

Grundlagen Rechnernetze und Verteilte Systeme (GRNVS)

IN0010 – SoSe 2026

Prof. Dr.-Ing. Georg Carle

Lorenz Lehle, Christian Dietze, Stefan Lachnit, Manuel Simon, Markus Sosnowski

Lehrstuhl für Netzarchitekturen und Netzdienste
School of Computation, Information and Technology
Technische Universität München

Einordnung im ISO/OSI-Modell

Sitzungsschicht

Darstellungsschicht

Anwendungsschicht

Literaturangaben

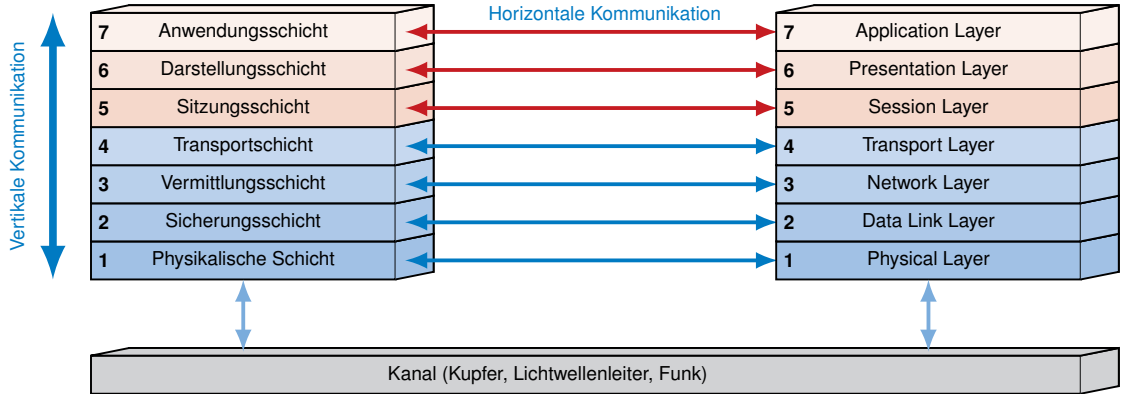
Einordnung im ISO/OSI-Modell

Sitzungsschicht

Darstellungsschicht

Anwendungsschicht

Literaturangaben



- Dienste der Sitzungs- und der Darstellungsschicht sind in einzelnen Fällen in Form standardisierter Protokolle implementiert.
- In vielen Fällen sind die Funktionen der Sitzungs- bzw. der Darstellungsschicht in die jeweilige Anwendung integriert.

⇒ Eine Differenzierung ist oft schwierig

Modell und Realität

- Die Standards der ITU-Serie X.200 beschreiben Dienste der sieben OSI-Schichten sowie Protokolle zur Erbringung dieser Dienste.
- Die in diesen Standards vorgenommene Strukturierung ist nützlich.
- Etliche OSI-Protokolle haben in der Praxis kaum Bedeutung.
- Oft ist keine strikte Trennung zwischen Sitzungs-, Darstellungsschicht und Anwendungsschicht möglich.
- Im [Internet Model \(RFC 1122\)](#)¹ werden alle diese Funktionen der Anwendungsschicht zugeordnet.

Im Folgenden werden wir

- die Aufgaben der Sitzungs- und Darstellungsschicht erläutern,
- beispielhaft einige Protokolle kennenlernen, deren Funktionen den Schichten 5, 6 und 7 zugeordnet werden können,
- sowie wichtige Protokolle der Anwendungsschicht erläutern.

¹ Das Internet-Modell ist ein alternatives Schichtenmodell (vgl. ISO/OSI-Modell), bei dem die Schichten 1 und 2 sowie 5-7 jeweils zusammengefasst sind. Dies entspricht in vielen Fällen einer praxisnäheren Einteilung, wenngleich die Zusammenfassung manchmal eine starke Vereinfachung darstellt. Letzteres trifft aber sogar auf einzelne Schichten des OSI-Modells zu, z. B. kann innerhalb von Schicht 2 die Medienzugriffskontrolle klar von der Adressierung unterschieden werden, weswegen man manchmal auch von MAC- und LLC-Sublayer (Logical Link Control) spricht.

Einordnung im ISO/OSI-Modell

Sitzungsschicht

- Dienste der Sitzungsschicht

- Realisierung der Funktionalität der Sitzungsschicht

- Transport Layer Security (TLS)

Darstellungsschicht

Anwendungsschicht

Literaturangaben

Gemäß Standard X.200 sind Dienste der Sitzungsschicht (engl. [Session Layer](#)) entweder verbindungsorientiert oder verbindungslos:

verbindungsorientiert (engl. „connection-oriented“)

- Es wird eine Verbindung zwischen den Kommunikationspartnern aufgebaut.
- Die Verbindung bleibt dabei über die Dauer einzelner Transfers (oder Verbindungen) der Transportschicht hinweg bestehen.
- Wie bei TCP-Verbindungen kann auch hier zwischen den Phasen [Verbindungsaufbau](#), [Datentransfer](#) und [Verbindungsabbau](#) unterschieden werden.

verbindungslos (engl. „connection-less“)

- Daten werden im Wesentlichen nur an die Transportschicht durchgereicht.
- Es wird keine Verbindung aufgebaut und kein Zustand zwischen den Kommunikationspartnern gehalten.

Hinweis

Eine Verbindung der Sitzungsschicht ([Session](#)) ist nicht gleichbedeutend mit einer Verbindung der Transportschicht. Eine Session kann beispielsweise nacheinander mehrere TCP-Verbindungen beinhalten oder Funktionen wie Bestätigungen unserem UDP-Chat hinzufügen, welche im Fall des TCP-Chats automatisch durch die Transportschicht erbracht werden.

Session

Eine **Session** beschreibt die Kommunikation zwischen mindestens zwei Teilnehmern mit definiertem Anfang und Ende sowie sich daraus ergebender Dauer.

Um für die dienstnehmende Schicht (Darstellungsschicht) eine Dialogführung zu ermöglichen, müssen gegebenenfalls mehrere Transportschicht-Verbindungen verwendet und kontrolliert werden. Dies kann auch die Behandlung abgebrochener und wiederaufgenommener TCP-Verbindungen beinhalten.

Im verbindungsorientierten Modus werden verschiedene Dienste angeboten:

- **Aufbau** und **Abbau** von Sessions,
- **normaler** und **beschleunigter Datentransfer**¹,
- Token-Management zur **Koordination** der Teilnehmer,
- **Synchronisation** und Resynchronisation,
- **Fehlermeldungen** und **Aktivitätsmanagement**, sowie
- **Erhaltung** und **Wiederaufnahme** von Sessions nach Verbindungsabbrüchen.

¹ Expedited Data Transfer: Wird verwendet für dringliche Daten, z. B. für Alarmer oder Interrupts

Beispiel: HTTP (Hyper Text Transfer Protocol)¹ → Details später

- HTTP in seiner ursprünglichen Form ist zunächst zustandslos.
- Zwischen unterschiedlichen Anfragen und Antworten besteht daher zunächst kein Zusammenhang.
- Cookies ermöglichen es, dass eine Sitzung über mehrere Anfragen und Antworten, Interaktionen und TCP-Verbindungen hinweg bestehen bleibt.
- Cookies sind kleine Datenfragmente, welche von einem Webserver (bzw. einem damit verbundenen Anwendungsprozess) an einen Webclient übertragen und dort gespeichert werden können.
- Dadurch wird es möglich, zeitlich getrennte oder von unterschiedlichen Adressen stammende Anfragen einem bestimmten Client und damit einem bestimmten Nutzer zuordnen zu können.
- HTTP wird üblicherweise der Anwendungsschicht (Schicht 7) zugeordnet, beinhaltet aber auch Funktionen der Darstellungs- und Sitzungsschicht (Schichten 6/5).

¹ Bei HTTP handelt es sich um ein Protokoll der Anwendungsschicht.

TLS (Transport Layer Security)

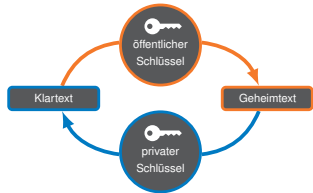
- TLS ist ein Protokoll zur sicheren Übertragung von Daten über einen verbindungsorientierten Transportdienst.
- Es ist die Grundlage u.a. für HTTP~~S~~.
- Es bietet unter anderem
 - Authentifizierung² („ist mein Gegenüber der, für den ich ihn halte“),
 - Integritätsschutz (Schutz vor Manipulation von Daten) und
 - Verschlüsselung (Vertraulichkeit, d. h. Schutz vor unberechtigtem Mitlesen).
- Es gibt unzählige kryptographische Verfahren, viele sind unsicher. Client und Server müssen sich auf die bestmöglichen Parameter einigen, die beide implementieren ([Kompatibilität](#) vs. [Sicherheit](#)).
- Sitzungen können mit Hilfe von Session-IDs über mehrere TCP-Verbindungen hinweg erhalten bleiben.
- Während die Funktionen zur Sitzungsverwaltung und -wiederaufnahme eher der Sitzungsschicht zugeordnet werden, gehören die Verschlüsselungsfunktionen zur Darstellungsschicht.

² Nicht zu verwechseln mit [Autorisierung](#), der Prüfung von Zugriffsberechtigungen.

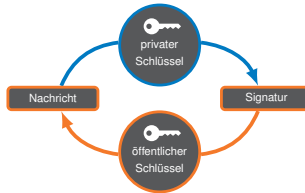
Transport Layer Security (TLS)

Authentifizierung durch X.509 Zertifikate

Bei der **Public-Key-Kryptographie** gibt es ein Schlüsselpaar: einen geheimen **Private Key** und einen öffentlichen **Public Key**. Dies ermöglicht sowohl Verschlüsselung als auch Signaturen.



Asymmetrische Verschlüsselung



Elektronische Signaturen

X.509-Zertifikate [2] sind standardisierte Public Keys, die mit zusätzlichen Informationen versehen und durch Signaturen kryptographisch abgesichert sind.

Problem: Ist mein Gegenüber wirklich der, für den ich ihn halte?

- Kennt und vertraut man dem Zertifikat des Gegenübers, kann dieser **authentifiziert** werden indem er mit seinem **Private Key** die aktuelle **Sitzung** signiert.
- Dies funktioniert nur, solange der Private Key geheim bleibt!

X.509 Certificate

Data		
Version	Serial Number	Signature Algorithm
3	1	SHA-256 with RSA
Subject (Inhaber)		
CN=net.in.tum.de		
Issuer (Aussteller)		
C=US, O=Let's Encrypt, CN=R3		
Valid Not Before		Valid Not After
17. Januar 2024 05:02		16. April 2024 05:02
X509v3 Extension: Alternative Names		
*.net.cit.tum.de, *.net.in.tum.de, net.cit.tum.de, net.in.tum.de		
Subject Public Key Info		
RSA 2048: AA:BB:12:17:5F:57:47:1D:43:72:15:DB [...]		
Signature		
F1:BA:2C:DE:E6:03:F5:91:F3:EF:B8:17:69:AF:C3 [...]		

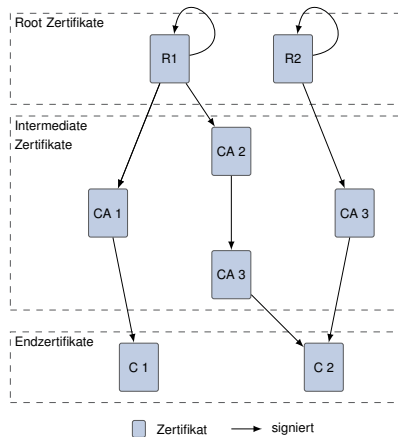
Vereinfachtes X.509 Zertifikats für net.in.tum.de

Transport Layer Security (TLS)

Eine globale X.509 Public-Key-Infrastruktur (PKI)

Es ist praktisch **unmöglich**, jedes Zertifikat einzeln zu kennen und individuell zu vertrauen.

- **Idee:** Vertraut man einem Zertifikat, kann man auch allen Zertifikaten vertrauen, die von diesem signiert wurden.
- Zertifikate können andere Zertifikate signieren und somit Vertrauen weiterreichen, es entsteht eine **Public-Key-Infrastruktur (PKI)**.
- Man unterscheidet:
 - **Root / Wurzel Zertifikate**, diese sind auf dem Betriebssystem oder im Browser vorinstalliert und durch sich selbst signiert.
 - **Intermediate / CA Zertifikate** gehören den zahlreichen **Certificate Authorities (CAs)**, die weitere Zertifikate ausstellen dürfen.
 - **End- / Leaf Zertifikate** dienen der Authentifizierung, dürfen keine weiteren Zertifikate ausstellen.
- Ein Zertifikat ist gültig, wenn eine Signaturkette (**Chain of Trust**) bis zu einem Root-Zertifikat besteht.
 - Server sollten alle benötigten Intermediate-Zertifikate zusammen mit dem Endzertifikat bereitstellen.
 - Bei dem sogenannten **Cross-Signing** wird der selbe Public Key von mehreren Parteien signiert (im Beispiel **CA 3**), dadurch können mehrere Chain of Trusts existieren.



Beispiel einer PKI

Transport Layer Security (TLS)

Der TLS Handshake

Das Herzstück von TLS ist der initiale **Handshake**¹, bei dem die Authentifizierung erfolgt und Parameter ausgehandelt werden. Können sich die Parteien nicht einigen, wird der Handshake abgebrochen. Wichtige Punkte:

- Der Algorithmus für den **Schlüsselaustausch** und der gemeinsamen geheimen Schlüssel,
- Das **Signaturverfahren** um die Zertifikate zu prüfen (und die konkreten Zertifikate), und
- Die Algorithmen für die **symmetrische** Verschlüsselung.

TLS nutzt **asymmetrische** Kryptographie, um einen gemeinsamen geheimen Schlüssel auszuhandeln. Für die eigentliche Datenübertragung wird jedoch eine **symmetrische** Verschlüsselung verwendet, da diese wesentlich effizienter ist.

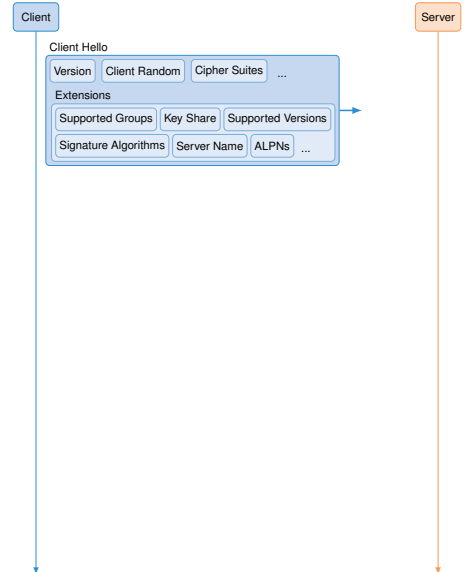
Hinweis

Internet Messungen ergaben, dass manche Middleboxen (z.B. Router) unbekannte TLS Versionen blockieren. Deshalb hat man TLS 1.3 so spezifiziert, dass es wie ein TLS 1.2 Handshake aussieht.

¹ Eine anschauliche Darstellung finden Sie unter <https://tls13.xargs.org>

Client Hello

- Der Start vom Handshake. Der Client schlägt die kryptographischen Algorithmen vor, die er unterstützt. Die Reihenfolge bestimmt die Präferenz.
- Die Client Hello **Nachricht** enthält u.a.:
 - Die **TLS Version** des Handshakes. In TLS 1.3 wird hier Version 1.2 versendet.
 - Eine Liste an **Cipher Suites**. Dies sind eine spezifizierte Kombination aus kryptographischen Algorithmen. Alle TLS 1.3 ciphers gelten als sicher. Für $\text{TLS} \leq 1.2$ gibt es auch unsichere ciphers, diese sollte man deaktivieren!
 - Die **Supported Groups** Extension enthält eine Liste an Algorithmen für den Schlüsselaustausch (in ≤ 1.2 war dies Teil der cipher suite).
 - In der **Key Share** Extension kann der Client den Schlüsselaustausch opportunistisch vorberechnen.
 - Erst die **Supported Versions** Extension verrät dem Server, dass es sich um einen TLS 1.3 Handshake handelt.
 - Die unterstützten **Signatur Algorithmen**. Dies hilft dem Server das richtige Zertifikat auszuwählen.
 - Der **Server Name** enthält die angefragten Domain. Wenn ein Server mehrere Domains bedient, kann er dadurch das richtige Zertifikat auswählen.
 - **ALPNs** (**Application Layer Protocol Negotiations**) enthalten eine Liste an unterstützten Layer 7 Protokollen, z.B. HTTP/1 oder HTTP/2. Dadurch können höhere Layer Zeit sparen.



Transport Layer Security (TLS)

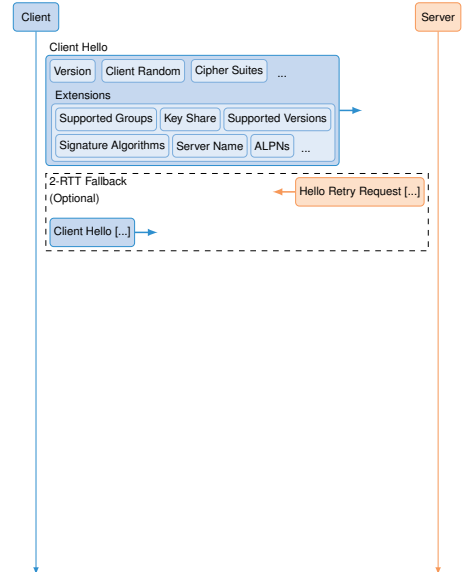
Der TLS 1.3 Handshake

2-RTT Fallback

- Wenn der Server mit dem Key Share des Client einverstanden ist, wird dieser Schritt übersprungen.
- Ansonsten informiert er den Client über den passenden Algorithmus durch eine **Hello Retry Request** Nachricht.
- Der Client kann nun ein neues **Client Hello** generieren, mit dem passenden Key Share.

Hinweis

Dieser Schritt passiert in der Regel nur selten, da es aktuell nur wenige Algorithmen gibt, diese alle als sicher gelten, und es oft wichtiger ist Daten schnell an den Nutzer zu geben als die eigene Präferenz einzuhalten.

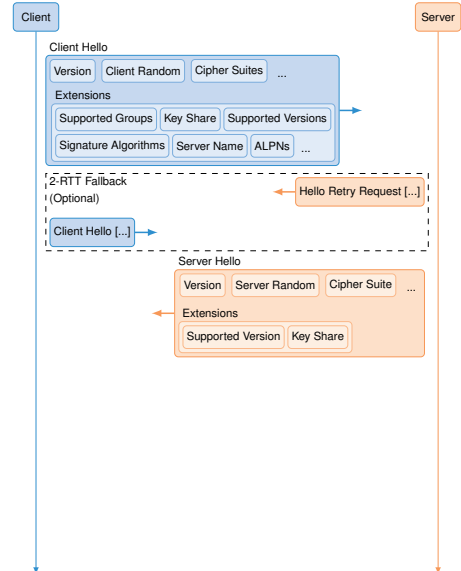


Transport Layer Security (TLS)

Der TLS 1.3 Handshake

Server Hello

- Die Antwort auf das Client Hello.
- Der Server wählt hier jeweils einen der vorgeschlagenen Algorithmen aus.
- Da es unverschlüsselt ist, enthält es nur die nötigen Informationen für den Schlüsselaustausch (um das abhören von Meta-Daten zu vermeiden)



Transport Layer Security (TLS)

Der TLS 1.3 Handshake

Change Cipher Spec

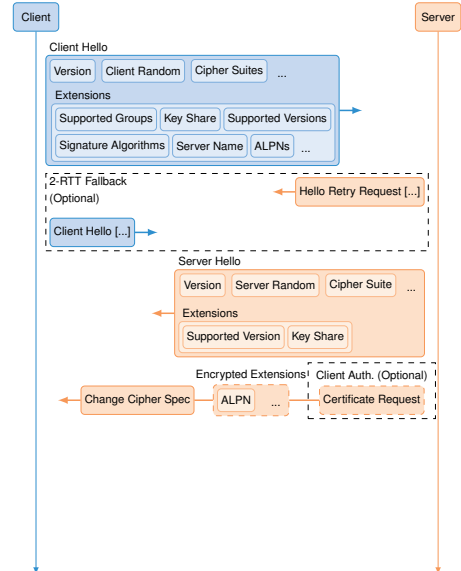
- Eine theoretisch unnötige Nachricht. Sie dient dazu Middleboxen vorzutäuschen, dass es sich um TLS 1.2 handelt.

Encrypted Extensions

- Beliebige weitere Informationen, wie das ausgewählte Layer 7 Protokoll.

Client Authentication

- Falls der Server auch den Client authentifizieren möchte, schickt er eine [Certificate Request](#) Nachricht. Für normale Websites unüblich.



