Grundlagen Rechnernetze und Verteilte Systeme

Kapitel 5 – Sitzungs-, Darstellungs- und Anwendungsschicht

SoSe 2015

Technische Universität München
Fakultät für Informatik

Fachgebiet für Betriebssysteme
Prof. Dr. Uwe Baumgarten, Sebastian Eckl

Lehrstuhl für Netzarchitekturen und Netzdienste
Stephan M. Günther, Johannes Naab, Marcel von Maltitz

7. Juli 2015

Worum geht es in diesem Kapitel?

Einordnung im ISO/OSI-Modell

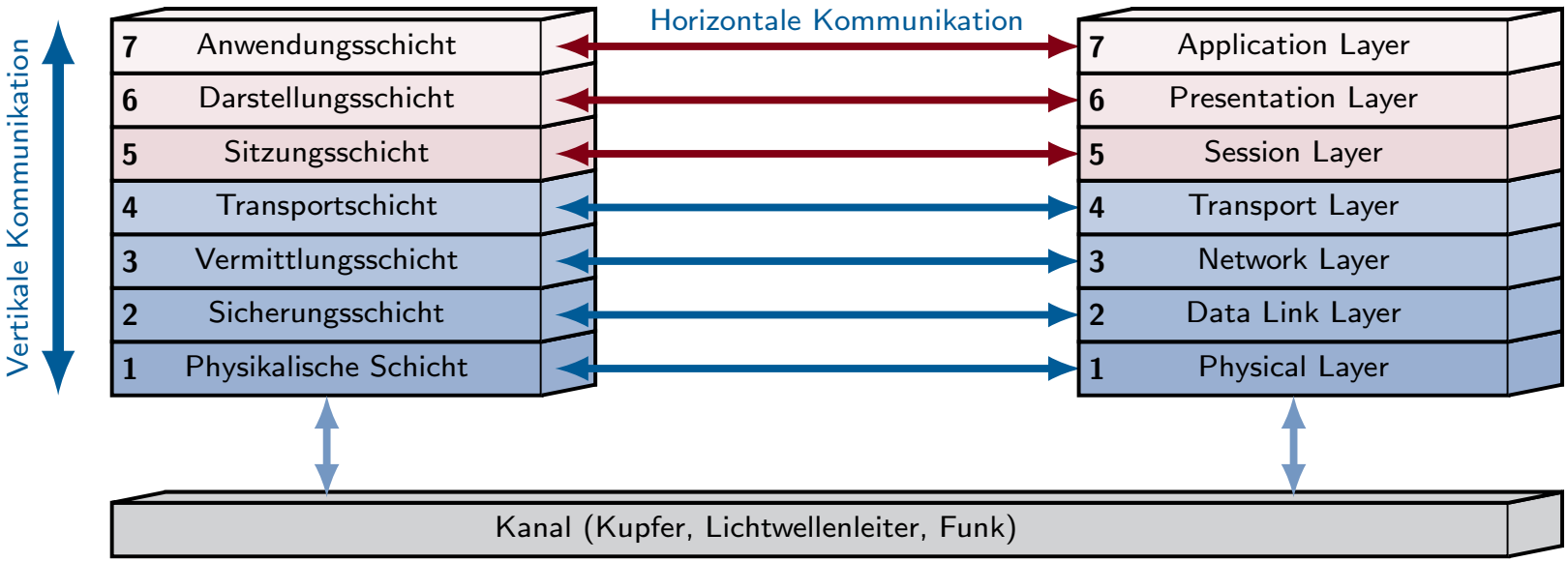
Sitzungsschicht
- Dienste der Sitzungsschicht
- Realisierung der Funktionalität der Sitzungsschicht

Darstellungsschicht
- Aufgaben der Darstellungsschicht
- Zeichensätze und Kodierung
- Kodierung
- Strukturierte Darstellung
- Datenkompression

Anwendungsschicht
- Domain Name System (DNS)
- Uniform Resource Locator (URL)
- HyperText Transfer Protocol (HTTP)
- Simple Mail Transfer Protocol (SMTP)
- File Transfer Protocol (FTP)

