NAMENSDIENSTE

SYSTEMTECHNIK (DEZENTRALE SYSTEME & SYSTEMINTEGRATION UND INFRASTRUKTUR)

12. OKTOBER 2015

STEFAN ERCEG & MARTIN KRITZL
5BHITT

Inhalt

1 Einführung[1]

In Computersystemen, vor allem bei verteilten Systemen, ist die Verwendung von Namensdiensten unumgänglich. Sie bewerkstelligen sowohl die gemeinsame Nutzung und Identifikation von Ressourcen als auch die Referenzierung auf den Standort einer Entität. Dabei wird zur Benennung dieser Einheiten ein künstlicher Name verwendet, welcher durch den Namensdienst in eine Adresse aufgelöst wird. [1, p. 208] [2, p. 159] [3, p. 413]

„Ein Dienstnutzer kann auf diese Weise mit Hilfe des logischen Namens auf einen Dienst zugreifen, ohne dessen genaue Adresse zu kennen bzw. an eine bestimmte Instanz des Dienstes mit fester Adresse gebunden zu sein.“ (A. Schill und T. Springer, Verteilte Systeme, p. 159)

Durch die unabhängige Erreichbarkeit zwischen den Clients und der Entitäten kann ein dynamisches Netz an Services zur Verfügung gestellt werden. Aufgrund der abstrakten Benennung wird eine Ortsänderung der Entität vom Client nicht bemerkt. Dieser Aspekt bietet zusätzlich die Möglichkeit einen replizierten und lastausgeglichenen Dienst zu implementieren. [2, p. 159]

Durch die Abstrahierung durch einen künstlichen Namen wird nicht nur die Abhängigkeit verringert, sondern es kann auch zur besseren Lesbarkeit des Benutzers führen. Das bekannteste Beispiel dafür stellen der URLs im World Wide Web dar. [1, p. 208]

Die Implementierung von Namenssystemen ist grundsätzlich in zwei Kategorien einzuteilen, die beide jeweils ihre Vor- als auch Nachteile haben.
Lineare Benennung besteht aus einer Ansammlung von Objekten, welche hierarchisch gesehen alle auf derselben Ebene angesiedelt sind. Die Kommunikation innerhalb dieses Systems ist auf verschiedene Arten implementiert worden.
Unter der hierarchischen Benennung versteht man wiederrum ausgehend von einem Wurzelknoten die Weiterleitung über Verzeichnisknoten in untere Ebenen, bis man die gewünschte Entität gefunden hat. [1, pp. 211, 223, 224]

[1] geschrieben von Erceg & Kritzl

2 Lineare Benennung[2]

2.1 Aufbau

Bei der linearen Benennung sind alle verfügbaren Knoten nicht hierarchisch, sondern auf gleicher Ebene vorhanden. Somit existiert keine Struktur beim Aufbau des Systems, sondern die Entitäten werden den Knoten auf verschiedenster Weise zugeteilt.

Im Folgenden werden 4 Implementierungen der linearen Benennung genauer beschrieben.

[1, p. 211]

2.2 Implementierungsarten

2.2.1 Broadcasting und Multicasting

An das angeschlossene Netzwerk wird eine Broadcast- oder Multicastanfrage geschickt, die in weiterer Folge von demjenigen Rechner beantwortet wird, dem die Adresse der gesuchten Entität bekannt ist. Als Antwort erhält der Client die genaue Adresse der gewünschten Entität und kann diese damit direkt ansprechen.

Der Vorteil dieser Variante ist die unkomplizierte Auflösung des Namens, da man keine Algorithmen berechnen bzw. Routen folgen muss. Jedoch ist die Verwendung in größeren Netzwerken performancemäßig impraktikabel. Dies kommt aufgrund der großen Datenmenge und der Abwicklung von Anfragen, die nicht in die Zuständigkeit des Servers fallen, zu Stande. Mit der Zuständigkeit wird beschrieben, ob dem Server die Adresse der gewünschten Entität bekannt ist.

[1, p. 212]

[2] geschrieben von Erceg & Kritzl

2.2.2 Zeiger zur Weiterleitung

Bei dieser Implementierungsart werden Zeiger zur Lokalisierung der gesuchten Entität verwendet. Zuerst wird über einen herkömmlichen Namensdienst die jeweilige Ressource lokalisiert und geprüft, ob auf dieser Adresse die Entität liegt. Ist die Entität dort nicht zu finden, hat diese den Standort geändert und einen Zeiger auf die neue Adresse hinterlassen.

Die Implementierungsart weist einen großen Vorteil, jedoch auch einige Nachteile auf. Vorteilhaft ist hier die Schlichtheit aufgrund der sequentiellen Nachverfolgung der Entität. Dadurch können jedoch lange Ketten entstehen, welche die Performance verringern. Diese können jedoch durch geeignete Gegenmaßnahmen gekürzt werden. Somit wird ein Server, auf den keine Referenz mehr existiert, aus der Kette genommen. Dazu wird der Pointer der vorherigen Instanz direkt mit dem nachfolgenden Server verbunden und damit ein Glied der Kette entfernt.

Ebenfalls müssen die Pointer vom Server solange wie nötig gespeichert werden. Deswegen kann es vorkommen, dass Zeiger aus diversen Gründen gelöscht werden bzw. nicht mehr existieren und deshalb die Ermittlung der Adresse nicht mehr durchgeführt werden kann.

[1, pp. 213, 214]

2.2.3 Heimatgestützte Ansätze

Zur Lokalisierung wird ein stationärer Heimatstandort verwendet, der im Normalfall direkt die Adresse der gesuchten Entität zurückgibt, unter der Voraussetzung, dass dieser sich am selben Standort befindet. Als Heimatstandort wird meist der Ort gewählt, an dem die Entität erstellt wurde.

Sollte sich die Entität nicht mehr an der alten Stelle befinden, bekommt diese eine sogenannte Care-of-Adresse, welche an den stationären Heimstandort übermittelt wird. Somit wird die Kommunikation dieser beiden Orte gewährleistet und ein Client kann über den Heimatstandort die Adresse der entfernten Entität feststellen.

Im Vergleich zu den „Zeigern zu Weiterleitung“ ist die zu überwindende Entfernung bis zur Ermittlung der Entität wesentlich kürzer, da nur maximal zwei Glieder vorhanden sein können. Trotzdem kann diese zusätzliche Distanz eine wesentliche Verlängerung der Latenz bedeuten. Zudem muss sichergestellt werden, dass der Heimatstandort ständig erreichbar ist und seinen Standort nicht ändern kann.

