Einführung in

Das Domain Name System des Internets DNS

(für Juristen, Journalisten, Kriminalisten und solche, die es wissen wollen)

Hadmut Danisch

Version 0.1 (7. September 2009) – *Alpha-Release*
urn:nbn:de:0107-einfdns-0-19

Hadmut Danisch
Hofäckerallee 13c
85774 Unterföhring
Deutschland
hadmut@danisch.de
www.danisch.de

Inhaltsverzeichnis

1 Einleitung

1.1 Warum und Wozu?

Ich hatte gerade Lust, eine Einführung in das DNS zu schreiben. Und ich hatte gerade etwas Zeit.

Es steckt aber doch noch etwas mehr Motivation dahinter. Noch vor einigen Jahren war das DNS etwas, womit sich nur einige wenige Spezialisten tiefer beschäftigt haben. Die meisten Internet-Nutzer haben es einfach benutzt, ohne näher drüber nachzudenken – oder ohne überhaupt zu merken, daß da etwas ist. Die typische Internet-Anwendungen, die man in der Laiensphäre häufig für das Internet selbst hält, allen voran Web-Surfen, E-Mail, Chat usw. funktionieren halt einfach so (mehr oder weniger). Daß da noch eine andere, eher unauffällige Anwendung, das DNS, im Hintergrund versteckt mitläuft, bemerkte man bis vor kurzem nicht. Die meisten Internet-Nutzer wußten nichts vom DNS, und die Internet-Nicht-Nutzer schon gar nicht.

Das hat sich geändert. Inzwischen ist das DNS ins Bewußtsein und die Diskussion gerückt. Zuerst etwas durch Phishing, eine Art von Angriffen auf Bankkonten, die teilweise auch über DNS-Angriffe ablaufen. Besonders aber durch die Diskussion über Internet-Sperren – Kinderpornographie und anderes. Die Forderungen und Erwartungen aus der Politik sind hoch, die Sachkunde ist niedrig und die Medienergüsse sind oberflächlich und seicht wenn es um Technik geht. Journalisten und Juristen neigen dazu, das gegenseitige Zitieren und Abschreiben über das Informieren aus erster Hand zu setzen. Von Juristen wird die rechtskräftige und zitierbare Aussage in einem Urteil höher bewertet als die Frage, ob der Richter es dabei eigentlich selbst verstanden hat und ob das so stimmt. Es wird viel geschrieben, und noch mehr abgeschrieben und zitiert. Manchmal ist es verblüffend zu sehen, wie eine an einer Stelle in Umlauf gesetzte Formulierung oder Argumentation ihren Weg durch veschiedene Publikationen bis hin zu Parlamentsschriftsachen finden.

Das führt zu einem gefährlichen Effekt, denn die (Rechts- und Presse-)Meinungen, die sich über das DNS gebildet haben, haben oft nur noch wenig mit dessen technischen Eigenschaften zu tun, sondern mit losgelösten, in der Literatur hin- und herschwappenden und sich selbst verstärkenden Meinungen. Juristen neigen berufsbedingt dazu, die höhere Instanz weit höher zu gewichten als die Realität, und die Aussage eines Juristen höher als die eines Technikers. Oder anders gesagt: *Papier ist geduldig.*

Und so verbreiten sich unrichtige, unvollständige, irreführende, laienhafte oder unglücklich formulierte Aussagen über das DNS rekursiv und unaufhaltsam weiter. Sogar in Gutachten für die Bundesregierung zu Internet-Sperren und allerlei regierungsinternen und öffentlich-politischen Texten stand viel Unsinn zum DNS. Man merkt da schon, wer das DNS halbwegs verstanden hat und wer nur nachplappert, was er irgendo aufgelesen hat.

Ich muß aber leider auch zugeben, daß man daraus nicht immer einen Vorwurf machen kann, denn eine gute Dokumentation, in der man sich informieren kann, konnte ich bisher auch nicht benennen. Das ist nämlich das Übel meiner eigenen Zunft, daß sie komplizierte Dinge baut, und diese entweder nicht, schlecht oder – wie im Falle DNS – zwar vollständig und präzise, aber für Nicht-Spezialisten unlesbar dokumentiert. Es gibt nur wenig Literatur zum DNS, und die ist meist schwer verständlich, für Techniker und Informatiker geschrieben und auf Administration ausgelegt. Als schlechtes Beispiel kann man sich den deutschen oder den englischen Wikipedia-Eintrag zum „Domain Name System“ anschauen. Es ist alles technisch richtig, was darin steht, aber jemandem, der nicht schon vorher weiß, was das DNS ist, hilft es kein Stück weiter. Ein typische Informatiker-Dokumentation, die Wissen nur dem vermittelt, der es schon hat. Es gibt sogar Vorlesungen und Skripte zum DNS, aber auch die sind meist hartes Brot und nicht selten selbst nur oberflächlich oder gar fehlerhaft, weil irgendein Dozent universitätstypisch von jetzt auf gleich mal irgendwelche Folien zu irgendwas, was er selbst nicht so richtig verstanden hat, zusammenrührt, und das Ziel weder Verständnis noch technische Befähigung, sondern Transport prüfungsfähigen Stoffes ist.

Ab und zu findet man oberflächliche Kurzzusammenfassungen über DNS im Umfang von ein bis zwei Sätzen bis höchstens ein oder zwei Seiten. Management-Blabla, Powerpoint-Futter, vom Niveau „DNS ist das Telefonbuch des Internet“. Zum Verständnis untauglich.

Aber auf die Frage, wo man das mal nachlesen kann, wußte ich bisher auch nichts zu empfehlen.

So gesehen bleibt Juristen und Journalisten ja eigentlich auch nichts anderes übrig, als voneinander abzuschreiben. Sonst ist ja nichts da. Also war die Überlegung das, womit ich normalerweise das DNS mündlich erkläre, aufzuschreiben. Ich bin – am Rande bemerkt – Informatiker und habe seit etwa 20 Jahren beruflich auch mit dem DNS zu tun. Also schon seit der Frühzeit des DNS an.

1.2 Aufbau und Inhalt

Diese Einführung ist so aufgebaut, daß sie vorne einfach beginnt und nach hinten immer technischer, und damit auch konkreter und anspruchsvoller wird. Die Absicht dahinter ist, daß man vorne anfängt zu lesen und dann aufhört,

sobald man keine Lust mehr hat, es zu schwierig wird oder man gefunden hat, was man sucht. Wenn man hinten angekommen ist (sobald die vorliegende Einführung fertig geschrieben ist), hat man dann von DNS schon mehr verstanden als die meisten Informatiker.

Diese Einleitung ist aber nur dazu gedacht, das DNS zu *verstehen*. Sie ist nicht für Administration und Betrieb gedacht, denn dazu muß man noch einiges mehr wissen, was hier zu weit führen würde. Insbesondere möchte ich hier nicht auf die Eigenheiten konkreter Implementierungen, die Syntax von Zonendateien und dergleichen eingehen.

Diese Einleitung ist auch keine Monographie im bibliographischen Sinn. Ich schreibe sie nicht fertig und geben sie dann heraus, wie man es mit einem Buch macht. Ich schreibe gelegentlich mal dies und mal das, und immer wenn ich es für angebracht halte, gebe ich eine neue Version heraus, wie bei Software. Aus gewissen Zeitgründen geben ich die ersten Versionen sogar heraus, bevor die letzten Kapitel fertig sind.

Und weil die E-Books gerade in Schwung kommen und ich mir demnächst auch eines kaufen will, wird es auch eine epub-Version geben. Sobald ich herausgefunden habe, wie man das macht und die nötige Software dazu gefunden oder selbst geschrieben habe.

1.3 Eigene Experimente

Die wichtigste Zutat zum Verständnis des DNS ist der Aha-Effekt, den man durch Ausprobieren erzielt. Ich rede dabei lieber von Experimenten als von Übungen, weil sich das genauso motivierend anhört wie „iß das, das ist gesund“. Es geht hier nicht darum, irgendetwas zu üben. Es geht darum, etwas auszuprobieren und eigene Variationen anzustellen. Bis man merkt, daß man es verstanden hat. Die Experimente beginnen in Abschnitt 4.1, bis dorthin kommt man mit Lesen.

Reicht es denn nicht, einfach diese Einführung nur zu lesen? Nein:

- Diese Einführung ist statisch. DNS dagegen ist dynamisch und die Daten verändern sich ständig. Nicht nur, daß ich diese Einleitung nicht aktuell halten könnte, solange sie als Papier oder PDF erscheint. *Der Leser soll verstehen, daß DNS eine sich ständig verändernde große Datenstruktur ist.* Und dazu gehört die Erkenntnis, daß man DNS nicht auf Papier drucken kann, sondern daß man immer im lebenden DNS die aktuellen Daten selbst abrufen muß.
- Ich werde die meisten Erläuterungen am Beispiel meiner eigenen Domain danisch.de vornehmen, denn an irgendetwas muß ich es ja erklären. Das heißt aber nicht, daß jeder sich (nur) meine Domain anschauen soll. Der Leser möge sich beliebige, auch ausländische Domains heraussuchen und die Beispiele analog mit anderen Domains nachvollziehen.

Daß ich für Erläuterungen meine eigene Domain verwende liegt vor allem daran, daß ich da nicht von Änderungen überrascht werde und keinen Ärger mit dem Domaininhaber befürchten muß, wie wenn ich fremde Domains verwende.

Der Leser möge aber unbedingt alle Beispiele und Experimente mit irgendwelchen anderen Domains nachvollziehen. Probieren Sie herum!

1.4 Quellen und weitere Informationen

Der Vorteil des Internet und seiner originären Protokolle ist: Die Dokumentation ist frei verfügbar. Der Nachteil ist: Das Zeugs ist so geschrieben, daß der Nichtfachmann nicht viel davon hat.

Die Internet Engineering Task Force IETF, unter deren Leitung die Entwicklung des Internet verläuft, gibt sogenannte „Request for Comments“, RFCs, heraus, in denen das alles detailliert beschrieben wird. Man kann sie kostenlos unter

http://www.ietf.org/rfc.html

abfragen, indem man die Nummer eingibt. Von Bedeutung sind hier folgende RFCs:

RFC 1034 und 1035 stammen aus dem Jahr 1987, wurden von Paul Mockapetris geschrieben und beschreiben die Struktur und das Protokoll von DNS und sind auch heute noch gültig. Da steht genau drin, wie es funktioniert. Daran sieht man auch, daß das DNS in seiner heutigen Form bereits über 20 Jahre alt ist.

RFC 882, 883 und 973 Ursprünglich wurde DNS in einem ersten Anlauf von Mockapetris schon 1983 in den RFC 882 und 883 beschrieben. Es zeigt sich aber bald, daß man noch einiges überarbeiten und ergänzen mußte. Die Erkenntnisse aus diesem ersten Testlauf wurden in RFC 973 beschrieben. Die beiden RFCs 882 und 883 wurden dann durch 1034 und 1035 ersetzt. Sie sind heute nicht mehr von Bedeutung und veraltet, und nur dann von Interesse, wenn man sich für die Geschichte und Entstehung des Internet interessiert.

RFC 2782 führte die SRV-Records ein (Abschnitt 4.7).

1.5 Fehler und Haftungsausschluß

Natürlich bin ich nicht frei von Fehlern. Obwohl ich jahrelang beruflich auch DNS-Server installiert und Domains konfiguriert, und bei der IETF sogar schon DNS-Erweiterungen entworfen habe, bin ich auch nicht bis ins allerletzte Detail mit jeder Einzelheit vertraut.

Außerdem ist das ein reines Lust-, Laune- und Freizeitprojekt ohne kommerziellen Hintergrund, ich verdiene daran nichts. Deshalb sind die Zeit und die Sorgfalt, die ich hineinstecken kann und will, begrenzt. Ich kann daher keinerlei Haftung und Gewähr für die Richtigkeit und Vollständigkeit übernehmen. Wer's liest und benutzt ist selbst schuld.

Trotzdem bin ich natürlich für Korrekturen, Hinweise auf Fehler, Ergänzungen, Kommentare, Verbesserungsvorschläge usw. dankbar und nehme sie gerne entgegen (hadmut@danisch.de).

2 Was ist das DNS?

2.1 Begriffsentwirrung

Auf die Frage, was das DNS genau ist, muß man eigentlich mehrere Antworten geben, denn es vermischen sich darin – wie auch beim Begriff Internet – verschiedene Bedeutungen. Diese Bedeutungen werden meist nicht präzise differenziert und ergeben sich oft nur aus dem Kontext des Sprechers oder durch Rückfragen – wenn dem Sprecher die Unterschiede selbst klar sind. Dabei ist die Abgrenzung der einzelnen Bedeutungen weder einfach, noch erschließt sie sich intuitiv.

Man sollte beim DNS folgende Bedeutungen auseinanderhalten:

- DNS als Funktionskomponente und „kleiner unsichtbarer Dienstleister“ vieler Dienste im Internet, und damit *die Funktionen*, die das DNS für diese Dienste erbringt, hauptsächlich die Namensauflösung und einige andere Funktionen, wie sie im Kapitel 4 beschrieben werden,
- der Datenbestand, der im DNS gespeichert wird und ebenfalls im Kapitel 4 beschrieben wird,
- die für das DNS charakteristische logische, hierarchisch verteilte Datenspeicher- und Serverstruktur, die im Kapitel 5 betrachtet wird,
- das DNS als große Datenbank mit besonderen Eigenschaften, wie im nachfolgenden Unterkapitel beschrieben wird,
- das DNS als Kommunikationsprotokoll, also die Festlegung einer gewissen Sprache, mit der Rechner miteinander über das Internet sprechen können, um DNS-Daten auszutauschen, wie in Kapitel 7 beschrieben,
- die Software, mit der das DNS betrieben wird, hauptsächlich die diversen DNS-Server und die Abfragesoftware, die in dieser Einführung nicht näher betrachtet werden, und deren bekanntester Vertreter der OpenSource-DNS-Server *bind* sein dürfte,
- das DNS als länderübergreifende weltumspannende Anwendung und Verwaltungsfunktion des Internet und damit als Machtinstrument und Quelle internationaler politischer Streitigkeiten, auf die hier nicht näher eingegangen wird,
- das DNS als Hauptangriffspunkt für Internet-Sperren und als politisch gebrauchtes Buzzword, wie in Kapitel 9 betrachtet.

2.2 Warum gibt es das DNS?

Das DNS wurde in den Grundzügen 1983 von Paul Mockapetris entworfen und zunächst in den RFCs 882 und 883 beschrieben. Aktualisiert wurden diese Entwürfe 1987 durch die RFCs 1034 und 1035, die im wesentlichen das auch heute noch verwendete DNS beschreiben, und die noch immer maßgeblich sind, obwohl es inzwischen eine Reihe von Ergänzungen und Verbesserungen gab. Das DNS ist damit inzwischen über 20 Jahre alt – und damit älter als das World Wide Web, dessen Entwicklung erst 1989 seine ersten Anfänge nahm und erst in den Neunzigern und mehr noch in den Nullern in Schwung kam.

Das DNS war die Lösung eines Problems, das mit dem damals gerade in seiner Frühentwicklungsphase befindlichen Internets entstanden war – nämlich daß man jedem Rechner, der bislang nur einen Namen hatte, nun auch eine Internet-Adresse (IP-Adresse) zuordnen mußte, und umgekehrt.

Das hatte im wesentlichen drei Gründe.

Der erste Grund ist, daß die Rechner schon vor dem Internet Namen hatten[1], der Name war also das wichtigere und ältere Merkmal war, und die IP-Adresse erst nachträglich dazu kam, man also beides brauchte.

Der zweite Grund ist, daß man schon früh bemerkte, daß man mit IP-Adressen allein nicht gut zurecht kam. Einmal, weil sie sich zu häufig änderten und man nicht mitbekam, daß ein Rechner mit dieser IP-Adresse eigentlich derselbe ist, der gestern noch eine andere hatte. Wie sollte man das vernünftig mitteilen, wenn es für den Rechner nicht noch einen abstrakten, dauerhaften Namen gab? Außerdem mußte man bei Änderung der IP-Adresse eines Rechners an zu vielen Stellen komplexe Änderungen vornehmen, was fehleranfällig war. Deshalb erkannte man schon frühzeitig die Notwendigkeit *symbolischer Namen*[2] ein. Man gab den Rechnern Namen, die ihre Funktion beschreiben oder die man sich sonstwie merken konnte, benannte sie nach Planeten, Comicfiguren, Raumschiffen und Whisky-Sorten. Damit mußte man sich die IP-Adressen nicht merken, sich bei Änderungen im Netzwerk nicht umgewöhnen, Konfigurationen nicht ändern.

Der dritte Grund ist, daß Informatiker leidenschaftlich gerne abstrahieren und Zuordnungsfunktionen bauen. Das macht Spaß, bringt Ordnung und wird als

[1] Vor Einführung des Internet war die Telekommunikation über Modems und normale Terminal-Sitzungen üblich, man wählte sich etwa mit einem Terminal – ein einfaches Datensichtgerät mit Bildschirm, Tastatur und 80x25 Textdarstellung – und einem Modem über das Telefonnetz beim Zentralrechner ein, meldete sich mit Benutzernamen und Passwort an, und konnte auf diesem Weg über beliebige Distanz arbeiten. Auf gleiche Weise konnten auch Rechner miteinander kommunizieren und per UUCP auch E-Mail, News und anderes übertragen, ohne jegliches Internet. Dafür waren Namen notwendig, also gab es diese symbolischen Namen schon vor dem Internet und nicht erst, wie oft behauptet, um sich Internet-Adressen nicht merken zu müssen.

[2] Genauer gesagt alphanumerische mnemotechnische Namen[3][4], also solche, die aus beliebig wählbaren Buchstaben und Ziffern für Rechner bestanden und „sprechende“ Namen hatten.

technischer und wissenschaftlicher Fortschritt angesehen. Außerdem sind Informatiker Leute, die – schon aus gesunder Faulheit – versuchen, alles was sich ändern kann, immer nur an einer einzigen Stelle zu speichern und ändern zu müssen, und Änderungen zu automatisieren. Es ist ein Vorteil, wenn man bei der Änderung einer IP-Adresse nicht erst lange überlegen muß, an wievielen Stellen man die Adressen ändern muß, sondern weiß, daß man sie nur an einer einzigen zentralen Stelle, nämlich der Zuordnung von symbolischem Namen zur IP-Adresse, ändern muß und damit alles erledigt hat. Informatiker machen das eben so, und es hat sich bewährt.