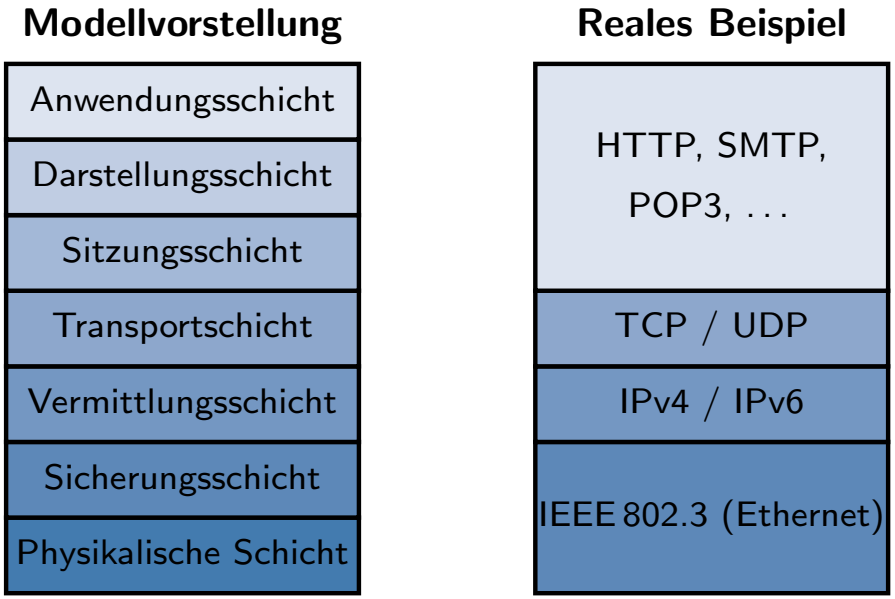
Zusammenfassung

Einordnung im ISO/OSI-Modell



Modell und Realität

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



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 Sitztung-, Darstellungs- und Anwendungsschicht möglich.
- Im Internet Model (RFC 1122) 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 und 6 zugeordnet werden können,
- sowie wichtige Protokolle der Anwendungsschicht erläutern.

Übersicht

Sitzungsschicht

Die Sitzungsschicht bietet nach X.200 in zwei grundlegend verschiedene Betriebisarten:

1. verbindungsorientert (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.
2. 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 ist nicht gleichbedeutend mit einer Verbindung der Transportschicht. Eine Session kann beispielsweise nacheinander mehrere TCP-Verbindungen beinhalten.

Dienste der Sitzungsschicht

Definition (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 wiederaufgenommer TCP-Verbindungen beinhalten.

Im verbindungsorientierten Modues 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: Dringliche Daten, z. B. Alarme oder Interrupts

Realisierung der Funktionalität der Sitzungsschicht

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

- HTTP ist zunächst zustandslos.
- Zwischen mehreren Anfragen und Antworten besteht 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 einer Website (bzw. Anwendung) auf den Client übertragen und dort gespeichert werden können.
- Dadurch wird es möglich, zeitlich getrennte oder von unterschiedlichen Adressen stammende Anfragen einem bestimmten Client und sogar einem bestimmten Nutzer zuordnen zu können.
- HTTP wird überlicherweise der Anwendungsschicht (Schicht 7) zurgeordnet, beinhaltet aber auch Funktionen der Darstellungs- und Sitzungsschicht (Schichten 6/5).
- Eine strikte Zuordnung ist nur schwer möglich.

Beispiel 2: TLS (Transport Layer Security)[1]

- TLS ist ein Protokoll zur verschlüsselten Übertragung von Daten.
- Es ist die Grundlage beispielsweise für HTTPS.
- Es bietet unter anderem
  - Authentifizierung,
  - Integritätsschutz und
  - Vertraulichkeit (Verschlüsselung).
- Beim Verbindungsaufbau werden zunächst die Kommunkationsparameter der Sitzung (z. B. Algorithmen, Schlüssel, Authentifizierung) ausgehandelt.
- Sitzungen können mit Hilfe von Session-IDs oder Session-Tickets über mehrere TCP-Verbindungen hinweg erhalten bleiben.
- Während die Funktionen zur Verwaltung und Wiederaufnahme von Sessions der Sitzungsschicht zuzuordnen sind, fallen insbesondere die Verschlüsselungsfunktionen in den Bereich der Darstellungsschicht.