[1, pp. 215, 216]

2.2.4 Verteilte Hash-Tabellen

Wie es bei den weitergeleiteten Zeigern der Fall ist, wird auch bei verteilten Hash-Tabellen eine Menge an Knoten zur Auflösung der Adresse einer Entität verwendet. Die Verknüpfung der jeweiligen Knoten erfolgt durch Implementierung eines Ringes. [1, pp. 216, 217]

2.2.4.1 Mechanismus

Für die verteilten Hash-Tabellen bestehen mehrere Implementierungsarten. Wir betrachten in diesem Kapitel die Chord-Implementierung genauer.

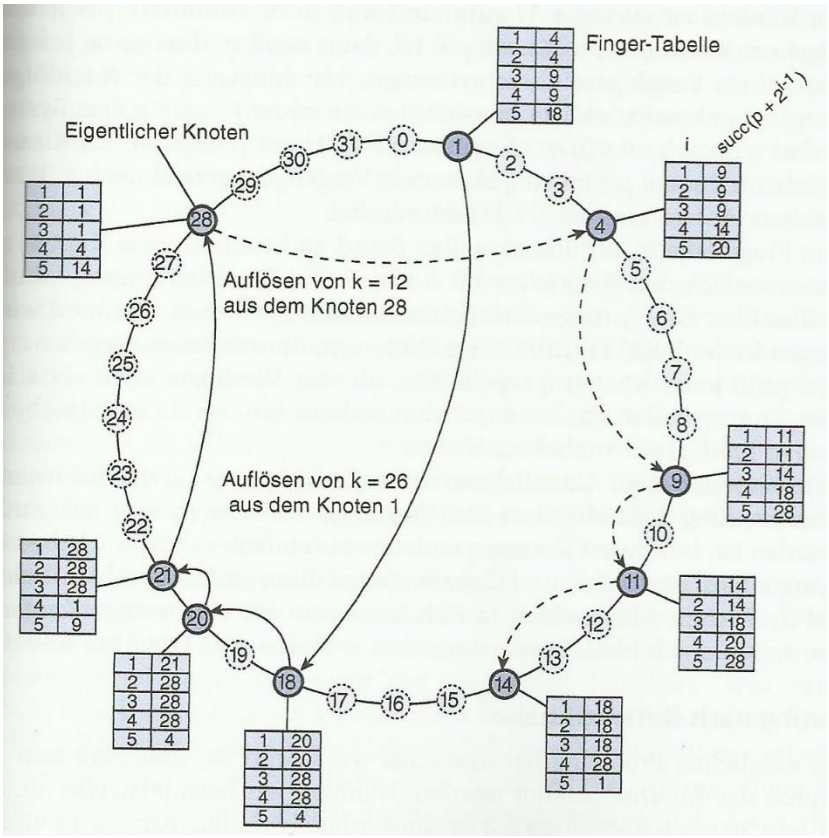

Abbildung 1: Chord-System [1, p. 217]

Jedem Knoten im Ring wird eine bestimmte Anzahl an Entitäten zugewiesen. Der Einflussbereich des jeweiligen Knotens liegt zwischen dessen darunterliegenden Knoten und dem eigenen. In der Abbildung 1 sind beispielsweise die Schlüssel 2, 3 und 4 im Einflussbereich des Knotens 4, da Knoten 1 sein darunterliegender ist.

Die Finger-Tabelle ist je nach Knoten `p` unterschiedlich. Sie besitzt 2 Spalten: In der einen wird ein fortlaufender Index `i` gespeichert, in der anderen bestimmte Werte durch die Formel $succ(p + 2^{i-1})$ ermittelt. Mit dieser Formel werden die exponentiellen Nachbarn berechnet. Sollte der Knoten mit dem berechneten Wert nicht existieren, wird der nächsthöhere verfügbare Knoten gewählt.

Jeder Knoten behält seine Nachbarn stets im Auge und bekommt dortige Änderungen sofort mit und kann darauf die Finger-Tabelle aktualisieren. Die gewünschte Entität wurde gefunden, wenn die ID des Knoten größer als der Schlüssel der gesuchten Entität ist. Bis zu diesem gesuchten Knoten wird immer der nächstniedrigere Knoten zu dem gewünschten Schlüssel angesprochen.

[1, pp. 216, 217], [4, pp. 5, 6]

2.2.4.2 Optimierung

Ein allgemeines Problem bei der Verwendung von verteilten Hash-Tabellen können die weiten Übertragungsstrecken zwischen den jeweiligen Knoten darstellen. Somit kann es zu häufigen Nachrichtenübertragungen über weite Strecken bei der Suche nach einer bestimmten Entität kommen, da die Nachbarn geografisch weit voneinander entfernt sind.

Aus diesem Grund wird versucht, eine topologiebasierte Zuweisung von Knotenschlüssel zu implementieren. Dadurch sollen die Abstände zwischen den logisch aneinander liegenden Knoten verringert werden. Dies ist problematisch, da die Topologie eines Ringes schwer auf eine Fläche anzuwenden ist.

Es besteht die Möglichkeit, einem Knoten nicht nur einen einzelnen Nachfolger, sondern mehrere in einer Liste zu speichern. Dadurch können Ausfälle einzelner Knoten leichter überbrückt werden. Dies hat zur Folge, dass sich die Finger-Tabellen der einzelnen Knoten häufiger aktualisieren.

[1, pp. 218, 219]

3 Hierarchische Benennung[3]

3.1 Namensräume

Namensräume dienen zur Strukturierung aller Namen, die vom Dienst erkannt werden. [1, p. 223], [3, p. 418]