Man erfand daher die Namensauflösung (Name Service) als Programmfunktion, die während des Programmlaufes die Zuordnung von symbolischen Rechnernamen zu IP-Adressen und umgekehrt vornahm, zunächst noch ohne DNS.

Das war anfangs sehr leicht, denn es gab nur ganz wenige Universitäts- und Militärrechner (hauptsächlich in den USA) und noch weniger Organisationen, die am Internet teilnahmen. Rechner waren sehr teuer und selten, und es änderte sich wenig. Man konnte das damals einfach als Liste in eine Datei schreiben und diese regelmäßig an alle Rechner im Internet verteilen – quasi wie ein Telefonbuch. Dann hatte jeder, der am Internet teilnahm, diese Liste, und kannte zu jedem Rechnernamen dessen IP-Adresse[5].

Das ging nicht lange gut. Die Liste wurde mit der wachsenden Zahl der Rechner am Internet schnell viel zu lang, um sie noch ordentlich verwalten zu können – und Speicherplatz und Kommunikationsbandbreite waren damals knapp und kostbar. Änderungen an Hostnamen und IP-Adressen wurden immer häufiger notwendig, so daß man auch mit der Aktualisierung nicht mehr nachkam. Im Prinzip steigt der Aufwand für die Verteilung dieser Telefonbuch-Liste *mit der dritten Potenz der Zahl der Rechner* und damit viel, viel schneller als die Zahl der Rechner, denn je mehr Rechner man hat, desto länger wird die Liste, an desto mehr Rechner muß man sie verteilen und desto öfter muß man sie verteilen, weil es mehr Änderungen gibt. Weil aber schon die Anzahl der Rechner überlinear stieg, explodierte der Aufwand für die Verteilung der Liste förmlich. Dazu kam der Verwaltungsaufwand für die Erstellung der Liste und die steigende Zahl von Fehlern und Korrekturen. Und das, obwohl das Internet damals noch viel, viel kleiner als heute war, nur einige hundert oder wenige tausend Rechner umfaßte. Außerdem wollten Administratoren in der Lage sein, nach Änderungen am Netzwerk sofort wieder funktionsfähig zu sein und nicht erst auf die nächste Verteilung der Liste zu warten.

Mit dem Fortschritt des Internet kamen neue Funktionen, die mit einer einfachen Liste nicht mehr abzubilden waren. Man wollte beispielsweise angeben können, wohin E-Mail abzuliefern wäre. Außer der Zuordnung von Hostnamen zu IP-Adressen hätte man deshalb noch eine zweite Tabelle verwalten müssen, nämlich die der Rechner, bei denen die E-Mail für die verschiedenen

[5] So entstand übrigens die hosts-Datei, die es unter Unix und heute unter Linux und Windows gibt.

Empfänger abgeliefert werden sollte. Das bedeutete noch mehr Verwaltungsaufwand und noch mehr Aktualisierungen.

Man erkannte, daß es mit einer solche Liste aller Internet-Adressen nach dem Prinzip eines Telefonbuchs nicht mehr weiterging. Eine Lösung des Problems mußte her.

Man machte sich dazu die Ursache des Problems, das gerade entstehende und wachsende Internet, zunutze und führte ein auf dem Internet basierendes System ein, bei dem der, der zu einem Hostnamen die IP-Adresse (oder umgekehrt) wissen wollte, schnell und einfach nach der *jeweils aktuellen Zuordnung* fragen konnte, und es dabei vermieden werden sollte, daß veraltete Daten herumliegen. So sorgte man dafür, daß immer *frische* Daten verwendet werden.

Außerdem sollte die zentrale Verwaltung abgeschafft werden, weshalb man es einführte, daß jeder die Zuordnung seiner Rechner zu seinen IP-Adressen selbst verwalten konnte. Das brachte die Notwendigkeit, diesen ganzen großen Namens- und IP-Adressbereich (man sagt auch Namensraum und Adressraum) so in viele kleine Zuständigkeiten aufzuteilen, daß es nicht zu Kollisionen, Überlappungen, Streitigkeiten kommt. Man erfand dazu ein hierarchisch unterteiltes und delegierbares Datenbanksystem. Die Domain-Namen und das DNS waren erfunden.

Damit ging zwei ganz wesentliche Neuerungen und Unterschiede zum alten Zustand einher:

- Es werden nicht mehr wie zuvor *alle Daten auf Vorrat* übertragen, sondern nur noch die Daten, die tatsächlich verwendet werden, und zwar erst dann, wenn sie *angefragt* werden.
- Es gibt keine zentrale Verwaltung aller Daten mehr. Der Anbieter der Daten liefert dezentral und ohne größeren Umweg direkt an den Nutzer der Daten.

  Wenn man so will, dann kann man DNS von seinem Aufbau her auch als eines der ersten Peer-to-Peer-Netzwerke im Internet ansehen[6].

Das DNS macht damit das Führen der bis dahin notwendigen Listen überflüssig und löste die damit verbundenen Probleme.

2.3 DNS als große verteilte Datenbank

Das DNS ist somit zunächst mal ein Datenbanksystem, das alle Eigenschaften einer klassischen Datenbank erfüllt: Da ist ein großes Ding, in das irgendwelche Leute irgendwo Informationen einfüllen und an das irgendwo anders

[6] Aber nicht als das erste überhaupt, denn auch die Vorgänger der Internet-Dienste wie UUCP, das in den Siebzigern entstand, und sogar das Internet selbst könnte man als Peer-to-Peer-Netzwerk ansehen.

irgendwelche anderen Leute Fragen stellen und aus dem Datenbestand Antworten erhalten – oder auch nicht.

Datenbanken sind heute allgegenwärtig, vor allem in Form von sogenannten relationalen Datenbanken, oft auch als SQL-Datenbank bezeichnet, die – stark vereinfach gesagt – gleichartige Daten zentral in großen Listen speichern. Solche relationalen Datenbanken erfüllen aber die Anforderungen, die man hier stellen muß, nicht. Außerdem bezieht sich der Begriff der relationalen Datenbank auf die Art und Weise, wie Daten *strukturiert gespeichert* werden. Beim DNS ist aber nicht so wichtig (und übrigens auch nicht standardisiert), wie man die Daten speichert, sondern *wie man sie zwischen Rechnern austauscht, wie man sie zwischen den Rechnern verteilt und daß sie mit einem Namen assoziiert sind.*.

Der Schwerpunkt beim DNS liegt also weniger auf der Datenbank und der Speicherung selbst, das kann man machen wie man Lust hat, als vielmehr auf der Sprache – dem Protokoll – zum Austausch der Daten zwischen den Rechnern, also zwischen der Datenbank und dem der fragt (genauer ab Seite 62), und der Art und Weise, wie man die Zuständigkeiten dafür verteilt (ab Seite 50) und die Daten findet. DNS ist also eine große, weltweit verteilte Datenbank, die auf der Zusammenarbeit vieler Rechner beruht und dazu diese Zusammenarbeit durch Festlegung gewisser Prinzipien und einer einheitlichen Sprache der Rechner untereinander reglementiert und ermöglicht.

Das ist ganz typisch für solche verteilten Systeme, daß man nicht spezifiziert, wie Daten gespeichert werden, und das jedem selbst überläßt, aber genau spezifiziert wie die Sprache zwischen den Rechnern aussieht und welche inhaltlichen Anforderungen erfüllt werden müssen. Man überläßt es jedem, wie er das Problem löst, solange gewährleistet ist, daß er ordnungsgemäß mit den anderen kooperiert[7].

Wir betrachten nun die besonderen Anforderungen an das DNS und die Eigenschaften des DNS als Datenbank. Erst wenn man sie kennt, kann man viele Eigenschaften des DNS nachvollziehen und verstehen. Viele öffentlich erhobene Forderungen nach Eingriffen in das DNS beruhen auf Unkenntnis der Hintergründe.

Es ist ein *verteiltes* System: Es gibt also nicht eine große zentrale Datenbank, in der alles gespeichert wird, sondern sehr viele, sehr kleine Datenbanken, die jede nur ein kleines Fragment davon speichern, aber die dafür gut miteinander vernetzt sind.

Es ist ein *delegiertes* System: Weil man eine zentrale Verwaltung mit riesigem Verwaltungsaufwand, schleppender Bearbeitung und hoher Fehlerquote vermeiden wollte, hat man das DNS so gebaut, daß jeder die Daten, für die er zuständig und verantwortlich ist (seine Domain), selbst verwalten *und die Daten dazu direkt verteilen* kann. Daß man also die

[7] Das ist auch bei vielen anderen verteilten Diensten der Fall, beispielsweise bei LDAP, E-Mail, dem World Wide Web.

technische Verwaltung einer Domain an deren rechtlich Verantwortlichen (oder Inhaber) *delegieren* kann.

Es ist ein *dezentrales* System: Es gibt keine zentrale Verwaltungsstelle für die im DNS gespeicherten Daten. Eigentlich ist das keine neue Aussage, sondern nur ein Folge oder Umformulierung der ersten beiden Punkte. Aber weil zunehmend erwartet wird, daß Eingriffe in das DNS vorgenommen werden, erscheint es mir wichtig zu betonen, daß es im ganzen DNS keine einzelne, zentrale Stelle gibt, die man zu Sperrungen verpflichten könnte. Jede Stelle im DNS ist nur für einen kleinen Teil der Funktionalität zuständig.

Es ist ein *hierarchisches* System: Um diese Verteilung der Daten, die administrativen Zuständigkeiten und Autorisierungen sauber und ohne Kollisionen an verschiedene Leute delegieren zu können, und denen die Möglichkeit zu geben, ihren Bereich in weitere Teile zu unterteilen und ihrerseits an Dritte oder Unterabteilungen weiterdelegieren zu können, hat man eine hierarchische Organisationsform geschaffen. Die Hierarchie steuert nicht nur die Delegation, sondern ist damit auch die logische Struktur, nach der Daten abgelegt werden. Anfragen an das DNS richten sich immer nach der Stelle im Hierarchiebaum. Fragen an das DNS lauten stets „Welche Daten sind für diese Stelle in der Hierarchiestruktur hinterlegt“. Diese Hierarchie bezeichnet man (auch) als das Domain Name System. Mehr dazu ab Seite 50.

Die Hierarchie als Organisationsform ist damit sowohl Teil der Anforderungen, als auch Teil der gewählten Lösung des Problems.

Es ist ein *assoziatives* System: Im Gegensatz zu beispielsweise einer relationalen Datenbank (der meistverbreitete Datenbanktyp), in der Daten einfach ohne zwingendes Suchkriterium aufgelistet werden, ist das DNS ein sogenannter *Assoziativer Speicher*. Das heißt, daß Daten immer an einen Namen gebunden sind und danach gefunden werden. Daten können im DNS nicht einfach so herumliegen wie in einer relationalen Datenbank. Sie sind immer an einen Domainnamen gebunden, und der Name ist das (einzige) Suchkriterium.

Es ist kein *universelles* System: Damals sah man nicht die Notwendigkeit, beliebige Daten speichern zu können. Das wäre beim damaligen Stand der Technik auch viel zu aufwendig und damit zu teuer geworden, wenn nicht gar teils unmöglich gewesen und entsprach nicht dem Stand der Technik und des Wissens. Deshalb hat man das, was man speichern kann, auf einige wenige Daten- oder Record-Typen beschränkt. Das DNS kennt also beispielsweise keine Datentypen für Bestellnummern, Temperaturen, Geldbeträge, sondern nur die, die man damals bei seiner Entwicklung als für Internetdienste notwendig erachtete. Wir betrachten diese Datentypen genauer im Kapitel 4.

Es ist ein *redundantes* und *hochverfügbares* System: Das DNS kann seinen Zweck nur erfüllen, wenn es *immer, rund um die Uhr* verfügbar ist. Rechner sind aber nicht immer in Betrieb. Sie gehen mal kaputt, müssen

mal gewartet oder an einen anderen Ort gebracht werden, manchmal muß man sie ersetzen. Manchmal hat die Software Fehler und stürzt ab, und manchmal macht der Administrator Fehler. Damit das nicht immer gleich zum Ausfall des DNS führt, ist das gesamte DNS so aufgebaut, daß *jede* Funktion des DNS durch mehrere Rechner erfüllt werden kann, die sich gegenseitig ersetzen, und in der Regel auch durch mehrere Rechner erfüllt wird. Man kann für eine Funktion im DNS immer mehrere Rechner angeben, die dann gleichberechtigt dieselbe Funktion erfüllen. Wer Daten abrufen will, bekommt diese Rechner genannt und sucht sich beliebig einen dieser Rechner heraus. Antwortet der nicht, fragt man einfach den nächsten. Man braucht für DNS deshalb auch keine hochverfügbaren Rechner oder Cluster, weil DNS selbst schon die nötigen Funktionen mitbringt, um mit gewöhnlichen Rechnern Hochverfügbarkeit zu erbringen. Dabei hat das den Vorteil, daß die Rechner, die gemeinsam eine Funktion erfüllen, nicht einmal in der Nähe zueinander stehen müssen, wie bei einem Cluster. Man kann die Rechner auch auf verschiedene Internet-Provider, verschiedene Länder oder gar verschiedene Kontinente verteilen – womit man auch eine gewisse Redundanz gegenüber politischen und gesetzlichen Einflüssen, Feuer oder Naturkatastrophen sowie größeren Netzwerkausfällen erzielen kann.

Es ist ein *schnelles* System mit schnellen Aktualisierungen: Ein Problem, das DNS lösen mußte, war die schleppende Aktualisierung der Daten nach Änderungen oder neuen Zuordnungen von Internet-Adressen. Mit DNS können Änderungen sehr schnell verteilt werden, weil derjenige, der für die Daten originär zuständig ist, nicht nur die Daten selbst, sondern dazu auch deren „Haltbarkeit“, die sogenannte Time To Live (TTL) sekundengenau angeben kann, und damit, wie lange die Daten als „frisch“ gelten bis sie erneut angefordert werden müssen.

Es bleibt einem selbst überlassen, ob man für die Daten seiner Domain eine Haltbarkeit von Sekunden, Stunden, Tagen oder sogar Monaten bestimmen will. Je kürzer die Haltbarkeit, desto früher verfallen die Daten wieder bei denen, die danach gefragt haben, desto eher müssen sie erneut danach fragen und desto schneller führt eine Änderung der Daten zu einer Aktualisierung. Die Kehrseite dessen ist, daß es damit auch zu mehr Anfragen und damit zu einer höheren Belastung der fragenden und der antwortenden Rechnern kommt. Es ist eine Frage der Abwägung im Einzelfall, welche Haltbarkeit man für die eigenen Daten angibt. Typischerweise liegen diese Haltbarkeitsfristen zwischen einer Viertel Stunde und zwei Wochen.

Es ist ein *schnelles, kleines, billiges* System: Man muß sich daran erinnern, daß zu der Zeit, als das DNS entwickelt wurde, das Internet verglichen mit heute steinzeitlich langsam und extrem teuer war, und daß die Rechner verglichen mit heute sehr teuer, leistungsschwach und mit sehr wenig Speicher ausgestattet waren. Anders als heute hat man damals versucht, jedes einzelne Byte einzusparen und jeden möglichen Geschwindigkeitsvorteil herauszuholen. DNS basiert daher auf dem Aus-

tausch sehr kleiner und teils komprimierter Datenpakete, in denen die Daten mit hoher Packungsdichte verstaut sind. Das führt leider zu allerhand Einschränkungen und schlechten Eigenschaften (siehe Kapitel 8), aber eben auch dazu, daß das DNS so flott ist und selbst auf den langsamen Rechnern von damals schon war, daß man im normalen Betrieb als Nutzer davon kaum etwas oder nur wenig bemerkt.

DNS entlastet den Endrechner: Eine weitere Folge dessen, daß Rechner damals im Vergleich zu heute sehr teuer, sehr schwach und mit sehr wenig Speicher ausgestattet waren, ist, daß das DNS so gebaut ist, daß es den End-Rechner, der das DNS nutzen will (also beispielsweise den Arbeitsplatz-PC auf dem Schreibtisch) möglichst entlastet und dazu mit möglichst wenig Speicher-, Netzwerk- und Rechenaufwand belastet.

Man hat das DNS dazu so gebaut, daß auf den einzelnen Endnutzer-Rechnern nur ein sehr kleine Software-Ergänzung notwendig ist, und man im lokalen Netzwerk – in Firmen, Universitäten, Organisationen – nur einen einzigen zentralen DNS-Rechner braucht, der die ganze Arbeit des Zusammensuchens der Daten durch Abklappern der Hierarchie und das Zwischenspeichern der Daten übernimmt, und den die Endrechner nur fragen müssen. So mußte man in einem LAN diesen Aufwand nur einmal betreiben und die Kosten nur einmal aufbringen. Weil diese zentralen Rechner für die anderen Rechner im LAN die Aufgabe der Auflösung von DNS-Anfragen übernehmen, werden sie auch als *Resolver* bezeichnet.

Obwohl heutige Rechner so stark und so billig sind, daß sie diese DNS-Abfragen leicht und schnell selbst erledigen könnten, hat sich diese Struktur bis heute erhalten – nicht zuletzt weil man es in Betriebssystemen wie Windows nie geändert hat. Noch heute fragen Rechner in Firmennetzwerken (LANs) den zentralen Nameserver, den sie per DHCP oder per fester Konfiguration genannt bekommen.

Dabei hat sich diese Struktur, die ursprünglich nur für das LAN gedacht war, sogar weiter ausgebreitet. Wer sich als Privatkunde mit einem Internetproviderprovider verbindet, erhält über PPP oder DHCP ebenfalls die Angabe, wo der Nameserver des Providers steht, obwohl die Leistungsfähigkeit heutiger Rechner dies eigentlich überflüssig machen würden.

Tatsächlich ist es heute (besonders unter Linux, selbst auf Kleinstrechnern) sehr einfach, seinen eigenen DNS-Resolver zu betreiben und auf die Nutzung des DNS-Servers des Internet-Providers zu verzichten. Es macht im Bereich der Privat- und kleinen Firmenkunden aber kaum jemand.

DNS ermöglicht Zwischenspeichern von Daten: Eine weitere Eigenschaft des DNS, eine Konsequenz aus den Sparmaßnahmen und den damals langsamen Netzwerken, und der hierarchischen Delegation ist, daß DNS-Antworten auf Nutzerseite zwischengespeichert werden können und sollen.