[1] Dient hier lediglich als Beispiel, wird im Rahmen der Vorlesung aber nicht eingehender behandelt. Man sollte allerdings wissen, wozu es dient.

Übersicht

Darstellungsschicht

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.[2]

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

[2] Unter Syntax versteht man die Darstellung von Daten nach bestimmten Regeln (Grammatik). Werden Daten durch Bedeutung ergänzt, spricht man von Information (Aufgabe der Semantik).

Aufgaben der Darstellungsschicht

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
  - Plattformunabängige, einheitliche Darstellung
  - Übersetzung zwischen verschiedenen Datenformaten
  - Serialisierung von Binärdaten (Übersetzung von binären Datenformaten in Textdarstellungen)

Wie auch bei der Sitzungschicht gilt hier: Protokolle lassen sich meist nicht eindeutig der Darstellungsschicht zuordnen, da sie häufig auch Funktionen anderer Schichten erfüllen.

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.

Zeichensätze und Kodierung

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 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 Darstellung 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 kontextabängig (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 rekodieren, so dass sie in einem Zeichensatz darstellbar werden.

[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 (aber genauso wenig standardisiert sind wie das Byte). Diese sind letztendlich durch die Registergröße der Prozessoren beschränkt. Da der Speicher aber byteweise adressiert wird, ist es prozessorabhängig, in welcher Reihenfolge die einzelnen Oktette geladen werden. Folglich ist die Byte Order von essentieller Bedeutung.

- Ein Zeichensatz verknüpft ein Zeichen mit einem Codepoint.
- 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 sind Steuerzeichen.
  - Steuerzeichen sind z. B. Zeilenumbruch, Tabulator, und Protokollzeichen.
  - Ursprünglich wurden darüber Terminals und Drucker angesteuert.
- ISO-8859-15 definiert 256 Codepoints.
  - Zeichen 0 – 127 entsprechen dem ASCII Zeichensatz.
  - ISO-8859-15 war vor Unicode der im westeuropäschen Sprachraum ein verbreiteter Zeichensatz.

- Ein Zeichensatz verknüpft ein Zeichen mit einem Codepoint.
- 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 sind Steuerzeichen.
  - Steuerzeichen sind z. B. Zeilenumbruch, Tabulator, und Protokollzeichen.
  - Ursprünglich wurden darüber Terminals und Drucker angesteuert.
- ISO-8859-15 definiert 256 Codepoints.
  - Zeichen 0 – 127 entsprechen dem ASCII Zeichensatz.
  - ISO-8859-15 war vor Unicode der im westeuropäschen Sprachraum ein verbreiteter Zeichensatz.

Unicode

- Unicode in Version 8.0 definiert 120 737 Zeichen.
- Die Größe des Coderaums (Anzahl möglicher Zeichen) beträgt bei Unicode 1 114 112 Zeichen.
- Unicode hat das Ziel, alle Schriftkulturen und Zeichensysteme abzubilden.
- Die Codepoints 0 – 255 entsprechen ISO-8859-1.
- Unicode definiert, wie Zeichen normalisiert, sortiert und verglichen werden sollen.
- Im Gegensatz zu eingeschränkten Zeichensätzen wie z. B. ASCII und ISO-8859-15 wird der Zeichensatz bei Unicode regelmäßig aktualisiert und erweitert.

Kodierung

Zur Übertragung müssen 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).
- 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 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. Am häufigsten wird hier hierfür UTF-8 verwendet, welches kompatibel zu ASCII[2] ist.