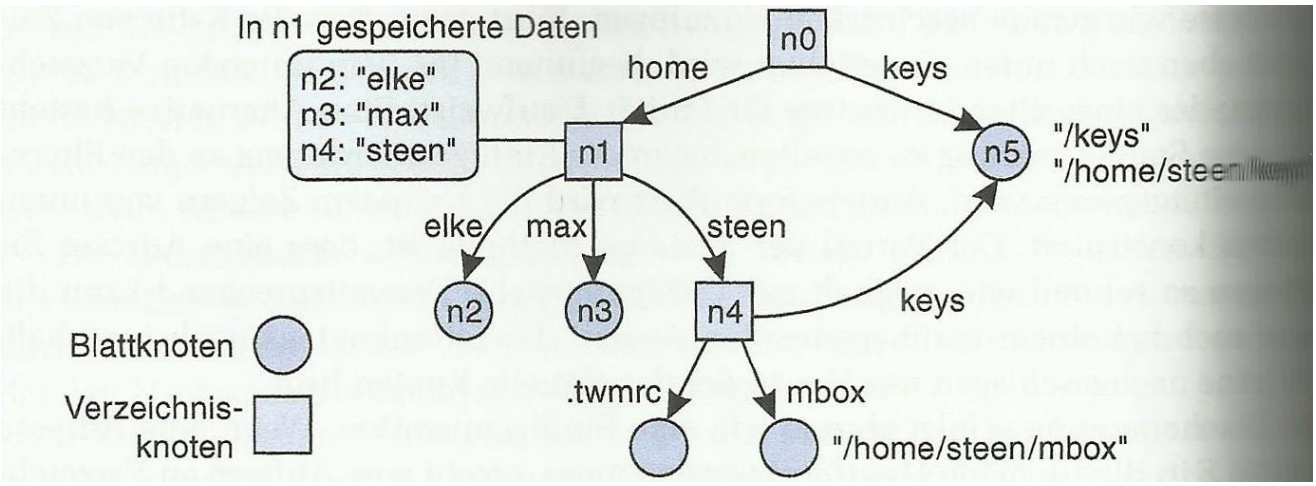

Abbildung 2: allgemeiner Namensgraph mit einem einzelnen Wurzelknoten [1, p. 224]

Die obere Abbildung zeigt mehrere Verzeichnis- und Blattknoten. Der Unterschied zwischen den beiden Arten ist folgender:

- **Verzeichnisknoten** besitzen eine oder mehrere ausgehende Kanten, welche alle mit einem bestimmten Namen deklariert sind. Jeder Verzeichnisknoten speichert in einer sogenannten Verzeichnistabelle die Kantenbeschriftung und die Bezeichnung des Knoten von jeder einzelnen ausgehenden Kante.
- **Blattknoten** stellen hingegen eine Entität dar und besitzen keine ausgehenden Kanten. Sie speichern Informationen zu der Entität, wie beispielsweise die Adresse oder der Zustand.

Bei Abbildung 2 stellt Knoten „n0“ den Wurzelknoten des Namensgraphen dar. In der Regel besitzt nahezu jedes Namenssystem nur einen einzelnen Wurzelknoten.

[1, pp. 223, 224]

Namensdienste werden standardgemäß durch einen Namensserver implementiert. Befindet sich das verteilte System in einem lokalen Netzwerk, kann ein Namensdienst durch einen einzigen Namensserver umgesetzt werden. Bei einer Implementierung eines Namensraumes, bei der das System sehr umfangreich ist, viele Entitäten existieren und diese zudem vielleicht sogar weit voneinander entfernt sind, wird die Umsetzung jedoch auf mehrere Namensserver verteilt. [1, p. 230]

[3] geschrieben von Erceg

3.1.1 Unterteilung in Schichten

Da Namensräume bei sehr großen verteilten Systemen gewöhnlich hierarchisch strukturiert werden, haben Cheriton und Mann ihn im Jahre 1989 in folgende drei logische Schichten unterteilt:

- Globale Schicht

  Diese besteht aus jenen Knoten, die die höchste Ebene des Namensraumes darstellen. Darunter fallen der Wurzelknoten und vor allem seine Kindknoten, da ihm diese logisch ziemlich nahe stehen. Ein besonderes Merkmal dieser Knoten stellen deren Verzeichnistabellen dar, weil diese sehr selten geändert werden.

- Administrationsschicht

  Die Knoten, die unter diese Schicht fallen, bilden Gruppen von Entitäten ab, welche alle derselben Organisation angehören. Ein Verzeichnisknoten kann beispielsweise die Abteilung einer Organisation darstellen. Änderungen treten zwar öfters als bei der globalen Schicht auf, jedoch nicht so oft wie bei der Managementschicht.

- Managementschicht

  Diese Schicht kann im Gegensatz zu den anderen beiden nicht nur von Systemadministratoren, sondern auch von den einzelnen Clients verwaltet werden. Aufgrund dieser Tatsache befinden sich hier Knoten, die regelmäßig geändert werden. Hier sind mehrere Arten von Knoten zu finden, beispielsweise jene, die Hosts im lokalen Netzwerk oder benutzerspezifische Verzeichnisse und Dateien darstellen. Ebenso gehören zu dieser Schicht die Knoten dazu, die Dateien, welche gemeinsam für Bibliotheken genutzt werden, wiederspiegeln.

[1, pp. 230, 231]

3.2 Namensauflösung

Namensauflösung beschreibt im Allgemeinen den Durchlauf eines oder mehrerer Namensgraphen. Dies geschieht durch das nacheinander folgende Nachschlagen der Informationen eines Pfadnamens. [1, p. 226], [3, p. 422]

3.2.1 Implementierungsarten

3.2.1.1 Iterativ

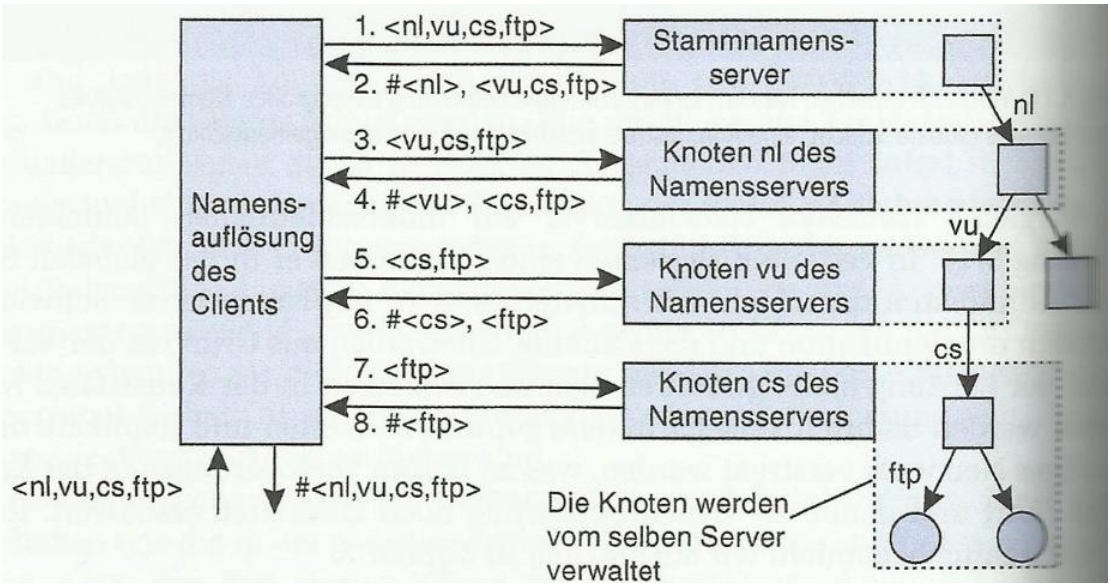

Abbildung 3: Prinzip der iterativen Namensauflösung [1, p. 234]

Die Voraussetzung hier ist, dass der lokale Namensserver (Resolver) des Clients die Adresse des Root-Servers kennt. Sobald dem Resolver diese bekannt ist, wird folgendermaßen vorgegangen:

1.) Der Name, der aufzulösen ist, wird an den Root-Server gesendet.
2.) Dieser versucht den Namen soweit es geht aufzulösen.
3.) Der Root-Server gibt dem Client die von ihm aufgelöste Bezeichnung und die Adresse des Namensservers, welcher für den weiteren Teil der Auflösung zuständig ist, zurück.
4.) Der Client gibt den restlichen Pfadnamen an diesen Namensserver weiter.
5.) Dieser Namensserver handelt genauso wie der Root-Server.
6.) Der Resolver des Clients sendet den Pfadnamen solange an die jeweiligen Namensserver, bis dieser komplett aufgelöst wurde.