So lange Daten zwischengespeichert sind und ihr Haltbarkeitsdatum (Time To Live) noch nicht abgelaufen ist, müssen spätere Anfragen nicht erneut gestellt werden.

Gibt der Urheber von DNS-Daten an, daß „seine“ Daten zwei Wochen haltbar sind, kann man die Daten, die man erhalten hat, zwei Wochen lang verwenden und muß nicht erneut fragen. Das spart Zeit und (damals teuren) Netzwerkverkehr.

Im Zusammenwirken stellt das DNS damit einen *Directory-Service* dar[8], also einen Dienst, der *zu einem Namen* mehrere Daten abspeichert. Der Name wird dabei als (Haupt-)Suchschlüssel genutzt. Man bezeichnet solche Speichertypen auch als *Dictionary* oder etwas abstrakter *Assoziativen Speicher*.

Die wesentliche Eigenschaft des DNS ist damit: Man fragt das DNS, indem man in der Frage einen Domain-Namen nennt und bekommt als Antwort die Daten, die zu diesem Domain-Namen gespeichert sind.

2.4 Probleme des DNS

Leider ist das DNS alles andere als perfekt. Es entstand zu einer Zeit mit völlig anderen Randbedingungen und Anforderungen als man sie heute hat. Man konnte die Sicherheitsprobleme von heute damals nicht einmal erahnen und verfügte nicht über den Stand des Wissens und die Erfahrung von heute. Zudem waren die damaligen Ressourcen an Speicher, Rechenleistung, Übertragungskapazität so begrenzt, daß man versuchte, Probleme immer auf dem einfachsten und günstigsten Weg zu lösen. Weder war damals absehbar, welche weitreichende Entwicklung das Internet in 25 Jahren nehmen würde, noch der paradoxe Umstand, daß das Internet sich gerade in Bezug auf das DNS nicht weiterentwickeln und DNS auch nach 25 Jahren noch in Betrieb sein würde. Es war nicht vorhersehbar, daß das Internet heute in fast jedem Privathaushalt und für Dienste wie Telebanking verwenden, und daß man dafür immer noch das alte DNS einsetzen würde.

Aus heutiger Sicht leidet DNS unter einer Vielzahl von Einschränkungen, Problemen und Schwächen:

- DNS ist technisch veraltet.
- DNS ist unsicher und leicht fälschbar. Dies wird genauer im Kapitel 8 beleuchtet.

[8] Wer dabei an das Active Directory von Microsoft, oder generell das Lightweight Directory Access Protokoll LDAP denkt, auf dem das AD beruht, oder sogar dessen Vorläufer X.500, aber auch die im Unix-Bereich früher üblichen Dienste wie YP (Yellow Pages) oder NIS (Network Information Service) denkt, der liegt genau richtig. Auch das sind alles ähnliche Dienste, die gleiche oder verwandte Leistungen wie DNS erbringen. Sie unterscheiden sich aber in zwei wesentlichen Eigenschaften: Erstens sind diese Dienste jünger, moderner und leistungsfähiger als DNS. Zweitens sind sie für das LAN und nicht für das Internet gedacht und können es damit nicht ersetzen.

- DNS ist nicht rechtssicher.
- DNS ist nicht vertraulich – es gibt keine verschlüsselte Übertragung der Daten, und man kann am DNS-Verkehr leicht erkennen, welche Webseiten etwa jemand besucht.
- DNS ist nicht leicht um neue Datentypen zu erweitern. Möchte man neue Datentypen einführen, muß man diese nicht nur zentral registrieren lassen, sondern muß die Software der DNS-Server und der DNS-Clients aufwändig erweitern, was auf breiter Ebene nahezu unmöglich ist. Zwar sind manche DNS-Resolver in der Lage, auch unbekannte Datentypen abzufragen, zu speichern und weiterzugeben, aber dann kann die Datenkompression nicht verwendet werden, weil diese voraussetzt, daß die Software entsprechend erweitert wurde. Ohne Kompression stößt man jedoch schnell an die Datengrenze von 512 Byte.
- DNS-Pakete und damit die Anfragen und Antworten dürfen nicht größer als 512 Byte werden. Werden sie es doch, muß man als darunterliegendes Protokoll von UDP auf TCP wechseln, was oftmals durch Firewalls usw. verhindert wird. Manche Rechner haben aber so viele Hostnamen oder IP-Adressen, daß die Antwort auf eine DNS-Anfrage nicht mehr in diese 512 Byte paßt, was oft zu DNS-Störungen führt.
- Der Zeichensatz ist auf ASCII begrenzt und dafür für viele nichtenglischsprachige Länder mit Einschränkungen verbunden. (Siehe aber den Abschnitt zu IDN ab Seite 61)

Wenn DNS aber so viele Schwächen hat, warum hat man es dann nicht durch etwas besseres ersetzt? Beispielsweise würde das erheblich modernere (aber auch aufwendigere und viel langsamere) LDAP fast alle Schwächen von DNS beheben.

Dem stehen eigentlich drei Gründen entgegen:

Der erste Grund ist, daß es einen ungeheuren Aufwand bedeuten würde, DNS zu ersetzen. Man hat es zu lange versäumt, das DNS durch etwas neueres zu ersetzen, zumal die Schwächen von DNS erst in den letzten Jahren – etwa durch Cache-Poisoning – so richtig schmerzlich ans Licht kamen. Inzwischen gibt es über eine Milliarde Internet-Nutzer und damit Internet-taugliche Rechner in vermutlich ähnlicher Größenordnung. Wie sollte man die alle aktualisieren? Viele Rechner werden nicht mehr gewartet, sind auf uralten Softwareständen oder dürfen aus Gründen der Zertifizierung nicht aktualisiert werden. Freilich könnte man ein neues System einführen, das eine Schnittstelle zum alten DNS hätte. Viel wäre aber nicht gewonnen, wenn die meisten dann weiterhin über DNS arbeiteten. Der Aufwand, ein neues System zu etablieren, steht derzeit wirtschaftlich (noch) in keinem Verhältnis zu den Schäden, die durch die Schwächen von DNS entstehen, denn der Arbeitsaufwand und damit die Kosten wären gigantisch. Zumal man ein neues System erst einmal haben müßte.

Der zweite Grund ist politischer Natur. Die USA üben die Macht über den Wurzelknoten des DNS aus. Das hat sich so ergeben, weil DNS und Internet in

den USA entwickelt wurden. Europa hat das einfach verschlafen und sich auf die Rolle des späten Konsumenten beschränkt. Die Einführung eines neueren, besseren Systems würde diese zentrale Rolle der USA aber in Frage stellen. Deshalb hat man kein Interesse daran.

Der dritte Grund ist, daß es niemanden gibt, der dafür zuständig wäre, ein neueres System zu entwickeln und einzuführen. *Wer sollte das tun?* Die Politik hat das Internet jahrzehntelang ignoriert und bis vor kurzem nicht als kritische Infrastruktur wahrgenommen, sondern es jahrelang einfach den Universitäten und der Industrie überlassen. Die Universitäten sind nicht in der Position, nicht in der Lage (und wohl auch nicht befähigt), ein neues System zu entwickeln. Die Industrie hat auch kein Interesse daran, weil sich damit in der Anfangsphase kein Geld verdienen und kein Marktvorteil erzielen ließe. Die Dominanz von Windows ist so hoch, daß eine solche Umstellung ohne Microsoft nicht möglich wäre. Eine Umstellung unter Beteiligung von Microsoft dürfte aber ebenfalls am groben Geschäftsgebaren von Microsoft scheitern, wie vor einigen Jahren schon das gescheiterte Unterfangen gezeigt hat, einen einfachen Spam-Schutz über das DNS zu implementieren (Seite 61 ff.). Die IETF schließlich, die formal zwar dafür zuständig wäre, handelt letztlich auch nicht aus eigenem Antrieb und mit eigenen Mitteln, sondern nur auf Bedarf der Industrie und nur mit Mitteln der Industrie. Solange von da nichts kommt, macht die IETF auch nichts.

Deshalb stehen die Chancen derzeit schlecht, DNS durch ein moderneres System zu ersetzen. Man versucht stattdessen, durch ständige Ergänzungen, Erweiterungen, Workarounds und Kniffe, das bestehende DNS möglichst rückwärtskompatibel auszubauen, was man aber nur als Flickwerk bezeichnen kann.

3 Fragen, Irrtümer und Folklore

3.1 Häufig gestellte Fragen

3.1.1 Was macht das DNS?

Das DNS beantwortet Fragen, die man stellt.

Die Fragen entsprechen dabei einem ganz einfachen und strikten Grundschema. Sie bestehen nur aus zwei Angaben, nämlich dem Domain-Namen als Fully Qualified Domain Name (vgl. Frage 3.1.4), und dem Daten- bzw. Record-Typ, nach dem man fragt (vgl. Kapitel 4). Dabei kann der Record-Typ auch *any* lauten, was einfach bedeutet, daß man Records *aller* Typen haben möchte.

Die Antwort besteht entweder aus einer Fehlermeldung (Server antwortet nicht, keine Daten vorhanden, ...) oder der Liste der angefragten Records, der Angabe, von welchen Authoritative Servern (vgl. Seite 57) die Antwort ursprünglich stammt, und einige ergänzende, eigentlich nicht angefragte Records, etwa für die IP-Adressen dieser Nameserver.

3.1.2 Gibt es nur ein DNS oder viele?

Gute Frage.

Im Prinzip kann man beliebig viele Namensräume aufspannen. Weil das DNS hierarchisch aufgebaut ist und sich die Zuständigkeiten daraus ergeben, wer was an wen delegiert, und dies alles von der Wurzel des Hierarchiebaumes ausgeht, kann man einfach einen neuen DNS-Baum aufschlagen, indem man eine neue Wurzel anfängt. Technisch gesehen könnte jeder Internet-Benutzer sein eigenes DNS gründen und betreiben. (Wir kommen später in Abschnitt 5.2 darauf zurück, wie und warum das möglich ist.)

Vor einigen Jahren haben Leute das tatsächlich versucht. Sie fingen einfach an, eine eigene DNS-Wurzel zu gründen und Domainnamen (hauptsächlich die begehrten .com-Namen) die schon anderweitig vergeben waren, einfach noch einmal neu und an andere zu vergeben. Das Verblüffende daran ist, daß das *technisch* sogar funktioniert, denn das DNS ist ja nicht Bestandteil des Internet, sondern selbst nur ein gewöhnlicher Dienst, der darauf aufbaut.

Das Hindernis dabei liegt nicht darin, daß es technisch nicht funktionieren würde, sondern daß man davon nichts hat – außer juristischem Ärger. Weil es

ziemliche Verwirrung bringt und niemand einen Nutzen daraus hat, wenn unter einem Domainnamen in einem anderen DNS jemand anderes auftaucht, der sich darüber geärgert hat, daß ihm im „echten“ DNS jemand den Domainnamen weggeschnappt hat, fehlte es an der Akzeptanz, niemand wollte das benutzen. Es ist, also würde jemand an die Bevölkerung einfach neue Telefonbücher verteilen wollen, in denen unter „Elektro-Meier“ die Telefonnummer eines anderen steht, der auch gerne „Elektro-Meier“ heißen würde.

Dadurch handelt man sich rechtliche Probleme ein, denn die Nutzung von Domainnamen unterliegt dem Marken- und Namensrecht, und wenn jemand einen bekannten Namen hat und Inhaber der zugehörigen Domain ist, muß er es nicht hinnehmen, daß da jemand ein alternatives DNS aufmacht, in dem er die Namen an andere vergibt.

Das Unterfangen hat sich aber längst erledigt und ist nicht mehr relevant. Trotzdem sollte man das Prinzip verstanden haben, denn es kommt noch immer gelegentlich vor, daß böse Buben per Trojaner oder Bot-Software das DNS in den Rechnern ihrer Opfer verändern, um Angriffe wie Phishing und ähnliches vorzubereiten.

Man sollte verstanden haben, daß nichts und niemand einen zwingt, das DNS überhaupt zu verwenden. Gerade in sicherheitsrelevanten Bereichen betreibt man Rechner oder einzelne Dienste oft ganz ohne DNS. Man kann es bleiben lassen, wenn man will. Aber die Teilnahme am einheitlichen, großen, weltumspannenden DNS hat eben Vorteile. Und die hat man nicht, wenn man seine eigene Suppe kocht.

Außerdem gehört es zum Wissen eines IT-Sicherheitsfachmannes, wie man beispielsweise für ein besonders abgesichertes Firmennetz ein völlig von der Außenwelt abgetrenntes und eigenständiges DNS aufbaut. E-Mail und World Wide Web sind dann nur noch über Proxies und Relays zu erreichen, was ein gewünschter Effekt sein kann. Die Anbindung eines Firmennetzes an das weltweite DNS ist zwar Standard und allgemeinüblich, bringt aber auch gewisse Gefahren mit sich. Das würde hier aber zu weit führen.

Passieren könnte es allerdings, daß das einheitliche DNS an politischen Streitigkeiten zerbricht und in mehrere Einzelteile gespalten wird (vgl. Frage 3.1.7).

3.1.3 Was ist ein »vollqualifizierter Domainname«?

Im beschlossenen, aber noch nicht in Kraft getretenen Gesetz über Kinderpornographiesperren, auch bekannt als das Gesetz zur Bekämpfung der Kinderpornographie in Kommunikationsnetzen oder „Zugangerschwerungsgesetz“ (ZugErschwG) findet sich gleich in § 1 Absatz 1 Satz 1 die Aussage

> *Das Bundeskriminalamt führt eine Liste über vollqualifizierte Domainnamen, Internetprotokoll-Adressen und Zieladressen von Telemedienangeboten, die Kinderpornographie . . .*

Weil eine Begriffsdefinition dort fehlt und der Begriff »Vollqualifizierter Domainname« zum ersten Mal in einem deutschen Gesetz auftaucht, und auch sonst bisher in dieser deutschen Form nicht geläufig war, stellt sich die Frage, was das ist.

Es ist genaugenommen ein doppelter Fehler.

Es ist ein Übersetzungsfehler und ein inhaltlicher Fehler. Den inhaltlichen Fehler werden wir in der nächsten Frage betrachten.

»Vollqualifizierter Domainname« ist eine unglückliche deutsche Übersetzung des englischen Begriffs des Fully Qualified Domain Names (FQDN). Das englische Wort qualify ist nicht bedeutungsgleich mit dem deutschen Wort qualifizieren, das hat man mit der Beißzange übersetzt. Zwar heißt das deutsche Wort »qualifizieren« zumindest laut eines einzigen meiner Wörterbücher auch kennzeichnen, aber mehr im Sinne von einordnen, und diese Bedeutung ist so selten, daß sie im Sprachgebrauch nicht vorkommt und in vielen Wörterbüchern auch nicht. Meine aktuelle Version des Duden nennt diese Bedeutung auch nicht (mehr), und die Bedeutungs- und Synonymwörterbücher kennen sie auch nicht. Qualifizieren bedeutet ausbilden, befähigen, heranziehen, Befähigung nachweisen. Es gibt qualifizierende Berufsausbildungen und qualifiziertes Personal, man kann sich für das Finale im 100-Meter-Lauf qualifizieren, und man kann unqualifizierte Bemerkungen ablassen. Aber Domainnamen kann auf deutsch nicht qualifizieren.

Wie wir im nächsten Abschnitt zum FQDN sehen werde, wäre eine dessen Bedeutung treffendere deutsche Übersetzung ein *Absoluter* oder *Vollständiger* Domainname als Gegensatz zu einem relativen Domainnamen gewesen. Aber wie so viele falsche Übersetzungen aus dem Englischen wird der Begriff sich als Brachial-Anglizismus[1] wohl etablieren. Allein schon über das Auftauchen in einem Gesetz wird er zwangsläufig Eingang in die Juristensprache und damit in Literatur und Rechtsprechung finden. Richtiger wird er dadurch nicht.

3.1.4 Was ist ein Fully Qualified Domain Name (FQDN)?

In der vorhergehenden Frage haben wir gesehen, daß der »vollqualifizierte Domainname« im Zugangserschwerungsgesetz keine gute Übersetzung des englischen »Fully Qualified Domain Name« ist. Aber selbst wenn wir annehmen, daß es eine exakte Übersetzung wäre und wir den englischen und im DNS gebräuchlichen und wichtigen Begriff des FQDN einsetzen könnten, paßt die Bedeutung nicht in das Gesetz. Wie aus der Intention des Gesetzes und den anderen Elementen der vom BKA geführten Liste hervorgeht, wollte der Gesetzgeber, daß die zu sperrenden Rechner (Webserver) aufgelistet werden. Das besagt der Begriff FQDN aber nicht.

[1] Am Ende des Tages wird jeder den Begriff benutzen ohne zu realisieren, daß er falsch ist, und niemand wir mehr die Entstehung erinnern.

Was ist ein »Fully Qualified Domain Name«?

Wir haben gesagt, daß das DNS hierarchisch organisiert ist, und alle Daten nach einem einzigen Schlüssel, nämlich dem Hostnamen abgelegt sind. Dieser Name besteht aus mehreren Bestandteilen, die durch Punkte verbunden sind, beispielsweise

```
erdbeere.danisch.de
```

Wir sehen, daß die oberste Hierarchieebene `de`, die für Deutschland *rechts* steht, da ist. Da fehlt nichts mehr um weltweit eindeutig zu sein. Das ist ein *kompletter Domainname*. Ein FQDN. Ein solcher FQDN bezeichnet eine eindeutige Stelle im Hierarchiebaum und damit den kompletten Pfad von der Wurzel bis zu dieser Stelle.

Das bedeutet aber nicht, daß ein FQDN drei Teile haben muß. Auch

```
danisch.de
```

und die Domain für Deutschland

```
de
```

ja sogar die eigentlich noch oberhalb von Deutschland liegende Domain mit dem *leeren Namen* sind alle FQDN. Weil sie weltweit eindeutig sind, weil da auf deren rechter Seite nichts mehr fehlt.

Es gibt im Gegensatz dazu auch *relative* Domainnamen. Und die haben viele von uns schon einmal benutzt, ohne es zu merken. Sind wir in einem Firmen-LAN, dann können wir lokale Rechner wie Asterix, Erdbeere oder Printserver direkt ansprechen, ohne dabei den vollen Domainnamen angeben zu müssen. Warum?

