embarassing, but as far as I know there have never
been any implementations.  The stumbling blocks have been (a) 99% of
the DNS implementations out there are based on BIND, (b) nobody has
volunteered to implement the SNMP code in BIND, because (c) the SNMP
community keeps finding ways to avoid standardizing any mechanism that
could be used to allow a system daemon like BIND's named to be managed
by an external SNMP daemon.

With the advent of the AgentX subagent protocol, it's possible that
somebody will finally get around to this.  Then again, I said the same
thing about SNMPv2-classic proxy, so don't hold your breath.

I have gotten a few nibbles on this from time to time, but no
confirmed reports of implementations.

Someday, in theory, one of the people in my shop is going to implement
this for the DNS code in Epilogue's Attache Plus protocol stack, but
that's not going to be of much interest to anybody outside the
embedded systems community.

--Rob
```

Index

Literaturverzeichnis

[1] Albitz, P., Liu, C. *DNS and BIND, 2nd edition.* O'Reilly & Associates, Inc. Januar 1997

[2] Larry Wall and Randal L. Schwartz. *Programming Perl.* O'Reilly & Associates, Inc. 1992

[Hauck94] Hauck, Heiko. *Integration des Managements des Domain Name Systems in die vorhandene Managementumgebung des Leibniz Rechenzentrums (LRZ).* Diplomarbeit, Fakultät für Informatik, Technische Universität München, November 1994

[Hege93] Hegering, H.G, und Abeck, S. *Integriertes Netz- und Systemmanagement*, Addison-Wesley (Deutschland) GmbH, 1993

[Hege95] Hegering, H.G. *Vorlesung Netz- und Systemmanagement*, Technische Universität München, 1995

[RFC 822] Crocker, D.H. *Standard for the Format of ARPA Tnternet Text Messages, RFC822*, University of Delaware, August 1982

[RFC 1034] Mockapetris, P., *Domain Names - Concepts and Facilities, STD 13, RFC 1034*, USC/Information Sciences Institute, November 1987

[RFC 1035] Mockapetris, P., *Domain Names - Implementation and Specification, STD 13, RFC 1035*, USC/Information Sciences Institute, November 1987

[RFC 1101] Mockapetris, P., *DNS Encoding of Network Names and Other Types, RFC 1101* USC/Information Sciences Institute, April 1989.

[RFC 1123] Braden, R., *Requirements for Internet Hosts – Application and Support, RFC 1123* Internet Engineering Task Force, Oktober 1989

[RFC 1155] Rose, M., McCloghrie, K., *Structure and Identification of Management Information for TCP/IP-based Internets* Performance Systems International, Hughes LAN Systems, Mai 1990

[RFC 1183] Everhart, C., Mamakos, L., Ullmann, R. and P. Mockapetris, Editors, *New DNS RR Definitions, RFC 1183*, Transarc, University of Maryland, Prime Computer, USC/Information Sciences Institute, October 1990

[RFC 1340] Reynolds, J. , Postel, J. *Assigned Numbers, RFC 1340*, ISI, July 1992

[RFC 1442] Case, J., McCloghrie, K., Rose, M. and Waldbusser, S. *Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)*, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993

[RFC 1537] Beertema, P., *Common DNS Data File Configuration Errors, RFC 1537*, CWI, October 1993

[RFC 1611] Austein, R. und Sapiera, J. *DNS Server MIB Extensions, RFC 1611*, Epilogue Technology Corporation, Digital Equipment Corporation, Mai 1994

[RFC 1612] Austein, R. und Sapiera, J. *DNS Resolver MIB Extensions, RFC 1611*, Epilogue Technology Corporation, Digital Equipment Corporation, Mai 1994

[RFC 1706] Manning, B., und R. Colella, *DNS NSAP Resource Records, RFC 1706*, USC/ISI and NIST, October 1994.

[RFC 1713] Romao, A., *Tools for DNS debugging, RFC 1713* FCCN November 1994

[RFC 1912] Barr, D., *Common DNS Operational and Configuration Errors, RFC 1912*, Pennsylvania State University, February 1996

[RFC 1982] Elz, R., Bush, R. *Serial Number Arithmetic, RFC 1982* University of Melbourne, RGnet, Inc., August 1996

[RFC 2181] Elz, R., Bush, R., *Clarifications to the DNS Specification, RFC 2181*, University of Melbourne, RGnet, Inc. July 1997

[RFC 2182] Elz, R., Bush, R., Bradner, S. and Patton, M., *Selection and Specification of Secondary DNS Servers, RFC 2182*, University of Melbourne, RGnet, Inc. Harvard University, July 1997

[draft-ietf-http-state-mgmt-05] Kristol, D.M. und Montulli, L. *HTTP State Management Mechanism, INTERNET DRAFT* HTTP Working Group, Bell Laboratories, Lucent Technologies, Netscape Communications, November 1996

[ripe-119] Lord, A., Terpstra, M., *RIPE Database Template for Networks and Persons, ripe-119* RIPE Network Coordination Centre, Amsterdam

[ripe-141] Caslav, P., Kuehne, M., Orange, C., *European IP Address Space Request Form, ripe-141* RIPE Network Coordination Centre, Amsterdam

[ripe-142] Caslav, P., Kuehne, M., Orange, C., *Supporting Notes for the European IP Address Space Request Form, ripe-142* RIPE Network Coordination Centre, Amsterdam

[ripe-157] Magee, A.M.R., *RIPE NCC Database Documentation, ripe-157* RIPE Network Coordination Centre, Amsterdam, May 1997

[3] Meier, Karsten, *Zugestopft: Vermeiden von Sicherheitslöchern in CGI- Skripten, c't Magazin für Computertechnik, S. 346ff* Verlag Heinz Heise, Hannover, März 1998

[4] Lffler, G., Sand, M. und Wessendorf, G. *Endspurt - IPv6 vor der Fertigstellung, iX Magazin für professionelle Informationstechnik, S. 132ff* Verlag Heinz Heise, Hannover, Mai 1998

[RFC 2065] Eastlake, D., and C. Kaufman, *Domain Name System Security Extensions, RFC 2065*, CyberCash, Iris, January 1997.

[RFC 2136] Vixie, P., S. Thomson, Y. Rekhter, and J. Bound, *Dynamic Updates in the Domain Name System (DNS UPDATE), RFC 2136*, ISC, Bellcore, Cisco, DEC, April 1997.

[RFC 2230] Atkinson, R., *Key Exchange Delegation Record for the DNS, RFC 2230*, NRL, October 1997.

[RFC 1712] Farrell, C., M. Schulze, B. Pleitner, and D. Baldoni, *DNS Encoding of Geographical Location, RFC 1712*, Curtin University of Technology, October 1994.

[RFC 1995] Ohta, M., *Incremental Zone Transfer in DNS, RFC 1995*, Tokyo Institute of Technology, August 1996.

[RFC 1996] Vixie, P., *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY), RFC 1996*, ISC, August 1996.

[RFC 2052] Gulbrandsen, A., and P. Vixie, *A DNS RR for Specifying the Location of Services (DNS SRV), RFC 2052*, Troll Technologies, Vixie Enterprises, October 1996.

[RFC 2163] Allocchio, C., *Using the Internet DNS to Distribute MIXER Conformant Global Address Mapping (MCGAM), RFC 2163*, GARR-Italy, July 1977.

[RFC 2168] R. Danie1, M. Mealling., *Resolution of Uniform Resource Identifiers using the Domain Name System, RFC 2168*, Los Alamos National Laboratory, Network Solutions, Inc., June 1997.