[1, pp. 233, 234], [3, p. 423]

3.2.1.2 Rekursiv

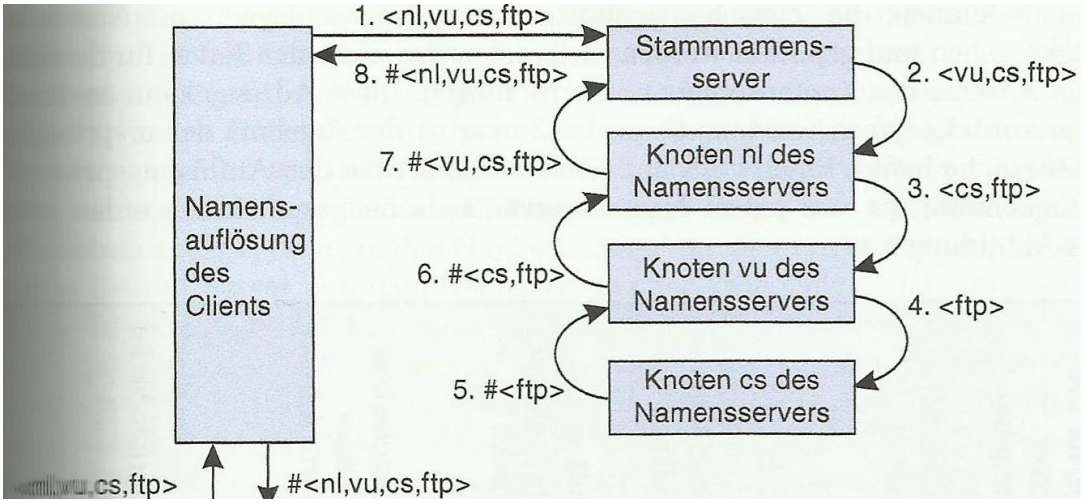

Abbildung 4: Prinzip der rekursiven Namensauflösung [1, p. 235]

Im Gegensatz zum iterativen Ansatz sendet der Resolver des Clients bei der rekursiven Namensauflösung den aufzulösenden Namen an den Root-Server und bekommt ihn aufgelöst auch wieder vom Root-Server zurück. Somit wird nicht jedes einzelne Zwischenergebnis an den Resolver zurückgegeben, sondern der Pfadname vom Root-Server beginnend an die jeweiligen Namensserver weitergeleitet, bis der Pfad vollständig aufgelöst wurde und der Root-Server das Ergebnis zurückschicken kann.

Vorteile dieses Ansatzes:

- Steigerung der Leistung, da sich die Namensserver untereinander ihre Adressen austauschen und nach einiger Zeit diese gelernt haben
  → effektivere Zwischenspeicherung des Ergebnisses
- kostengünstigere Kommunikation hinsichtlich der geografischen Lage, da der Resolver des Clients den Pfadnamen nur an den Root-Server senden muss

Nachteile dieses Ansatzes:

- größere Last für die Namensserver

[1, pp. 234, 235, 237], [3, p. 424]

3.2.1.3 Closure-Mechanismus

Unter dem Closure-Mechanismus versteht man das Suchverfahren nach dem Anfangsknoten, bei der die Namensauflösung beginnen soll. Begonnen wird hier bei einem Wurzelknoten, welcher sich immer weiter in die untere Ebenen bewegt, bis die gesuchte Entität gefunden wurde. [1, pp. 226, 227]

3.2.1.4 Verlinken und Mounting

Aliase kommen bei der Namensauflösung oft zum Einsatz. Darunter versteht man das Vergeben eines anderen Namens für dieselbe Entität.

Aliase können bei hierarchischen Benennungen im Wesentlichen auf zwei verschiedene Arten implementiert werden:

1.) Verwendung von Hard Links

Abbildung 2 veranschaulicht die Verwendung von Hard Links. Dort stellen die beiden Pfadnamen `/keys` und `/home/steen/keys` harte Links dar, die zu Knoten „n5“ verweisen.

Bei diesem Ansatz verweisen mehrere absolute Pfadnamen auf einen bestimmten Knoten im Namensgraphen.

2.) Verwendung von Symbolic Links

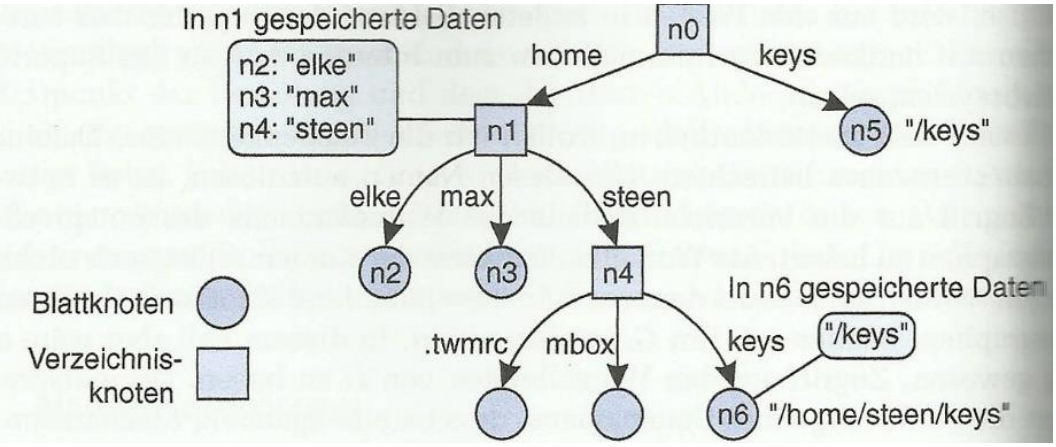

Abbildung 5: symbolische Links in einem Namensgraphen [1, p. 228]

Abbildung 5 veranschaulicht die Verwendung von Symbolic Links. Hier stellt der Pfadname `/home/steen/keys` einen symbolischen Link zu Knoten „n5“, dessen Pfadname `/keys` ist, dar.