Normalerweise sind an Rechnern in LANs immer ein oder mehrere lokale Domains als Default-Domain eingestellt. Erkennt der Rechner, daß ein aufzulösender Name unvollständig ist (was aber nicht leicht ist, siehe dazu die nächste Frage), dann fügt er diese Default-Domain hinten an um den Namen aufzulösen. Wir geben beispielsweise im Webbrowser ein, daß wir auf den Rechner Erdbeere wollen, und das DNS macht zur Auflösung der IP-Adresse daraus beispielsweise `erdbeere.danisch.de`. Das heißt, `erdbeere` war kein vollständiger, absoluter Domainname, sondern ein unvollständiger Name, relativ zur lokalen Netzwerkdomain. `erdbeere` ist kein FQDN. `erdbeere.danisch.de` ist dagegen einer (man beachte dazu aber die nächste Frage um zu sehen, daß es auch ein relativer sein könnte).

Um das mal ganz salopp auszudrücken: FQDN bedeutet, daß ein Domainname auf der *rechten* Seite vollständig ist. Es bedeutet nicht, daß er auf der *linken* Seite vollständig ist. FQDN bedeutet auch nicht, daß darunter ein Rechner, Webserver oder überhaupt irgendetwas gespeichert ist. FQDN bedeutet, daß der Name eindeutig ist, aber es bedeutet nicht, daß er etwas bedeutet.

Genau deshalb wird im Englischen manchmal auch der Begriff des FQHN, des Fully Qualified Host Name verwendet. Der wäre für das Gesetz viel besser gewesen, aber auch nicht ganz passend, weil ein FQHN auch eine IP-Adresse sein kann, und die betrachtet das Gesetz ja noch einmal separat.

Nun erkennt man, daß der Gesetzentwurf für die Kinderpornographiesperre an dieser Stelle mißlungen ist, wenn FQDN in die Sperrliste aufgenommen werden. Zwar ist das insofern richtig, als man zu sperrende Domainnamen auf jeden Falls als FQDN (und nicht nur als relative Namen) angeben müßte. Aber es besagt nichts darüber, ob nur bestimmte Rechner oder ganze Domains gesperrt werden können. Und gerade das hätte in einem solchen Gesetz geregelt werden müssen. Vielleicht dachte man, das getan zu haben. Vielleicht hat man so weit gar nicht gedacht.

Diese anzuhängende Default-Domain (unter Windows auch als DNS-Suffix bezeichnet) kann man am Rechner übrigens fest einstellen. Meist wird sie aber vom DHCP-Server bei der Anmeldung am Netzwerk vergeben. Unter Linux steht das dann in `/etc/resolv.conf`, unter Windows kann man sich das mit dem Befehl `ipconfig` anzeigen lassen. Schauen Sie mal rein!

3.1.5 Wie unterscheidet man einen FQDN von einem relativen Domainnamen?

Woran erkennt denn ein Rechner, ob ein Domainname relativ oder absolut ist? Wenn ich also Rechner wie `aa.bb` oder `www.danisch.de` anspreche, beispielsweise mit dem Webbrowser, woher weiß dann der Rechner, ob er da noch seine lokale Domain anhängen soll oder nicht?

Zwei Gedanken drängen sich auf:

- Man könnte alle Namen ohne Punkt (also die, die aus nur einem Namensteil bestehen) als relativ und alle mit Punkt (also die, die aus mindestens zwei Namensteilen bestehen) als absolut ansehen. Gute Idee, aber leider nicht richtig. Obwohl es gelegentlich so implementiert wird und es unter Unix aus Not sogar einen Parameter gibt, mit dem man das einstellen kann[2]. Auch `www.danisch.de` kann ein relativer Domainname sein, an den hinten noch die lokale Domain angehängt wird.

  Würde man Namen aus nur einer Komponente immer als relativ ansehen, könnte man niemals die Informationen für die Länderdomains wie `de` oder die generic domains wie `com`, `edu` usw. abfragen, weil der Rechner dann immer versuchen würde, diese um die eigenen Domains zu ergänzen.
- Einfach ausprobieren: Zur Sicherheit erst relativ, dann absolut versuchen. Zwar macht man es in der Realität so, es ist aber leider auch falsch. Und führt dazu, daß wenn es einen Namen sowohl relativ, als auch absolut gibt, ich den absoluten nicht mehr erreichen kann.

[2] Siehe man 5 resolver, options ndots

In der Realität kann man relative und absolute Domainnamen genau betrachtet nicht unterscheiden. Der Grund dafür ist schiere Schlamperei.

Eigentlich sieht das DNS eine genaue Unterscheidung vor, denn absolute Domainnamen sind quasi *relativ zur obersten leeren (!) Hierarchiewurzel*. Und deshalb müssen sie hinten immer mit einem Punkt enden. (Präzise gesagt, muß hinter dem Punkt noch ein leerer Name folgen, also nichts.) Das heißt daß

- `www.danisch.de` entgegen obiger Aussage eigentlich nur ein relativer und damit **kein** FQDN ist.
- `www.danisch.de.` ist ein absoluter Name und damit ein FQDN. **Weil es mit einem Punkt endet.**

Das ist so ähnlich wie mit Pfaden im Dateisystem: `/usr/bin/nslookup` ist ein absoluter, `bin/nslookup` ein relativer Pfad. Nur daß hier die Wurzel *links* steht.

Und tatsächlich findet man dies bei einigen wenigen Anwendungen auch so. Beim Nameserver bind wird rigoros alles relativiert und mit einer Domain ergänzt, was keinen Punkt am Ende hat, den man leicht vergessen kann. Auch in den Regelsätzen des Mail-Programms sendmail findet sich die Schreibweise mit dem Punkt am Ende. Das ist eigentlich korrekt.

Dann aber kamen die ersten Anwendungen, bei denen der Benutzer manuell den Namen eingibt, und schon gewöhnte man sich an, den Punkt am Ende wegzulassen – weil es ja augenscheinlich funktionierte. Daß da mehrere DNS-Anfragen vorausgingen, merkte man ja nicht. Und dann kamen das World Wide Web und die Ausbreitung des Internet, und man kannte nur noch die Variante ohne Punkt am Ende. Manche Browser akzeptieren es nicht einmal, wenn man den Punkt dranschreibt. Die meisten Anwender würden es auch nicht verstehen. Deshalb ist diese Schreibweise in der Praxis nahezu bedeutungslos geworden und fast untergegangen. Und deshalb gibt es eigentlich in der Praxis außer in den Nameservern selbst gar keine FQDN mehr.

Man bezeichnet aber – nachlässig – einen Namen dann als FQDN, wenn rechts (außer dem Punkt) nichts mehr drankommt. Die Schwierigkeiten, relative und absolute Domainnamen auseinanderzuhalten ergeben sich eigentlich nur aus einer gewissen Schlamperei.

Woran man den nachlässigen Umgang mit dem Punkt am Ende bemerken kann, sehen wir am Beispiel der NS-Records (Abschnitt 4.10).

3.1.6 Was sind Top- und Second-Level-Domains?

Top-Level-Domains sind die Hierarchieebene ganz oben (unter der leeren Wurzel, vgl. vorhergehender Abschnitt) bzw. ganz rechts in einem vollständigen Domainnamen. Die Top-Level-Domains sind

- Die generic TLD sind unter anderem .com, .gov, .net, .org, .edu, .arpa, .biz, .info, .mil, es gibt noch einige weitere.
- Die ccTLD sind die zweistelligen Countrycodes des ISO 3166, also beispielsweise .de, .at, .ch für Deutschland, Österreich, Schweiz.
- Die Verwaltungsorganisation des Internet ICANN will in Kürze weitere TLDs an Bewerber vergeben. Vermutlich bekommen wir dann auch TLDs wie .google oder .ebay.

Eine Second-Level-Domain ist die, die direkt unterhalb der TLD angesiedelt ist und an Firmen vergeben wird, also beispielsweise danisch.de oder example.com.

Allerdings gibt es Länder, die unterhalb ihrer eigenen TLD noch eine organisatorische Ebene analog den generic TLDs eingeführt haben:

- Großbritannien und Neuseeland haben unter ihrer TLD eine solche Ebene eingezogen, sie aber von drei auf zwei Buchstaben gekürzt, statt com wird also nur co geschrieben usw.: www.google.co.uk und www.google.co.nz
- Australien verwendet eine dreibuchstabige Zwischenebene: www.google.com.au

Für diese Länder müßte man die individuell vergebenen Domains eigentlich als Third-Level-Domains bezeichnen. Das ist aber umständlich und mißverständlich, denn so bezeichnet man eigentlich eine vom Inhaber einer Second-Level-Domain intern vergebene Subdomain. Deshalb wird in diesen Ländern die individuell vergebene Domain meistens auch als Second-Level-Domain (eben eine Stufe unter der Landesverwaltung) bezeichnet, obwohl sie formal an dritter Stelle steht.

3.1.7 Wer verwaltet das DNS?

Obwohl das DNS dezentrale und verteilt ist, hat es doch eine Wurzel, die verwaltet werden muß. Dafür ist die ICANN zuständig, eine amerikanische Firma mit dem Segen der US-Regierung. Auch für die generic Top Level Domains ist die ICANN zuständig, siehe auch www.icann.org.

Die Länderdomains nach ISO 3166 wie de für Deutschland werden jedoch an die jeweiligen Registrierstellen der Länder delegiert. In Deutschland ist das die DENIC (www.denic.de).

Die Verwaltung der Wurzel durch die USA führt derzeit zunehmend zu Verstimmungen zwischen manchen Ländern, vor allem mit China und Russland. Es könnte mittelfristig zu einem Auseinanderbrechen des bisher einheitlichen DNS führen.

3.2 Typische Irrtümer und Legenden

3.2.1 Das Internet funktioniert nur mit dem DNS

Falsch. Das kann schon deshalb nicht stimmen, weil das Internet vor dem DNS da war. Das Internet funktioniert auch ohne DNS. Genauer gesagt, das Internet hat mit dem DNS überhaupt nichts zu tun und funktioniert mit numerischen Internet-Adressen.

Aber viele, nahezu alle Telekommunikationsdienste, die auf dem Internet beruhen, nutzen auch das DNS. Die bekanntesten sind das World Wide Web und E-Mail. (Dazu unten mehr.) Weil viele Leute aber den Fehler machen, diese Dienste mit dem Internet gleichzusetzen, nehmen sie häufig an, daß das Internet das DNS benötige um zu funktionieren.

3.2.2 Das DNS ist das Telefonbuch des Internet

Nicht ganz falsch. Aber mehr falsch als richtig.

Wie im Unterkapitel 2.2 beschrieben hat man zunächst versucht, die Namensauflösung mit Listen zu lösen, die man an alle Teilnehmer verteilte und die dem Prinzip des Telefonbuchs entsprechen. Weil man aber sehr schnell gemerkt hat, daß die Datenorganisation nach Art eines Telefonbuches das Problem nicht brauchbar lösen kann, hat man das DNS erfunden, das die frühere Telefonbuch-Methode ersetzte.

Damit hat das DNS schon gewisse Ähnlichkeit mit einem Telefonbuch, denn es hat ja die Stelle des früheren Telefonbuchs eingenommen. Man gibt einen Namen an und man bekommt eine Nummer.

Es ist aber eben kein Telefonbuch. Es ist ja gerade das Hauptkonstruktionsmerkmal des DNS, grundlegend anders als ein Telefonbuch zu sein, denn das Telefonbuch-Prinzip konnte das Problem nicht lösen. Man hat das DNS ja gerade eingeführt, um das Prinzip Telefonbuch abzulösen. Außerdem kann das DNS viel mehr als ein Telefonbuch.

Das DNS beantwortet zwar ähnliche Fragen wie ein Telefonbuch, aber es funktioniert völlig anders. Der Vergleich mit einem Telefonbuch taugt daher nur, um jemandem in erster Näherung zu erklären, wozu das DNS gut ist. Sobald es um Inhalt oder Funktionsweise des DNS geht, etwa wenn in der Politik über Internetsperren oder andere Eingriffe in die Funktion des DNS diskutiert wird, führt der Vergleich mit dem Telefonbuch unweigerlich in die Irre. Das DNS ist keine zentrale Sammlung wie ein Telefonbuch, sondern so etwas wie eine verteilte Telefonauskunft, bei der jeder Haushalt die Telefonauskunft für seine Familienmitglieder selbst betreibt und Anfragen beantwortet, wie er will.

Das DNS für die Diskussion um Internetsperren und andere Eingriffe wie ein Telefonbuch zu betrachten ist ungefähr so, als würde man ein Auto zur Reparatur zum Tierarzt bringen, weil es doch genauso wie sein Vorgänger Pferd ein Transportmittel ist und vier Beine hat.

Und wenn man ganz pingelig ist, unterscheidet sich das DNS auch in seiner Assoziationsstruktur von einem Telefonbuch. In einem Telefonbuch kann es beliebig viele Leute mit dem gleichen Namen geben, aber jeder bekommt normalerweise nur eine Telefonnummer. Im DNS dagegen kann es jeden Namen nur ein einziges Mal geben, aber dafür können sie auch mehrere IP-Adressen (und andere Daten) tragen. Telefonbuch heißt, viele Namen mit je einer Nummer, DNS heißt ein Name mit vielen Nummern.

3.2.3 Das DNS erleichtert das „Merken“

Man liest oft, das DNS wäre dazu da, daß man sich die Internet-Adressen nicht merken müßte, quasi eine Gedächtnisstütze, um dem Menschen das Websurfen zu vereinfachen.

Das ist zwar auch ein Nebeneffekt des DNS, aber so kurz gegriffen und so unvollständig, daß man es als falsch ansehen muß. Es ist, als würde man ein Auto als ein Ding definieren, das hupt und Scheiben wischt. Das tut es zwar, aber das geht am Zweck und Wesen des Autos weit vorbei.

- Es geht nicht darum, sich die Information *als Mensch* zu merken. Zwar beinhaltet das DNS auch ein „Merken“ im technischen Sinne (Caching), aber das führen Computer aus und die brauchen keine Gedächtnisstützen.
- Das „Merken“ ist nicht das Problem, man darf das nicht vermenschlichen und nur aus seiner Lebenserfahrung sehen.

  Das erste Problem ist, daß man etwas überhaupt erst einmal in Erfahrung bringen muß, bevor man es sich merken kann. Das schnell, effizient und aktuell zu tun ist Aufgabe des DNS, nicht das Merken zu unterstützen. Oder anders gesagt: DNS löst nicht das Problem, wie man sich etwas merkt, sondern wie man damit anfängt und an die Informationen kommt, die man sich merken will.

  Das zweite Problem ist, daß man selbst dann, wenn man sich etwas gemerkt hat, nicht weiß, ob die Information noch aktuell ist und noch stimmt. Sich etwas zu merken ist gar nicht so nützlich, wie es erscheint, denn sich etwas zu merken, von dem man nicht weiß, ob es noch aktuell ist, ist kaum nützlicher als es sich nicht zu merken. Deshalb löst das DNS auch das Problem, von gemerktem Wissen zu wissen, ob es noch aktuell ist. Oder anders gesagt: DNS löst nicht das Problem des Merkens, sondern das, wie man damit wieder aufhört.

> Diese beiden Probleme, das effiziente Erlangen der Information und das geordnete Vergessen und Aktualisieren, sind viel, viel schwieriger als das Merken der Information an sich. Das Merken ist trivial.

Auch diese Legende über das DNS stimmt so also (so) nicht.

3.2.4 Hostnamen sind immer dreiteilig und beginnen mit dem Dienstnamen

Oft hört man aus dem Gespräch und manchmal liest man sogar, daß die Auffassung vorherrscht, daß Host-Namen oder die Namen von Webservern, FTP-Servern und ähnlichem immer dreiteilig wären und mit dem Dienstnamen anfangen, also wie `www.danisch.de` oder `ftp.danisch.de` aussehen.

Das stimmt nicht. Das sieht nur so aus, weil irgendwer damit angefangen hat und alle es nachgemacht haben. Es ist eine Sitte, mehr nicht. Das DNS beziehungsweise der DNS-Server nennen auf Anfrage zu einem Domain-Namen eine IP-Adresse. Wofür und für welche Anwendung man die verwendet, und wie man die Anwendung nennt, ist dem DNS-Server egal.

Außer bei einigen Spezialanwendungen und bei SRV-Records (Abschnitt 4.7) gibt es keine Vorgaben, wie ein Hostname auszusehen hat. Es gibt lediglich die Vorgabe, daß jede einzelne Namenskomponenten (die Namensstücke zwischen den Punkten) maximal 63 Zeichen und der gesamte Name maximal 255 Zeichen lang sein darf. Auf jeden beliebigen Domainnamen, den man so bilden kann, kann man auch IP-Adressen legen und ihn für Dienste wie World Wide Web, FTP, E-Mail verwenden. Welchen Dienst man darauf legt, und ob der Name auf www oder ftp oder etwas anderes lautet, ist dem DNS völlig egal. Namenskomponenten wie www und ftp haben keinerlei besondere Bedeutung. Wenn man eine Domain erst einmal hat, kann man diese weitgehend unterteilen. Hostnamen wie

- a.b.c.d.e.f.g.h.i.j.danisch.de
- server.oben.muenchen.deutschland.erde.milchstrasse.danisch.de

würden ganz genauso funktionieren. Aber es geht auch kürzer, man kann auf seinen Domainnamen (z. B. danisch.de) ebenfalls IP-Adressen legen und das für Webserver oder andere Dienste verwenden.

Übrigens haben Großbritannien, Australien und Neuseeland schon zweikomponentige Länderdomains (z. B. .co.uk, .co.nz, .com.au), dort sind die Namen also alle eine Komponente länger als bei uns.

Besonders ärgerlich ist aber, daß nicht nur manche Laien, sondern auch manche „Profis“ (oder die sich dafür halten) diesem Irrtum unterliegen. Das führt zu allerlei Verwirrung und Sicherheitsproblemen. Häufig will oder muß man die

Daten auf Webseiten von verschiedenen Servern zusammensuchen. Dämlicherweise beantragen die Macher der Webseiten dafür häufig neue Second-Level-Domains, anstatt ihre normale Domain zu verwenden und weiter zu unterteilen. Wenn also beispielsweise eine Firma XYZ die domain xyz.com hat und einen zweiten Server für Videosequenzen oder Javascript verwenden will, kommen leider viele auf die dumme Idee, dafür eine neue Domain wie xyz-video.com zu beantragen oder sowas in der Art. Das führt aber dazu, daß der Benutzer sich nicht so sicher sein kann, ob xyz-video.com wirklich zur Firma XYZ gehört oder nicht doch Teil eines Angriffs ist und von bösen Buben beantragt wurde. Es wäre sinnvoller, dafür Unterdomains zu bilden, also den Server beispielsweise server.video.xyz.com zu nennen, was für den Benutzer eindeutig wäre und außerdem Geld sparen würde. Aber nicht jede Internet-Agentur und nicht jeder Web-Designer verstehen ihr Handwerk.