Transport Layer Security (TLS)

Der TLS 1.3 Handshake

Certificate

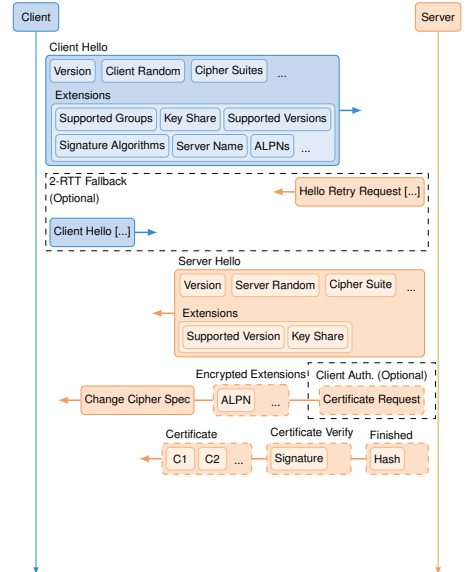
- Hier schickt der Server das eigene und alle Zertifikate, die der Client benötigt um die [Chain of Trust](#) zu berechnen.

Certificate Verify

- Diese Nachricht enthält eine Signatur von dem Server Zertifikat über ausgewählte Felder des bisherigen Handshakes. Dadurch beweist der Server den Besitz des private Keys zu dem Zertifikat.

Server Finished

- Diese Nachricht schließt den Handhake ab. Sie enthält einen abschließenden Hash über den bisherigen Handshake. Dies verhindert, dass ein Dritter den Handshake manipuliert hat.



Transport Layer Security (TLS)

Der TLS 1.3 Handshake

Client Change Cipher Spec

- Auch der Client schickt diese leere Nachricht

Client Authentication

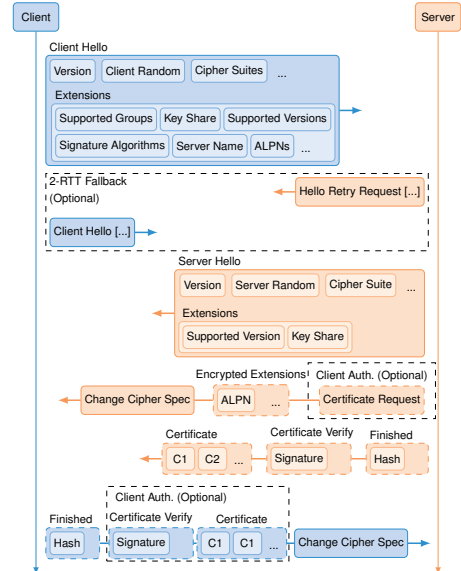
- Falls der Server den Client authentifizieren möchte, muss dieser analog zum Server eine **Certificate** und **Certificate Verify** Nachricht schicken.
- Manche Unternehmen stellen z.B. eigene Zertifikat aus, um den Zugang zu internen Webseiten zu kontrollieren.

Client Finished

- Der Abschluss des Handshakes, ebenfalls mit einem Hash über den bisherigen Handshake
- Ab jetzt können Daten verschlüsselt ausgetauscht werden

Ausblick

Quanten Computer (QC) können theoretisch heutige asymmetrische Kryptographie brechen. Möchte man TLS dagegen absichern, muss man im Schlüsselaustausch (Supported Groups und Key Share) post-quanten sichere Alternativen einsetzen. Um Man-in-the-Middle Angriffe durch QC zu verhindern, auch die Signaturen in den Zertifikaten und die Handshake Signatur.



Kapitel 5: Sitzungs-, Darstellungs- und Anwendungsschicht

Einordnung im ISO/OSI-Modell

Sitzungsschicht

Darstellungsschicht

- Aufgaben der Darstellungsschicht

- Zeichensätze und Kodierung

- Kodierung

- Strukturierte Darstellung

- Datenkompression

Anwendungsschicht

Literaturangaben

Die Aufgabe der **Darstellungsschicht** (engl. **Presentation Layer**) ist es, den Kommunikationspartnern eine einheitliche Interpretation der Daten zu ermöglichen, d. h. Daten in einem einheitlichen Format zu übertragen.

Der Darstellungsschicht sind grundsätzlich folgende Aufgaben zugeordnet:

- die Darstellung der Daten (Syntax),
- die Datenstrukturen zur Übertragung der Daten
- die Darstellung der Aktionen an diesen Datenstrukturen, sowie
- Datentransformationen.

Hinweis

Die Darstellung auf Schicht 6 muss nicht der Darstellung auf Schicht 7 (Anwendungsschicht) entsprechen. Die Darstellungsschicht ist für die **Syntax** der Nutzdaten verantwortlich, die **Semantik** verbleibt bei den Anwendungen.¹

- Anwendungen sollen syntaxunabhängig miteinander kommunizieren können.
- Anwendungsspezifische Syntax kann von der Darstellungsschicht in eine einheitliche Form umgewandelt und dann übertragen werden.

¹ Unter **Syntax** versteht man die Darstellung von Daten nach bestimmten Regeln (**Grammatik**). Wird Daten eine Bedeutung zugeordnet, spricht man von Information (Aufgabe der **Semantik**).

Den grundlegenden Aufgaben der Darstellungsschicht lassen sich konkrete Funktionen zuordnen:

- **Kodierung** der Daten
 - Übersetzung zwischen Zeichensätzen und Codewörtern gemäß standardisierter Kodierungsvorschriften
 - Kompression von Daten vor dem Senden (Entfernung von unerwünschter Redundanz)
 - Verschlüsselung
- **Strukturierte Darstellung** von Daten
 - plattformunabhängige, einheitliche Darstellung
 - Übersetzung zwischen verschiedenen Datenformaten
 - Serialisierung strukturierter Daten für die Übertragung

Wie auch bei der Sitzungsschicht gilt hier: Der Darstellungsschicht zuordenbare Protokolle erfüllen häufig auch Funktionen anderer Schichten.

Beispiel: TLS

- Die Verschlüsselungsfunktionen von TLS können der Darstellungsschicht zugeordnet werden.
- Funktionen wie Verbindungsaufbau und -abbau sowie Authentifizierung und Autorisierung fallen hingegen in den Bereich der Sitzungsschicht.

Daten liegen in einer von zwei Formen vor:

(1) **Textzeichen** bzw. **Symbole** in lesbarer Form („human readable“), z. B. Buchstaben, Textrepräsentation von Zahlen, Sonderzeichen. . .

- Ein **Zeichensatz** ist eine Menge textuell darstellbarer Zeichen sowie deren Zuordnung zu einem **Codepoint**¹.
- Wie die Codepoints eines bestimmten Zeichensatzes in binärer Form (also mittels einer Sequenz von Bits) dargestellt werden wird durch **Kodierungsvorschriften** festgelegt.
- Für einen Zeichensatz können ggf. mehrere mögliche Kodierungen existieren.

(2) **Binäre Daten** (also eine Sequenz von Bits), z. B. binäre Darstellungen von Buchstaben, Zahlen und Symbolen aber auch Bilder, Musik, Filme, etc. in digitaler Form. . .

- Ein **Datum** ist eine für Computer verarbeitbare „Einheit“, d. h. eine kurze Sequenz von Bits, deren Länge meist ein Vielfaches von 8 bit ist.²
- Was ein Datum repräsentiert – eine Zahl, ein Zeichen, einen Teil davon oder doch ein Bild – ist kontextabhängig (vgl. Kapitel 1 „Information und deren Bedeutung“).
- Ist bekannt, dass es sich bei den vorliegenden Daten um Text handelt, welche Kodierung verwendet wurde und um welchen Zeichensatz es sich handelt, lassen sich die binären Daten leicht wieder in Textzeichen übersetzen.

Hinweis: Binäre Daten (z. B. ein Bild) können nicht ohne Weiteres mit einem Zeichensatz dargestellt werden. Hierzu gibt es eigene Kodierungsvorschriften, die es ermöglichen, Binärdaten zu kodieren, so dass sie mit einem Zeichensatz dargestellt werden können.

¹ Ein Codepoint ist eine eindeutige „Kennzahl“ für höchstens ein Textzeichen (nicht alle Codepoints müssen vergeben sein).

² Die kleinste im Speicher adressierbare Einheit ist für heutige Computer ein Oktett, also ein Block von 8 bit, welches im Sprachgebrauch als Byte bezeichnet wird. Üblich sind daneben noch die Größen 16 bit, 32 bit und 64 bit, welche häufig als Word, Double Word bzw. Quad Word bezeichnet werden. Speicher wird byteweise adressiert, wobei es prozessorabhängig ist, in welcher Reihenfolge die einzelnen Oktette geladen werden. Folglich ist die **Byte Order** von essentieller Bedeutung.

- Ein **Zeichensatz** besteht aus einer Menge von Zeichen, bei der jedes **Zeichen** mit einem **Codepoint** verknüpft ist.
- Beispiele für Zeichensätze sind **ASCII**, **ISO-8859-15 (ISO-8859-1 mit €)** und **Unicode**.
- Ein Zeichensatz kann **druckbare Zeichen** sowie **Steuerzeichen** enthalten.
 - ASCII definiert 128 Zeichen.
 - Davon sind die Zeichen 0 – 31 und 127 **Steuerzeichen**.
 - Beispiele für Steuerzeichen sind Zeilenumbruch, Tabulator und Protokollzeichen.
 - Ursprünglich wurden darüber Drucker und Terminals angesteuert.
- ISO-8859-15 definiert 256 Codepoints.
 - Dabei entsprechen die Codepoints 0 – 127 dem ASCII Zeichensatz.
 - Weitere Codepoints sind mit Sonderzeichen europäischer Sprachen sowie dem Eurozeichen verknüpft.
 - ISO-8859-15 war vor Unicode ein im westeuropäischen Sprachraum weit verbreiteter Zeichensatz.

- Ein **Zeichensatz** besteht aus einer Menge von Zeichen, bei der jedes **Zeichen** mit einem **Codepoint** verknüpft ist.
- Beispiele für Zeichensätze sind **ASCII**, **ISO-8859-15** (**ISO-8859-1** mit €) und **Unicode**.
- Ein Zeichensatz kann **druckbare Zeichen** sowie **Steuerzeichen** enthalten.
 - ASCII definiert 128 Zeichen.
 - Davon sind die Zeichen 0 – 31 und 127 **Steuerzeichen**.
 - Beispiele für Steuerzeichen sind Zeilenumbruch, Tabulator und Protokollzeichen.
 - Ursprünglich wurden darüber Drucker und Terminals angesteuert.
- **ISO-8859-15** definiert 256 Codepoints.
 - Dabei entsprechen die Codepoints 0 – 127 dem ASCII Zeichensatz.
 - Weitere Codepoints sind mit Sonderzeichen europäischer Sprachen sowie dem Eurozeichen verknüpft.
 - ISO-8859-15 war vor Unicode ein im westeuropäischen Sprachraum weit verbreiteter Zeichensatz.
- **Unicode**
 - Unicode ist ein Zeichensatz, der in Version 13.0 143 859 Zeichen definiert.
 - Die Größe des Coderaums (Anzahl möglicher Zeichen) beträgt bei Unicode 1 114 112 Zeichen, gegliedert in 17 Ebenen mit je 2^{16} Zeichen.
 - Unicode hat das Ziel, alle Schriftkulturen und Zeichensysteme abzubilden.
 - Die Codepoints 0 – 255 entsprechen ISO-8859-1.
 - Im Gegensatz zu eingeschränkten Zeichensätzen wie z. B. ASCII und ISO-8859-15 wird der Zeichensatz bei Unicode regelmäßig aktualisiert und erweitert.
 - Es sind mehrere Zeichencodierungen definiert, die Codepoints für alle Unicode-Zeichen beinhalten, mit denen die Codepoints auf konkrete binäre Darstellungen abgebildet werden, u. a. UTF-32LE (Little Endian), UTF-32BE (Big Endian), UTF-16LE und UTF-16BE und UTF-8.

Zur Übertragung müssen Zeichen (bzw. die entsprechenden Codepoints) **kodiert** werden. Man unterscheidet zwischen

- **Fixed-Length Codes**, bei welchen alle Zeichen mit Codewörtern derselben Länge kodiert werden, z. B. ASCII (7 bit) oder UCS-2 (16 bit) und
- **Variable-Length Codes**, bei denen Zeichen mit Codewörtern unterschiedlicher Länge kodiert werden, z. B. UTF-8 (1 – 4 B).¹

Die möglichen Kodierungsverfahren hängen dabei vom jeweiligen Zeichensatz ab:

- ASCII und ISO-8859-15 definieren die Kodierung gemeinsam mit dem Zeichensatz.
 - Codewörter sind bei ASCII 7 bit lang, wobei das highest-order Bit eines Oktetts stets 0 ist.
 - ISO-8859-15 verwendet 8 bit lange Codewörter.
- Unicode definiert keine Kodierung sondern nur einen Zeichensatz. Häufig wird zur Kodierung von Unicode **UTF-8** verwendet, welches (mit Einschränkungen) kompatibel zu ASCII² ist.

¹ **Morse-Zeichen** sind ein weiteres Beispiel: Hier werden häufiger auftretende Zeichen kürzeren Codewörtern zugewiesen.

² Die Codewörter 0 – 127 entsprechen ASCII.

Unicode Transformation Format (UTF-8)

UTF-8 kodiert den Unicode Zeichensatz abhängig vom Codepoint mit 1 – 4 B langen Codewörtern:

Unicode-Bereich	Länge	binäre UTF-8 Kodierung	kodierbare Bits
U+0000 – U+007F	1 B	0xxxxxxx	7
U+0080 – U+07FF	2 B	110xxxxx 10xxxxxx	11
U+0800 – U+FFFF	3 B	1110xxxx 10xxxxxx 10xxxxxx	16
U+10000 – U+1FFFFF	4 B	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	21

- Die Darstellung U+xxxx ist lediglich eine Notation der Codepoints für Unicode. Die hexadezimalen Ziffern geben dabei den Wert der **kodierten Bits** eines Codeworts an.
- Bei Codewörtern länger als 1 B gibt die Anzahl der führenden 1-en vor der ersten 0 im ersten Oktett die Länge des Codeworts an.
- Die beiden highest-order Bits aller nachfolgenden Oktette eines Codeworts sind 10.
- Bei Codewörtern, die nur aus einem Oktett bestehen, ist das highest-order Bit stets 0 (vgl. ASCII).

Eigenschaften:

- UTF-8 ist rückwärts-kompatibel zu ASCII: ASCII kodierter Text ist valides UTF-8 und kann dementsprechend ohne Konvertierung als Unicode interpretiert werden.
- UTF-8 ist präfixfrei¹ und selbstsynchronisierend.
- Nicht alle Kombinationen sind gültige Codewörter (Kompatibilitätsgründe mit UTF-16).

¹ Kein gültiges Codewort ist ein echtes Präfix eines anderen Codeworts.

Beispiel 1: Kodierung des Umlauts ä

- **Unicode:** Codepoint 228, Codewort in UTF-8: U+00E4 = 11000011 10100100 („LATIN SMALL LETTER A WITH DIAERESIS“)
- **ISO-8859-1 und ISO8859-15:** Codepoint 228, Codewort: 11100100
Obwohl beide Zeichensätze eine Teilmenge von Unicode sind, ergeben sich bei Unicode bei der Zeichencodierung andere Codewörter.
- **ASCII:** Keine Kodierung möglich, da Umlaute nicht Teil des Zeichensatzes ist.

Beispiel 2: Kodierung des Eurosymbols €

- **Unicode:** Codepoint 8364, Codewort in UTF-8: U+20AC = 11100010 10000010 10101100
- **ISO-8859-15:** Codepoint 164, Codewort 10100100
- **ISO-8859-1 und ASCII:** Keine Kodierung möglich

Beispiel 1: Kodierung des Umlauts ä

- **Unicode:** Codepoint 228, Codewort in UTF-8: U+00E4 = 11000011 10100100 („LATIN SMALL LETTER A WITH DIAERESIS“)
- **ISO-8859-1 und ISO8859-15:** Codepoint 228, Codewort: 11100100
Obwohl beide Zeichensätze eine Teilmenge von Unicode sind, ergeben sich bei Unicode bei der Zeichencodierung andere Codewörter.
- **ASCII:** Keine Kodierung möglich, da Umlaute nicht Teil des Zeichensatzes ist.

Beispiel 2: Kodierung des Eurosymbols €

- **Unicode:** Codepoint 8364, Codewort in UTF-8: U+20AC = 11100010 10000010 10101100
- **ISO-8859-15:** Codepoint 164, Codewort 10100100
- **ISO-8859-1 und ASCII:** Keine Kodierung möglich

Vom verwendeten Zeichensatz nicht unterstützte Zeichen können häufig mittels **Zeichen-Entität-Referenzen** kodiert werden:

- XML erlaubt beispielsweise die Kodierung beliebiger Unicode-Zeichen durch Angabe des Codepoints, z. B. ä
- HTML erlaubt die Kodierung häufiger verwendeter Zeichen mittels **named entities**, z. B. ä
- $\text{\LaTeX}$ erlaubt die Kodierung verschiedener Zeichen auf ähnliche Art, z. B. \“a¹

Die Syntax ist aber abhängig vom jeweils verwendeten Protokoll (HTTP, SMTP → später) bzw. der Anwendung.

¹ Vertippt man sich und schreibt Verschlu\“sslung anstelle von Verschlu\“usslung, ergibt das dann in den Folien Verschlüsselung.

Damit Anwendungen Daten austauschen können, müssen diese eine einheitliche Syntax für die ausgetauschten Daten verwenden. Möglichkeiten hierfür sind:

- (gepackte)¹ structs / serialisierte Speicherbereiche

Daten werden so wie sie im Speicher vorliegen übertragen. Integration zwischen verschiedenen Systemen schwierig, weil diese die selben Datenstrukturen (und Compiler) verwenden müssen. Erweiterungen/Änderungen sind nur dann möglich, wenn alle beteiligten Systeme gleichzeitig aktualisiert werden.

- Ad-hoc Datenformate

Datenformat wird bei Bedarf „entworfen“. Problematisch sind hierbei die Dokumentation, Eindeutigkeit, Fehlerfreiheit (wie gut und sicher kann das Format geparsed werden) und Erweiterbarkeit.

- Strukturierte Serialisierungsformate wie XML oder JSON.

¹ In C bleibt es dem Compiler überlassen, wie viel Speicher ein struct tatsächlich belegt. Compiler-spezifische Keywords (bei gcc `__attribute__((packed))`) erzwingen, dass ein struct den minimal notwendigen Speicher belegt.

Damit Anwendungen Daten austauschen können, müssen diese eine einheitliche Syntax für die ausgetauschten Daten verwenden. Möglichkeiten hierfür sind:

- (gepackte)¹ structs / serialisierte Speicherbereiche

Daten werden so wie sie im Speicher vorliegen übertragen. Integration zwischen verschiedenen Systemen schwierig, weil diese die selben Datenstrukturen (und Compiler) verwenden müssen. Erweiterungen/Änderungen sind nur dann möglich, wenn alle beteiligten Systeme gleichzeitig aktualisiert werden.

- Ad-hoc Datenformate

Datenformat wird bei Bedarf „entworfen“. Problematisch sind hierbei die Dokumentation, Eindeutigkeit, Fehlerfreiheit (wie gut und sicher kann das Format geparsed werden) und Erweiterbarkeit.

- Strukturierte Serialisierungsformate wie XML oder JSON.

Beispiel: JavaScript Object Notation (JSON)

- Definiert in ECMA-404 [1] und RFC 7159 [4].
- Ursprünglich von JavaScript abgeleitet, mittlerweile aber sprachenunabhängiges Datenformat.
- Daten werden struktuiert und in lesbarer („human-readable“) Form als Text übertragen.

¹ In C bleibt es dem Compiler überlassen, wie viel Speicher ein struct tatsächlich belegt. Compiler-spezifische Keywords (bei gcc `__attribute__((packed))`) erzwingen, dass ein struct den minimal notwendigen Speicher belegt.

Strukturierte Darstellung

JSON

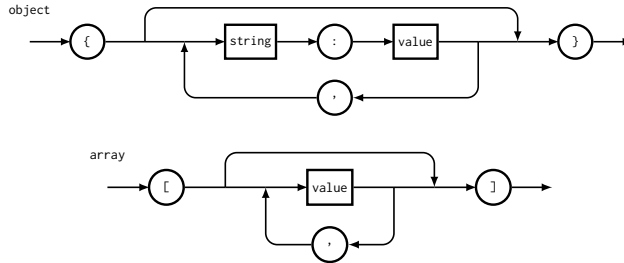
JSON definiert die folgenden Datentypen: number, string, boolean, array, object, null

- Zwischen den Elementen kann (beliebiger) Whitespace eingefügt werden.
- JSON wird im allgemeinen als UTF-8 kodiert.