Hierbei wird die Entität als Blattknoten angesehen. Dieser Knoten speichert anstatt des Zustandes der Entität den absoluten Pfadnamen.

Da die Namensauflösung nicht nur in einem einzelnen Namensraum, sondern auch über mehrere Namensräume hinausgehen kann, wird jetzt noch ein zusätzlicher Begriff erklärt. Als **Mountpoint** wird jener Verzeichnisknoten bezeichnet, der den Bezeichner eines Verzeichnisknoten aus einem anderen Namensraum speichert. Meistens wird dazu der Wurzelknoten verwendet.

Folgende Informationen sind beim Mounting eines fremden Namensraumes in einem verteilten System notwendig:

1.) Der Name eines Zugriffsprotokolls

    ist erforderlich, um eine Kommunikation mit dem Server des fremden Namensraumes herzustellen

2.) Der Servername

    ist erforderlich, um unter einer bestimmten Adresse den Server erreichen zu können

3.) Der Name des Mountpoint im fremden Namensraum

    ist erforderlich, um die Auflösung des Namens des Mountpoint nach einem Knotenbezeichner durchführen zu können

[1, pp. 227, 228]

4 Anwendungen

4.1 Lineare Anwendungen[4]

4.1.1 Peer-to-Peer

Ein Peer-to-Peer-System stellt eine Anwendung der linearen Benennung dar. Bei solch einem System werden die Knoten einer horizontalen Verteilung unterzogen. Dabei werden alle verfügbaren Entitäten in logisch gleichwertige Teile gegliedert und ermöglichen somit eine Lastverteilung. Grundsätzlich wird zwischen zwei verschiedenen Ansätzen unterscheiden:

- strukturierte Peer-to-Peer-Architekturen

  Hier wird meistens das Prinzip der verteilten Hash-Tabellen umgesetzt. Es wird versucht, den Schlüssel der jeweiligen Entitäten mittels einer Entfernungsmetrik einem geeigneten Knoten zuzuweisen.

- unstrukturierte Peer-to-Peer-Architekturen

  Bei dieser Variante werden die Schlüssel der jeweiligen Entitäten zufällig dem Knoten zugewiesen. Die Verteilung der Nachbarn wird ebenfalls zufällig gewählt.

[1, p. 62 bis 67]

[4] geschrieben von Erceg & Kritzl

4.2 Hierarchische Anwendungen

4.2.1 DNS[5]

DNS ist das wohl bekannteste Beispiel für hierarchische Benennung, welches sich durch die Zunahme der Hosts, die auf Adressen zugreifen wollten, entwickelt hatte. Die Hauptaufgabe von DNS ist die Umwandlung einer URL/Domain in eine IP-Adresse und umgekehrt. [5, p. 182]

4.2.1.1 Aufbau der Domains

Die Struktur von Domains ist wie folgt unterteilt:

- Top-Level Domain:
  - Jedes Land hat eine Eigene (z.B.: `.de` für Deutschland)
  - Für bestimmte Bereiche sind ebenfalls Top-Level Domains vorhanden
- Second-Level Domain:
  - Beschreibt den Namen der Domain
- Subdomain (optional):
  - Beschreibt eine untergeordnete Domain, wie zum Beispiel die Sprache der Webseite (z.B.: www.**en**.name.at)

- Hostname:
  - Spiegelt den Verwendungszweck wieder (z.B. www oder mail)

[5, pp. 183 - 185]

Für Domainnamen sind alle Groß- und Kleinbuchstaben, Zahlen und Bindestriche erlaubt. Jedoch ist die Auflösung der Domain nicht Case-Sensitiv, obwohl die Deklaration von Domains aufgrund von IP-Adressen bei der Übertragung nicht normalisiert wird, um vielleicht in Zukunft Domains Case-Sensitiv zu machen. [5, p. 185]

4.2.1.2 Zonen

In einer sogenannten Zonen-Datei ist die technische Umsetzung des hierarchischen Aufbaus festgehalten. Dazu wird zuerst festgelegt für welchen Bereich der Gesamtstruktur diese Zone zuständig ist und wie die ausgelagerte Namensauflösung weitergeleitet werden muss. Das Kernstück besteht aus der Definition wie Domains in IP-Adressen oder umgekehrt aufgelöst werden sollen. Zusätzlich sind noch Eigenschaften wie die maximale Gültigkeit der Einträge, eine Seriennummer und die Erreichbarkeit des Administrators dieser Zone. [5, p. 186]

[5] geschrieben von Kritzl

4.2.1.3 Übertragung

Die Übertragung geschieht im Normalfall über Port 53 per UDP. Sollte die Antwort über UDP, durch die begrenzte Länge, nicht ausreichend sein, kann diese per TCP wiederholt werden. Zonen-Transfers müssen in jedem Fall über TCP erfolgen. [5, p. 191]

4.2.1.4 Suche

Eine Suche auf einem DNS-Server kann entweder rekursiv oder nicht rekursiv geschehen. Die Weiterleitung auf untergeordnete Domains wird immer vom Server zur Verfügung gestellt.

Sollte eine Anfrage rekursiv von einem Client gefragt werden und der Server kann dies auch bereitstellen, sorgt sich der Server selber um die gesamte Weiterleitung einer Anfrage, die sich auf eine andere Domain bezieht. Somit braucht sich der Client nicht weiter darum kümmern.

Sollte eine Anfrage jedoch nicht rekursiv erfolgen und eine andere Domain gesucht werden, dann antwortet der DNS-Server nur mit einem Verweis zu einem anderen Server, der die Anfrage weiter bearbeiten kann. Somit muss der Client eine erneute Abfrage auf den neuen Server starten, bis er die richtige Antwort erhält.

Um diese Vorgänge effizienter zu machen, werden Einträge für eine definierte Zeit zwischengespeichert.