3.2.5 DNS kann Sperr- und Fehlermeldungen ausgeben

Im Zusammenhang mit der Sperrung von Kinderpornographie wurde der Wunsch geäußert, daß die DNS-Server doch bei Anfragen entsprechende Sperr- oder Fehlermeldung (gar in Form von Webseiten) ausgeben sollten.

Das ist nicht möglich. Wenn man einen DNS-Server nach der IP-Adresse zu einem Namen fragt, gibt es nur vier Möglichkeiten:

- Der Server antwortet mit IP-Adressen.
- Der Server delegiert und sagt, man soll einen anderen Server fragen.
- Der Server sagt, daß er zu einem Namen keine Daten hat. Entweder, weil er zuständig ist und deshalb weiß, daß die angefragten Daten nicht existieren. Oder weil er nicht zuständig ist und deshalb zu dem angefragten Namen nichts weiß. Oder weil er zwar von seinem Administrator so konfiguriert wurde, daß er selbst suchen und fragen geht, wenn jemand ihn etwas fragt, wofür er nicht selbst zuständig ist, aber andere Server ihm gesagt haben, daß die Daten nicht existieren.
- Der Server antwortet gar nicht.

Zwar können aus dem DNS auch Fehlermeldungen kommen, aber die kommen nicht vom Server, sondern werden vom Rechner des Fragenden selbst erzeugt. Das DNS kann also nicht selbst solche Sperrmeldungen oder „Stopp-Seiten“ ausgeben.

3.2.6 Internetnutzer müssen den DNS-Server des Providers verwenden

In den verschiedenen Diskussionen um Internetsperren wurde häufig die Auffassung geäußert, daß der „normale“ Internet-Nutzer stets den DNS-Server

seines Internet-Providers nutzt und Eingriffe in diesen Server deshalb auf alle Kunden wirken.

Es kursierten dann geheimnisvolle Anleitungen auf Webseiten oder gar als Youtube-Video, wie man den Rechner auf andere DNS-Server umstellt und so den manipulierten Provider-DNS-Server umgeht. Manche sagten, daß man mit solchen Methoden die Sperren umgehen könnte, andere taten das als Hacker-Tricks besonders versierter Internetnutzer ab, manche hielten das für schon leicht kriminelles Hackertum, weil die Anleitung dazu vom Chaos Computer Club kam, und einige verstiegen sich sogar dazu, diese Umgehung gleich als strafbare Unterstützung von Kinderpornographie zu geißeln.

Alles Quatsch und dummes Zeug.

Die Nutzung eines vom Provider bereitgestellten DNS-Servers ist technisch gesehen eigentlich die Ausnahme, nicht die Regel.

Eine „normale“ Internet-Anbindung ist nämlich nicht die Einwahl per Modem, ISDN, DSL, UMTS usw., sondern eine Standleitung mit fest zugewiesenen IP-Adressen. DNS ist eigentlich das alleinige Problem des Kunden. Und sofern der Kunde ein eigenes Netzwerk betreibt (etwa weil er eine Firma oder Universität oder so etwas ist), betreibt er meist auch einen eigenen DNS-Resolver, der ihm alle DNS-Anfragen auflöst ohne jemals auf den DNS-Server des Providers zuzugreifen.

Zwar bieten die Provider seit einiger Zeit diesen Firmenkunden auch an, den DNS-Server des Providers zu verwenden, das ist aber eine freiwillige Dienstleistung, ein Service den man nutzen kann, aber nicht muß.

Wer ein Firmennetzwerk an das Internet anbindet, der sollte sogar tunlichst darauf achten, den Provider-DNS-Server *nicht* zu verwenden, aus Sicherheitsgründen. Ich habe rund 10 Jahre lang Forschungseinrichtungen, Firmenkunden, Behörden und dergleichen an das Internet angeschlossen und dabei stets eigenständige DNS-Resolver aufgebaut.

Erst mit dem Aufkommen der Einwahl-Provider für den Klein- und Privatkundenbereich und der Einführung des Protokolls PPP für Modem, ISDN, DSL, UMTS usw. und den Beschränkungen des Windows-Betriebssystems kam es auf, daß so viele Internet-Nutzer den vom Provider angebotenen DNS-Server nutzen. PPP nennt diesen bei der Einwahl und der Rechner des Nutzers konfiguriert sich automatisch so, den angebotenen Server zu verwenden.

Das muß man aber nicht. Man kann durchaus, wie in den diversen Anleitungen des CCC oder auf Youtube zu sehen, einen beliebigen anderen DNS-Server verwenden. Oder man verwendet überhaupt keinen externen DNS-Resolver, sondern setzt einen eigenen auf, was dank OpenSource und der Leistungsfähigkeit heutiger Rechner ohne weiteres selbst auf Taschencomputern möglich ist. Jede ordentliche Linux-Distribution bringt solche Programme mit, deren Installation kaum mehr als ein Fingerschnippen beansprucht.

Die häufig gehörten Aussagen, daß das Umgehen der DNS-Server des Providers etwa durch die Umstellung auf einen anderen DNS-Server nur Machenschaften versierter Benutzer oder gar kriminelle Akte wären, sind reiner Blödsinn.

3.2.7 Mit DNS-Sperren kann man Webseiten sperren

Stimmt nicht.

Gelegentlich wird die Auffassung vertreten, der Zugriff auf Webseiten laufe durch den DNS-Server hindurch, als wäre er quasi so etwas wie ein Web-Proxy oder ein zentraler Dreh- und Angelpunkt des World Wide Web.

Der Zugriff eines Webbrowsers auf eine Webseite erfolgt in zwei Schritten:

1. Falls der Teil des Hostnamens im URL[3] ein Domain-Name und nicht schon eine IP-Adresse ist, führt der Rechner eine Namensauflösung durch, um die IP-Adresse in Erfahrung zu bringen.
2. Der Browser greift über das HTTP-Protokoll auf die Seite zu.

Wie man leicht sieht, können sich DNS-Sperren – wenn überhaupt – nur auf den ersten Schritt auswirken, weil der zweite Schritt, der eigentliche Zugriff, mit DNS nichts zu tun hat.

Und selbst auf den ersten Schritt haben die DNS-Sperren nur unter sehr engen Randbedingungen Auswirkung:

- Es muß überhaupt eine Namensauflösung stattfinden. Enthält der URL im Host-Teil bereits eine IP-Adresse, findet die Namensauflösung gar nicht erst statt. Ein probates Mittel um auf einfache Weise Sperren zu umgehen (vgl. aber den nächsten Abschnitt 3.2.8).
- Nicht jede Namensauflösung erfolgt über DNS, es gibt mehrere Methoden. Die einfachste Methode ist, den Namen und die zugehörige IP-Adresse (sofern man sie aus anderen Quellen kennt) unter Windows, Unix, Linux in die hosts-Datei zu schreiben. Damit kommt es zu einer Namensauflösung ohne daß DNS überhaupt zum Einsatz käme. Damit bleiben DNS-Sperren wirkungslos.
- Wie im vorigen Abschnitt 3.2.6 beschrieben, besteht für den Nutzer kein Zwang, den DNS-Server des Providers zu benutzen. Und in vielen Fällen ist es sogar der Normalfall, das nicht zu tun. Auf solche Nutzer wirkt sich eine DNS-Sperre im Provider-DNS-Server auch nicht aus.

[3] Vereinfacht gesagt das, was zwischen dem http:// und dem ersten / kommt, beispielsweise www.danisch.de in http://www.danisch.de/index.html oder das 1.2.3.4 in http://1.2.3.4/index.html

Also haben DNS-Sperren nur unter gewissen Voraussetzungen Auswirkung auf den Webseiten-Zugriff. Das mag zwar in manchen Fällen durchaus dazu führen, daß der klassische Privatnutzer tatsächlich nicht mehr normal und ohne weiteres auf die Webseite kommt. Aber der Begriff „sperren“ heißt nicht, daß man nur ein paar Nutzer abhält und die auch nur, solange sie sich das gefallen lassen. Der Begriff „Sperre“ ist für solche Teilstörungen nicht gerechtfertigt.

3.2.8 Internet-Dienste funktionieren auch direkt mit Internet-Adressen

Auf Seiten der Gegner von DNS-Internetsperren hört man gelegentlich, daß man alle Dienste auch direkt mit IP-Adressen betreiben kann, beispielsweise eine IP-Adresse direkt in einen URL schreiben kann.

Zwar im Prinzip, aber nicht ganz richtig.

Der Domainname wird nicht immer nur zum Auflösen der IP-Adresse verwendet. Manchmal hat er noch weitere Funktionen, für die er gebraucht wird:

- Sogenannte „Virtuelle Webserver“ sind ein Weg, um Geld zu sparen und mehrere logische Webserver auf einer einzigen Maschine laufen zu lassen. Weil der Browser in seiner Anfrage nicht nur den hinteren Teil des URLs mit dem Pfad, sondern auch den angefragten Hostnamen selbst mit angibt, kann der Webserver, der auf der Maschine läuft, erkennen, mit welcher Webseite der Benutzer eigentlich kommunizieren wollte. So kann ein Webserver, der auf nur einer Maschine mit nur einer IP-Adresse so tun, als wäre er ein ganzer Haufen verschiedener Webserver. Obwohl alle die gleiche IP-Adresse haben. Man nennt dies „virtuelle Webserver“, weil es so aussieht, als stünden da ganz viele, obwohl nur einer dasteht. Manche Webserver führen mehrere tausend dieser virtuellen Webserver auf einer Maschine mit nur einer IP-Adresse.

  Gibt man aber im URL direkt die IP-Adresse an, dann funktioniert das nicht mehr, weil der Webbrowser und der Webserver nicht mehr erkennen können, welchen der virtuellen Webserver der Benutzer ansprechen wollte.

  Es funktioniert nur dann, wenn auf dem Rechner (bzw. dieser IP-Adresse) nur ein einziger Webserver läuft, so daß es nicht zu Mehrdeutigkeiten kommen kann.

- Selbst wenn der Zugriff über eine IP-Adresse funktioniert, weil auf der Maschine unter dieser IP-Adresse nur ein Server läuft und die Anfrage nicht mehrdeutig ist, kann es zu Problemen kommen. Denn auch gewissen Server-Software (z. B. für Foren oder Blogs) verwendet den Haupt-Domainnamen des Servers, und auch andere eingebundene Daten wie Bilder, Style-Sheets, JavaScript-Dateien oder der Verweis auf

andere Seiten auf diesem Server können auf URLs mit dem regulären Hostnamen statt einer IP-Adresse basieren. Damit könnte der Browser zwar die erste HTML-Seite holen, aber könnte schon deren weitere Bestandteile nicht mehr holen.

- Manchmal verwendet man auch HTTPS, also HTTP über SSL. Dabei wird das Zertifikat in der Regel nicht auf die IP-Adresse, sondern auf den Domainnamen des Servers ausgestellt. Spricht man den Server aber über seine IP-Adresse an, kann der Browser nicht mehr erkennen, ob das gelieferte Zertifikat auch zu seiner Anfrage paßt.
- Bei E-Mail ist der Teil rechts des '@' der Domainname, der für die Zustellung verwendet wird. Zwar unterstützen die meisten Mailsysteme es auch, daß man dort eine IP-Adresse in eckigen Klammern angibt und liefern die E-Mail dorthin aus. Aber auch bei E-Mail ist es so, daß Server häufig virtuell für mehrere Domains zuständig sind und der Server dann an einer e-Mail-Adresse wie `info@[1.2.3.4]` nicht mehr erkennen könnte, welche der diversen Domains und damit welche Mailbox gemeint ist.

Die Liste ließe sich weiterführen und um ähnliche Probleme bei anderen Diensten ergänzen.

Das heißt, daß viele Internet-Dienste eben doch nicht ohne Namen und ohne Namensauflösung auskommen.

Das macht aber nichts, denn wie in den vorigen Abschnitten beschrieben kann man die Namensauflösung durchaus auch ohne DNS oder mit anderen DNS-Servern durchführen. Anstatt URLs und e-Mail-Adressen von Domainnamen auf Internet-Adressen umzustellen ist es viel einfacher und unproblematischer, sie unverändert zu lassen und die Zuordnung zwischen Name und Adresse in der hosts-Datei vorzunehmen.

4 Welche Informationen findet man im DNS?

In diesem Kapitel betrachten wir, *was* im DNS gespeichert wird, aber noch nicht *wo* man es speichert und *wie* man es findet, das kommt erst im nächsten Kapitel dran.

Außerdem werden wir nicht alle Datentypen im DNS betrachten, denn viele sind veraltet und überflüssig. Wir betrachten nur die, die heute tatsächlich in Gebrauch sind.

4.1 Probieren geht über Studieren

Die wichtigste Zutat zum Verständnis des DNS ist der Aha-Effekt. Deshalb muß man selbst mal im DNS experimentieren.

Außerdem erläutere ich immer am Beispiel meiner eigenen Domain `danisch.de`, weil ich da keinen Ärger mit dem Domain- oder Namensinhaber bekomme und nicht von plötzlichen Änderungen überrascht werde, aber natürlich ist es doof, wenn man das DNS immer nur anhand meiner eigenen Domain betrachtet. Der Leser ist also angehalten, jedes Beispiel, in dem `danisch.de` vorkommt, mit irgendwelchen anderen Domains nachzuvollziehen und sich die Unterschiede anzuschauen.

Die Experimente durchzuführen ist dabei ganz einfach und völlig harmlos. Ein gewöhnlicher Rechner in der Standardkonfiguration, mit Windows (ab XP) oder Linux reicht. Man braucht nicht mal Zusatzsoftware oder irgendwelche Einstellungen. Es kostet nichts, und man kann nichts kaputtmachen.

Eine Einschränkung gibt es jedoch: Je nachdem, in welchem Netzwerk man sich mit seinem Rechner befindet, kann es zu Einschränkungen kommen, je nachdem was die Netzwerkadministration an lokalem DNS eingestellt und an DNS-Verkehr auf der Firewall ausfiltert. Besonders ab dem Kapitel über Delegation kann es leicht passieren, daß die Beispiele aus einem Firmennetz heraus nicht möglich sind[1]. Im Zweifelsfall kann man die Beispiele von einem

[1] Eigentlich ist es sogar so, daß einige oder unter Umständen sogar alle Beispiele aus Firmennetzen heraus nicht nachvollziehbar sein sollten, denn wenn der Administrator sein Handwerk versteht und es nicht triftige Gründe gibt, den DNS-Verkehr freizugeben, sollte der direkte DNS-Verkehr von den Arbeitsplatzrechnern in das Internet gesperrt sein und über einen zentralen DNS-Resolver laufen oder sogar gar keine Internet-DNS-Daten in das LAN gegeben werden. Was das alles bedeutet, dazu kommen wir später.

privaten Internet-Anschluß zu Hause ausprobieren, da sollte das ohne weiteres funktionieren.

4.1.1 Übungen unter Microsoft Windows

Unter Windows (ab XP) sind die Experimente ganz einfach, es ist nämlich alles schon da, was man zu experimentieren braucht. Das Programm `nslookup` reicht, mehr brauchen wir nicht. Es ist allerdings kein graphisches Programm, sondern ein kommandozeilenorientiertes.

Wir rufen also mit der Maus unter dem Menü „Start“ den Punkt „Ausführen“ auf. Es öffnet sich ein kleines Fenster, das fragt, was auszuführen sei. Je nach Vorliebe und Vorkenntnissen gibt man dort direkt `nslookup` ein oder öffnet eine Kommandoshell, indem man `cmd` eingibt um dort dann `nslookup` aufzurufen.

Es meldet sich nslookup ungefähr so mit blinkendem Cursor:

```
Standardserver:  dnsserver.danisch.de
Address:  10.0.2.3

>
```

Natürlich steht dort ein anderer Server als bei mir, und es kann sogar eine Fehlermeldung kommen, daß der Servername nicht gefunden wurde, aber Hauptsache, hinter Address: steht eine Internet-Adresse aus Zahlen.

Damit kann man loslegen.

Für die Experimente in diesem Kapitel muß man nur drei Befehle für nslookup kennen:

exit
: Damit beendet man nslookup wieder.

Abfrage zu einem Domainnamen
: Wir geben den Domainnamen einfach ein, und nslookup gibt die Antwort. Beispiel:

```
> www.danisch.de
Server:  dnsserver.danisch.de
Address:  10.0.2.3

Nicht autorisierte Antwort:
Name:    www.danisch.de
Address:  88.198.248.38
```

(Das bedeutet, daß uns hier der Nameserver 10.0.2.3 die non-authoritative Auskunft gegeben hat, daß zu www.danisch.de die IP-Adresse 88.198.248.38 gehört.)

set type = ...

Um unterschiedliche Daten abzufragen muß man vorher mit diesem Befehl den Datentyp setzen, nachdem man fragen will. Dies wird in den nachfolgenden Abschnitten zu den Datentypen jeweils erläutert. Daß man nach dem Start von nslookup direkt eine Anfrage stellen kann ohne diesen Befehl eingegeben zu haben liegt daran, daß der Abfragetyp A voreingestellt ist. Will man nach allen verfügbaren Daten fragen, gibt man den Typ `any` an. Beispiel:

```
> set type=any
```

4.1.2 Übungen unter Linux/Unix

Auch unter Linux gibt es nslookup (eigentlich kommt es sogar daher und wurde auf Windows portiert), das allerdings seit längerer Zeit nicht mehr weiterentwickelt wird und daher inzwischen sogar meist hinter den Funktionen der Windows-Version zurückbleibt. Die meisten Beispiele kann man unverändert auch mit der Linux- oder Unix-Version durchführen.

Das modernere Tool ist jedoch `dig`. dig ist auch nicht interaktiv, sondern nimmt seine Befehle als Kommandozeilenparameter, gibt das Ergebnis aus und terminiert wieder. Näheres zur dig unter der man-page von dig. Prinzipiell sollte man sich für dig merken:

- Der abgefragte typ wird hinter dig auf der Kommandozeile angegeben.
- Der abgefragte datentyp wird hinter dem angefragten Namen angegeben. Gibt man nichts an, wird – wie bei nslookup – der Datentyp A abgefragt.