[1] Morse-Zeichen sind ein weiteres Beispiel: Hier werden häufiger auftretenden Zeichen kürzere Codewörter 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 ledliglich eine Notation der Codepoints für Unicode. Die hexadezimalen Ziffern geben dabei den Wert der kodierten Bits eines Codeworts an.
- Bei Codewörtern, die länger als 1 B sind, 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 präfixfrei[1] und selbstsychronisierend.
- 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 andere Codewörter
- ASCII: Keine Kodierung möglich, da Umlaute nicht Teil des Zeichensatzes ist.

Beispiel 2: Kodierung des Eurosymbols €

- Unicode: Codepoint 8384, Codewort in UTF-8: `U+00E4 = 11100010 10000010 10101100`
- ISO-8859-15: Codepoint 164, Codewort `10100100`
- ISO-8859-1 und ASCII: Keine Kodierung möglich

[1] Vertippt man sich und schreibt `Verschlu\"sselung` anstelle von `Verschl\"usselung`, ergibt das dann in den Folien Verschlußselung.

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 andere Codewörter
- ASCII: Keine Kodierung möglich, da Umlaute nicht Teil des Zeichensatzes ist.

Beispiel 2: Kodierung des Eurosymbols €

- Unicode: Codepoint 8384, Codewort in UTF-8: `U+00E4 = 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. `&#288;`
- HTML erlaubt die Kodierung häufiger verwendeter Zeichen mittels named entities, z. B. `&auml;`.
- LaTeXerlaubt die Kodierung verschiedener Zeichen auf änliche Art, z. B. `\"a`[1]

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

[1] Vertippt man sich und schreibt `Verschlu\"sselung` anstelle von `Verschl\"usselung`, ergibt das dann in den Folien Verschlußselung.

Strukturierte Darstellung

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 JSON oder XML.

[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

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 JSON oder XML.

Beispiel: JavaScript Object Notation (JSON)

- Definiert in ECMA-404 [1] und RFC 7159 [3].
- 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.

JSON

JSON definiert die folgenden Datentypen:

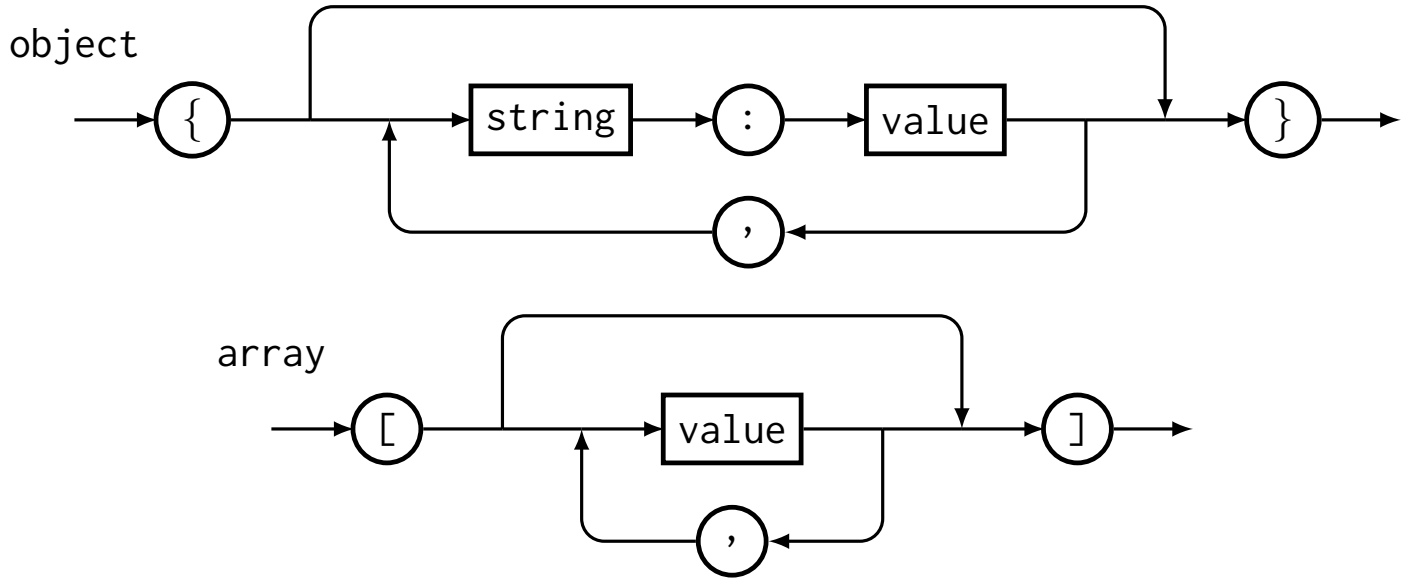
- number
- string
- boolean
- array
- object
- null