[5, pp. 192, 193]

4.2.1.5 Ausfallsicherheit

Um eine Ausfallsicherheit von DNS-Servern zu erreichen, werden Master-Slave Konfigurationen verwendet. Diese sollen jedoch für den Administrator keinen zusätzlichen Aufwand darstellen. Dabei stellt der Slave nur eine Kopie des Master-Servers da, um eine zusätzliche Möglichkeit zur Abfrage zu schaffen. Die einzige Einschränkung eines Slave-Servers ist die Verweigerung von Schreibzugriffen, da diese nur auf dem Master-Server geschehen dürfen und eine Konsistenz der Daten zu gewähren. Die Aktualisierung der Slave-Server erfolgt in einem definierten Intervall, in dem die Version des Masters überprüft wird. [5, pp. 192, 193]

4.2.1.6 Sicherheit / DNSSEC

DNS ist aufgrund der unverschlüsselten Übertragung immer wieder ein Ziel von Angriffen. Dabei werden meist die Antworten des DNS-Servers so abgeändert, dass der Client auf eine ganz andere Seite geleitet wird. [5, p. 291]

4.2.1.6.1 Verfügbarkeit

Angriffe können ebenfalls bewirken, den DNS einer Firma auszuschalten und damit die Namensauflösung nach außen zu verhindern. [5, p. 292]

Eine weitere Folge ist die „Unsichtbarkeit“ von draußen, da diese Domain nicht mehr aufgelöst werden kann. Dieser Aspekt tritt aber meist erst Schritt für Schritt ein, da die Domain an verschiedenen Stellen zwischengespeichert ist und erst mit Ablauf einer gewissen Zeit eine erneute Namensauflösung fordert. [5, pp. 292, 293]

Eine weitere Möglichkeit bei DNS-Servern die rekursive Namensauflösung unterstützen, ist die Auflösung eines eigenen DNS-Servers mit einer sehr großen Zonen-Datei. Damit speichert der anzugriefende DNS-Server diese Einträge zwischen und antwortet auf Anfragen dieser Einträge. Nun werden sehr viele Anfragen mit gefälschtem Absender an den Server gesendet und erreicht damit eine Überlastung desselben. Durch die ständige Rückmeldung des Servers hat es den Anschein, als würde dieser der Angreifer und nicht das Opfer sein. [5, pp. 293, 294]

4.2.1.6.2 Manipulation der Antworten

Eine Antwort eines DNS-Servers kann so abgeändert werden, sodass die IP-Adresse auf einen komplett anderen Server oder ins leere zeigt. Dadurch kann eine bestimmte Website quasi offline geschaltet werden, da diese niemand mehr findet. [5, p. 294]

Eine weitaus schlimmere Variante ist die Umleitung auf einen eigenen Server, der genauso wie der originale Server aussieht. Dadurch können sogar Benutzerdaten über eine Anmeldemöglichkeit mitprotokoliert werden. Nach der Anmeldung wird einfach wieder auf die ursprüngliche Website weitergeleitet, um unerkannt zu bleiben. [5, pp. 294, 295]

4.2.1.6.3 Vertraulichkeit

Für einen Angreifer ist es meist von Vorteil die Netzwerkstruktur seines Zieles zu kennen und ist deswegen bemüht die Zonen-Datei des dortigen DNS-Servers zu bekommen. Dazu würde sich vor allem der Zonentransfer zwischen zwei DNS-Servern eignen, da hier die Zone über das Netzwerk geschickt wird. Genau aus diesem Grund sollte diese Kommunikation verschlüsselt sein, obwohl diese dadurch sehr rechenintensiv wird. [5, pp. 296, 297]

4.2.1.6.4 Risikosenkende Architektur

Durch einen geeigneten Aufbau der Netzwerkstruktur lassen sich Risiken, wie oberhalb beschrieben, bis zu einem gewissen Grad verhindern. Grundsätzlich sollte eine strenge Trennung des inneren (LAN) und äußeren (Internet) Teil vorhanden sein. Meist ist die Schnittstelle ein Server, der eine Demilitarisierte Zone darstellt. Somit kann bei einem Angriff schlechter auf den inneren Bereich zugegriffen werden, da diese durch den DMZ-Server getrennt sind. Zusätzlich ist die Verwendung einer Firewall und eines Proxy-Servers anzuraten. [5, pp. 297, 298]

4.2.1.6.5 Transportsicherheit

Die Kommunikation zwischen DNS-Servern für den Austausch von Zonen-Dateien sollte vor Manipulation geschützt sein. Zur Authentisierung einer Nachricht können zwei verschiedene Implementierungen verwendet werden:

TSIG: Aus einer Nachricht und einem privaten Schlüssel, den beide Partner kennen und sicher aufbewahren, wird ein Hash-Wert generiert. Die Nachricht und der erzeugte Hash-Wert werden zum Empfänger übertragen. Dieser bildet aus der Nachricht und dem geheimen Passwort wieder einen Hash-Wert und vergleicht diesen mit dem übergebenen. Stimmen beide überein, wurde keine Veränderung vorgenommen. Die Nachricht ist in sich aber nicht verschlüsselt und kann deswegen mitgelesen werden.

TKEY und Sig(0): Genauso wie bei der TSIG-Variante wird die Nachricht mit einer Signatur, also mit einem Hash-Wert versehen, der aber mit einem Public-Key generiert und mit einem Private-Key wieder aufgelöst werden kann. Diese Implementierung erhöht zwar die Sicherheit, benötigt aber eine höhere Rechenleistung.

[5, pp. 301 - 304]

4.2.1.6.6 DNSSEC

Durch DNSSEC soll eine Authentisierung der Echtheit der Antwort eines DNS-Servers am Client geprüft werden. Bei anderen Applikationen wird dies über eine verschlüsselte Verbindung gehandhabt, jedoch wäre dies bei einer DNS-Abfrage ein viel zu großer Overhead, da auch die Unlesbarkeit einer Antwort nicht notwendig ist. [5, pp. 304, 305]

Die Struktur der Vertrauensbeziehungen wird in die Struktur des DNS eingebunden. Angefangen von der Rootzone werden alle darunterliegenden Top-Level Domains von diesem signiert und können durch einen bekannten Public Key verifiziert werden. Dieses Schema geht so weit hinunter, bis einzelne Zonen oder Subdomains signiert sind. Somit wird jeder Resource Record mit einer Signatur versehen, sodass der Client durch einen Public Key die Korrektheit der Antwort feststellen kann. Bei der Delegation von DNS-Einträgen werden ebenfalls Signaturen zur Authentisierung der Antworten verwendet. [5, pp. 305, 308]

Jede Signatur eines Resource Record hat ebenfalls die Information, ab wann diese gültig ist und wie lange diese verwendet werden darf. Weitere zwei Felder beinhalten den Key Tag des Schlüssels, der bei der Erstellung vergeben wurde, und den Domainnamen, bei dem die Signatur aktualisiert werden kann. [5, pp. 307, 308]

4.2.2 LDAP[6]

Im Grunde ist LDAP nichts weiter als ein Verzeichnisdienst und wird deshalb hier auch so genannt.
Unter einem Verzeichnisdienst wird die hierarchische Datenhaltung von Entitäten verstanden. Dadurch können bestimmte Teile dieses Verzeichnisbaumes der meist ebenfalls hierarchisch orientierten Organisation zugeteilt werden. Somit können beispielsweiße Teilfirmen auf einen kleinen Ausschnitt des gesamten Systems zugreifen und administrieren. Dies kann so weit gehen, dass jeder Benutzer seinen eigenen Bereich für sich zu Verfügung hat. [6, pp. 5,6]