Die Ausgabe von dig ist deutlich „echter“ weil eine direktere Darstellung des Inhaltes des DNS-Paketes:

```
% dig www.danisch.de

; <<>> DiG 9.5.1-P2 <<>> www.danisch.de
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57391
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;www.danisch.de. IN A

;; ANSWER SECTION:
www.danisch.de. 4757 IN A 88.198.248.38

;; AUTHORITY SECTION:
danisch.de. 4757 IN NS robotns2.second-ns.de.
danisch.de. 4757 IN NS ns1.first-ns.de.
danisch.de. 4757 IN NS dns.rackland.de.
danisch.de. 4757 IN NS robotns3.second-ns.com.

;; ADDITIONAL SECTION:
robotns2.second-ns.de. 65710 IN A 213.133.105.6
ns1.first-ns.de. 0 IN A 213.239.242.238
dns.rackland.de. 83957 IN A 88.198.248.38
robotns3.second-ns.com. 152110 IN A 193.47.99.3

;; Query time: 13 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu Sep  3 15:54:14 2009
```

```
;; MSG SIZE  rcvd: 235
```

Sieht schon ganz anders aus als die Ausgabe von nslookup. Wir wir später noch sehen werden, entspricht dies genau dem Aufbau und Inhalt des DNS-Datenpaketes mit der Antwort des Servers. dig ist zwar etwas schwieriger als nslookup, dafür aber „näher dran“. Im Gegensatz zu nslookup sieht man auch die Time-to-Live (hier 4757 Sekunden für www.danisch.de).

Leider kann es allerdings sein, daß nslookup und dig bei einer Linux-Standardinstallation – je nach Distribution – nicht schon automatisch vorinstalliert werden und als optionale Software nachinstalliert werden müssen.

4.2 Domainnamen als Suchschlüssel

Einige wesentliche Eigenschaften des DNS haben wir schon erwähnt, andere müssen wir hier erwähnen:

- Alle Daten werden nach Domain-Namen gespeichert. Um nach Daten zu fragen, muß man den Domain-Namen angeben, für den man nach Daten sucht, und man muß den Datentyp angeben, nach dem man fragt.
- Domain-Namen sind maximal 255 Zeichen lang und bestehen aus einzelnen, durch Punkte getrennten Namensteilen. Jeder Namensteil darf maximal 63 Zeichen lang sein und nur aus Klein- oder Großbuchstaben (a-z, A-Z), Ziffern (0-9), Underscore und Minus-Zeichen bestehen, und muß dabei mit einem Buchstaben anfangen. Zwar können wir Groß- und Kleinbuchstaben verwenden, aber DNS unterscheidet sie nicht. WWW.DaNiScH.de ist das Gleiche wie www.danisch.de.
- Die Domain-Namen sind hierarchisch geordnet, die Wurzel steht rechts. Solange die Namen nicht an die Längenbegrenzung von 255 Zeichen stoßen, kann man beliebig Unterdomains bilden.

  Beispiel: www.danisch.de ist eine Unterdomain von danisch.de, diese von de und diese wiederum von der leeren Domain .
- Das DNS kann zu jedem Domain-Namen Daten verschiedener Typen und nicht nur eines Typs speichern. Ein und derselbe Domain-Name kann also A, AAAA, TXT und andere Records haben.
- Das DNS kann zu jedem Domain-Namen auch mehrere Daten desselben Typs speichern. Ein Domain-Name kann also auch viele A-Records und viele TXT-Records usw. haben.

4.3 Die Hierarchiebäume im DNS

Aus diesen Domainnamen und Hierarchien werden (derzeit) drei Hierarchie-Bäume gebildet:

Die normalen generic und country-code TLDs , also die normalen Domain-Namen, wie wir sie aus dem Internet kennen, die unter .com, .edu, .gov und so weiter und unter den Länderdomains wie .de liegen. Deshalb kann man eigentlich auch nicht von einem einzelnen Hierarchiebaum sprechen, sondern eher von einem Wäldchen.

Außer bei der Besprechung der PTR-Records werden in diesem gesamten Kapitel diese „normalen“ Domainnamen betrachtet.

Rückwärtsauflösungen für Internet-Adresse sind für besondere Anfragen nötig, die im Abschnitt über PTR-Records betrachtet werden. Sie liegen unter der Domain `arpa.`

Hierarchien für Telefonnummern wurden unlängst eingeführt, um auch zu Telefonnummern Daten im Internet zu speichern. Das wird in dieser Einführung aber nicht betrachtet.

4.4 A-Records: Internet-Adressen finden

Der meistverwendete und intuitiv wichtigste Datentyp ist der A-Record. Er speichert zu einem Domain-Namen die IPv4-Adresse, die fast alle Internet-Dienste verwenden, beispielsweise das World Wide Web, aber auch Dienste wie FTP und viele andere. So ermittelt auch der Web-Browser, unter welcher IP-Adresse er den in einem URL genannten Server erreichen kann.

Weil dieser Typ so häufig ist, ist er unter nslookup auch voreingestellt, man kann das also auch weglassen, falls man den Abfragetyp nicht mit `set type=...` geändert hat. Auch dig verwendet den A-Records sofern man keinen anderen Typ angibt.

Beispiel mit nslookup:

```
> set type=a
> www.danisch.de
Server:  UnKnown
Address:  10.0.2.3

Nicht autorisierte Antwort:
Name:    www.danisch.de
Address:  88.198.248.38
```

Übung: Bestimmen Sie die IPv4-Adressen der von Ihnen bevorzugt besuchten Webseiten. Probieren Sie es aus! Finden Sie auch Hostnamen, auf denen mehr als eine IPv4-Adresse liegt?

Den umgekehrten Weg, nämlich von einer IP-Adresse zurück zu einem Namen betrachten wir unten in Abschnitt 4.12.

4.5 AAAA-Records: Internet-IPv6-Adressen finden

Im Prinzip das gleiche wie bei den A-Records für IPv4, nur eben mit den längeren IPv6-Adressen. Und weil IPv6-Adresse mit 128 Bit viermal so lang sind wie IPv4-Adressen mit 32 Bit, hat man den Record-Typ spassigerweise AAAA genannt.

Ein Beispiel mit nslookup:

```
> set type=aaaa
> www.ipv6.org
...
Nicht autorisierte Antwort:
www.ipv6.org    canonical name = shake.stacken.kth.se
shake.stacken.kth.se    AAAA IPv6 address = 2001:6b0:1:ea:202:a5ff:fecd:13a6
...
```

4.6 MX-Records: E-Mail-Server finden

Wenn wir eine e-Mail an jemanden schicken, woher weiß unser Mail-Server, an welchen Rechner er sie schicken muß? Er nimmt aus der Mail-Adresse des Empfängers den Teil rechts vom @-Zeichen und verwendet ihn als Domain-Namen. Dann fragt er den MX-Record ab, MX steht für Mail-Exchange.

Auch hier wieder ein Beispiel mit nslookup. Wir wollen abfragen, an welchen Rechner E-Mail ausgeliefert wird, wenn Sie eine E-Mail an hadmut@danisch.de schicken:

```
> set type=mx
> danisch.de
...

Nicht autorisierte Antwort:
danisch.de  MX preference = 10, mail exchanger = mail.rackland.de
...
mail.rackland.de        internet address = 88.198.248.38
...
```

Mail für die Domain danisch.de wird also an einen Rechner namens mail.rackland.de geliefert. Dessen IP-Adresse wird praktischerweise gleich mitgeliefert. Ein Mailserver könnte also direkt mit der Auslieferung per SMTP an diese Adresse beginnen.

Wir wir sehen, enthält die Antwort aber zusätzlich die Angabe `preference = 10`. Was bedeutet das? Nun, E-Mail ist ein wichtiger Dienst, der immer zur Verfügung stehen sollte. Deshalb geben viele Domain-Inhaber nicht nur einen, sondern mehrere Mail-Server an, die Mail für den Empfänger entgegennehmen können. Manchmal sind das aber auch nur angemietete fremde Server, die man nicht so gerne verwendet oder die einfach zusätzlichen Aufwand betreiben, weshalb die nur zur Reserve eingesetzt werden sollen und nur, wenn die regulären Server gerade nicht verfügbar sind.

Deshalb gibt diese preference an, welche Mail-Server der Absender zuerst probieren soll. Erst versucht er die, mit dem niedrigsten preference-Wert, dann die nächsthöheren und so weiter.

Aufgabe: Bestimmen Sie für Ihre häufigsten Mail-Partner, welche Mail-Rechner für diese E-Mail empfangen und in welcher Reihenfolge. Fragen Sie ab, auf welchen Rechnern die Bundesregierung unter bundesregierung.de Mail empfängt.

Sollte es für die Empfängerdomain übrigens keine MX-Record geben, fragt der Mail-Server des Absenders ersatzweise nach dem A-Record, also direkt der IP-Adresse zu der Domain. Deshalb können Internet-Sperren, die eigentlich den Abruf von Webseiten blockieren sollen, durchaus auch den E-Mail-Verkehr stören.

4.7 SRV-Records: Server für Dienste finden

Wir haben eben gesehen, daß es für E-Mail einen eigenen Record-Typ, den MX-Record gibt, der für diesen Dienst verschiedene Server und dazu Prioritäten angeben kann. Das hat sich als praktisch und nützlich erwiesen, aber ist leider auf den E-Mail-Dienst beschränkt. Man wollte diese Art von Record auch für andere, neuere Dienste habe, aber nicht den Fehler wiederholen, sich auf einen bestimmten Dienst zu beschränken und festzulegen. Deshalb wollte man einen neuen Datentyp für *beliebige* Dienste einführen. Der Benutzer, der DNS-Daten anfragt, muß also noch den Dienst angeben, für den er fragt.

Das wirft ein Problem auf: DNS sieht nicht vor, daß man weitere Bestandteile in die Frage einbaut, man kann nur nach einem Domain-Namen und einem Datentyp fragen. Und für jeden Dienst einen neuen Datentyp zu definieren wollte man ja gerade vermeiden. Also muß der Dienst, nach dem man fragt, in den Domain-Namen mit hineingepackt werden. Man hat dies darüber gelöst, daß man vor den eigentlichen Domain-Namen noch den Dienst und das

Protokoll als Subdomain-Namen mit verpackt und hat damit nachträglich den SRV-Record in RFC 2782 eingeführt.

Man setzt vor den eigentlich abgefragten Domainnamen ein Underscore, den Dienstnamen, einen Punkt, einen weiteren Underscore, den Protokollnamen (TCP oder UDP) und noch einen Punkt.

Wie auch beim MX-Record wird zusätzlich zu jedem angegebenen Host-Namen noch eine Priorität angegeben, zusätzlich dazu aber noch ein weiterer Zahlenwert für die Gewichtung der Server innerhalb einer Priorität, um unterschiedlichen Leistungsfähigkeiten der Server Rechnung zu tragen.

Obwohl überaus nützlich, werden die SRV-Records bisher nur für wenige Anwendungen eingesetzt:

- Im Chat-Dienst Jabber werden die Server über SRV-Records gefunden. Die „Subdomain“ für den Dienst lautet `_JABBER._TCP`.
- Beim Voice-over-IP-Protokoll (Telefonieren über das Internet) SIP kann man statt Telefonnummern auch Empfängeradressen anrufen, die wie E-Mail-Adressen aussehen. In diesem Fall wird der Server des Angerufenen – ähnlich wie bei E-Mail – aus dem Teil rechts des ’@’-Zeichens bestimmt. Dazu dient die „Subdomain“ `_SIP._UDP`.
- In Windows-Netzwerken, die mit Active Directory verwaltet werden, finden die Rechner die Server zu den Netzwerk-Domains ebenfalls über die SRV-Records:
  - `_LDAP._TCP` für den AD-Server selbst
  - `_KERBEROS._TCP`, `_KERBEROS._UDP`, `_KPASSWD._TCP` und tt _KPASSWD._UDP für den Kerberos-Dienst
  - `_GC._TCP` für den Global Catalog

Auch hier haben wir wieder ein Beispiel. Der Chaos Computer Club betreibt einen Jabber-Server. Die Benutzeradressen dafür lauten auf ...@jabber.ccc.de. Wir wollen herausfinden, welche IP-Adresse der Jabber-Server des CCC hat:

```
> set type=srv
> _jabber._tcp.jabber.ccc.de
...
Nicht autorisierte Antwort:
_jabber._tcp.jabber.ccc.de      SRV service location:
          priority       = 5
          weight         = 0
          port           = 5269
          svr hostname   = jabberd.jabber.ccc.de

...
jabberd.jabber.ccc.de   internet address = 217.10.10.196
...
```

Es ist schade, daß SRV-Records erst nachträglich eingeführt wurden und deshalb heute kaum verwendet werden, denn eigentlich ist das eine ganz famose und leistungsfähige Erfindung. MX-Records wären damit als Spezialfall überflüssig, und auch das World Wide Web hätte profitiert, wenn Browser nicht den A-Record für einen angegebenen Hostnamen, sondern den SRV-Record einer angegebenen Domain für HTTP abfragen würden.

Übung: Wenn Sie sich an einem Windows-Rechner befinden, der in einem Windows-Netzwerk angemeldet ist, bestimmen Sie zunächst den Domainnamen des Netzwerkes und fragen dann ab, welche Rechner die AD-Server für ihre Domain sind[2]

4.8 TXT-Records: Allgemeine Informationen abfragen

Man hat auch einen TXT-Record definiert, der einfach eine beliebige Zeile Text speichert, ohne dafür eine konkrete Verwendung vorzuschreiben. Gedacht war das eigentlich für menschenlesbare Informationen, nicht für Maschinen.

DNS ist aber nicht so besonders gut konstruiert und leidet an einigen Entwurfsfehlern, die es sehr schwer machen, neue Datentypen einzuführen. Deshalb „mißbraucht“ man gelegentlich diese TXT-Records um neue Daten abzulegen. Ein Beispiel dafür sind SPF-Records gegen E-Mail-Fälschung (vgl. Abschnitt 6).

Beispiel:

Wir wollen den SPF-Record für danisch.de abfragen:

```
> set type=txt
> danisch.de
...
Nicht autorisierte Antwort:
danisch.de      text =

        "v=spf1 ip4:88.198.248.38 -all"
```

4.9 SOA-Records: Verwaltungsinformationen

Der SOA-Record enthält Steuerinformationen für die Secondary Server (vgl. Frage 5.6), ist für den Nutzer ohne Bedeutung und zum Verständnis uninteressant.

[2] Falls Sie den Domainnamen Ihres Netzwerkes nicht kennen, können Sie ihn mit `echo Userdomain=%USERDOMAIN% und UserDNSDomain=%userdnsdomain%` anzeigen. Auch der mit `ipconfig` angezeigte DNS Suffix stimmt meist damit überein.

4.10 NS-Records: Nameserver finden und Delegation nachverfolgen

Wir haben schon mehrfach erwähnt, daß das DNS ein verteiltes System ist, das nicht zentral arbeitet, sondern die Zuständigkeiten durch Delegation an verschiedene Server verteilt. Wie das genau funktioniert, werden wir erst in Kapitel 5 betrachten. Da wir jetzt aber trotzdem schon über den dafür benötigten Datentyp gestolpert sind, schauen wir in uns auch an.

Der Nameserver-Record gibt den Namen des zuständigen Nameservers an, und weil es meistens mehrere Nameserver gibt, gibt es meist auch mehrere Nameserver-Records.

Der Nameserver-Record hat für Internet-Dienste, Anwendungsprogramme und Benutzer normalerweise keinen Nutzen, sondern dient der Delegation im DNS und hat damit nur eine DNS-interne Funktion. Trotzdem kann man ihn abfragen wie jeden anderen Datentyp auch.

Beispiel:

Bestimmen Sie die Nameserver für meine Domain danisch.de:

```
> set type=ns
> danisch.de
...
Nicht autorisierte Antwort:
danisch.de      nameserver = ns1.first-ns.de
danisch.de      nameserver = robotns2.second-ns.de
danisch.de      nameserver = robotns3.second-ns.com
danisch.de      nameserver = dns.rackland.de
...
```

Bestimmen Sie außerdem die Nameserver für die Domain de von Deutschland. Dabei könnte folgendes passieren:

```
> de
Server:  UnKnown
Address:  10.0.2.3

*** de wurde von ... nicht gefunden: Server failed
```

Was ist da schief gegangen? Wir hatten im Abschnitt 3.1.5 beleuchtet, daß man aufgrund von Nachlässigkeiten oft den Punkt am Ende eines Domainnamens wegläßt, der eigentlich relative von vollständigen Domainnamen unterscheidet. Weil die Domain de nur aus einem Teil besteht, hat nslookup sie nicht als Fully Qualified Domain Name erkannt, sondern als relativen Domainnamen aufgefasst und unter der lokalen Domain gesucht. Es könnte ja sein, daß man in seinem Netzwerk eine Unterdomain oder einen Rechner namens

de hat und den gemeint hat. Die Anfrage war deshalb mehrdeutig. Hängen wir, wie es eigentlich richtig ist, den Punkt ans Ende von de, dann klappts:

```
> de.
...
Nicht autorisierte Antwort:
de      nameserver = C.DE.NET
de      nameserver = L.DE.NET
de      nameserver = A.NIC.de
de      nameserver = Z.NIC.de
de      nameserver = S.DE.NET
de      nameserver = F.NIC.de
```

Übung 1: Bestimmen Sie die Nameserver der Domains irgendwelcher Firmen, die Ihnen gerade einfallen.

Übung 2: Bestimmen Sie die Root-Name-Server des DNS, also die Nameserver, die in der Hierarchie-Wurzel ganz oben sitzen und die Hierarchiebaum aufspannen. Hinweis hierzu: Der Domainname der DNS-Wurzel ist der leere Name, an dessen Ende ein Punkt steht. Also nur ein Punkt. Geben Sie bei nslookup also einfach nur einen Punkt als Frage ein. Was fällt Ihnen an den Namen der Root-Name-Server auf?

4.11 CNAME-Records: Alias-Namen

Nicht alles läßt sich in eine strikte Hierarchie pressen. Manchmal ist es erforderlich, den Anfragenden auf einen anderen Namen zu verweisen, damit er dort weitersucht. Dafür ist der CNAME-Record da: Er liefert einfach einen neuen Domainnamen, an dem der Suchende mit der Suche fortfahren soll. CNAME steht für „Canonical Name„ und besagt, daß das der Verweis auf den eigentlich richtigen Namen ist.