Beispiel:

```
{
  "Image": {
    "Width": 800,
    "Height": 600,
    "Title": "View from 15th Floor",
    "Thumbnail": {
      "Url": "http://www.example.com/image/481989943",
      "Height": 125,
      "Width": 100
    },
    "Animated" : false,
    "IDs": [116, 943, 234, 38793]
  }
}
```

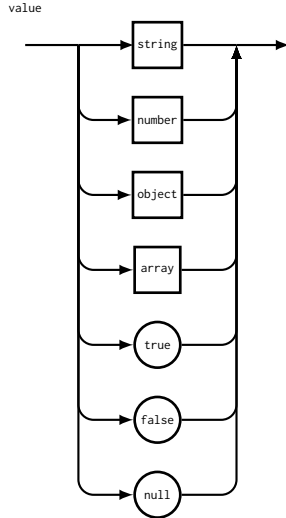
Strukturierte Darstellung JSON Objects und Arrays



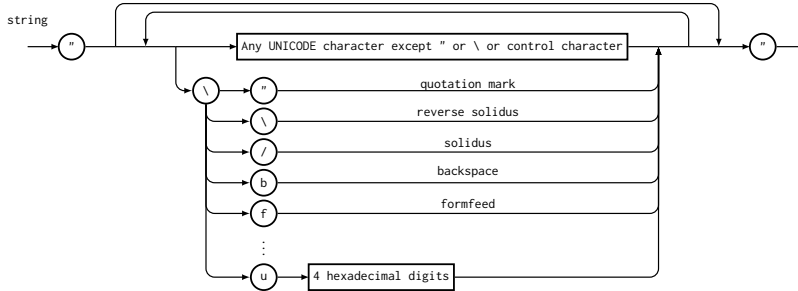
- JSON-Syntax, in RFC4627 über Augmented BNF spezifiziert, kann auch über Syntax-Diagramm spezifiziert werden.
- Ein **JSON-Objekt** ist eine ungeordnete Sammlung von **Key/Value-Paaren**.
- Der Key (Schlüssel) ist ein Unicode-String.
- Die Values (Werte) können unterschiedliche Typen aufweisen.
- Eine Sammlung von JSON-Objekten ist konzeptuell vergleichbar mit Hashmaps bzw. Dictionaries verschiedener Programmiersprachen.
- Arrays sind (geordnete) Listen (leere Listen sind erlaubt).
- Bei der Übertragung bleibt die Reihenfolge der Elemente innerhalb einer Liste erhalten.
- Listen können wiederum Werte von unterschiedlichen Typen aufweisen.

Strukturierte Darstellung

JSON Values

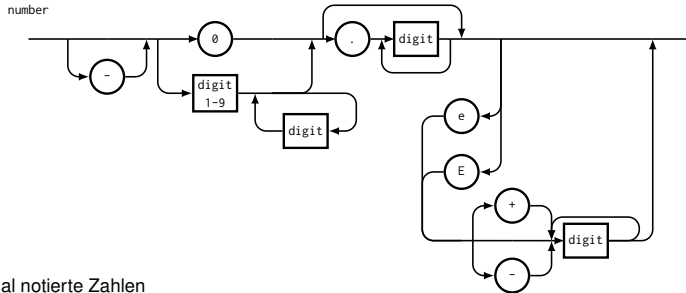


- Werte (values) können in JSON selbst wieder beliebige Typen sein.
- Booleans werden durch `true` und `false` repräsentiert.
- `null` entspricht dem Nullpointer (`null`, `None`, `nil`, `NUL`, etc.) der verschiedenen Sprachen.



- Strings können beliebige Unicode-Zeichenketten sein.
- Strings werden gequoted (also in der Form `"<text>"` notiert) und können leer sein.
- Steuerzeichen müssen escaped werden.
- Da Strings aus Unicode-Zeichen bestehen, können beliebige Binärdaten nicht direkt kodiert werden (es muss unterscheidbar sein, ob es sich um einen String oder um Binärdaten handelt).
- Das Problem kann durch Serialisierung der Binärdaten z. B. mittels Base64 [6] umgangen werden¹, oder mittels BSON (Binary JSON).

¹ Anything is human readable when base64 encoded and wrapped in XML ;)



- Numbers sind dezimal notierte Zahlen
- Es wird nicht zwischen Ganzzahlen und Gleitkommazahlen unterschieden.
- Hintergrund hierfür: JavaScript – worauf JSON ja ursprünglich basierte – kennt nur Gleitkommazahlen.
- Es ist implementierungsabhängig, ob Ganzzahlen und Gleitkommazahlen unterschieden werden und mit welcher Präzision diese abgebildet werden.
- Implementierungsabhängig ist auch, ob die Zahlenwerte 42, 4.2e1 und 42.0 als gleich angesehen werden.¹

¹ Ein Grund mehr, niemals Gleitkommazahlen auf Gleichheit zu testen.

Bei Kompressionsverfahren muss unterscheiden werden:

1. Verlustfreie Komprimierung (engl. lossless compression)

- Komprimierte Daten können verlustfrei, d. h. exakt und ohne Informationsverlust, wiederhergestellt werden.
- Verlustfrei komprimierte Dateiformate sind beispielsweise ZIP, PNG¹ (Bilder), FLAC² (Musik), ...

2. Verlustbehaftete Komprimierung (engl. lossy compression):

- Komprimierte Daten können im Allgemeinen nicht wieder exakt rekonstruiert werden.
- Es tritt also ein Verlust von Information bei der Komprimierung auf.
- Dafür ermöglichen diese Verfahren meist höhere und in Abhängigkeit des Verlustfaktors variable Kompressionsraten.
- Verlustbehaftet komprimierte Dateiformate sind beispielsweise MP3, MPEG, JPEG, ...

¹ Portable Network Graphics

² Free Lossless Audio Codec

Viele Protokolle komprimieren Daten vor dem Senden (Quellenkodierung).

Häufig nutzen Dateiformate den Huffman-Code als Backend bzw. in Kombination mit weiteren Verfahren, z. B. ZIP / DEFLATE, JPEG.

Grundlegende Idee der Huffman-Kodierung:

- Nicht alle Textzeichen treten mit derselben Häufigkeit auf, z. B. tritt der Buchstabe „E“ in der deutschen Sprache mit einer Häufigkeit von 17,4 % gefolgt von „N“ mit 9,8 % auf.
- Anstelle Zeichen mit uniformer Codewortlänge zu kodieren (z. B. ASCII-Code), werden **häufigen Zeichen kürzere Codewörter** zugewiesen.
- Die Abbildung zwischen Zeichen und Codewörtern bleibt dabei eindeutig und umkehrbar, weswegen es sich um ein verlustloses Kompressionsverfahren handelt.

Konstruktion

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftretswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.¹

z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30

¹ Andernfalls würden die Aussagen zur Optimalität des Huffman-Codes im Allgemeinen nicht mehr zutreffen.

Konstruktion

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftretswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

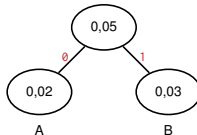
z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30



Konstruktion

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftrittswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

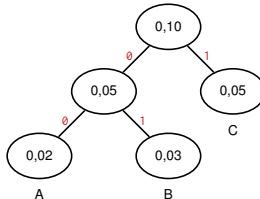
z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30



Konstruktion

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftretswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

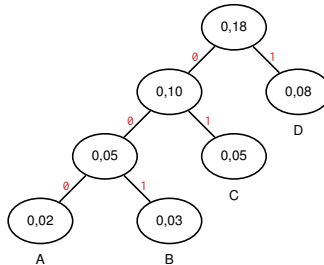
z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30



Konstruktion

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftretswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

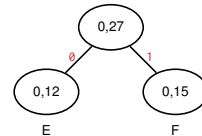
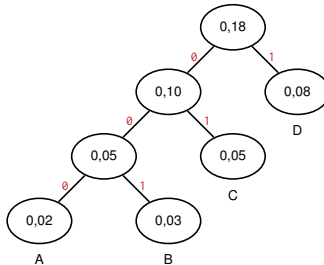
z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30



Konstruktion

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftretswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

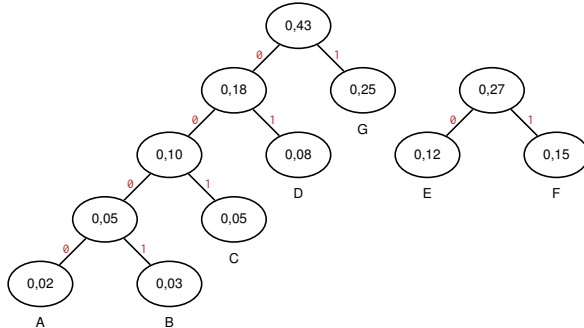
z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30



Konstruktion

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftretswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

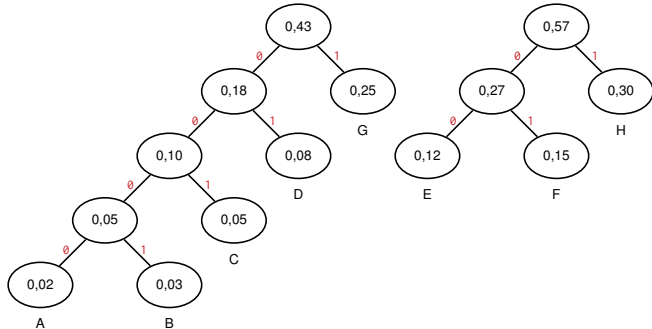
z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30



Konstruktion

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftretswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

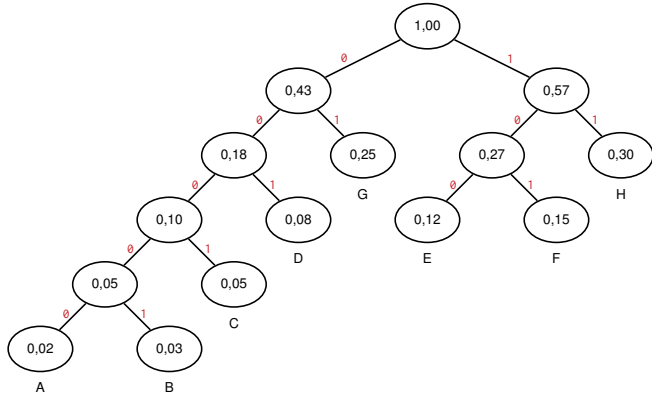
z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30



Konstruktion

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftretswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

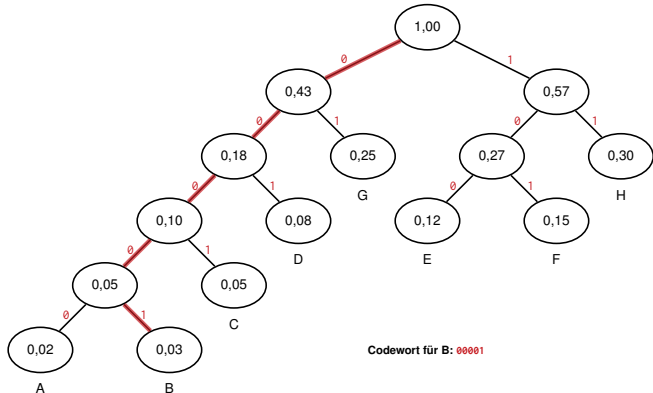
z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30



Konstruktion

- Gegeben Sei das Alphabet $\mathcal{A} = \{A, B, C, D, E, F, G, H\}$ sowie Auftretswahrscheinlichkeiten $\Pr[X = z]$ für alle Zeichen $z \in \mathcal{A}$.
- Es sei außerdem vorausgesetzt, dass die einzelnen Zeichen unabhängig voneinander auftreten.

z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30



Durchschnittliche Codewortlänge

z	$\Pr[X = z]$	Huffman-Code	Länge $l_H(z)$	Uniformer Code
A	0,02	00000	5	000
B	0,03	00001	5	001
C	0,05	0001	4	010
D	0,08	001	3	011
E	0,12	100	3	100
F	0,15	101	3	101
G	0,25	01	2	110
H	0,30	11	2	111

- Uniformer Code:

$E[l_U(z)] = 3,0$, da alle Codewörter gleich lang sind

- Huffman-Code:

$$E[l_H(z)] = \sum_{z \in \mathcal{A}} \Pr[X = z] l_H(z) = 2,6$$

$\Rightarrow$ Die Einsparung beträgt $1 - \frac{E[l_H(z)]}{E[l_U(z)]} \approx 13\%$

Anmerkungen

- Statische Huffman-Codes sind darauf angewiesen, dass die Auftretswahrscheinlichkeit der Zeichen den Erwartungen entspricht.
- Zeichenhäufigkeiten können dynamisch bestimmt werden, allerdings muss dem Empfänger dann das verwendete **Codebuch** mitgeteilt werden.
- Der Huffman-Code ist ein **optimaler** und **präfixfreier** Code.

Definition (Optimaler Präfixcode)

Bei einem **präfixfreien Code** sind gültige Codewörter niemals Präfix eines anderen Codeworts desselben Codes. Ein **optimaler** präfixfreier Code minimiert darüber hinaus die mittlere Codewortlänge

$$\sum_{i \in \mathcal{A}} p(i) \cdot |c(i)|,$$

wobei $p(i)$ die Auftretswahrscheinlichkeit von $i \in \mathcal{A}$ und $c(i)$ die Abbildung auf ein entsprechendes Codewort bezeichnen.

- Es handelt sich zudem um einen **Variable-Length Code**, d. h. Codewörter haben unterschiedliche Länge.
- Die Huffman-Codierung zählt zu den sogenannten **Entropy-Encoding** Verfahren.
- Huffman-Codierung kann grundsätzlich nicht nur für einzelne Zeichen sondern auch für Zeichenfolgen angewandt werden. Längere Codewörter können infolge der Komplexität zum Bestimmen des Codebuchs aber zu Problemen führen.

Datenkompression

Familie der Run-length Codes

Grundlegende Idee

- Daten weisen häufig Wiederholungen einzelner Zeichen oder Gruppen von Zeichen auf.
- Anstatt bei Wiederholungen jedes Zeichen bzw. jede Zeichengruppe erneut zu kodieren, geschieht dies nur einmal.
- Zur Rekonstruktion wird an den betroffenen Stellen die **Anzahl der Wiederholungen** kodiert.
- Ob einzelne Bits, Zeichen oder ganze Sequenzen kodiert werden, hängt vom jeweiligen Code ab.

Eigenschaften

- Verlustfreie Kompression
- Einfach (auch in Hardware) implementierbar
- Im Allgemeinen nicht optimal

Verwendung

- In verschiedenen Bildkompressionsverfahren
- Fax
- Analog- und ISDN-Modems

Kapitel 5: Sitzungs-, Darstellungs- und Anwendungsschicht

Einordnung im ISO/OSI-Modell

Sitzungsschicht

Darstellungsschicht

Anwendungsschicht

- Domain Name System (DNS)

- Uniform Resource Locator (URL)

- Hyper Text Transfer Protocol (HTTP)

- Simple Mail Transfer Protocol (SMTP)

- File Transfer Protocol (FTP)

Literaturangaben

Die **Anwendungsschicht (Application Layer)** ist die oberste Schicht der 7 Schichten des ISO/OSI-Referenzmodells¹ höchste Schnittstelle zwischen Anwendungen und dem Netzwerk. Protokolle der Anwendungsschicht stellen spezifische Dienste bereit.

Beispiele:

- **Domain Name System (DNS)** – Auflösung sog. vollqualifizierter Domännennamen² in IP-Adressen und umgekehrt.
- **Hyper Text Transfer Protocol (HTTP)** – Protokoll zum Transfer von Webseiten und Daten.
- **File Transfer Protocol (FTP)** – Protokoll zum Transfer von Daten von und zu Servern.
- **Simple Mail Transfer Protocol (SMTP)** – Dient dem Versandt von Emails sowie der Kommunikation zwischen Mailservern.
- **Post Office Protocol (POP)** und **Internet Message Access Protocol (IMAP)** – Abrufen von bzw. Zugriff auf Emails. Protokolle, mit denen ein Mail User Agent auf zugestellte Emails zugreifen kann.
- **Telnet** – Einfaches Protokoll zur interaktiven Kommunikation mit einem anderen Host (vgl. TCP Chat Client).
- **Secure Shell (SSH)** – Verschlüsselte entfernte Anmeldung an einem Host.
- **Simple Network Management Protocol (SNMP)**
Dient dem Monitoring und dem Management von Netzkomponenten.

¹ In vielen Fällen sind Protokolle der Anwendungsschicht Bestandteil der Anwendungen selbst, also innerhalb einer Anwendung implementiert und nicht Bestandteil des Betriebssystems oder einer Middleware.

² Umgangssprachlich auch als „Webadressen“ bezeichnet.

Domain Name System (DNS)

Motivation:

- Möchte ein Nutzer (Mensch) einen Computer adressieren, z. B. beim Aufruf einer Webseite, will er sich gewöhnlich nicht dessen IP-Adresse merken müssen.
- Stattdessen adressiert man das Ziel üblicherweise mittels eines hierarchisch aufgebauten Namens, z. B. www.google.com.

Das **Domain Name System (DNS)** besteht aus drei wesentlichen Komponenten:

1. Der Domain Namespace

- ist ein hierarchisch aufgebauter Namensraum, und
- hat eine baumartige Struktur.

2. Nameserver

- speichern Informationen über den Namensraum,
- jeder Server kennt nur kleine Ausschnitte des Namensraums.

3. Resolver sind Programme,

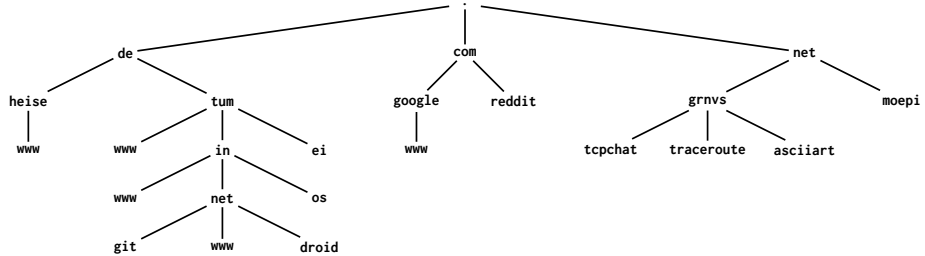
- die durch Anfragen an Nameserver Informationen aus dem Namespace extrahieren, und
- anfragenden Clients bzw. Anwendungen zur Verfügung stellen.

Details: siehe entsprechende RFCs [7, 8, 3, 5]

Gleichzeitig abstrahiert DNS von IP-Adressen, d. h. anstelle die IP-Adresse eines Servers z. B. im Emailprogramm konfigurieren zu müssen, kann sein Name angegeben werden. Die IP-Adresse kann sich damit sogar ändern, ohne dass die Konfiguration des Mailprogramms geändert werden muss.

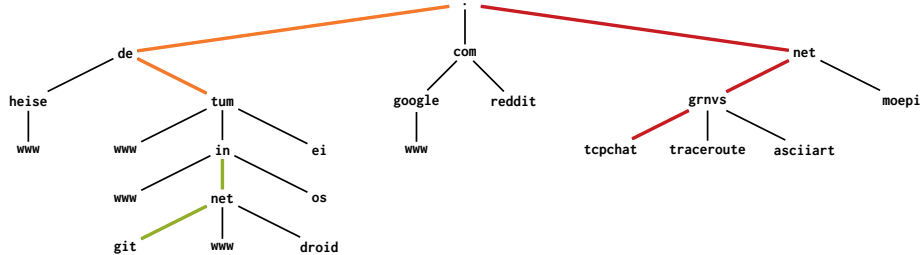
Domain Name System (DNS)

Domain Namespace (Beispiel)



Domain Name System (DNS)

Domain Namespace (Beispiel)

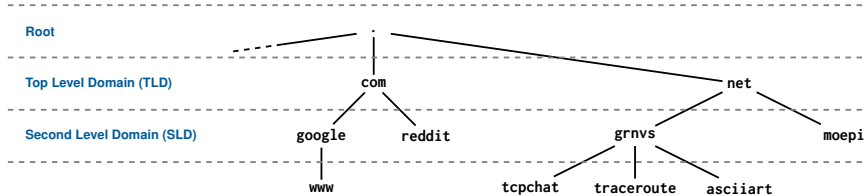


- Ein **Label** ist ein beliebiger Knoten im Namespace.
- Ein **Domain Name** ist eine Sequenz von Labels:
 - Ein **Fully Qualified Domain Name (FQDN)** besteht aus der vollständigen Sequenz von Labels ausgehend von einem Knoten bis zur Wurzel und endet mit einem Punkt, z. B. **tum.de.** oder **tcpchat.grnvs.net.**
 - Endet er nicht mit einem Punkt, handelt es sich zwar ebenfalls um einen Domain Name, allerdings ist dessen Angabe relativ ausgehend von einem anderen Knoten als der Wurzel, z. B. **git.net.in**.
 - Ein FQDN kann als **Suffix** für einen nicht-qualifizierten Namen verwendet werden, z. B. ergibt **git.net.in** zusammen mit dem FQDN **tum.de.** einen neuen FQDN **git.net.in.tum.de.**
 - Ob ein FQDN existiert (z. B. in eine Adresse aufgelöst werden kann), bleibt zunächst offen.