Der Zugriff auf die verschiedenen Abschnitte wird wie bei Dateisystemen mit absoluten, als auch relativen Pfaden, genauso wie einen Namen, der die Entitäten durch andere in dieser Ebene unterscheidet, ermöglicht. [6, pp. 25,26]

Die Identifikation eines Eintrages in einer Hierarchie-Schicht wird über den relativen Namen gewährleistet. Dieser kann sich aus einen oder mehreren Attributen zusammensetzen. Um einen Eintrag in dem gesamten Verzeichnisbaum zu referenzieren, ist die zusätzliche Angabe des Pfades notwendig. [6, pp. 26-28]

Durch den objektorientierten Ansatz, können Erweiterungen - Controls genannt - in Form von Objekten implementiert und verwendet werden. Um diese zusätzlichen Möglichkeiten abzufragen, werden diese sowohl serverseitig als auch clientseitig in die Message-Envelope gespeichert. [6, p. 33]

4.2.2.1 Ausfallsicherheit[7]

Eine Partitionierung des Verzeichnisses auf mehrere Server ermöglicht eine bessere Administration und Lastverteilung des Systems. Dies ist kein Problem, da die Standardisierung von X.500 nur die Client-Server und nicht auf die Server-Server Kommunikation definiert. Somit ermöglicht man die transparente Nutzung von LDAP, egal wie die Aufteilung tatsächlich aussieht. [6, p. 35]

4.2.2.2 Performance[8]

Genauso ist die Replikation von Einträgen auf mehreren Servern möglich. Dies kann aufgrund der Lastverteilung, schnelleren Zugriff aufgrund von Nähe oder der erhöhten Ausfallsicherheit des Gesamtsystems betrieben werden. In den meisten Fällen wird dies durch eine Master-Slave Konfiguration durchgeführt, die sich auf eine bestimmte Ebene des Verzeichnisbaumes beziehen kann. Somit könnte zum Beispiel jede Abteilung eines Betriebs einen eigenen Master-Slave Aufbau haben. [7, pp. 56,62]

[6] geschrieben von Erceg & Kritzl
[7] geschrieben von Kritzl
[8] geschrieben von Kritzl

Die Kommunikation zwischen LDAP-Server und Client verläuft asynchron um die Latenzzeit zu vermindern. Die in unterschiedlicher Reihenfolge ankommenden Antworten, können mit einer MessageID der Abfrage zugeordnet werden und damit die asynchrone Verwendung möglich machen. [6, pp. 31-33]

4.2.2.3 Objektorientierter Aufbau[9]

Durch ein objektorientiertes Datenmodell können in einem X.500-kompatiblen Verzeichnis Elemente eines Datensatzes jederzeit durch eine einfache Vererbung mit weiteren Eigenschaften versehen werden. Dafür wird eine neue Objektklasse erstellt, die die neue Information beinhaltet, und damit den Datensatz erweitert. [6, pp. 6,20]

Mehrfachvererbung ist auf der Instanz Ebene ebenso möglich, jedoch nicht wie in anderen OO-Sprachen die Vererbung von Klassen. Dazu wird ein spezielles Attribut objectClass verwendet. Zudem muss jeder Eintrag der Klasse top (für einen tatsächlichen Eintrag) oder der Klasse alias (für einen Verweis auf einen anderen Eintrag) angehören. [6, p. 21]

Es gibt 3 verschiedene Typen von Objektklassen die definiert werden können:

- Structural: Ist als Basis-Klasse für andere, erbende Klassen geschaffen. Diese darf nur einmal in der Objekthierarchie vorhanden sein
- Auxiliary: Jede normale Klasse, die Eigenschaften einer Structural-Klasse erweitert. Diese muss von einer Basis-Klasse abgeleitet sein
- Abstract: Eine Klasse, die ni9cht instanziiert werden kann. Muss durch eine Auxiliary-Klasse abgeleitet werden, um diese verwenden zu können. Damit spielt diese Klasse eine Vorgaberolle. Schon implementierte Beispiele wären „top“ und „alias“.

[6, pp. 23,24]

4.2.2.4 Kommunikation[10]

Durch die ehemalige Problematik verschiedene Systeme ohne standardisierte Protokolle miteinander kommunizieren zu lassen, musste statt der aufwändigen Kopplung dieser Architekturen eine festgelegte Kommunikation geschaffen werden. Zu diesem Zweck wurde der X.500 Standard definiert. Dort wurde die Verschlüsselung, Authentifizierung, Replikation und Verwaltung von Verzeichnisdaten festgelegt. [6, pp. 8,9]

Das Directory Access Protocol basierte jedoch auf dem OSI-Protokoll-Stack, welches von den meisten Clients nicht implementiert war und diese durch die Komplexität dieses Protokolls überlastet hätte. Deswegen wurde LDAP Lightweight Directory Access Protocol spezifiziert, welches auf dem TCP-Schicht aufsetzt und damit den OSI-Overhead eliminiert. [6, p. 11]

[9] geschrieben von Kritzl
[10] geschrieben von Kritzl

4.2.2.5 Datenmodell[11]

Der Aufbau des X.500 Datenmodels - und dadurch auch von LDAP - ist durch eine hierarchische Struktur von Objekten gekennzeichnet. Ein Objekt besteht aus einem Namen und seinen Attributen, die mit anderen Objekten in Beziehung stehen. Die Attribute sind je nach ihrer Objektklasse vorgeschrieben und standardisieren diese damit. Genauso sind die Attribute über die Objektklassen hinweg standardisiert. Die Verwendung von schon erstellten und standardisierten Klassen und Attributen ist aufgrund der erhöhten Kompatibilität zu anderen Applikationen empfehlenswert. [6, pp. 17,18,21-23]

Attribute werden gesondert von den Objekten definiert, um diese öfters verwenden zu können. Diese beinhalten einen

- gekürzten Namen
- gegeben falls eine Beschreibung
- Eigenschaften wie Gleichheit und Ordnung des Attributs
- Die Deklaration, ob es sich um einen einzigen Wert oder um eine Menge von Werten handeln soll

[6, p. 19]

4.2.2.6 Authentifizierung[12]

Die Authentifizierung über einen LDAP-Server hat für den Client den Vorteil, sich nicht bei jeder Applikation oder jedem Service ein eigenes Konto erstellen und verwalten zu müssen, sondern die Authentifizierung über den LDAP-Server geschehen zu lassen. Somit ist der Administrationsaufwand der Applikationen, sowie der Mehraufwand des Users sich alle Daten zu merken, gemindert. [7, p. 71]