Das kann vielfältige Gründe haben. Manchmal möchte man einfach nur Ordnung halten und die Einstellungen für IP-Adressen an einer Stelle zentralisieren, obwohl der Rechner unter vielen Namen auftauchen soll.

Angenommen, man hat einen Rechner namens `server.beispiel.com`, für den man über einen A-Record auch die IP-Adresse angegeben hat. Nun möchte man, daß der Server, der verschiedene Aufgaben übernimmt, auch als Webserver agiert und unter dem gebräuchlichen Namen `www.beispiel.com` anzusprechen ist. Man könnte nun entweder auch für diesen Namen die IP-Adresse eintragen, müßte sie aber immer an mehreren Stellen ändern, falls die IP-Adresse einmal wechselt. Einfacher wäre es, unter `www.beispiel.com` einen CNAME-Record einzutragen, der auf `server.beispiel.com` verweist. Damit hat man die IP-Adresse nur an einer Stelle, und sie stimmt trotzdem für alle Namen.

Man kann damit auch in andere Domains verweisen. Hat man keine statische IP-Adresse, sondern nur einem normalen Heim-Internet-Anschluß mit dynamisch zugewiesenen IP-Adresse, dann würde es schwierig, hierfür immer einen Eintrag unter `www.beispiel.com` aktuell zu halten. Es gibt aber Anbieter wie dyndns.org, die einem ständig aktualisierte DNS-Einträge anbieten können. Die meisten Heim-Router können einen solchen Eintrag ständig auf die vergebene IP-Adresse aktualisieren, womit man einen DNS-Namen hat, der immer auf die aktuell vergebene IP-Adresse verweist. Die Sache hat einen Schönheitsfehler: Diese Anbieter aktualisieren nur DNS-Namen unter ihren eigenen Domains wie `hugosrouter.dyndns.org`, was nicht schön aussieht. Setzt man aber unter `www.beispiel.com` einen CNAME auf `hugosrouter.dyndns.org`, dann kann man den Rechner zu Hause trotz der dynamischen IP-Adressen immer auch unter diesem Namen ansprechen.

Auch den CNAME kann man mit nslookup abfragen. Fragt man aber nach anderen Record-Typen wie etwa einem A-Record, dann springt nslookup automatisch bei Angabe eines CNAME weiter zum nächsten Namen. Wir schauen uns das am Beispiel von `www.google.de` an. Zunächst bestimmen wir die IP-Adresse für www.google.de:

```
> set type=a
> www.google.de
...
Nicht autorisierte Antwort:
Name:    www.l.google.com
Addresses:  209.85.135.106, 209.85.135.104, 209.85.135.105,
          209.85.135.103, 209.85.135.147, 209.85.135.99
Aliases:  www.google.de, www.google.com
```

Da gibt es nun zwei Besonderheiten: Obwohl wir nach www.google.de gefragt haben, wird als Name in der Antwort www.l.google.com angegeben. Außerdem werden www.google.de und www.google.com als Aliases genannt. Wie kommt das? Schauen wir uns die CNAMEs an:

```
> set type=cname
> www.google.de
...
www.google.de   canonical name = www.google.com
...
```

www.google.de verweist also per CNAME auf www.google.com,

```
> www.google.com
...
www.google.com  canonical name = www.l.google.com
...
```

und das auf www.l.google.com, und da liegen erst die IP-Adressen.

4.12 PTR-Records: Namen finden

Dieser Record-Typ ist komplizierter als die anderen. Bisher hatten wir immer den Fall, daß wir einen Domainnamen angeben und dazu Daten wie die IP-Adresse abfragen. Manchmal steht man aber vor dem umgekehrten Problem, nämlich daß man zu einer gegebenen IP-Adresse den zugehörigen DNS-Namen finden will. Man erwartet, daß DNS das auch leistet.

Dabei stößt man auf ein Problem. Denn das DNS akzeptiert als Frage nur Domain-Namen, nicht IP-Adressen. Man muß IP-Adressen also so umformen, daß sie selbst als Suchschlüssel dienen können. Die Lösung ist, IP-Adressen auf einen eigenen DNS-Hierarchiebaum anzubilden und IP-Adressen selbst als Domainnamen darzustellen. Dann könnte man im DNS zu jeder IP-Adresse deren normalen DNS-Namen ablegen und abfragen.

Auf den ersten Blick erscheint so eine Umwandlung von IP-Adressen in besondere Domainnamen nicht schwer, denn eine IPv4-Adresse wie 88.198.248.38 sieht ja schon aus wie ein Domainname – einzelne Teile, die durch einen Punkt verbunden sind. Wir müssen nur die Regel fallen lassen, daß die Namensbestandteile mit einem Buchstaben anfangen müssen.

Man läuft dabei aber auf ein weiteres Problem. Auch IP-Adressen sind hierarchisch organisiert, nur mit dem Unterschied, daß die Wurzel *links* steht, während sie bei Domainnamen *rechts* steht. Würden wir die IP-Adresse 88.198.248.38 als Domainnamen auffassen, dann wäre 88.198 eine Unterdomain von 248.38, man müßte zuerst 38 delegieren, dann 248 und so weiter. Das ist aber falsch, denn 248.38 ist ein Unterbereich des Netzwerkes 88.198/16 und nicht umgekehrt. Das Problem läßt sich lösen, indem man die IP-Adresse *verkehrtherum* als Domain-Name schreibt: Die IP-Adresse 88.198.248.38 muß in Domain-Schreibweise also 38.248.198.88 lauten, dann stimmt es auch mit den Unterdomains. Außerdem hat man festgelegt, daß man IPv4-Adressen im DNS nicht unter der Wurzel (also dem leeren Domain .) sondern unter der Domain `in-addr.arpa` ablegt.

Allgemein gesagt: Der Domain-Name für die IP-Adresse w.x.y.z liegt unter z.y.x.w.in-addr.arpa.

Wieder ein Beispiel hierzu: Wir wollen den DNS-Namen für die IP-Adresse 88.198.248.38 bestimmen. Dazu müssen wir die Adresse „rumdrehen“ und .in-addr.arpa anhängen:

```
> set type=ptr
> 38.248.198.88.in-addr.arpa
...

38.248.198.88.in-addr.arpa      name = sklave3.rackland.de
...
```

Der Rechner heißt also sklave3.rackland.de. Diese Struktur hat den schönen Vorteil, daß man die Zuständigkeit für 198.88.in-addr.arpa an den Inhaber des Netzes 88.198/16 delegieren kann[3].

Übung: Bestimmen Sie für Ihre bevorzugt besuchten Webseiten zunächst die IP-Adresse und dann zu der IP-Adresse den Hostnamen. Stimmt er überein?

Auch für IPv6-Adressen funktioniert das Prinzip, nur mit zwei Unterschieden:

- Die Zuordnung liegt nicht unter `in-addr.arpa`, sondern unter `ip6.arpa`.
- Auch hier werden die IP-Adressen herumgedreht, dazu aber nicht in 8-Bit-Teile wie bei IPv4, sondern in 4-Bit-Teile unterteilt. Man teilt also in der Hex-Schreibweise für IPv6 die einzelnen Hex-Ziffern durch Punkte und dreht das Ganze herum. Aus der IPv6-Adresse 2003::4 wird also der Domain-Name 4.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.0.2.ip6.arpa.

Man bezeichnet die Zuordnung eines Domainnamens zu einer IP-Adresse über PTR-Records auch als *Rückwärtsauflösung*.

Eine dringende Warnung zu PTR-Records: Wer die Administrationsrechte für den Nameserver für die Rückwärtsauflösung zu seinen IP-Adressen hat, kann dort eintragen, was er will, was für allerlei Angriffe mißbraucht werden kann und auch häufig wird. Man sollte den Domainnamen, die ein PTR-Record liefert, weder bezüglich der Richtigkeit noch bezüglich ihrer Syntax trauen. Bevor man den von einem PTR-Record gelieferten Namen verwendet, sollte man ihn unbedingt auf syntaktische Korrektheit, auf Plausibilität und durch eine anschließende Vorwärtsauflösung über A oder AAAA-Records auf Richtigkeit überprüfen. Keinesfalls sollte man die Angaben aus PTR-Records (alleine) für Log-Dateien, forensische Zwecke oder als Beweismittel ansehen[4][5]. Mehr dazu in Abschnitt 8.

[3] Leider funktioniert das so nur mit Class A, Class B und Class C Netzen, aber nicht mit anderen CIDR-Netzen. In diesem Fall muß man für jedes Teilnetz eine eigene Subdomain anlegen und CNAMEs einfügen, was hier aber zu weit führen würde.

[4] Forensik

[5] Beweismittel

5 Wo liegen die DNS-Daten und wie findet man sie?

5.1 Weitere Experimente

Auch in diesem Abschnitt experimentieren wir wieder mit nslookup unter Windows und dig unter Linux. Dazu brauchen wir zwei neue Befehle bzw. Optionen.

Zum einen müssen wir rekursive Abfragen abschalten können (was das ist, bekommen wir später). Unter nslookup machen wir das mit dem Befehl

```
> set norecurse
```

und schalten sie mit

```
> set recurse
```

wieder ein. Bei dig geben wir auf der Kommandozeile die Option `+norecurse` an.

Und wir benötigen einen Weg um anzugeben, daß wir einen ganz bestimmten Nameserver (und nicht nur unseren eigenen) etwas fragen wollen. Unter nslookup geben wir den Server mit

```
> set server nameserver
```

an, unter dig geben wir den Server auf der Kommandozeile mit vorangestelltem @ an. Wir werden dies in den Beispielen sehen.

5.2 Root-Name-Server

Unter Frage 3.1.2 hatten wir schon die Überlegung erörtert, ob es nur ein oder viele DNS gibt. Die Antwort war Ja, man kann beliebig viele DNS-Bäume aufspannen und manche Angreifer tun das um einen bessern angreifen zu

können. Welchen DNS-Baum man verwendet hängt davon ab, bei welchen Wurzel-Servern (sogenannte Root-Name-Server) man seine Suche beginnt. Die Auswahl der Root-Name-Server entscheidet darüber, ob man im „richtigen“ DNS-Namensbaum ist. Und da es nur einen weltweit anerkannten und verwendeten DNS-Namensbaum gibt, wäre die Frage, wie man diese herausfindet. Wer bis hierher mitgelesen hat, weiß das nicht nur, sondern hat das auch schon selbst herausgefunden, nämlich in der Übung zu Abschnitt 4.10 über NS-Records. Machen wir's aber nochmal explizit. Die Wurzel im DNS besteht aus dem leeren Namen und wird durch einen Punkt abgeschlossen. Also fragen wir mit nslookup einfach mal die Nameserver für diesen leeren Namen ab:

```
> set type=ns
> .
Server:  UnKnown
Address:  10.0.2.3

Nicht autorisierte Antwort:
(root)  nameserver = M.ROOT-SERVERS.NET
(root)  nameserver = E.ROOT-SERVERS.NET
(root)  nameserver = I.ROOT-SERVERS.NET
(root)  nameserver = H.ROOT-SERVERS.NET
(root)  nameserver = J.ROOT-SERVERS.NET
(root)  nameserver = G.ROOT-SERVERS.NET
(root)  nameserver = F.ROOT-SERVERS.NET
(root)  nameserver = A.ROOT-SERVERS.NET
(root)  nameserver = D.ROOT-SERVERS.NET
(root)  nameserver = B.ROOT-SERVERS.NET
(root)  nameserver = K.ROOT-SERVERS.NET
(root)  nameserver = C.ROOT-SERVERS.NET
(root)  nameserver = L.ROOT-SERVERS.NET

M.ROOT-SERVERS.NET      internet address = 202.12.27.33
E.ROOT-SERVERS.NET      internet address = 192.203.230.10
I.ROOT-SERVERS.NET      internet address = 192.36.148.17
H.ROOT-SERVERS.NET      internet address = 128.63.2.53
H.ROOT-SERVERS.NET      AAAA IPv6 address = 2001:500:1::803f:235
J.ROOT-SERVERS.NET      internet address = 192.58.128.30
J.ROOT-SERVERS.NET      AAAA IPv6 address = 2001:503:c27::2:30
G.ROOT-SERVERS.NET      internet address = 192.112.36.4
F.ROOT-SERVERS.NET      internet address = 192.5.5.241
F.ROOT-SERVERS.NET      AAAA IPv6 address = 2001:500:2f::f
A.ROOT-SERVERS.NET      internet address = 198.41.0.4
A.ROOT-SERVERS.NET      AAAA IPv6 address = 2001:503:ba3e::2:30
D.ROOT-SERVERS.NET      internet address = 128.8.10.90
B.ROOT-SERVERS.NET      internet address = 192.228.79.201
```

Und schon haben wir die Liste der Root-Name-Server des weltweiten DNS, dazu deren IPv4- und einige IPv6-Adressen. Damit wissen wir, wo das weltweite DNS „anfängt“. Wie man sieht, hat irgendwer diesen Servern auch

einheitliche Namen gegeben. Sie bestehen alle aus einem Buchstaben und `.root-servers.net`.

Das eröffnet ein Henne-Ei-Problem: Wie kann man DNS nach den Root-Servern fragen, die man braucht um DNS in Gang zu setzen, wenn man zuerst einen funktionierenden DNS-Client haben muß, um danach fragen zu können, die Root-Server aber noch nicht kennt? Gute Frage.

Irgendwo sind sie dokumentiert. Und bei DNS-Serversoftware ist normalerweise eine Konfigurationsdatei mit dabei, in der sie stehen. Diese oben aufgelisteten Adressen muß man einem Nameserver, der als selbständiger Server jungfräulich startet, durch manuelle Konfiguration mitgeben. Schreibt man da etwas anderes herein, hängt man – gewollt oder ungewollt – im „falschen“ DNS (oder an gar keinem).

Der einfachste Weg ist aber der oben beschriebene, indem man einfach auf irgendeinem Rechner am Internet mit laufendem DNS nach den Servern der Root-Domain fragt. Ganz einfach.

5.3 Delegation

Nun wissen wir, wo die Wurzel unseres weltweiten DNS-Systems anfängt. Wie aber findet unser DNS-Resolver die angefragten Daten zu einem Domainnamen, etwa eine IP-Adresse, die ein Webbrowser haben will? Wir haben ja in Kapitel 2 festgestellt, daß DNS dezentral, verteilt und delegiert ist. Also ist nicht zu erwarten, daß die Daten bei den Root-Name-Servern gespeichert ist, die wir eben gefunden haben. *Wie also findet die DNS-Software heraus, wo die Daten zu einem Domainnamen liegen und wie man sie dort abholt?*

Probieren wir es einfach aus. Ich habe im vorangegangenen Abschnitt gesagt, daß man einem DNS-Server (der als Resolver arbeiten soll) diese Liste der Root-Name-Server per manueller Konfiguration mitgeben muß. Am Anfang ist der Nameserver also leer, weiß noch gar nichts über die Struktur des DNS und andere Server. Er kennt nur die Root-Name-Server. Im bleibt also gar nichts anderes übrig, als sich einen dieser Root-Name-Server herauszugreifen und den zu fragen. Probieren wir das mal indem wir mal ganz dreist die Root-Name-Server nach der IP-Adresse für `www.danisch.de` fragen (siehe hierzu auch die Abbildung 5.1 auf Seite 55:

```
> set type=a
> server h.root-servers.net
Standardserver:  h.root-servers.net
Address:  128.63.2.53

> www.danisch.de
Server:  h.root-servers.net
Address:  128.63.2.53
```

```
Name:    www.danisch.de
Served by:
- a.nic.de
          194.0.0.53
          de
- c.de.net
          208.48.81.43
          de
- f.nic.de
          81.91.164.5
          de
- l.de.net
          89.213.253.189
          de
- s.de.net
          195.243.137.26
          de
- z.nic.de
          194.246.96.1
          de
```

Etwas hochinteressantes ist da nun passiert. Der Root-Name-Server hat uns zwar die Frage nicht beantwortet, aber er hat uns an andere Name-Server verwiesen, und zwar die für die Domain `de`, also die für Deutschland. *An diese 6 Nameserver haben die Root-Name-Server die Domain für Deutschland delegiert.*

Wir greifen uns nun willkürlich einen dieser 6 Nameserver für de heraus, geben zur Klarheit gleich dessen IP-Adresse an und fragen einen von denen nach der IP-Adresse für `www.danisch.de` (die angegebene IP-Adresse des Servers 81.91.164.5 ist die IP-Adresse des Nameservers f.nic.de für Deutschland, wie wir aus der obigen Antwort des Root-Name-Servers wisse, bitte vergleichen[1]):

```
> set type=a
> server 81.91.164.5
Standardserver:  [81.91.164.5]
Address:  81.91.164.5

> www.danisch.de
Server:  [81.91.164.5]
Address:  81.91.164.5

Name:    www.danisch.de
Served by:
- robotns3.second-ns.com
```

[1] Das ist natürlich der Stand der Dinge zu dem Zeitpunkt, zu dem ich das hier schreibe. Sollten die IP-Adressen oder Servernamen zum Zeitpunkt des Experimentierens anders liegen, muß man sie natürlich entsprechend anpassen.

```
          danisch.de
- robotns2.second-ns.de

          danisch.de
- ns1.first-ns.de

          danisch.de
- dns.rackland.de
          88.198.248.38
          danisch.de
```

Schon wieder haben wir keine Antwort auf die Frage erhalten, wurden aber schon wieder an andere Server verwiesen. Weil die Deutschland-Domain de die Domain danisch.de weiterdelegiert hat, nämlich an die oben genannten vier Nameserver[2]. Wir kommen der Sache näher. Also fragen wir erneut, diesmal einen dieser vier Nameserver für danisch.de:

```
> set type=a
> server 88.198.248.38
Standardserver:  [88.198.248.38]
Address:  88.198.248.38

> www.danisch.de
Server:  [88.198.248.38]
Address:  88.198.248.38

Name:    www.danisch.de
Address:  88.198.248.38
```

Und jetzt hat es endlich geklappt, wir haben die Antwort erhalten. Im dritten Schritt haben wir eine Antwort erhalten. Zu `www.danisch.de` gehört die IP-Adresse 88.198.248.38. Zwei Besonderheiten fallen an dieser Antwort noch auf:

- Sowohl der Nameserver für danisch.de, den wir hier gefragt haben (dns.rackland.de), als auch www.danisch.de haben dieselbe IP-Adresse. Warum denn das? Na, weil derselbe Rechner sowohl den Webserver für mich und einige andere Domains wie rackland.de abgibt, als auch den DNS-Server. Probieren Sie das Spiel mit anderen Servernamen anderer Webseiten aus und Sie werden sehen, daß es meistens nicht so ist und die Adressen sich meistens unterscheiden.