Domain Name System (DNS)

Domain Namespace

Für die ersten drei Hierarchieebenen im Name Space sind eigene Bezeichnungen üblich:

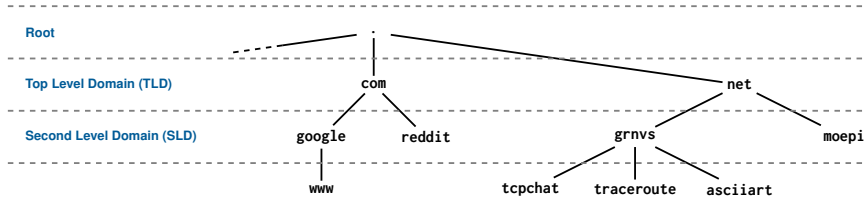


Darunter liegende Ebenen werden gelegentlich als **Subdomains** bezeichnet.

Domain Name System (DNS)

Domain Namespace

Für die ersten drei Hierarchieebenen im Name Space sind eigene Bezeichnungen üblich:



Darunter liegende Ebenen werden gelegentlich als **Subdomains** bezeichnet.

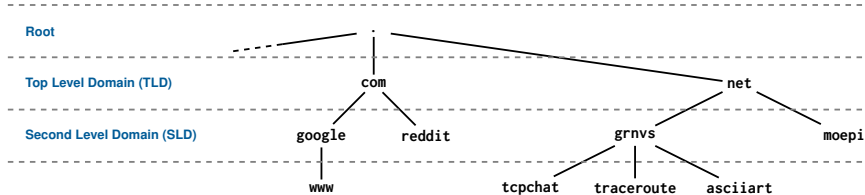
Vergabe von TLDs und SLDs

- Top Level Domains werden von der [Internet Corporation for Assigned Names and Numbers \(ICANN\)](#) vergeben.
- Second Level Domains werden von verschiedenen Registraren vergeben.
- Im Fall der länderspezifischen Top-Level-Domain (ccTLD) `.de` ist dies DENIC.

Domain Name System (DNS)

Domain Namespace

Für die ersten drei Hierarchieebenen im Name Space sind eigene Bezeichnungen üblich:



Darunter liegende Ebenen werden gelegentlich als **Subdomains** bezeichnet.

Vergabe von TLDs und SLDs

- Top Level Domains werden von der [Internet Corporation for Assigned Names and Numbers \(ICANN\)](#) vergeben.
- Second Level Domains werden von verschiedenen Registraren vergeben.
- Im Fall der länderspezifischen Top-Level-Domain (ccTLD) **.de** ist dies DENIC.

Zeichensatz

- Erlaubt sind nur Buchstaben (A–Z) und Zahlen sowie **-**, wobei letzterer nicht das erste oder letzte Zeichen sein darf.¹
- Zwischen Groß- und Kleinschreibung wird nicht unterschieden.

¹ Technisch gesehen könnte ein Eintrag im DNS alle möglichen Oktette enthalten.

Domain Name System (DNS)

Nameserver

Der Namespace wird

- in Form einer verteilten Datenbank
- von einer großen Anzahl von Servern gespeichert,
- wobei jeder Server nur einen kleinen Teil des gesamten Namespaces kennt.

Zu diesem Zweck ist der Namespace in **Zonen** unterteilt:

- Zonen sind **zusammenhängende Teilbäume** des Namespaces.
- Eine Zone kann daher mehrere Ebenen des Namespaces umfassen, aber keine Teilbäume ohne gemeinsame Wurzel.
- Nameserver bezeichnet man als **autoritativ** für die jeweiligen Zonen, die sie speichern.
- Dieselbe Zone kann auf mehreren Nameserver gespeichert sein.
- DNS sieht Mechanismen zum Transfer von Zonen zwischen autoritativen Nameservern vor.
- Dabei gibt es einen **primären Nameserver**, auf dem Änderungen an einer Zone vorgenommen werden können, sowie beliebig viele **sekundäre Nameserver**, welche lediglich über Kopien der Zone verfügen.

Nameserver erwarten eingehende Verbindungen auf Port UDP/TCP 53:

- Anfragen werden meist an UDP 53 gestellt.
- Anfragen, die größer als 512 B sind, werden typischerweise an TCP 53 gestellt.
- Zone Transfers finden immer über TCP 53 statt.

Domain Name System (DNS)

Resource Records

Die Informationen, die in einer Zone gespeichert sind, bezeichnet man als **Resource Records**:

- **SOA Record (Start of Authority)** ist ein spezieller Record, der die Wurzel der Zone angibt, für die ein Nameserver autoritativ ist.
- **NS Records** geben den FQDN eines Nameservers an. Dieser kann auch auf FQDNs in anderen Zonen verweisen.
- **A Records** assoziieren einen FQDN mit einer IPv4-Adresse.
- **AAAA Records** assoziieren einen FQDN mit einer IPv6-Adresse.
- **CNAME Records** sind Aliase, d. h. ein FQDN verweist auf einen "Canonical Name", der selbst wiederum ein FQDN ist.
- **MX Records** geben den FQDN eines Mailservers für eine bestimmte Domain an, welcher sich nicht notwendigerweise in derselben Zone befinden muss.
- **TXT Records** assoziieren einen FQDN mit einem String (Text). Kann für unterschiedliche Zwecke verwendet werden.
- **PTR Records** assoziieren eine IPv4- oder IPv6-Adresse mit einem FQDN (Gegenstück zu A bzw. AAAA Records).

Hinweise:

- Mehrere A oder AAAA Records (auch unterschiedlicher Zonen) können mit derselben IP-Adresse assoziiert sein.
- Für einen FQDN kann es maximal einen CNAME geben. Wenn ein CNAME existiert, handelt es sich um einen Alias, weswegen es keine weiteren Resource Records für den betreffenden FQDN mehr geben darf.
- Für eine Zone bzw. Domain gibt es üblicherweise mehrere NS bzw. MX Records.

Domain Name System (DNS)

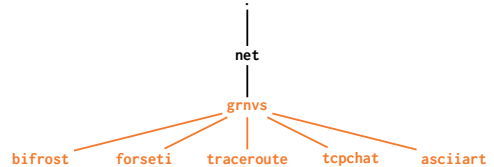
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN  SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                        164160 ; serial
                        1800   ; refresh (30 minutes)
                        300    ; retry (5 minutes)
                        604800 ; expire (1 week)
                        1800   ; nxdomain (30 minutes)
)
                        NS   bifrost.grnvs.net.
                        NS   forseti.grnvs.net.
                        A     131.159.15.21

$ORIGIN grnvs.net.
bifrost  A       129.187.145.241
forseti  A       78.47.25.36
        AAAA    2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute  A       89.163.225.145
           AAAA    2001:4ba0:ffec:0193::0
tcpchat     A       89.163.225.145
asciart     CNAME   svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



Domain Name System (DNS)

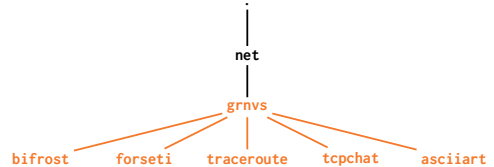
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
                    )
                    NS  bifrost.grnvs.net.
                    NS  forseti.grnvs.net.
                    A   131.159.15.21

$ORIGIN grnvs.net.
bifrost  A   129.187.145.241
forseti  A   78.47.25.36
        AAAA 2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute A   89.163.225.145
        AAAA 2001:4ba0:ffec:0193::0
tcpchat  A   89.163.225.145
asciart  CNAME svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- Setzt die **Time to Live**¹ der nachfolgenden Resource Records auf 1 d.

¹ Bitte nicht mit der TTL von IPv4-Paketen verwechseln.

Domain Name System (DNS)

Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

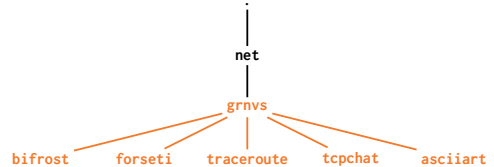
```
$TTL 86400 ; 1 day
grnvs.net. IN SOA bifrost.grnvs.net. hostmaster.grnvs.net. (
    164160 ; serial
    1800   ; refresh (30 minutes)
    300    ; retry (5 minutes)
    604800 ; expire (1 week)
    1800   ; nxdomain (30 minutes)
)
NS      bifrost.grnvs.net.
NS      forseti.grnvs.net.
A       131.159.15.21

$ORIGIN grnvs.net.
bifrost  A       129.187.145.241
forseti  A       78.47.25.36
AAAA    AAAA    2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute A       89.163.225.145
AAAA      AAAA    2001:4ba0:ffec:0193::0
tcpchat   A       89.163.225.145
asciart    CNAME   svm0012.net.in.tum.de.
```

- SOA Record für die Domain grnvs.net.

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



Domain Name System (DNS)

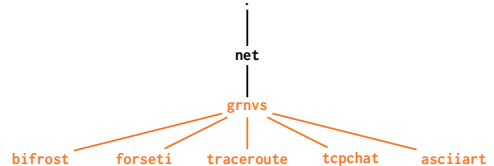
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA bifrost.grnvs.net. hostmaster.grnvs.net. (
    164160 ; serial
    1800 ; refresh (30 minutes)
    300 ; retry (5 minutes)
    604800 ; expire (1 week)
    1800 ; nxdomain (30 minutes)
)
NS bifrost.grnvs.net.
NS forseti.grnvs.net.
A 131.159.15.21

$ORIGIN grnvs.net.
bifrost A 129.187.145.241
forseti A 78.47.25.36
AAAA 2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute A 89.163.225.145
AAAA 2001:4ba0:ffec:0193::0
tcpchat A 89.163.225.145
asciart CNAME svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- SOA Record für die Domain grnvs.net.
 - serial gibt die Version der Zonendatei an.
 - Mit jedem Update der Zonendatei, z. B. beim Hinzufügen, Entfernen oder Modifizieren von Resource Records, muss diese inkrementiert werden.
 - Dient insbesondere bei Zonentransfers dazu festzustellen, ob ein Secondary veraltete Informationen besitzt.

Domain Name System (DNS)

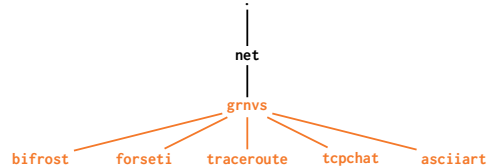
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
                    )
                    NS  bifrost.grnvs.net.
                    NS  forseti.grnvs.net.
                    A    131.159.15.21

$ORIGIN grnvs.net.
bifrost  A    129.187.145.241
forseti  A    78.47.25.36
        AAAA  2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute  A    89.163.225.145
           AAAA  2001:4ba0:ffec:0193::0
tcpchat    A    89.163.225.145
asciart    CNAME svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- SOA Record für die Domain grnvs.net.
 - refresh gibt das Zeitintervall an, in dem Secondaries versuchen, ihre lokale Kopie der Zonendatei mit dem Primary abzugleichen.

Domain Name System (DNS)

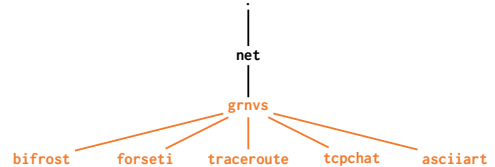
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
)
                NS  bifrost.grnvs.net.
                NS  forseti.grnvs.net.
                A    131.159.15.21

$ORIGIN grnvs.net.
bifrost  A    129.187.145.241
forseti  A    78.47.25.36
        AAAA  2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute  A    89.163.225.145
           AAAA  2001:4ba0:ffec:0193::0
tcpchat     A    89.163.225.145
asciart     CNAME svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- SOA Record für die Domain grnvs.net.
 - retry gibt das Zeitintervall an, in dem Secondaries versuchen, ihren Primary zu kontaktieren, falls der vorherige Versuch fehlgeschlagen ist.

Domain Name System (DNS)

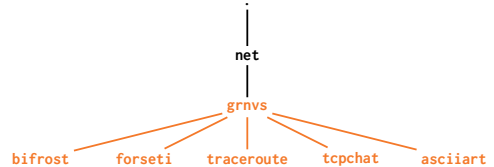
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
)
                    NS  bifrost.grnvs.net.
                    NS  forseti.grnvs.net.
                    A    131.159.15.21

$ORIGIN grnvs.net.
bifrost  A    129.187.145.241
forseti  A    78.47.25.36
        AAAA  2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute  A    89.163.225.145
           AAAA  2001:4ba0:ffec:0193::0
tcpchat    A    89.163.225.145
asciart     CNAME svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- SOA Record für die Domain `grnvs.net.`
 - `expire` gibt für Secondaries das maximale Zeitintervall seit dem letzten erfolgreichen Abgleich mit seinem Primary an, während dessen die Information des Secondaries als autoritativ gelten.
 - Sollte kein Kontakt mit dem Primary innerhalb dieser Zeit möglich sein, stellt der Secondary seinen Betrieb ein.

Domain Name System (DNS)

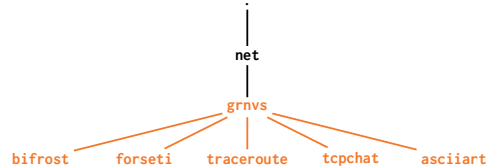
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
)
                    NS  bifrost.grnvs.net.
                    NS  forseti.grnvs.net.
                    A   131.159.15.21

$ORIGIN grnvs.net.
bifrost  A   129.187.145.241
forseti  A   78.47.25.36
        AAAA 2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute A   89.163.225.145
        AAAA 2001:4ba0:ffec:0193::0
tcpchat   A   89.163.225.145
asciart    CNAME svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- SOA Record für die Domain grnvs.net.
 - `nxdomain`² gibt an, wie lange anfragende Resolver in ihrem Cache halten dürfen, dass ein angefragter Resource Record nicht existiert.

² Dieses Feld hatte früher eine andere Bedeutung.

Domain Name System (DNS)

Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

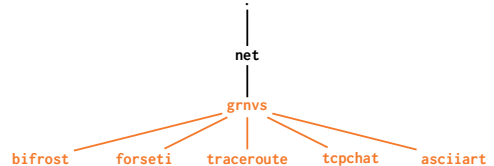
```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
                    )
```

```
NS  bifrost.grnvs.net.
NS  forseti.grnvs.net.
A   131.159.15.21
```

```
$ORIGIN grnvs.net.
bifrost  A   129.187.145.241
forseti  A   78.47.25.36
        AAAA 2a01:4f8:190:60a3::2
```

```
$TTL 3600 ; 1 hour
traceroute  A   89.163.225.145
           AAAA 2001:4ba0:ffec:0193::0
tcpchat     A   89.163.225.145
asciart     CNAME svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- Die NS Records geben die FQDNs der autoritativen Nameserver für diese Zone an.
- Deren FQDNs müssen nicht notwendigerweise dieselbe Endung wie die aktuelle Zone haben.

Domain Name System (DNS)

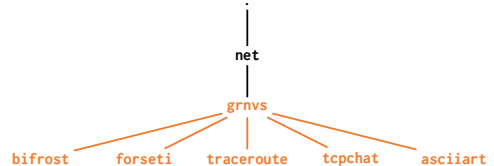
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
                    )
                    NS  bifrost.grnvs.net.
                    NS  forseti.grnvs.net.
                    A   131.159.15.21
```

```
$ORIGIN grnvs.net.
bifrost  A      129.187.145.241
forseti  A      78.47.25.36
         AAAA   2a01:4f8:190:60a3::2
```

```
$TTL 3600 ; 1 hour
traceroute  A      89.163.225.145
            AAAA   2001:4ba0:ffec:0193::0
tcpchat     A      89.163.225.145
asciart     CNAME  svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- A Record für den FQDN **grnvs.net.**
- Was geschieht, wenn dieser fehlt, sah man beim Versuch den FQDN **ei.tum.de.** im Browser aufzurufen.

Domain Name System (DNS)

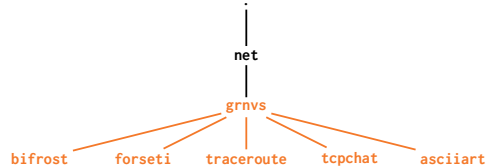
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
                    )
                    NS  bifrost.grnvs.net.
                    NS  forseti.grnvs.net.
                    A    131.159.15.21
```

```
$ORIGIN grnvs.net.
bifrost  A      129.187.145.241
forseti  A      78.47.25.36
         AAAA   2a01:4f8:190:60a3::2
```

```
$TTL 3600 ; 1 hour
traceroute  A      89.163.225.145
            AAAA   2001:4ba0:ffec:0193::0
tcpchat     A      89.163.225.145
asciart     CNAME  svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- Gibt an, dass alle nachfolgenden Domains in Resource Records, für die dieser Server autoritativ ist, relativ zu **\$ORIGIN** zu sehen sind.
- Spart Schreibarbeit bei den nachfolgenden Resource Records, die alle auf **grnvs.net.** enden.

Domain Name System (DNS)

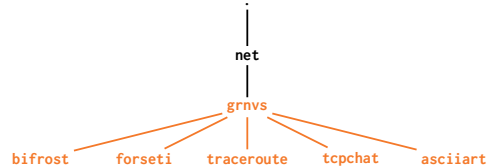
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
                    )
                NS  bifrost.grnvs.net.
                NS  forseti.grnvs.net.
                A    131.159.15.21

$ORIGIN grnvs.net.
bifrost   A      129.187.145.241
forseti   A      78.47.25.36
          AAAA   2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute A      89.163.225.145
          AAAA   2001:4ba0:ffec:0193::0
tcpchat   A      89.163.225.145
asciart    CNAME  svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- A Records für insgesamt vier Hosts.
 - Die ersten beiden sind für die beiden autoritativen Nameserver der Zone.
 - Die anderen beiden verweisen auf dieselbe IP-Adresse, d. h. **traceroute.grnvs.net.** und **tcpchat.grnvs.net.** sind in Wirklichkeit derselbe Host.

Domain Name System (DNS)

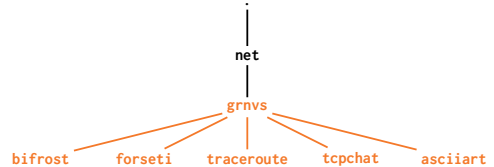
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
                    )
                    NS  bifrost.grnvs.net.
                    NS  forseti.grnvs.net.
                    A    131.159.15.21

$ORIGIN grnvs.net.
bifrost  A      129.187.145.241
forseti  A      78.47.25.36
        AAAA   2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute  A      89.163.225.145
        AAAA   2001:4ba0:ffec:0193::0
tcpchat     A      89.163.225.145
asciart     CNAME  svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- AAAA Records für zwei Hosts (IPv6).

Domain Name System (DNS)

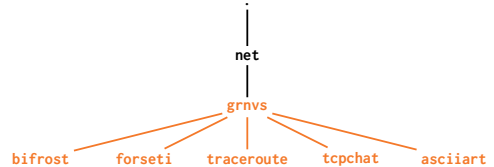
Die Resource Records einer Zone werden auf Nameservern in Form von **Zone Files** gespeichert:

```
$TTL 86400 ; 1 day
grnvs.net. IN SOA  bifrost.grnvs.net. hostmaster.grnvs.net. (
                    164160 ; serial
                    1800  ; refresh (30 minutes)
                    300   ; retry (5 minutes)
                    604800 ; expire (1 week)
                    1800  ; nxdomain (30 minutes)
                    )
                    NS  bifrost.grnvs.net.
                    NS  forseti.grnvs.net.
                    A   131.159.15.21

$ORIGIN grnvs.net.
bifrost  A       129.187.145.241
forseti  A       78.47.25.36
          AAAA   2a01:4f8:190:60a3::2

$TTL 3600 ; 1 hour
traceroute  A       89.163.225.145
            AAAA   2001:4ba0:ffec:0193::0
tcpchat     A       89.163.225.145
asciiart    CNAME   svm0012.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zum abgebildeten Zone File korrespondierende Teil ist hervorgehoben):



- CNAME Record, der auf eine unserer virtuellen Maschinen verweist.
- Das Ziel `svm0012.net.in.tum.de.` liegt außerhalb dieser Zone.

Domain Name System (DNS)

Resolver

Resolver sind Server, die Informationen aus dem DNS extrahieren und das Ergebnis an den anfragenden Client zurückliefern.

- Da das DNS einer verteilten Datenbank entspricht und kein einzelner Nameserver alle Zonen kennt, sind i. A. mehrere Anfragen notwendig.
- Resolver fragen dabei schrittweise bei den autoritativen Nameservern der jeweiligen Zonen an.
- Das Ergebnis wird an den anfragenden Client zurückgegeben und kann (hoffentlich unter Beachtung der TTL im SOA Record der jeweiligen Zone) gecached werden.
- Stellt ein Client innerhalb dieser Zeit nochmal dieselbe Anfrage, kann diese aus dem Cache beantwortet werden.