4.2.2.7 Rechteverwaltung[13]

Die Zugriffsrechte Instanzen des Verzeichnisbaumes sind ebenso hierarchisch angeordnet. Diese werden aber noch in eine Deny- und Allow-Liste getrennt. Bei einer Anfrage eines Clients wird zuerst die Deny-Liste von speziellen Regeln, also unterste Ebene, bis zur obersten Ebene vorgegangen. Sollte der Zugriff verweigert sein, wird die Anfrage abgebrochen. Sollte keine Deny-Regel gefunden werden wird die Allow-Liste in genau derselben Reihenfolge überprüft. Sollte in einer Ebene der Zugriff erlaubt sein, wird der Request des Clients ausgeführt, sollte jedoch keine Allow-Regel zutreffen, ist der Client für diese Anfrage nicht privilegiert. [7, pp. 82,84]

[11] geschrieben von Kritzl
[12] geschrieben von Kritzl
[13] geschrieben von Kritzl

4.2.2.8 Anwendung[14]

4.2.2.8.1 Active Directory

„Active Directory ist ein auf dem LDAP-Protokoll aufgebautes, auf verteilten Datenbanken beruhendes Verzeichnis. Diesem Verzeichnis liegen eine Vielzahl an Möglichkeiten zugrunde, die jedoch weder alle verarbeitet noch in Anspruch genommen werden müssen." (O. Kümmel, Active Directory: Handbuch für Administratoren, p. 11)

Active Directory kann seine Funktionalität unter anderem für folgende Einsatzgebiete gewährleisten:

- Verwaltung aller Computer, Server, Drucker, Dateifreigaben uvm.
- Verwaltung von Unternehmen, die hierarchisch gegliedert sind, geografische verteilte Standorte besitzen und eine Vielzahl an IP-Subnetzen definiert haben
- Bereitstellung von Sicherheitsaspekten wie beispielsweise Passwort-Richtlinien in Bereichen des Unternehmens

Active Directory ist nur bei Windows Server-Betriebssystemen ab Windows 2000 lauffähig. Außerdem basiert es auf dem X.500-Standard und ist nur eingeschränkt zu früheren Systemen, beispielsweise NT 4.0 Domänen, kompatibel.

[8, pp. 12, 14]

4.2.2.8.2 openLDAP

OpenLDAP ist eine freie Open-Source-Software und ist inzwischen Bestandteil der meisten aktuellen Linux-Distributionen. Es läuft jedoch auch auf Mac OS X und verschiedenen Windows-Versionen.

OpenLDAP ist…:

- skalierbar
  ➔ von einer geringen bis zu einer sehr großen Anzahl an Verbindungen

- replikationsfähig
  ➔ gleichmäßige Lastverteilung der Lesezugriffe durch Aufteilung eines bestimmten Datenbestandes auf mehrere Server

- partitionsfähig
  ➔ gleichmäßige Lastverteilung der Schreib- und Lesezugriffe durch Aufteilung eines bestimmten Datenbestandes auf mehrere Server

[6, pp. 79, 80]

[14] geschrieben von Erceg

4.2.2.8.3 Novell eDirectory

Novell eDirectory stellt eine Mischung aus LDAP und NDS (Novell Directory Service) dar. Der Verzeichnisdienst NDS basiert auf dem X.500-Standard und ist für die Betriebssysteme Netware 5, Windows NT/2000, Linux/Unix und Solaris verfügbar.

Die Architektur von Novell eDirectory setzt sich aus zwei Hauptkomponenten zusammen:

- DSA (Directory Service Agent)

  Diese stellen die Hauptkomponenten des Servers dar und speichern Informationen über die Verzeichnisse, Replikationen und den Daten selbst. Mittels der Verwendung von LDAP oder NDAP (Novell Directory Access Protocol) wird der Zugriff auf den Verzeichnisdienst ermöglicht.

- Directory Clients

  Mittels dieser Komponente wird anderen Novell-Produkten der Zugriff auf eDirectory gewährt.

[9], [10]

5 Glossar[15]

Entität

Unter einer Entität versteht man Ressourcen, wie z.B. Hosts, Drucker, Festplatten und Dateien, genauso wie Prozesse, Benutzer, Posteingänge, Internetseiten, Nachrichten und Netzwerkverbindungen. [1, p. 209]

Bezeichner

Ein Bezeichner ist eine eindeutige Identifikation einer Entität in dem gegebenen Namensraum. [1, p. 210]

horizontale Verteilung

Darunter versteht man eine unstrukturierte Verteilung der Entitäten in derselben Ebene. Dabei sind die Entitäten mit den anderen gleichgestellt. [1, p. 62]

Resource Record

Unter einem Resource Record wird jeder Eintrag in der Zonen-Datei verstanden. Dies kann ein Name Server Eintrag, eine Delegation, ein Alias oder einfach ein Adressen-Eintrag sein. Es ist aber noch eine Vielzahl anderer Einträge möglich. [5, p. 186]

[15] geschrieben von Erceg & Kritzl

6 Literaturverzeichnis

[1] A. S. Tanenbaum und M. van Steen, Verteilte Systeme: Prinzipien und Paradigmen, Pearson, 2008.

[2] A. Schill und T. Springer, Verteilte Systeme, Springer, 2012.

[3] G. Coulouris, J. Dollimore und T. Kindberg, Verteilte Systeme: Konzepte und Design, Pearson, 2002.

[4] H. Zhang, Y. Wen, H. Xie und N. Yu, Distributed Hash Table: Theory, Platforms and Applications, Springer, 2013.

[5] K. Agouros, DNS/DHCP: Grundlagen und Praxis, open source press, 2007.

[6] D. Klünter und J. Laser, LDAP verstehen, OpenLDAP einsetzen: Grundlagen und Praxiseinsatz, dpunkt, 2008.

[7] P. Gergen, Internetdienste, Addison-Wesley, 2002.

[8] O. Kümmel, Active Directory: Handbuch für Administratoren, vmi-buch, 2005.

[9] Elektronik-Kompendium.de, „Novell eDirectory," Elektronik-Kompendium.de, 1997, 2015. [Online]. Available: http://www.elektronik-kompendium.de/sites/net/0905031.htm. [Zugriff am 12 10 2015].

[10] K. Lipinski, „NDS (NetWare directory service)," ITWissen.info, 2015. [Online]. Available: http://www.itwissen.info/definition/lexikon/NetWare-directory-service-NDS.html. [Zugriff am 12 10 2015].

7 Abbildungsverzeichnis