- Zwischen den Elementen kann (beliebiger) Whitespace eingefügt werden.
- JSON wird im allgemeinen als UTF-8 kodiert.
- JSON Dokumente besitzen, anders als XML, kein explizites Schema.

Beispiel:

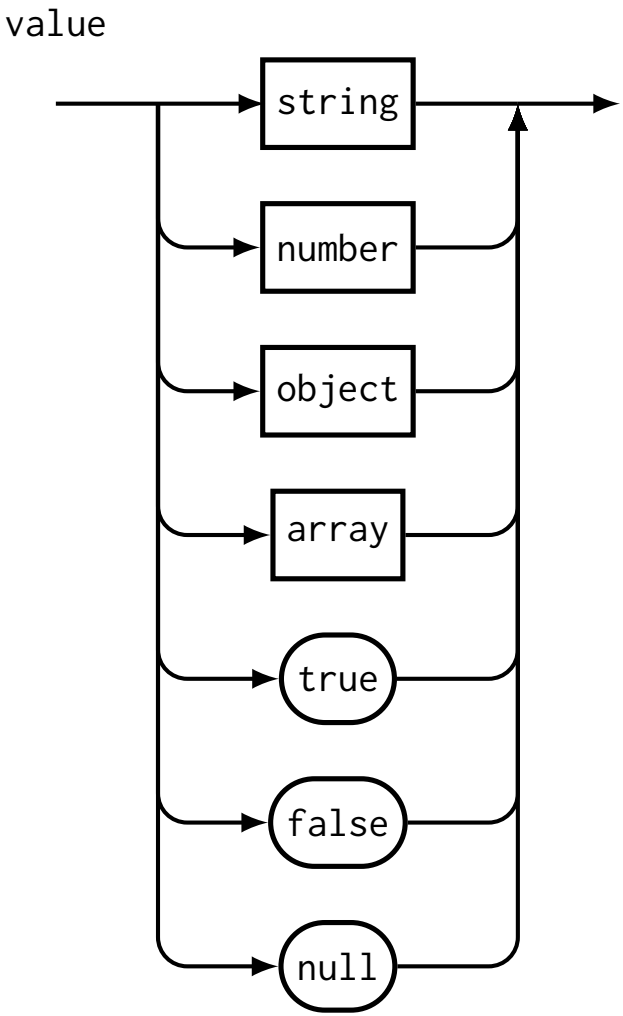
```
{
    "Image": {
        "Width":  800,
        "Height": 600,
        "Title":  "View from 15th Floor",
        "Thumbnail": {
            "Url":    "http://www.example.com/image/481989943",
            "Height": 125,
            "Width":  100
        },
        "Animated" : false,
        "IDs":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

JSON Objects und Arrays



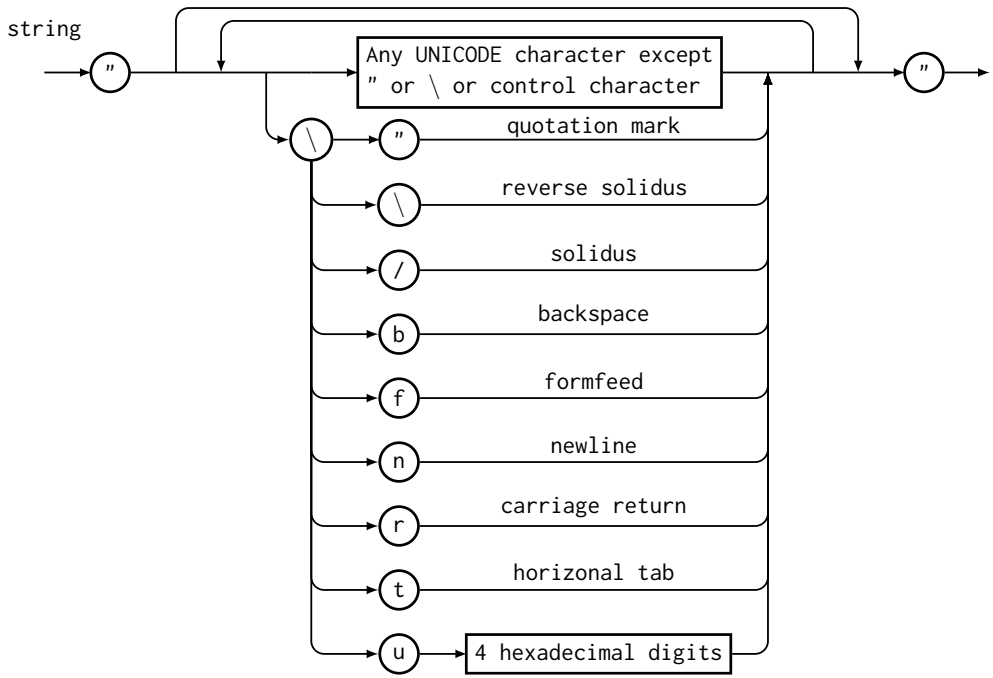
- Ein JSON-Objekt ist eine ungeordnete Sammlung von Key/Value-Paaren.
- Der Key (Schlüssel) ist ein Unicode-String.
- Die Values (Werte) können unterschiedliche Typen aufweisen.