(Öffentliche) Resolver sind i. d. R. für keine Zone selbst autoritativ.

Problem: Woher weiß ein Resolver, wo er anfangen soll?

Domain Name System (DNS)

Resolver

Resolver sind Server, die Informationen aus dem DNS extrahieren und das Ergebnis an den anfragenden Client zurückliefern.

- Da das DNS einer verteilten Datenbank entspricht und kein einzelner Nameserver alle Zonen kennt, sind i. A. mehrere Anfragen notwendig.
- Resolver fragen dabei schrittweise bei den autoritativen Nameservern der jeweiligen Zonen an.
- Das Ergebnis wird an den anfragenden Client zurückgegeben und kann (hoffentlich unter Beachtung der TTL im SOA Record der jeweiligen Zone) gecached werden.
- Stellt ein Client innerhalb dieser Zeit nochmal dieselbe Anfrage, kann diese aus dem Cache beantwortet werden.

(Öffentliche) Resolver sind i. d. R. für keine Zone selbst autoritativ.

Problem: Woher weiß ein Resolver, wo er anfangen soll?

- Resolver verfügen über eine statische Liste der 13 Root-Server¹, die für die Root-Zone autoritativ sind.
- Die Root-Zone wird von der [Internet Corporation for Assigned Names and Numbers](#) verwaltet. Änderungen bedürfen jedoch der Zustimmung durch das [US Department of Commerce](#).
- Betrieben werden die Root-Server von verschiedenen Organisationen, u. a. ICANN, Versign, U. S. Army, RIPE, NASA, etc.

¹ In Wirklichkeit handelt es sich dabei um hunderte Server, welche über die 13 IP-Adressen via Anycast erreichbar sind.

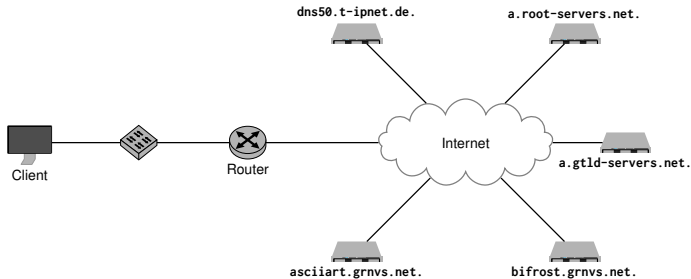
Domain Name System (DNS)

Ausschnitt der Root-Hints

```
;      This file holds the information on root name servers needed to
;      initialize cache of Internet domain name servers
;      (e.g. reference this file in the "cache . <file>"
;      configuration file of BIND domain name servers).
;
;      This file is made available by InterNIC
;      under anonymous FTP as
;          file           /domain/named.cache
;          on server      FTP.INTERNIC.NET
;      -OR-              RS.INTERNIC.NET
;
;      last update:      March 23, 2016
;      related version of root zone:  2016032301
;
; formerly NS.INTERNIC.NET
;
;
.      3600000      NS      A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.  3600000      A      198.41.0.4
A.ROOT-SERVERS.NET.  3600000      AAAA   2001:503:ba3e::2:30
;
; FORMERLY NS1.ISI.EDU
;
;
.      3600000      NS      B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.  3600000      A      192.228.79.201
B.ROOT-SERVERS.NET.  3600000      AAAA   2001:500:84::b
;
; FORMERLY C.PSI.NET
;
;
.      3600000      NS      C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.  3600000      A      192.33.4.12
C.ROOT-SERVERS.NET.  3600000      AAAA   2001:500:2::c
;
```

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

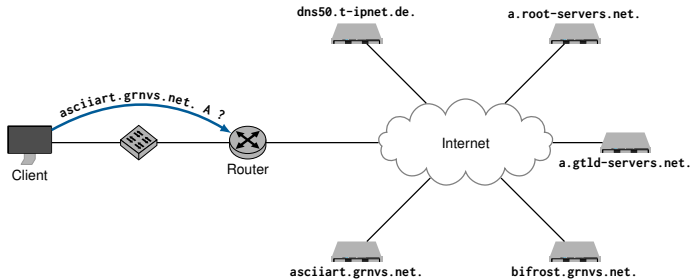
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

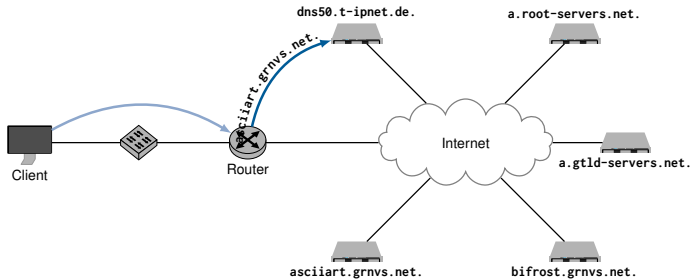
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

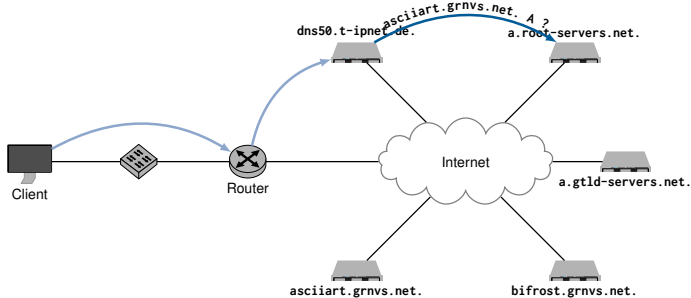
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

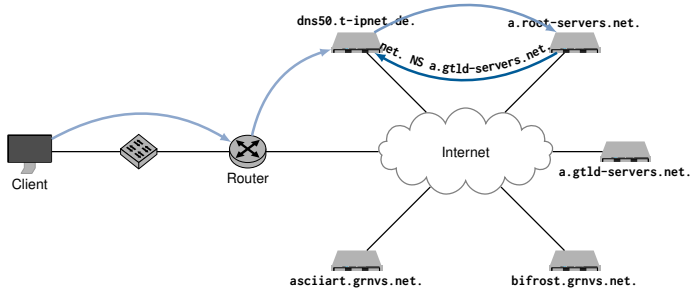
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

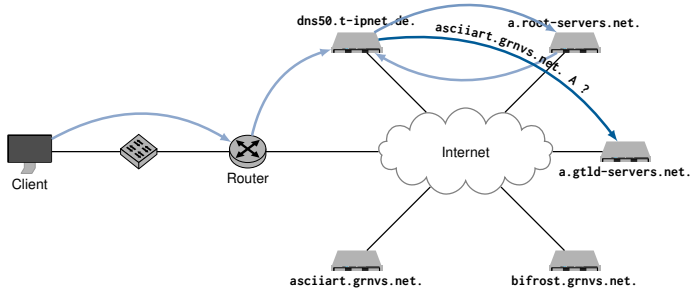
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

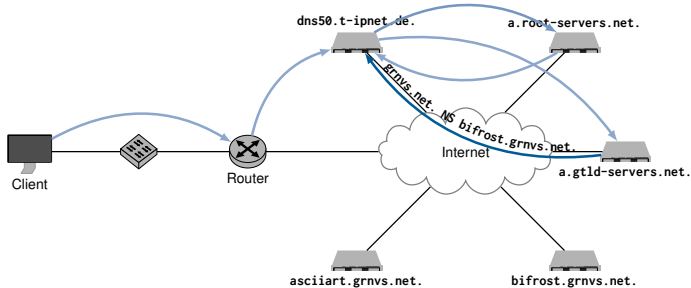
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

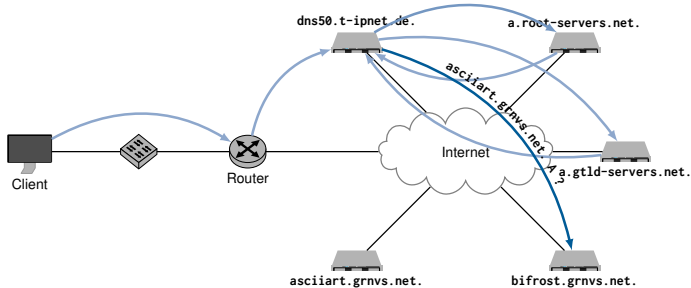
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

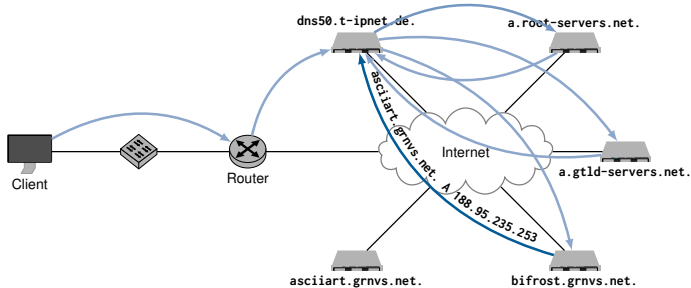
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

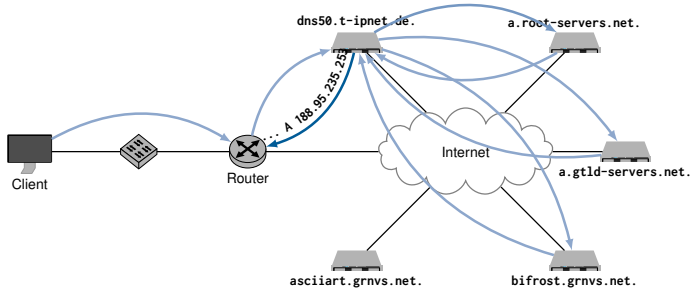
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

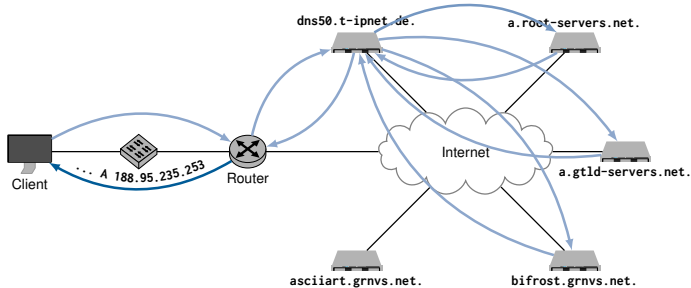
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Beispiel: Gewöhnlicher privater Internetanschluss eines motivierten Studenten, der zur Bearbeitung der 4. Programmieraufgabe mittels Webbrowser auf den Server mit FQDN `asciiart.grnvs.net.` zugreift.

- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.¹
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind **rekursiv (recursive queries)**.
- Die eigentliche Namensauflösung wird vom Resolver `dns50.t-ipnet.de.` mittels einer Reihe von **iterativen Anfragen (iterative queries)** erledigt.



¹ Diesen Vorgang bezeichnet man als **Forwarding**. Weswegen könnte das sowohl aus Providersicht als auch aus Nutzersicht vorteilhaft sein?

Das vorherige Beispiel war in einigen Punkten vereinfacht:

1. Der Resolver hätte von **bifrost.grnvs.net.** in Wirklichkeit keinen A Record mit der gesuchten IP-Adresse, sondern lediglich einen CNAME Record erhalten.

- Das liegt daran, dass laut der Zone File auf Folie 41 kein A Record sondern nur ein CNAME Record für **asciart.grnvs.net.** existiert (das ist nur dem Beispiel geschuldet).
- In Wirklichkeit hätte der Resolver im Anschluss mittels weiterer Anfragen einen autoritativen Nameserver für **svm0012.net.in.tum.de.** ermitteln müssen (der FQDN, den der CNAME Record liefert).
- Aus Sicht des Routers und des Clients hätte sich aber nichts geändert.

2. Die Antworten der Nameserver auf die iterativen Anfragen liefern lediglich die FQDNs der autoritativen Nameserver für eine Zone.

- Der Resolver erhält also z. B. im ersten Schritt von **a.root-servers.net.** lediglich den FQDN der für **net.** autoritativen Nameserver.
- Bevor der Resolver mit seiner eigentlichen Aufgabe fortfahren kann, muss er zunächst den entsprechenden A oder AAAA Record für **a.gtld-servers.net.** bestimmen.
- Wenn der FQDN des Nameservers in der gesuchten Zone liegt (wie es hier der Fall ist), müssen in der darüberliegenden Zone so genannte **Glue Records** eingetragen werden. Glue Records enthalten die IP Adresse der gesuchten Nameserver.
- Aus Sicht des Routers und Clients ändert sich auch in diesem Fall nichts.

Domain Name System (DNS)

Reverse DNS

Im DNS können mittels **PTR (Pointer) Records** auch FQDNs zu IP-Adressen hinterlegt werden. Dies bezeichnet man als **Reverse DNS**. Hierzu existiert für IPv4 und IPv6 jeweils eine eigene Zone:

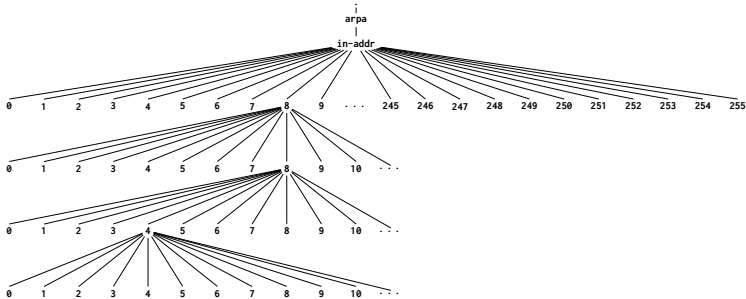
- **in-addr.arpa.** für IPv4
- **ip6.arpa.** für IPv6

Reverse DNS

Im DNS können mittels **PTR (Pointer) Records** auch FQDNs zu IP-Adressen hinterlegt werden. Dies bezeichnet man als **Reverse DNS**. Hierzu existiert für IPv4 und IPv6 jeweils eine eigene Zone:

- **in-addr.arpa.** für IPv4
- **ip6.arpa.** für IPv6

Für IPv4 wird der Namespace unterhalb von `in-addr.arpa.` durch die vier Oktette in umgekehrter Reihenfolge erzeugt:

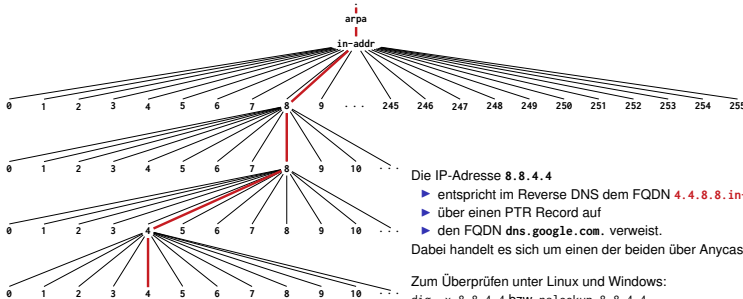


Reverse DNS

Im DNS können mittels **PTR (Pointer) Records** auch FQDNs zu IP-Adressen hinterlegt werden. Dies bezeichnet man als **Reverse DNS**. Hierzu existiert für IPv4 und IPv6 jeweils eine eigene Zone:

- **in-addr.arpa.** für IPv4
- **ip6.arpa.** für IPv6

Für IPv4 wird der Namespace unterhalb von **in-addr.arpa.** durch die vier Oktette in umgekehrter Reihenfolge erzeugt:



Für den Namespace unterhalb von `in-addr.arpa.` ergeben sich einige Einschränkungen:

- Da jede Ebene einem ganzen Oktett entspricht, gibt es maximal vier Ebenen.
- Subnetze, deren Präfixlänge nicht 8, 16, 24 oder 32 ist, können nicht in getrennten Zonen gespeichert werden (letzte entsprechen einzelnen IP-Adressen).
- Die Abbildung von Subnetzen anderer Größen ist nur mit Tricks möglich.

Für den Namespace unterhalb von `in-addr.arpa.` ergeben sich einige Einschränkungen:

- Da jede Ebene einem ganzen Oktett entspricht, gibt es maximal vier Ebenen.
- Subnetze, deren Präfixlänge nicht 8, 16, 24 oder 32 ist, können nicht in getrennten Zonen gespeichert werden (letzte entsprechen einzelnen IP-Adressen).
- Die Abbildung von Subnetzen anderer Größen ist nur mit Tricks möglich.

Der Namespace für IPv6 (unterhalb von `ip6.arpa.`) ist sehr ähnlich aufgebaut:

- Die Aufteilung findet an 4 bit-Grenzen anstelle ganzer Oktette statt.
- Dies erweitert die Möglichkeiten zur Aufteilung in getrennte Zonen.
- Der Namespace ist infolge der 128 bit langen IPv6-Adressen entsprechend größer.

Uniform Resource Locator (URL)

Mittels FQDN können wir

- mit Hilfe von DNS das Ziel einer Verbindung auf Schicht 3 identifizieren,
- aber weder das zu verwendende Anwendungsprotokoll angeben noch bestimmte Resource adressieren.

¹ Wenn Sie auf diese Art Benutzername und Kennwort angeben, wissen Sie hoffentlich, dass Sie Ihre Anmeldeinformationen genauso gut via Rundfunk bekanntgeben könnten . . .

² Innerhalb von URLs (bzw. URIs) wird der terminierende . eines FQDNs i. d. R. weggelassen.

Uniform Resource Locator (URL)

Mittels FQDN können wir

- mit Hilfe von DNS das Ziel einer Verbindung auf Schicht 3 identifizieren,
- aber weder das zu verwendende Anwendungsprotokoll angeben noch bestimmte Resource adressieren.

Uniform Resource Locator (URL) sind Adressangaben der Form

`<protocol>://[<username>[:<password>]@]<fqdn>[:<port>][/<path>][?<query>][#<fragment>]`

- `<protocol>` gibt das Anwendungsprotokoll an, z. B. HTTP(S), FTP, SMTP, etc.
- `<username>[:<password>]@` ermöglicht die optionale Angabe eines Benutzernamens und Kennworts.¹
- `<fqdn>` ist der vollqualifizierte Domain Name², der das Ziel auf Schicht 3 identifiziert.
- `:<port>` ermöglicht die optionale Angabe einer vom jeweiligen well-known Port abweichenden Portnummer für das Transportprotokoll.
- `/<path>` ermöglicht die Angabe eines Pfads auf dem Ziel relativ zur Wurzel `</>` der Verzeichnisstruktur.
- `?<query>` ermöglicht die Übergabe von Variablen in der Form `<variable>=<value>`. Variablen können mittels `&` konkateniert werden.
- `#fragment` ermöglicht es einzelne Fragmente bzw. Abschnitte in einem Dokument zu referenzieren.

¹ Wenn Sie auf diese Art Benutzername und Kennwort angeben, wissen Sie hoffentlich, dass Sie Ihre Anmeldeinformationen genauso gut via Rundfunk bekanntgeben könnten ...

² Innerhalb von URLs (bzw. URIs) wird der terminierende . eines FQDNs i. d. R. weggelassen.

Uniform Resource Locator (URL)

Mittels FQDN können wir

- mit Hilfe von DNS das Ziel einer Verbindung auf Schicht 3 identifizieren,
- aber weder das zu verwendende Anwendungsprotokoll angeben noch bestimmte Resource adressieren.

Uniform Resource Locator (URL) sind Adressangaben der Form

`<protocol>://[<username>[:<password>]@]<fqdn>[:<port>][/<path>][?<query>][#<fragment>]`

- `<protocol>` gibt das Anwendungsprotokoll an, z. B. HTTP(S), FTP, SMTP, etc.
- `<username>[:<password>]@` ermöglicht die optionale Angabe eines Benutzernamens und Kennworts.¹
- `<fqdn>` ist der vollqualifizierte Domain Name², der das Ziel auf Schicht 3 identifiziert.
- `:<port>` ermöglicht die optionale Angabe einer vom jeweiligen well-known Port abweichenden Portnummer für das Transportprotokoll.
- `/<path>` ermöglicht die Angabe eines Pfads auf dem Ziel relativ zur Wurzel `</>` der Verzeichnisstruktur.
- `?<query>` ermöglicht die Übergabe von Variablen in der Form `<variable>=<value>`. Variablen können mittels `&` konkateniert werden.
- `#<fragment>` ermöglicht es einzelne Fragmente bzw. Abschnitte in einem Dokument zu referenzieren.

Beispiele:

- `http://www.tum.de`
- `https://vcs.net.in.tum.de/svn/grnvss15/users/ma92les/exam/IN0010-20150612-0000.pdf`
- `https://www.mymail.alsoinsecure/mybox?user=student&password=notlongeryourscret`

¹ Wenn Sie auf diese Art Benutzername und Kennwort angeben, wissen Sie hoffentlich, dass Sie Ihre Anmeldeinformationen genauso gut via Rundfunk bekanntgeben könnten ...

² Innerhalb von URLs (bzw. URIs) wird der terminierende . eines FQDNs i. d. R. weggelassen.