[2] Daß der Server hier nur für einen der vier Nameserver die IP-Adresse mitgegeben hat, hat technische Gründe, die ich hier jetzt nicht weiter vertiefen will. Wir könnten sie über separate Anfragen auflösen, benutzen aber einfach die eine IP-Adresse, die wir bekommen haben.

- Bei den Experimenten in Kapitel 4 stand in den DNS-Antworten stets ein „Nicht autorisierte Antwort:". Das steht hier nicht mehr. Warum nicht? Weil das jetzt im Gegensatz zu früher eine sogenannte „Authoritative Answer“ war. Mehr dazu in Abschnitt 5.5.

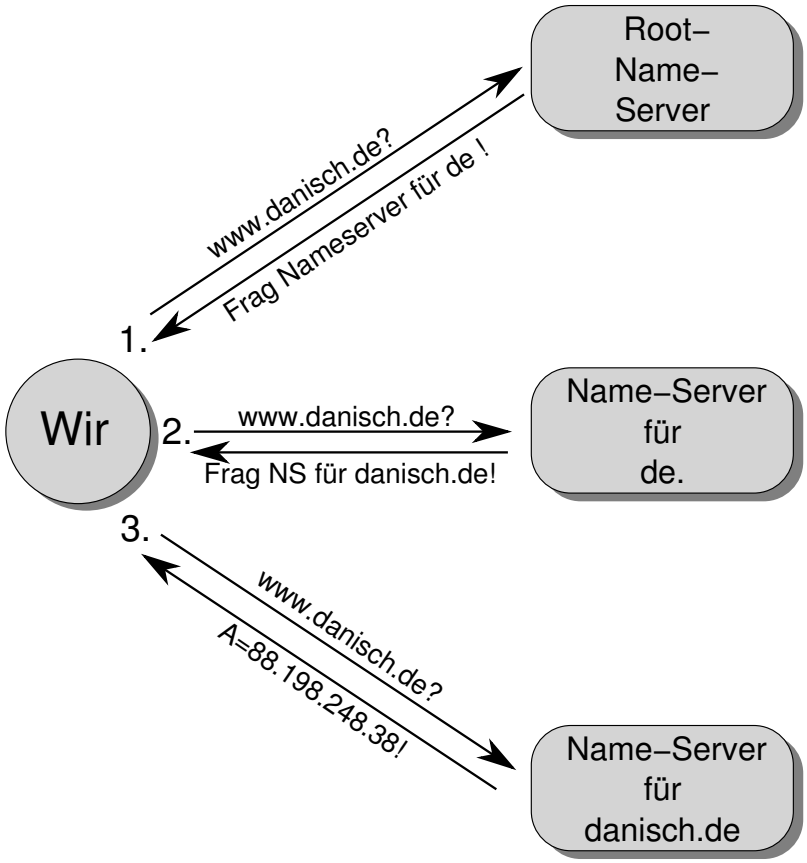

Abbildung 5.1: Die Delegation für www.danisch.de wie im Text dargestellt

Übung: Spielen Sie das Spiel von den Root-Name-Servern bis zum Auffinden der IP-Adresse mit verschiedenen Hostnamen verschiedener Länder oder anderer TLDs wie .com, .edu, .org durch.

Übung: Manche Massenhoster für Privatkundenwebseiten lassen sich deren Domains nicht delegieren, sondern tragen den A-Record für `www.*.de` direkt in der Domain für Deutschland ein. Finden Sie eine solche nicht-delegierte Domain?

Wer jetzt mitgedacht hat, dem ist vielleicht ein ganz subtiles Problem aufgefallen: Einer der Nameserver heißt `dns.rackland.de` und ist nicht nur für meine Domain danisch.de, sondern eben auch für die Domain rackland.de zuständig. Das führt zu einem Henne-Ei-Problem: Die IP-Adresse für `dns.rackland.de` erfährt man nur vom Nameserver für rackland.de, aber um den fragen zu können, müßte man dessen IP-Adressen kennen. Um die-

ses Problem zu lösen kopiert man die nötigen NS- und die A-Records in die Zone des Nameservers für die darüberliegende Domain. Obwohl die Domain rackland.de delegiert wurde und die Nameserver für de damit nicht mehr zuständig sind, enthalten sie den A-Record für `dns.rackland.de`, weil man ihn sonst nicht finden könnte. Diese zusätzlichen Records, die helfen, diese ganzen Delegationen „zusammenzukleben“ nennt man Glue Records.

Übung für Fortgeschrittene: Verifizieren Sie, daß die A-Records für `dns.rackland.de` sowohl auf den Servern, an die delegiert wurde, als auch als Glue Record vorliegt.

5.4 Iterative und Rekursive Abfragen

Gelegentlich hört man von iterativen und rekursiven DNS-Abfragen. Wie in Abschnitt 7 gezeigt wird, unterscheiden sich die *Anfragen* selbst nur in einem einzigen Bit. Der Fragende kann angeben, ob er die Antwort gerne rekursiv haben möchte oder nicht. Deshalb haben wir oben für Experimente auch die Option norecurse gesetzt, um genau dieses Bit abzuschalten.

Die Begriffe iterativ und rekursiv sind eigentlich nur für Informatiker direkt verständlich. Allgemeinsprachlich heißt eigentlich

iterative Anfrage: Sag mir nur, was Du gerade selbst weißt, ich mache die Arbeit des Herumfragens selbst.

rekursive Anfrage: Erledige Du bitte die Arbeite des Herumfragens und liefere mir nur das Endergebnis

Dieses Abklappern der Nameserver von den Root-Nameservern bis zu den Nameservern, die uns tatsächlich Antwort geben können, nennt man die interative Abfrage. Wir haben uns die Arbeit selbst gemacht, um das DNS besser zu verstehen.

Solche *interativen* Anfragen sind für Endnutzer des DNS aber nicht die Regel. Normalerweise hat man im Firmennetz einen zentralen Nameserver im LAN oder als Privatanwender den Nameserver des Providers, der bereit ist, diese Arbeit des Abklapperns für einen zu übernehmen, und dann nur das fertige Ergebnis zu liefern. Wenn man das möchte, daß ein anderer Nameserver für einen diese Arbeit übernimmt, dann stellt man eine *rekursive* Anfrage, also eine in der dieses Bit gesetzt ist und damit die Bitte signalisiert wird, der Server möge doch die lästige Arbeit des Abklapperns für einen erledigen.

Einen DNS-Server, der so konfiguriert ist, daß er bereit ist, für einen diese Arbeit zu erledigen, an den man also rekursive Anfragen stellen kann, nennt man Resolver, während man Server genaugenommen nur die nennt, die eigene Daten anbieten. Ein Server kann aber auch beides zugleich sein.

Diese Technik hat Vorteile. Es entlastet den Arbeitsplatzrechner und Rechenarbeit, Netzwerkverkehr und Speicheraufwand, was zur Zeit der Entwicklung

des DNS kostbar und teuer war. Laufen alle Anfragen einer Firma oder einer Universität über einen zentralen Resolver, dann hat man Geschwindigkeitsvorteile, weil der Resolver sich die Zwischenergebnisse merkt, bis deren mitgeliefertes Haltbarkeitsdatum, die Time to Live abgelaufen ist. Betrachten wir das Beispiel im letzten Abschnitt und die Abbildung 5.1 auf Seite 55. Ein Resolver, der gerade keine Daten gespeichert hat, würde genau so vorgehen, falls ihn jemand nach `www.danisch.de` fragt. Er würde die drei Schritte durchführen und das Endergebnis als Antwort zurückschicken.

Hätte aber kurz zuvor jemand anderes nach einer anderen deutschen Domain gefragt, etwa `www.beispiel.de`, dann hätte der Resolver noch gewußt, daß die Root-Server die Domain de an andere Nameserver delegiert hat und an welche. Er hätte also direkt diese gefragt und den ersten Schritt ausgelassen, weil er die Antwort auf die erste Frage schon kannte.

Hätte zufällig kurz zuvor jemand anderes einen anderen Domainnamen aus der Domain danisch.de abgefragt, beispielsweise weil mir jemand eine E-Mail geschickt hat, dann hätte der Resolver auch gewußt, daß die Domain danisch.de delegiert ist und an wen. Er hätte also auch den zweiten Schritt in Abbildung 55 weggelassen, weil er die Antwort schon wußte.

Und hätte kurz zuvor jemand anderes (oder wir selbst) schon auf `www.danisch.de` gesehen, dann hätte der Resolver gleich alle drei Schritte weggelassen und direkt mit der IP-Adresse geantwortet.

Der Vorteil eines Resolvers liegt also darin, daß er Antworten zwischenspeichert, und zwar umsomehr, desto mehr ihn um Auflösungen anfragen. Und je mehr er zwischengespeichert hat, desto schneller und effizienter kann er antworten. Rekursive Anfragen an einen Resolver zu stellen kann also viel schneller die Antwort bringen als wenn man sich selbst iterativ auf die Suche macht.

5.5 Was ist ein Authoritative Nameserver?

Wir haben im Kapitel 4 für fast jeden Record-Typ Experimente durchgeführt, und immer kam dabei mit der Antwort die (deutsche oder englische) Meldung

> Nicht autorisierte Antwort:

Eigentlich heißt es non-authoritative answer.

Betrachten wir aber im Abschnitt 5 das Abfragebeispiel auf Seite 54, in dem wir die Antwort für www.danisch.de bekommen haben, dann steht das da nicht dabei. *Das war authoritative answer.*

Wo ist der Unterschied?

Bei den Experimenten in Kapitel 4 haben wir einfach den nächstbesten Resolver gefragt, an dem unser Rechner eben gerade hing. Dieser Resolver konnte

die Anfragen nur beantworten, weil er selbst iterative Anfragen gestellt und die Antwort herausgefunden hatte. Er war aber nicht selbst für die abgefragten Domains zuständig, er war nicht vom Domaininhaber autorisiert, und sie wurden nicht an ihn delegiert. Deshalb kam die Antwort von einem nicht autorisierten Server, nur einem Resolver. Damit man das erkennt, wurde sie als non-authoritative markiert.

Das ist manchmal wichtig. Der Resolver beantwortet die Frage aus seinem Gedächtnis, solange die Time to Live nicht abgelaufen ist. Trotzdem könnte der Domaininhaber die Daten in der Zwischenzeit geändert haben. Trotz des Haltbarkeitsdatums sind die Daten im Resolver also nicht unbedingt frisch und aktuell. Dann ist ein Resolver natürlich anfälliger gegen Angriffe wie Cache Poisoning. Und dazu könnte natürlich der Betreiber des Resolvers selbst ein böser Bube sein, der einen angreifen will oder durch Sperrgesetze verpflichtet ist, veränderte Daten herauszugeben. Non-authoritative Answers sind also Auskünfte, die nicht aus erster Hand stammen.

Im Beispiel oben in Abschnitt 5 dagegen haben wir uns selbst durchgefragt, bis wir direkt bei den Nameservern der Domain danisch.de angekommen sind, und dann haben wir dort direkt gefragt. *Wir haben also nicht irgendeinen Zwischenspeicher gefragt, sondern bei den Servern, die der Domaininhaber direkt befüllt, an die delegiert wurde. Die Server, die autorisiert sind, für diese Domain zu sprechen. Die Daten aus erster Hand liefern.* Da ist die Antwort dann auch aktuell.

Aber Vorsicht: Nichts hält einen Non-Authoritative Server, den ein Angreifer unter seiner Kontrolle hat, davon ab, das authoritative Bit zu setzen, damit es so aussieht, als wäre es eine Antwort aus erster Hand.

5.6 Was sind Primary und Secondary Nameserver?

Eigentlich muß man nur wissen, daß sie zwei Konfigurationsvarianten für Authoritative Nameserver sind. Für das Verständnis des DNS ist die Unterscheidung ziemlich unbedeutend, man kann das problemlos überlesen. Wer es aber trotzdem wissen will:

Wir haben in Abschnitt 5 gesehen, daß es für Domains meist nicht nur einen, sondern viele Nameserver gibt: Die Root-Domain hat 13 Root-Name-Server, Die Deutschland-Domain de hat 6 Nameserver, und danisch.de hat 4 Nameserver. Normalerweise kann man eine Domain nur anmelden, wenn man mindestens 2 Nameserver betreibt, für Top Level Domains sogar mehr.

Das bedeutet, daß der Domaininhaber mehrere Nameserver mit Daten befüllen und damit auch bei jeder Änderung mehrere Nameserver aktualisieren muß. Das ist nicht nur lästig und aufwendig, erfahrungsgemäß ist es auch fehleranfällig und führt zu differierenden Daten. Und glauben Sie mir aus langjähriger Berufserfahrung: Authoritative Nameserver einer Domain, die unterschiedliche Daten halten, können ganz üble Effekte auslösen, die sehr schwer

einzugrenzen und zu erkennen sind. Man sollte sorgfältig darauf achten, daß die alle den gleichen Datenbestand haben. Ein Automatismus zum Abgleich wäre hilfreich.

Es gibt ein zweites Problem. Oft betreibt man selbst gar nicht so viele eigene DNS-Server, wie man für seine Domain haben möchte. Deshalb kauft man dies als Dienstleistung bei einem Provider ein, der auch Authoritative Nameserver für Kunden zur Verfügung stellt. Die sind dann meist für ganz viele Domains zuständig und damit viel wirtschaftlicher, als wenn man selbst mehrere Server betreiben würde. Außerdem haben solche zugekauften Serverdienste noch einen anderen, sehr wichtigen Vorteil: *Sie stehen woanders, in anderen Netzen, an anderen Orten, mit anderen Stromversorgungen.* Selbst wenn das eigene Netzwerk völlig ausfällt, funktionieren die externen Nameserver immer noch. Eine feine Sache. Auch für meine Domain danisch.de betreibe ich nur einen Nameserver selbst, die anderen drei sind eine von einem Provider zugekaufte Dienstleistung. Weil aber der Provider überhaupt nicht damit einverstanden wäre, wenn jeder seiner Kunden auf seinen Servern herumkonfiguriert, und ich keine Lust habe, jede Änderung über eine Webseite mit der Maus einzuklicken, braucht man einen Automatismus.

Um dieses Problem zu lösen hat man im DNS einen solchen Automatismus entwickelt, den sogenannten *Zonentransfer*. Der heißt so, weil man die Daten, die man auf einem Nameserver für eine Domain und nicht-delegierte Unterdomains einfüllt, als *Zone*, und bei manchen Nameservern diese Datei, in der sie stehen, als *Zonendatei* bzw. *Zonendaten* bezeichnet.

Der Zonentransfer ist eine Methode, mit dem die Authoritative Nameserver untereinander diese Daten abgleichen. Man muß nur noch einen Nameserver manuell konfigurieren, den *Primary Nameserver*. Die anderen Nameserver können sich die Daten dann von diesem per Zonentransfer herunterladen und sind dann auf demselben Datenbestand. Diese nennt man *Secondary Nameserver*. Da dieser Zonentransfer regelmäßig und automatisiert abläuft, kann man nicht vergessen, die Server auf den gleichen Stand zu bringen. Man kann also seinen eigenen Nameserver wie gewohnt mit Daten befüllen, und die Nameserver des Providers holen sich die Daten einfach per Zonentransfer. Feine Sache.

Für den Abfragenden im Internet spielt es aber keine Rolle, welcher der Nameserver ein Primary oder Secondary ist. Abgesehen von einer leichten Verzögerung bei der Änderung von Daten kann man von außen keinen Unterschied sehen. Die Unterscheidung zwischen Primary und Secondary ist nur für den Domaininhaber von Bedeutung, für den Rest der Welt ist er bedeutungslos, man kann das völlig ignorieren.

Es gibt dabei noch einen Spezialfall, den sogenannten *Hidden Primary Nameserver*. Auch das ist für Außenstehende bedeutungslos und betrifft nur den Domaininhaber und seinen Provider.

Wie ich eben beschrieben habe, ist eine typische Konstellation, daß man einen eigenen Nameserver betreibt und mit Daten befüllt, und sich mehrere externe

Secondary Nameserver dazu anmietet, die mit entsprechender Leistungsfähigkeit für das weltweite DNS den Nameserver für eine Domain spielen, und die sich die Zonendaten per Zonentransfer von dem Primary Nameserver herunterladen. Äußerlich sind die Nameserver durch nichts zu unterscheiden.

Nun kann es aber sein, daß man nicht möchte, daß der eigene Primary Nameserver weltweit sichtbar ist und von der ganzen Welt per DNS befragt wird. Gründe dafür gibt es viele. Vielleicht ist er zu schwach dafür. Vielleicht ist er nicht immer in Betrieb. Vielleicht will man das aus Sicherheitsgründen nicht. Vielleicht gefällt es einem einfach nicht.

Was wäre nun, wenn man die Secondary Nameserver des Providers unverändert die Daten von diesem Primary herunterladen läßt, aber für die Delegation an die Domain nur noch die Secondary Nameserver einträgt und nicht mehr den Primary? Betrachten wir nochmal die Abbildung 5.1 auf Seite 55. Der übergeordnete Nameserver würde Fragende nur noch an die Secondaries verweisen, nicht mehr an den Primary Nameserver. Niemand im Internet wüßte noch, daß es da einen Nameserver mehr gibt, als man von außen sieht, nur noch die Secondaries wären sichtbar. Trotzdem würden diese noch immer regelmäßig ihre Zonen vom Primary herunterladen. Unser Primary bekäme keinerlei Anfragen mehr von außen und wäre nur noch für die Zonentransfers der Secondaries zuständig. Einen solchen Primary Nameserver, der von außen nicht mehr sichtbar ist, nennt man *Hidden Primary*.

6 Erweiterungen, Neuerungen, Huckepacklösungen

Das Kapitel ist in Arbeit und wird in einer der nächsten Versionen erscheinen.

7 Das DNS-Protokoll

Das Kapitel ist in Arbeit und wird in einer der nächsten Versionen erscheinen.

8 Schwächen, Schwindel, Angriffe

Das Kapitel ist in Arbeit und wird in einer der nächsten Versionen erscheinen.

9 Internet-Sperren auf DNS-Basis

Das Kapitel ist in Arbeit und wird in einer der nächsten Versionen erscheinen.

Stichwortverzeichnis