Aber ich tippe doch nur `google.de` ein und es funktioniert trotzdem!

Es funktioniert in den meisten Fällen aus folgenden Gründen:

1. Sie tippen das in Ihrem Webbrowser ein. Ihr Webbrowser weiß, dass mit höchster Wahrscheinlichkeit eine Verbindung über HTTP(S) und nicht über FTP oder gar SMTP (Email) gewünscht ist.
2. Google hat A bzw. AAAA Records für `google.de` gesetzt, so dass sich der Aufruf genauso wie mit `www.google.de` verhält. Das ist nicht selbstverständlich:
Vergleichen Sie `https://tumonline.de` und `https://www.tumonline.de` (letzteres leitet auf `https://campus.tum.de/...` weiter).
3. Der Domain Name, der nicht mal ein FQDN ist, wird stillschweigend als solcher interpretiert.

Aber ich tippe doch nur `google.de` ein und es funktioniert trotzdem!

Es funktioniert in den meisten Fällen aus folgenden Gründen:

1. Sie tippen das in Ihrem Webbrowser ein. Ihr Webbrowser weiß, dass mit höchster Wahrscheinlichkeit eine Verbindung über HTTP(S) und nicht über FTP oder gar SMTP (Email) gewünscht ist.
2. Google hat A bzw. AAAA Records für `google.de` gesetzt, so dass sich der Aufruf genauso wie mit `www.google.de` verhält. Das ist nicht selbstverständlich:
Vergleichen Sie `https://tumonline.de` und `https://www.tumonline.de` (letzteres leitet auf `https://campus.tum.de/... weiter`).
3. Der Domain Name, der nicht mal ein FQDN ist, wird stillschweigend als solcher interpretiert.

Übrigens:

- Dass es sich um einen „Webserver“ handelt, entscheidet sich nicht daran, dass der FQDN `www.webserver.de` lautet.
- Der „Webserver“ könnte genauso gut unter dem FQDN `mail.mydomain.de` erreichbar sein.
- Nicht einmal die Portnummer entscheidet hierüber:
 - Natürlich ist es üblich, dass ein „Webserver“ über TCP 80 und ein Mailserver über TCP 25 erreichbar ist (well-known Ports).
 - Es hindert uns aber nichts¹ daran, einen „Webserver“ auf TCP 25 und einen Mailserver auf TCP 80 erreichbar zu machen.
 - Ob und wann das sinnvoll ist, sei dahingestellt.

¹ Root- bzw. Administratorrechte vorausgesetzt, da man andernfalls unter den gängigen Betriebssystemen keinen listening Socket < 1024 anlegen kann.

Das im Internet am häufigsten zur Datenübertragung zwischen Client und Server genutzte Protokoll ist [Hyper Text Transfer Protocol \(HTTP\)](#):

- HTTP definiert, welche Anfragen ein Client stellen darf und wie Server darauf zu antworten haben.
- Mit einem HTTP-Kommando wird höchstens ein „Objekt“ (Text, Grafik, Datei, etc.) übertragen.
- Kommandos werden als ASCII-kodierter Text interpretiert, d. h. es handelt sich um ein textbasiertes Protokoll.
- Eingehende HTTP-Verbindungen werden auf dem well-known Port TCP 80 erwartet.
- Bei HTTP 1.0 wird nach jedem Anfrage/Antwort-Paar die TCP-Verbindung wieder abgebaut.

Frage: Was ist die offensichtliche Problematik beim Abruf einer Webseite, welche aus vielen Teilen zusammengesetzt ist (z. B. eine große Anzahl kleiner Grafiken)?

Das im Internet am häufigsten zur Datenübertragung zwischen Client und Server genutzte Protokoll ist [Hyper Text Transfer Protocol \(HTTP\)](#):

- HTTP definiert, welche Anfragen ein Client stellen darf und wie Server darauf zu antworten haben.
- Mit einem HTTP-Kommando wird höchstens ein „Objekt“ (Text, Grafik, Datei, etc.) übertragen.
- Kommandos werden als ASCII-kodierter Text interpretiert, d. h. es handelt sich um ein textbasiertes Protokoll.
- Eingehende HTTP-Verbindungen werden auf dem well-known Port TCP 80 erwartet.
- Bei HTTP 1.0 wird nach jedem Anfrage/Antwort-Paar die TCP-Verbindung wieder abgebaut.

Frage: Was ist die offensichtliche Problematik beim Abruf einer Webseite, welche aus vielen Teilen zusammengesetzt ist (z. B. eine große Anzahl kleiner Grafiken)?

Für jedes Objekt muss eine neue TCP-Verbindung aufgebaut werden, was entsprechend Zeit kostet und bei vielen kleinen Objekten einen entsprechenden Overhead bedeutet.

Verbesserung mit HTTP 1.1:

- TCP-Verbindungen überdauern mehrere Anfragen.
- HTTP kann dennoch als zustandslos angesehen werden, da die einzelnen Anfragen voneinander unabhängig sind.

Hyper Text Transfer Protocol (HTTP)

HTTP Message Types und Headers

HTTP unterscheidet grundsätzlich zwischen zwei Nachrichtentypen: **Request** und **Response**

1. **Request** (vom Client zum Server), enthält

- eine **Method**, welche die vom Client gewünschte Aktion beschreibt, z. B. Übertragung einer bestimmten Resource vom Server zum Client,
- Pfad und Query-Parameter des **Uniform Resource Locator (URL)**, welcher die angefragte Resource näher beschreibt und
- eine Reihe weiterer Headerfelder, in denen unter anderem die folgenden Informationen enthalten sein können:
 - FQDN des angefragten Hosts ¹
 - Zeichensatz und Encoding, in dem die Antwort erwartet wird
 - Von wo eine Anfrage kam (z. B. Weiterleitung von einer anderen Webseite über den Referrer ²)
 - Den User-Agent, also die verwendete Client-Software

¹ Über dieses Feld kann der Server bei so genannten Virtual Hosts unabhängig von der IP-Adresse entscheiden, welche Daten ausgeliefert werden sollen

² In HTTP in falscher Schreibweise als Referer[sic]

Hyper Text Transfer Protocol (HTTP)

HTTP Message Types und Headers

HTTP unterscheidet grundsätzlich zwischen zwei Nachrichtentypen: **Request** und **Response**

1. **Request** (vom Client zum Server), enthält

- eine **Method**, welche die vom Client gewünschte Aktion beschreibt, z. B. Übertragung einer bestimmten Resource vom Server zum Client,
- Pfad und Query-Parameter des **Uniform Resource Locator (URL)**, welcher die angefragte Resource näher beschreibt und
- eine Reihe weiterer Headerfelder, in denen unter anderem die folgenden Informationen enthalten sein können:
 - FQDN des angefragten Hosts ¹
 - Zeichensatz und Encoding, in dem die Antwort erwartet wird
 - Von wo eine Anfrage kam (z. B. Weiterleitung von einer anderen Webseite über den Referrer ²)
 - Den User-Agent, also die verwendete Client-Software

2. **Response** (vom Server zum Client), enthält

- eine Status-Line (numerischer Code + Text zur Angabe von Fehlern),
- einen Response Header mit ggf. weiteren Optionen und
- den mittels CRLF (Carriage Return Line Feed) abgetrennten **Body**, welcher die eigentlichen Daten enthält.

¹ Über dieses Feld kann der Server bei so genannten Virtual Hosts unabhängig von der IP-Adresse entscheiden, welche Daten ausgeliefert werden sollen

² In HTTP in falscher Schreibweise als Referer[sic]

Hyper Text Transfer Protocol (HTTP)

HTTP Methods und Response Codes

Clients verwenden eine übersichtliche Anzahl unterschiedlicher Kommandos (Methods):

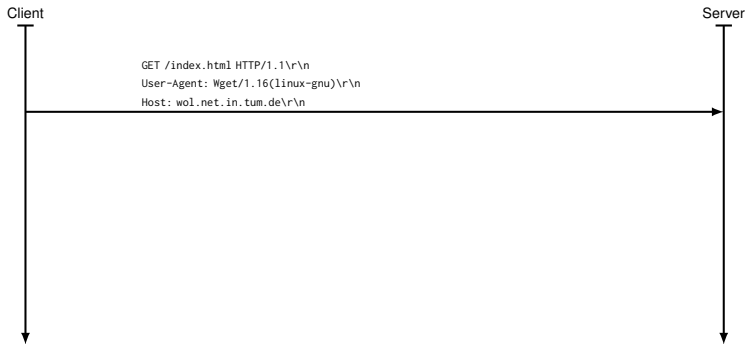
- **GET** – Anfrage zur Übertragung eines bestimmten Objekts vom Server
- **HEAD** – Anfrage zur Übertragung des Headers eines bestimmten Objekts (z. B. bestehen Webseiten aus mehreren Sektionen, wovon eine als Header bezeichnet wird)
- **PUT** – Übertragung eines Objekts vom Client zum Server, welches ggf. ein bereits existierendes Objekt überschreibt
- **POST** – Übertragung eines Objekts vom Client zum Server, welches ggf. an ein bereits existierendes Objekt angehängt wird (z. B. Anhängen von Text)
- **DELETE** – Löschen eines Objekts vom Server

Die Antworten des Servers geben mittels der Status-Line das Ergebnis der Anfrage an, z. B.

- | | |
|-----------------------------|--|
| • 200 – OK | • 404 – Not Found |
| • 3xx – Redirection | • 418 – I'm a teapot (RFC 2324) |
| • 400 – Bad Request | • 5xx – Server Error |
| • 401 – Unauthorized | • etc. |
| • 403 – Forbidden | |

Hyper Text Transfer Protocol (HTTP)

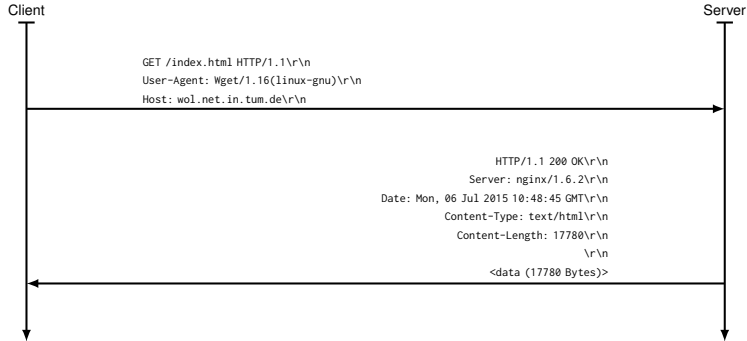
Beispiel: Zugriff auf `http://wol.net.in.tum.de/index.html`



- Client fordert mittel GET eine bestimmte Resource relativ zum Document Root an.
- Unter anderem sendet der Client den benutzten User-Agent, d. h. den verwendeten Browser.
- Host gibt noch einmal explizit den FQDN des Webservers an.¹

¹ Auf den ersten Blick ist dies überflüssig, da dieser ja bereits mittels DNS in eine IP-Adresse aufgelöst wurde und so das Ziel auf der Netzwerkschicht identifiziert. Das nochmalige Senden ermöglicht es allerdings, mehrere Webserver mit unterschiedlichen FQDNs bzw. URLs unter derselben IP-Adresse und Portnummer zugänglich zu machen.

Beispiel: Zugriff auf `http://wol.net.in.tum.de/index.html`



- Der Server antwortet mit dem entsprechenden Status Code.
- Zusätzlich werden verschiedene Optionen angegeben², hier insbesondere der Content-Type. Häufig werden auch das Content-Encoding und der Zeichensatz angegeben.
- Am Ende folgen die (entsprechend kodierten) Daten.

² Aus Platzgründen sind nicht alle Optionen abgebildet.

Hyper Text Transfer Protocol (HTTP)

Verschlüsselung bei HTTP

- HTTP selbst kennt keine Mechanismen zur Verschlüsselung übertragener Daten.
- Es besteht aber die Möglichkeit, Zwischen Transport- und Anwendungsschicht entsprechende Protokolle zu verwenden.

Hyper Text Transfer Protocol Secure (HTTPS)

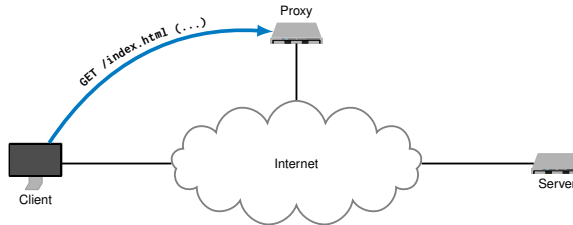
- Nutzt [Transport Layer Security \(TLS\)](#), ein Verschlüsselungsprotokoll oberhalb der Transportschicht.
- TLS ver- und entschlüsselt den Datentransfer.
- HTTP selbst bleibt unverändert.
- Zur Unterscheidung von unverschlüsselten Verbindungen wird TCP 443 anstelle von TCP 80 verwendet.
- Für TLS ist HTTP nur ein Datenstrom.

Hyper Text Transfer Protocol (HTTP)

HTTP Proxy

Manchmal ist es sinnvoll oder notwendig, dass ein Client keine Ende-zu-Ende-Verbindung zu einem HTTP-Server aufbaut, sondern einen dazwischenliegenden **Proxy** verwendet:

- Anstelle den Zielserver direkt zu kontaktieren, werden HTTP-Anfragen an einen HTTP-Proxy geschickt.
- Dieser nimmt die Anfragen entgegen, baut seinerseits eine neue Verbindung zum eigentlich Zielserver (oder einem weiteren Proxy) auf, und stellt die Anfrage anstelle des Clients.
- Die entsprechende Antwort wird zunächst dem Proxy gesendet, welcher sie (hoffentlich unmodifiziert) dem Client zustellt.
- Zur Unterscheidung von normalen HTTP-Servern verwenden Proxies häufig andere Portnummern wie z. B. TCP 3128.

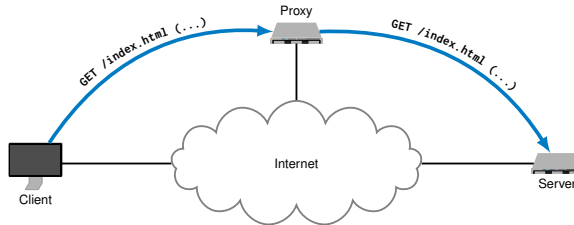


Hyper Text Transfer Protocol (HTTP)

HTTP Proxy

Manchmal ist es sinnvoll oder notwendig, dass ein Client keine Ende-zu-Ende-Verbindung zu einem HTTP-Server aufbaut, sondern einen dazwischenliegenden **Proxy** verwendet:

- Anstelle den Zielserver direkt zu kontaktieren, werden HTTP-Anfragen an einen HTTP-Proxy geschickt.
- Dieser nimmt die Anfragen entgegen, baut seinerseits eine neue Verbindung zum eigentlich Zielserver (oder einem weiteren Proxy) auf, und stellt die Anfrage anstelle des Clients.
- Die entsprechende Antwort wird zunächst dem Proxy gesendet, welcher sie (hoffentlich unmodifiziert) dem Client zustellt.
- Zur Unterscheidung von normalen HTTP-Servern verwenden Proxies häufig andere Portnummern wie z. B. TCP 3128.

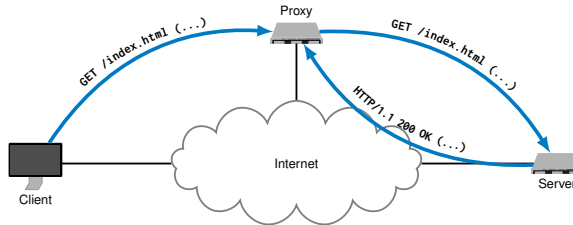


Hyper Text Transfer Protocol (HTTP)

HTTP Proxy

Manchmal ist es sinnvoll oder notwendig, dass ein Client keine Ende-zu-Ende-Verbindung zu einem HTTP-Server aufbaut, sondern einen dazwischenliegenden **Proxy** verwendet:

- Anstelle den Zielserver direkt zu kontaktieren, werden HTTP-Anfragen an einen HTTP-Proxy geschickt.
- Dieser nimmt die Anfragen entgegen, baut seinerseits eine neue Verbindung zum eigentlich Zielserver (oder einem weiteren Proxy) auf, und stellt die Anfrage anstelle des Clients.
- Die entsprechende Antwort wird zunächst dem Proxy gesendet, welcher sie (hoffentlich unmodifiziert) dem Client zustellt.
- Zur Unterscheidung von normalen HTTP-Servern verwenden Proxies häufig andere Portnummern wie z. B. TCP 3128.

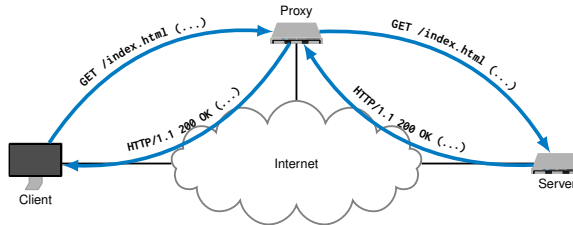


Hyper Text Transfer Protocol (HTTP)

HTTP Proxy

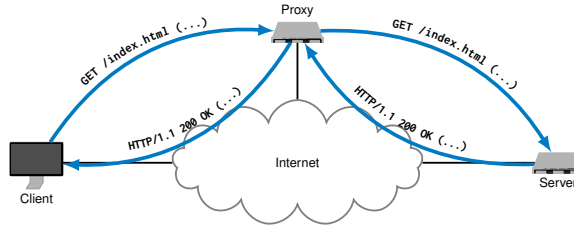
Manchmal ist es sinnvoll oder notwendig, dass ein Client keine Ende-zu-Ende-Verbindung zu einem HTTP-Server aufbaut, sondern einen dazwischenliegenden **Proxy** verwendet:

- Anstelle den Zielserver direkt zu kontaktieren, werden HTTP-Anfragen an einen HTTP-Proxy geschickt.
- Dieser nimmt die Anfragen entgegen, baut seinerseits eine neue Verbindung zum eigentlich Zielserver (oder einem weiteren Proxy) auf, und stellt die Anfrage anstelle des Clients.
- Die entsprechende Antwort wird zunächst dem Proxy gesendet, welcher sie (hoffentlich unmodifiziert) dem Client zustellt.
- Zur Unterscheidung von normalen HTTP-Servern verwenden Proxies häufig andere Portnummern wie z. B. TCP 3128.



Hyper Text Transfer Protocol (HTTP)

HTTP Proxy



- Der Client selbst bleibt hinter dem Proxy (weitgehend) verborgen, d. h. der Webserver sieht lediglich den Proxy, nicht aber den Client.¹
- Der Proxy kann Anfragen cachen, d. h. bei mehrfachen identischen Anfragen (auch unterschiedlicher Clients) können Inhalte aus einem Cache geliefert werden.
 - Sind die Inhalte noch aktuell?
 - Wie und wann müssen Inhalte aktualisiert werden?
- Proxies können auch **transparent** arbeiten, d. h. ohne Wissen der Clients.
 - In Firmennetzwerken häufig anzutreffen.
 - In einigen Fällen lassen sich Proxies sogar dazu „missbrauchen“ TLS-verschlüsselte Verbindungen zu überwachen.²

¹ Davon unbeeinträchtigt sind natürlich Informationen, die der Client mittels POST an den Server übermittelt.

² Dies setzt voraus, dass Clients das SSL-Zertifikat des Proxies als vertrauenswürdig einstufen, was sich im Rahmen administrierter Netzwerke leicht bewerkstelligen lässt.

Simple Mail Transfer Protocol (SMTP)

Das **Simple Mail Transfer Protocol (SMTP)** ist ein textbasiertes Protokoll und dient

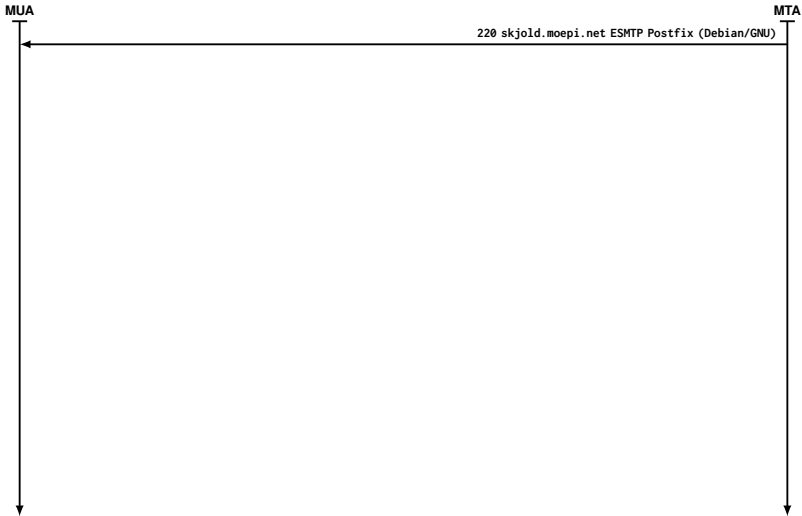
- dem Versenden von Emails, d. h. dem Transport vom **Mail User Agent (MUA)** zu einem **Mail Transfer Agent (MTA)**¹, sowie
- dem Transport von Emails zwischen MTAs.

In Empfangsrichtung werden i. d. R.

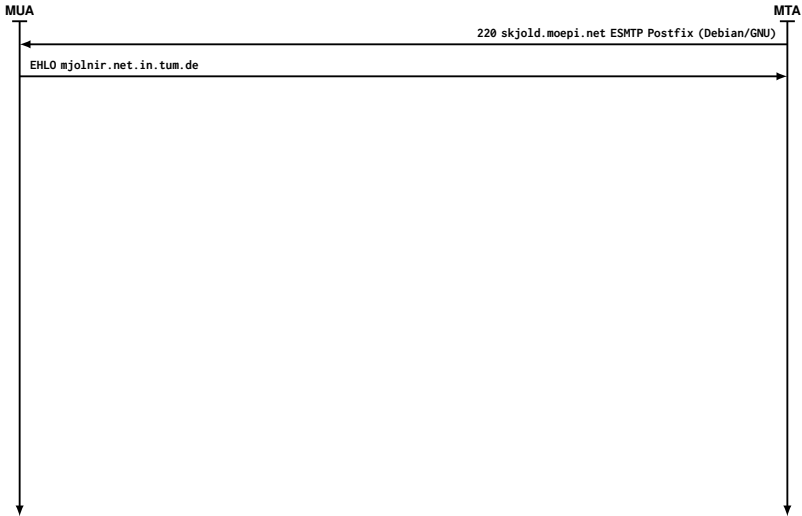
- **Post Office Protocol (POP)** oder
- **Internet Message Access Protocol (IMAP)** verwendet.

¹ Ein Email-Client besteht heute häufig aus einem MUA zusammen mit einem einfachen MTA. Letzterer sorgt für die lokale Annahme einer abgesendeten Mail. Sobald eine Verbindung zum nächsten MTA (meist dem konfigurierten SMTP-Server des jeweiligen Providers) möglich ist, wird die Email „versendet“. Ohne diesen integrierten MTA wäre das „Absenden“ einer Email ohne Internetverbindung nicht möglich.

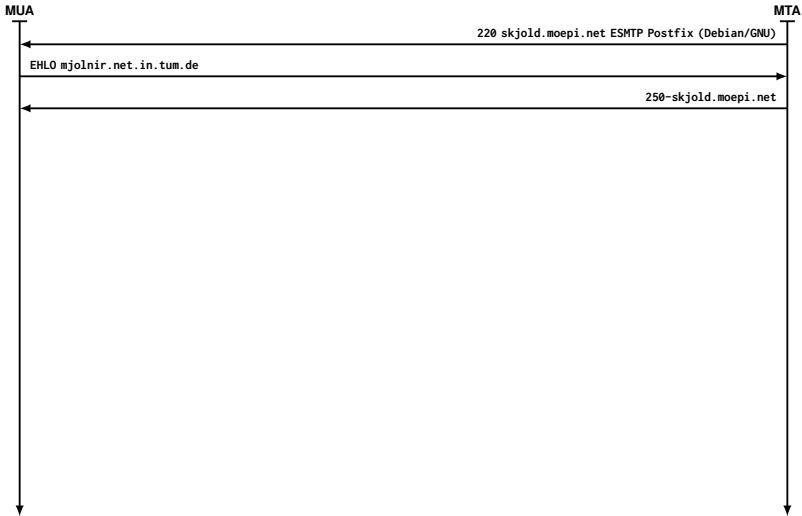
Beispiel: SMTP



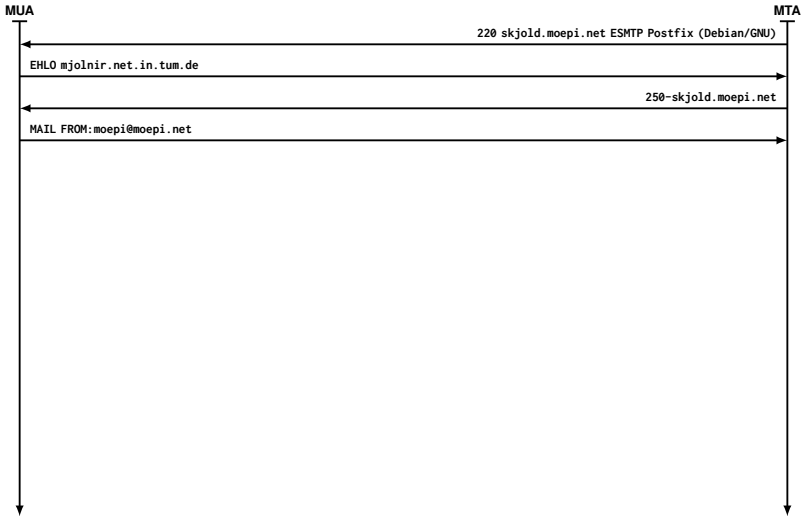
Beispiel: SMTP



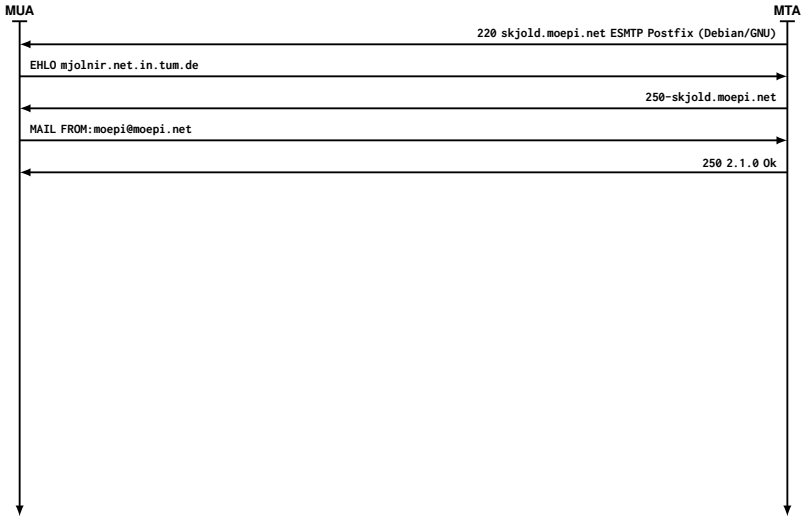
Beispiel: SMTP



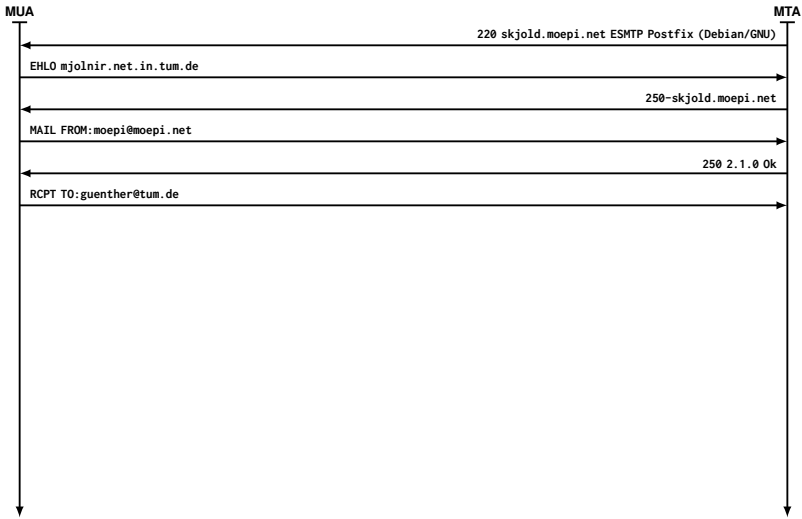
Beispiel: SMTP



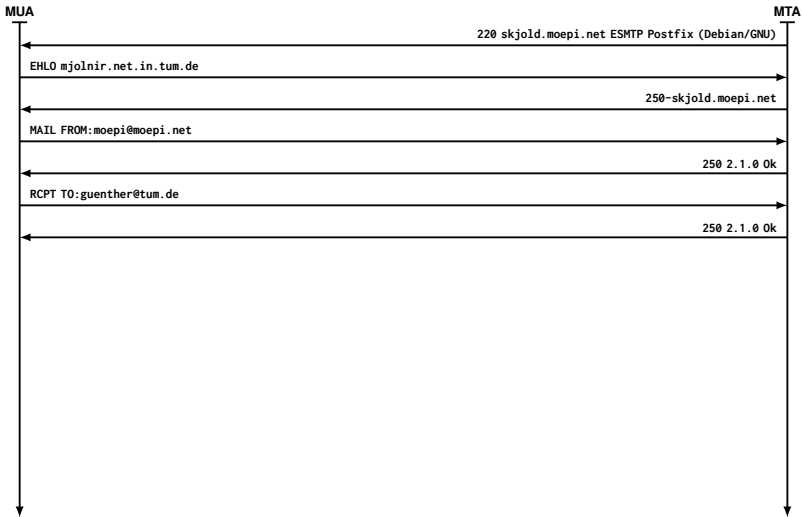
Beispiel: SMTP



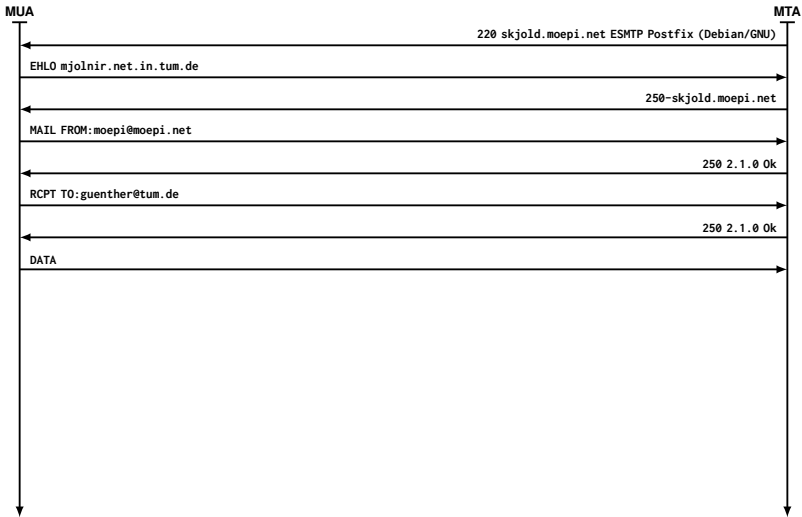
Beispiel: SMTP



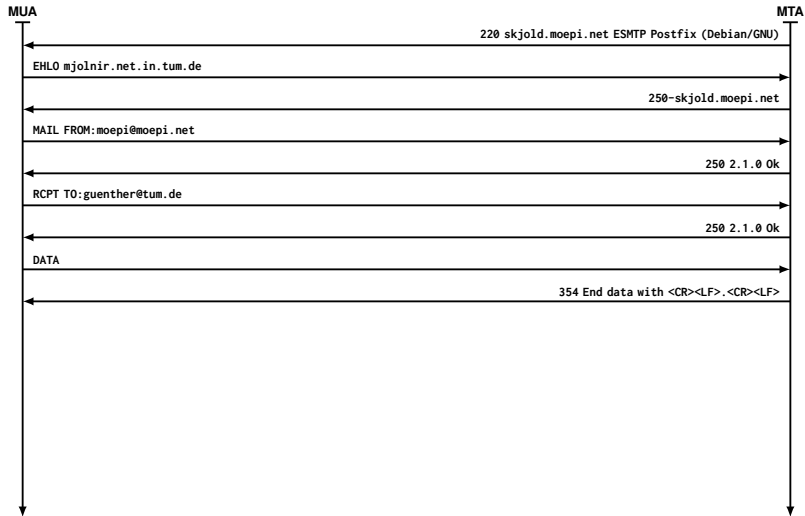
Beispiel: SMTP



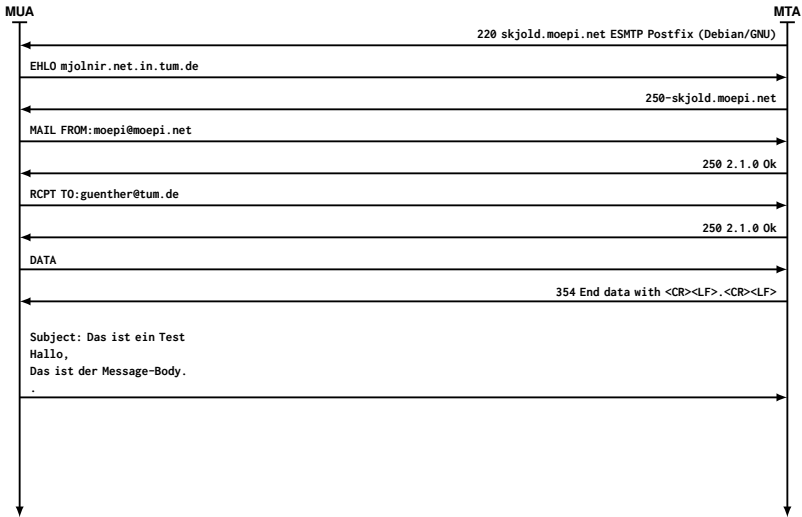
Beispiel: SMTP



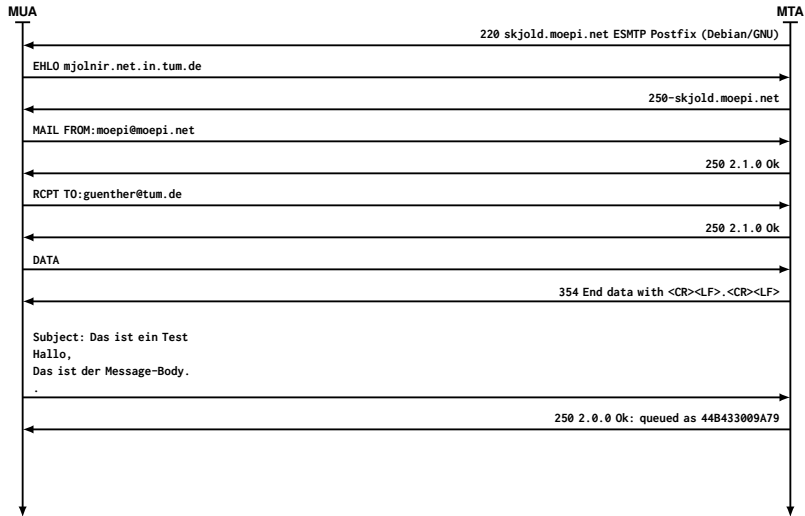
Beispiel: SMTP



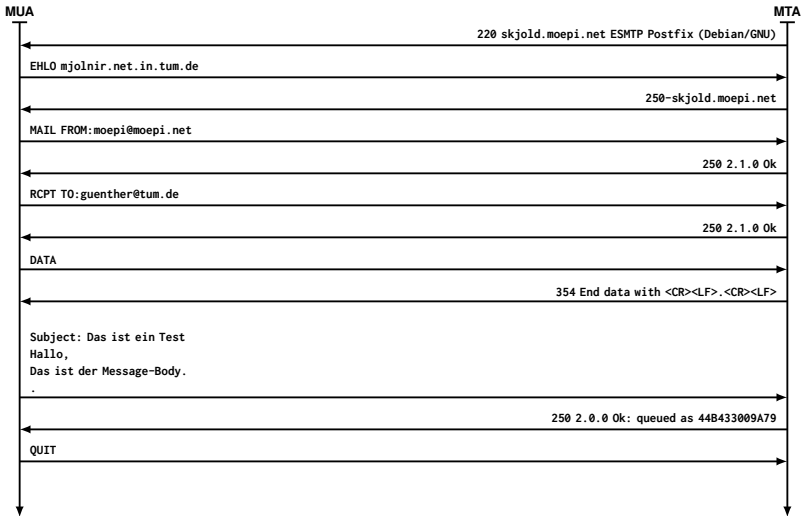
Beispiel: SMTP



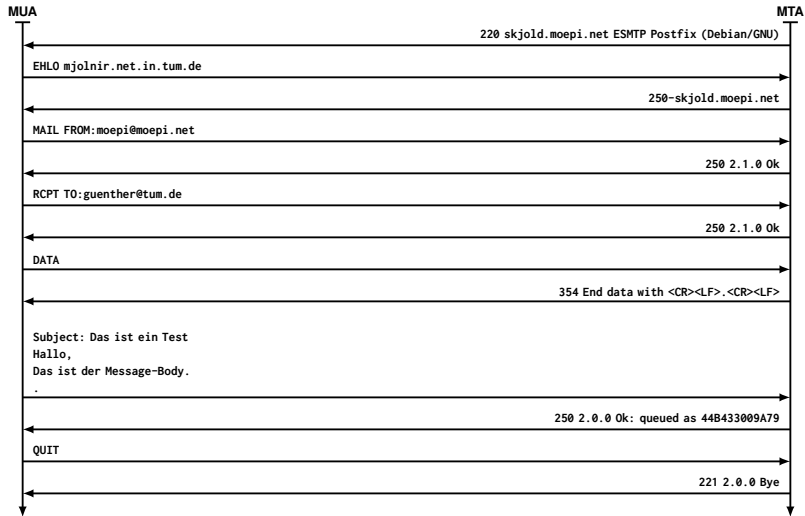
Beispiel: SMTP



Beispiel: SMTP



Beispiel: SMTP



Beispiel: SMTP (Fortsetzung)

Nach Erhalt der Email versucht der MTA diese zuzustellen:

- Die Empfängeradresse `guenther@tum.de` enthält den FQDN des Ziels.
- Mittels DNS werden die MX-Records der Domain `tum.de` identifiziert:

```
tum.de. 3600 IN MX 100 postrelay2.lrz.de.  
tum.de. 3600 IN MX 100 postrelay1.lrz.de.
```

Die beiden Einträge enthalten jeweils die TTL, Typ des DNS-Records, eine Präferenz (kleinere Werte bedeuten höhere Präferenz) sowie den FQDN des jeweiligen Mailservers.

- Der MTA wird nun eine weitere SMTP-Verbindung zu einem der beiden Mailserver aufbauen und versuchen, die Email weiterzureichen.
 - Falls erfolgreich, wird der MTA die Nachricht löschen. Der Absender wird nicht benachrichtigt.
 - Falls der Zielserver nicht erreichbar ist, wird der MTA es in regelmäßigen Zeitabständen wieder versuchen, nach einiger Zeit eine Delay-Notification an Absender zurücksenden und den Client im Fall einer Aufgabe benachrichtigen.
 - Falls der Zielserver die Annahme der Nachricht verweigert, wird der Absender mit dem entsprechenden Grund benachrichtigt.

Beispiel: SMTP (Fortsetzung)

Nach Erhalt der Email versucht der MTA diese zuzustellen:

- Die Empfängeradresse `guenther@tum.de` enthält den FQDN des Ziels.
- Mittels DNS werden die MX-Records der Domain `tum.de` identifiziert:

```
tum.de. 3600 IN MX 100 postrelay2.lrz.de.  
tum.de. 3600 IN MX 100 postrelay1.lrz.de.
```

Die beiden Einträge enthalten jeweils die TTL, Typ des DNS-Records, eine Präferenz (kleinere Werte bedeuten höhere Präferenz) sowie den FQDN des jeweiligen Mailservers.

- Der MTA wird nun eine weitere SMTP-Verbindung zu einem der beiden Mailserver aufbauen und versuchen, die Email weiterzureichen.
 - Falls erfolgreich, wird der MTA die Nachricht löschen. Der Absender wird nicht benachrichtigt.
 - Falls der Zielserver nicht erreichbar ist, wird der MTA es in regelmäßigen Zeitabständen wieder versuchen, nach einiger Zeit eine Delay-Notification an Absender zurücksenden und den Client im Fall einer Aufgabe benachrichtigen.
 - Falls der Zielserver die Annahme der Nachricht verweigert, wird der Absender mit dem entsprechenden Grund benachrichtigt.

Die Email wird auf dem Mailserver dem Empfänger zur Verfügung gestellt:

- **POP3** ermöglicht den „Abruf“ einer Email, welche im Anschluss vom Server gelöscht wird.
- **IMAP4** ermöglicht die Synchronisation einer Mailbox zwischen dem Server und einem oder mehreren Mailclients.

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```

From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbmGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHW7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

```

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```
From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFWuIEdhbGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit
```

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```

From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHW7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

```

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```

From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

```

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```

From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEhhbGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHW7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

```

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```

From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFWuIEdhbGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHw7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

```

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```

From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbmGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHW7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

```

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```

From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbmGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHW7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

```

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```

From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFWuIEdhbGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHW7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

```

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```

From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFuIEdhbmGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHW7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

```

Hi Stefan,
 wir haben fuer Freitag 11 Uhr einen Termin mit (...)

Message Header

Body

Simple Mail Transfer Protocol (SMTP)

Email (Beispiel):

```

From SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de Wed Jul 1 19:20:20 2015
Return-Path: <SRS0=Alge=HJ=net.in.tum.de=gallenmu@srs.mail.lrz.de>
Delivered-To: moepi@moepi.net
Received: from forwout2.mail.lrz.de (forwout2.mail.lrz.de [IPv6:2001:4ca0:0:103::81bb:ff83])
    by freya.moepi.net (Postfix) with ESMTPS id 452E430086F0
    for <moepi@moepi.net>; Wed, 1 Jul 2015 19:18:36 +0200 (CEST)
(...)
Received: from mail-out1.informatik.tu-muenchen.de (mail-out1.informatik.tu-muenchen.de [131.159.0.8])
    (using TLSv1 with cipher DHE-RSA-AES256-SHA (256/256 bits))
    (Client did not present a certificate)
    by postrelay1.lrz.de (Postfix) with ESMTPS
    for <guenther@tum.de>; Wed, 1 Jul 2015 19:18:33 +0200 (CEST)
Received: from grumpycat.net.in.tum.de (grumpycat.net.in.tum.de [131.159.20.204])
    by mail.net.in.tum.de (Postfix) with ESMTPSA id 5C75319110D9;
    Wed, 1 Jul 2015 19:18:30 +0200 (CEST)
Message-ID: <559420E5.2090101@net.in.tum.de>
Date: Wed, 01 Jul 2015 19:18:29 +0200
From: =?UTF-8?B?U2ViYXN0aWFWuIEdhbGxlbm3DvGxsZXI=?=
    <gallenmu@net.in.tum.de>
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.10; rv:31.0) Gecko/20100101 Thunderbird/31.7.0
MIME-Version: 1.0
To: Stefan <mail@example.com>
CC: =?UTF-8?B?U3RlcGhhbiBHW7xudGhlcg==?= <guenther@tum.de>,
    Johannes Naab <naab@net.in.tum.de>
Subject: [GRNVS-next] Antrittsgespraech
References: <21ed7f299038e5f6fdd48381fc3edfa7@localhost> <554B3F8A.7090403@gmx.de>
In-Reply-To: <554B3F8A.7090403@gmx.de>
Content-Type: text/plain; charset=utf-8
Content-Transfer-Encoding: 8bit

```

```

Hi Stefan,
wir haben fuer Freitag 11 Uhr einen Termin mit (...)

```

Message Header

Body

Hinweise:

- Ein MTA akzeptiert Emails i. d. R. nur für die eigenen Domains, nicht aber für fremde Ziel-Domains. Andernfalls spricht man von einem **Open Relay**, der schnell zum Versand von Spam missbraucht wird und auf Blacklists landet.
- Authentifizierung zwischen MTAs ist unüblich da technisch nicht umsetzbar.¹
- Sowohl SMTP als auch POP / IMAP können mittels TLS verschlüsselt werden. Die Portnummern ändern sich dabei (vgl. HTTP und HTTPS).
- Die Verbindungen zwischen MTAs können **opportunistisch verschlüsselt** werden, d. h. es wird kein Wert darauf gelegt die Gegenseite zu authentifizieren. Es wird lediglich der Transportweg verschlüsselt. Der Nutzer bzw. MUA hat hierauf keinen Einfluss.

¹ Ausnahme stellen MTAs dar, welche als sog. **Smarthost** oder **Relay-Host** arbeiten und Emails von anderen (bekannten) MTAs entgegennehmen, da diese selbst nicht in der Lage sind, Emails erfolgreich an andere MTAs zuzustellen. Beispiele sind MTAs mit dynamischen IP-Adressen, welche meist keine PTR-Records besitzen und infolge dessen häufig von anderen MTAs zurückgewiesen werden (Spamschutz).

Das [File Transfer Protocol \(FTP\)](#) [9] ist ein weiteres Protokoll zum Transfer von Daten (Text wie Binärdaten). Unterschiede zu HTTP:

- FTP nutzt zwei getrennte TCP-Verbindungen:
 1. Kontrollkanal zur Übermittlung von Befehlen und Statuscodes zwischen Client und Server.
 2. Datenkanal zur Übertragung der eigentlichen Daten.
- Der Kontrollkanal bleibt über mehrere Datentransfers hinweg bestehen, d. h. FTP ist [stateful](#).
- FTP erfordert grundsätzlich eine Art von Authentifizierung (anonymer Zugang mittels Benutzername `anonymous` und beliebigem Passwort sofern konfiguriert).

Das **File Transfer Protocol (FTP)** [9] ist ein weiteres Protokoll zum Transfer von Daten (Text wie Binärdaten). Unterschiede zu HTTP:

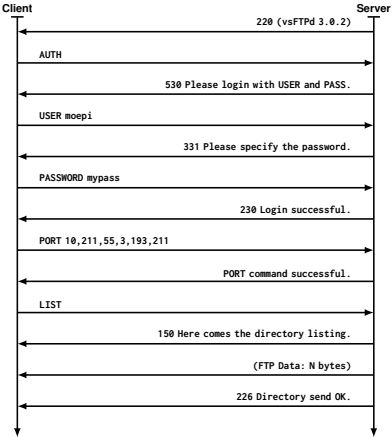
- FTP nutzt zwei getrennte TCP-Verbindungen:
 1. Kontrollkanal zur Übermittlung von Befehlen und Statuscodes zwischen Client und Server.
 2. Datenkanal zur Übertragung der eigentlichen Daten.
- Der Kontrollkanal bleibt über mehrere Datentransfers hinweg bestehen, d. h. FTP ist **stateful**.
- FTP erfordert grundsätzlich eine Art von Authentifizierung (anonymer Zugang mittels Benutzername `anonymous` und beliebigem Passwort sofern konfiguriert).

FTP arbeitet entweder im **active** oder **passive mode**:

- In beiden Fällen baut der Client den Kontrollkanal zum Server auf TCP 21 auf.
- Im **active mode** teilt der Client mittels des **PORT**-Kommandos dem Server eine zufällige Portnummer mit, auf der der Server vom Quellport TCP 20 eine neue TCP-Verbindung zum Client aufbaut, die als Datenkanal verwendet wird.
- Im **passive mode** sendet der Client das Kommando **PASV** über den Kontrollkanal und erhält vom Server IP-Adresse und Portnummer, zu der der Client eine zweite TCP-Verbindung aufbauen soll, die wiederum als Datenkanal verwendet wird.

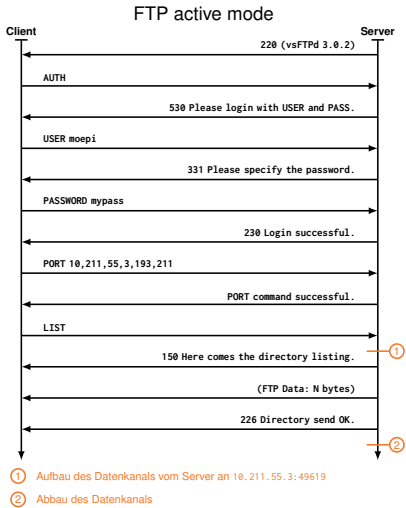
Beispiel:

FTP active mode



FTP passive mode

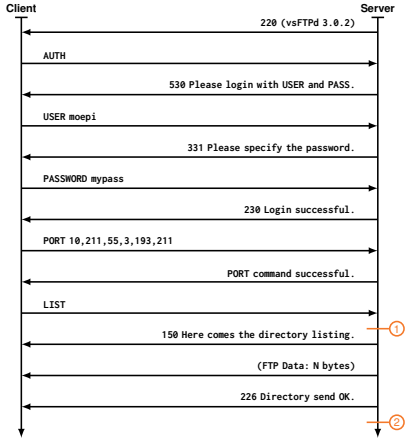
Beispiel:



FTP passive mode

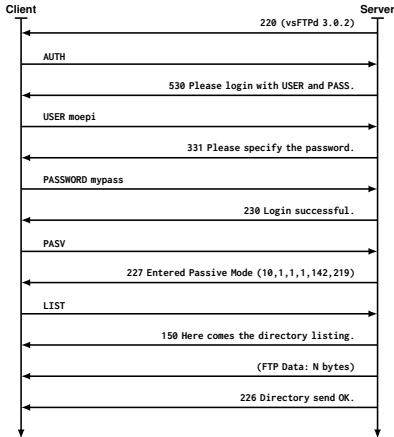
Beispiel:

FTP active mode



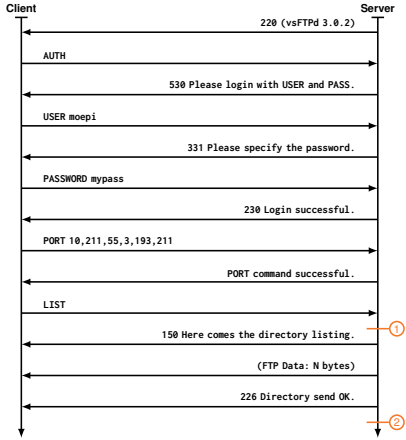
- ① Aufbau des Datenkanals vom Server an 10.211.55.3:49619
- ② Abbau des Datenkanals

FTP passive mode



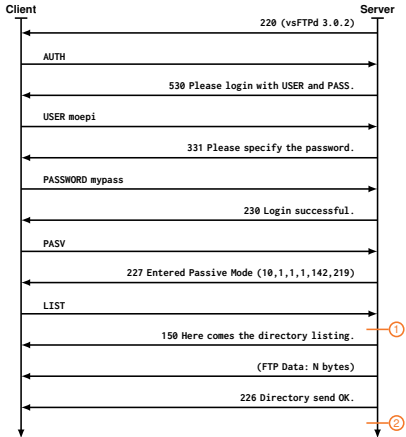
Beispiel:

FTP active mode



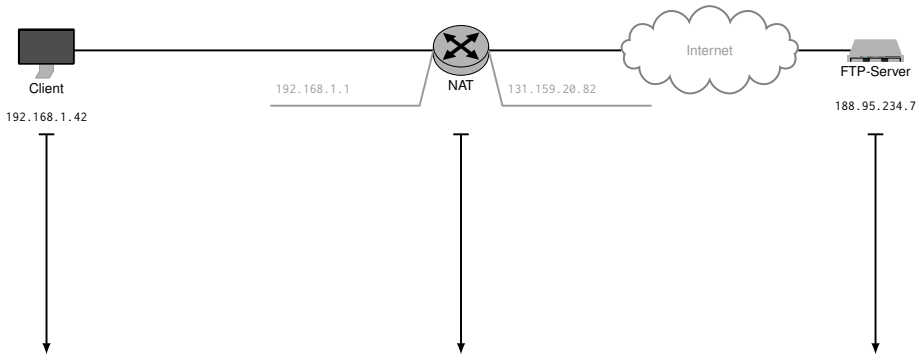
- ① Aufbau des Datenkanals vom Server an 10.211.55.3:49619
- ② Abbau des Datenkanals

FTP passive mode

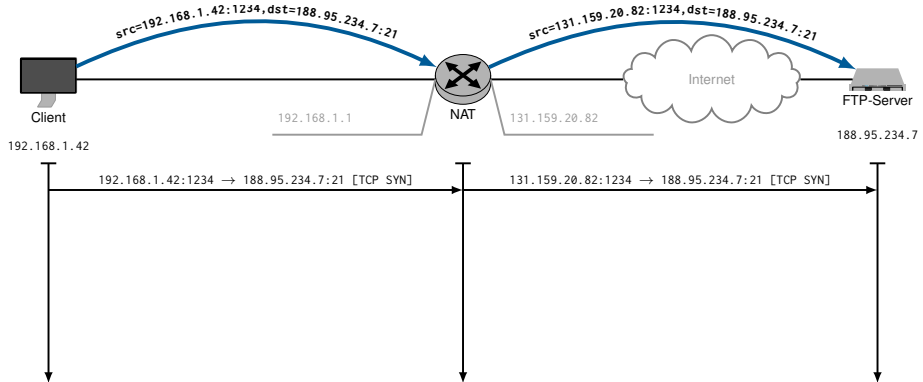


- ① Aufbau des Datenkanals vom Client an <server>:36571
- ② Abbau des Datenkanals

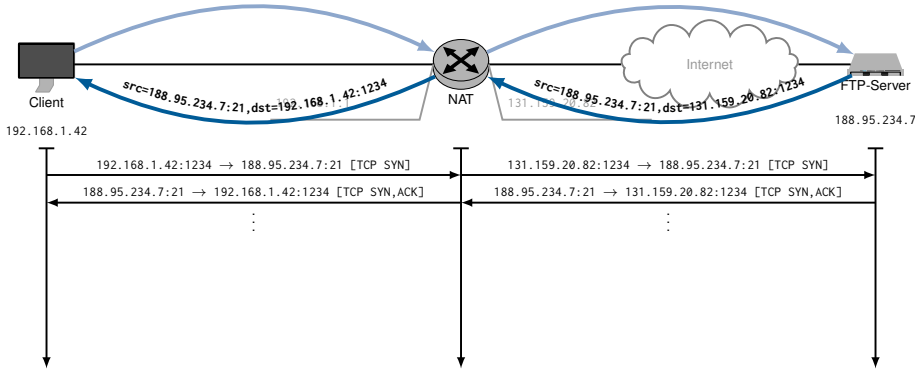
FTP und NAT: Was ist das Problem mit FTP active mode und NAT?



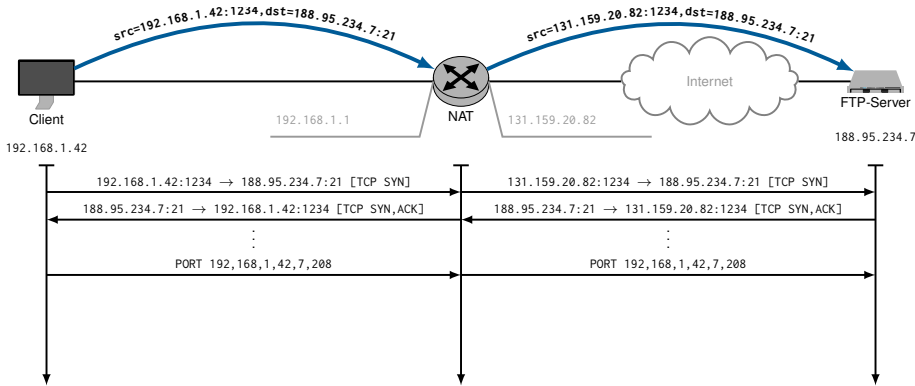
FTP und NAT: Was ist das Problem mit FTP active mode und NAT?



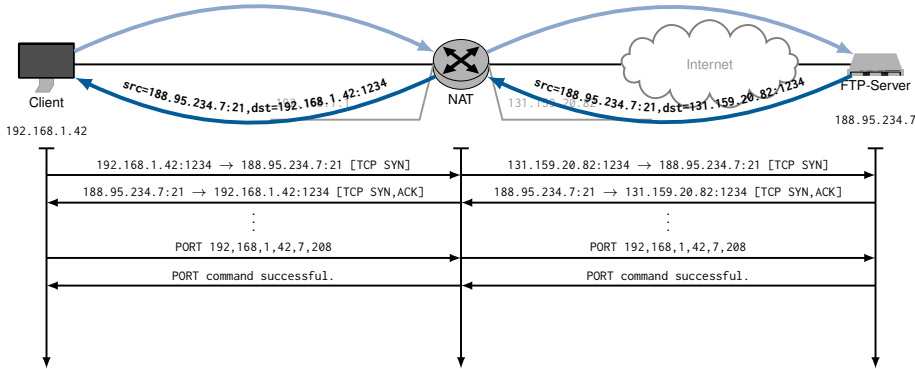
FTP und NAT: Was ist das Problem mit FTP active mode und NAT?



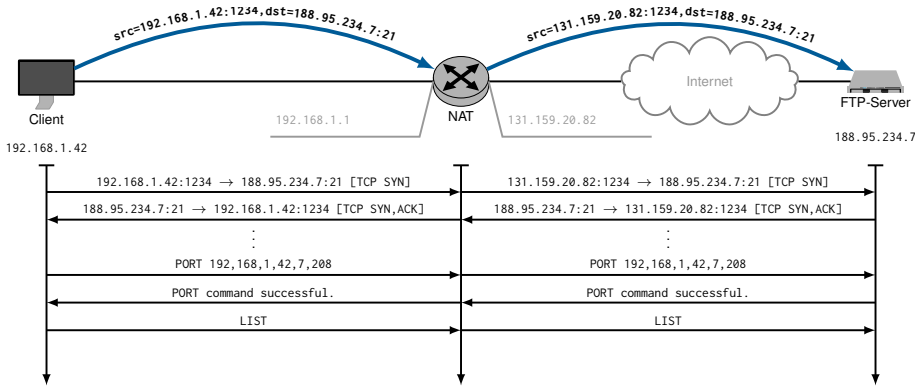
FTP und NAT: Was ist das Problem mit FTP active mode und NAT?



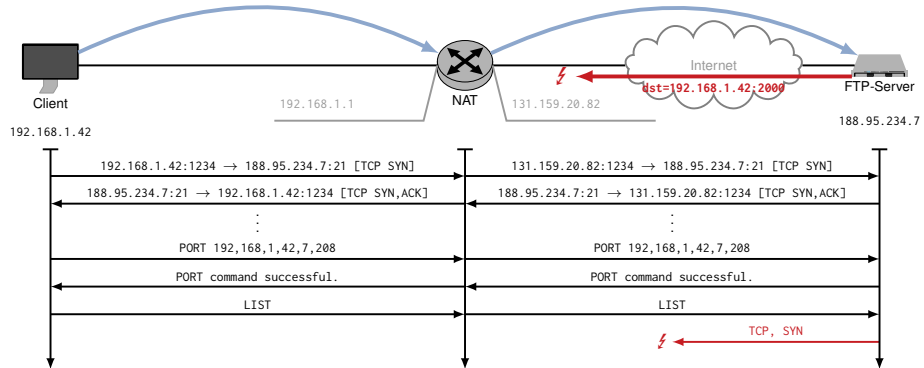
FTP und NAT: Was ist das Problem mit FTP active mode und NAT?



FTP und NAT: Was ist das Problem mit FTP active mode und NAT?



FTP und NAT: Was ist das Problem mit FTP active mode und NAT?



- Der Server versucht infolge des LIST-Kommandos eine Verbindung zu 192.168.1.42:2000 – wie zuvor über das PORT-Kommando ausgehandelt – aufzubauen.
- Das schlägt allein schon wegen der privaten IP-Adresse fehl.
- Selbst wenn die Adresse öffentlich erreichbar wäre, hätte das NAT keinen passenden Eintrag für eine eingehende Verbindung auf TCP 2000.

Lösungen:

1. Die NAT-Implementierung wird so erweitert, dass sie FTP unterstützt.

- NAT müsste die L7-PDU auf das Vorhandensein von PORT prüfen und bei Auftreten sowohl
- die private IP-Adresse durch eine öffentlich gültige ersetzen als auch
- einen Eintrag in der NAT-Tabelle für den entsprechenden Port vornehmen.

2. FTP passive mode

- Da hier der Server keine Verbindung zum Client aufbaut, sondern dieser eine zweite Verbindung zum Server aufbaut, sollte es mit NAT vorerst keine Probleme geben.
- Problematisch wird es, wenn der Server selbst hinter einem NAT steht, das lediglich TCP 21 an eine bestimmte private Adresse weiterleitet.
- Ebenfalls problematisch ist der Einsatz von [Firewalls](#), die nur Verbindungen zu bestimmten Ports erlauben.

Einordnung im ISO/OSI-Modell

Sitzungsschicht

Darstellungsschicht

Anwendungsschicht

Literaturangaben

- [1] [The JSON Data Interchange Format](#), 2017.
<https://ecma-international.org/publications-and-standards/standards/ecma-404/>.
- [2] [S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper.](#)
Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.
<https://tools.ietf.org/html/rfc5280>.
- [3] [R. Braden.](#)
Requirements for Internet Hosts – Communication Layers, 1989.
<https://tools.ietf.org/html/rfc1122>.
- [4] [T. Bray.](#)
The JavaScript Object Notation (JSON) Data Interchange Format, 2014.
<https://tools.ietf.org/html/rfc7159>.
- [5] [R. Elz and R. Bush.](#)
Clarifications to the DNS Specifications, 1997.
<https://tools.ietf.org/html/rfc2181>.
- [6] [S. Josefsson.](#)
The Base16, Base32, and base64 data encodings, 2006.
<https://tools.ietf.org/html/rfc4648>.
- [7] [P. Moackapetris.](#)
Domain Names – Concepts and Facilities, 1987.
<https://tools.ietf.org/html/rfc1034>.
- [8] [P. Moackapetris.](#)
Domain Names – Implementation and Specification, 1987.
<https://tools.ietf.org/html/rfc1035>.
- [9] [J. Postel and J. Reynolds.](#)
File Transfer Protocol (FTP), 1985.
<https://tools.ietf.org/html/rfc959>.