- Eine Sammlung von JSON-Objekten ist konzeptuell vergleichbar mit Hashmaps bzw. Dictionaries verschiedener Programmiersprachen.
- 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.

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.

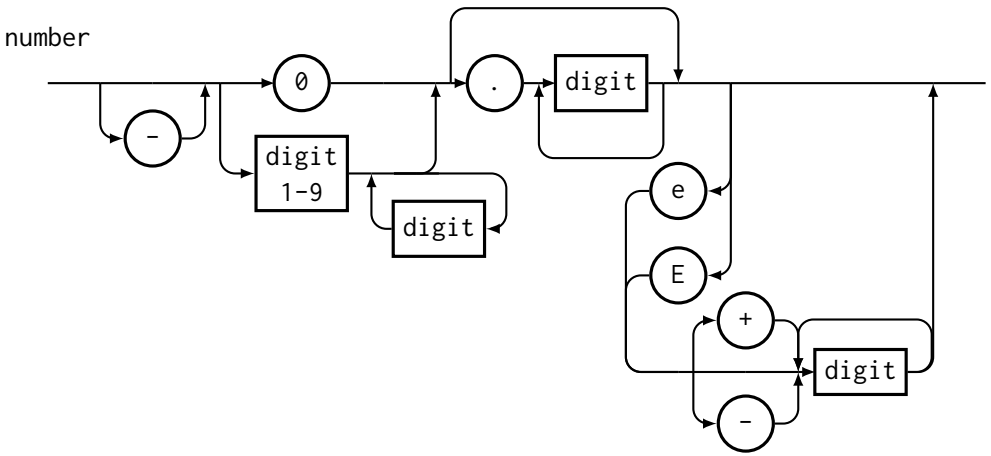
JSON Strings



- 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 Serialiserung der Binärdaten z. B. mittels Base64 [5] umgangen werden.[1]

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

JSON Numbers



- 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.

Datenkompression

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, PNG[2] (Bilder), FLAC[3] (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, ...

[2] Portable Network Graphics
[3] Free Lossless Audio Codec

Beispiel 1: Huffman-Code

- Viele Protokolle komprimieren Daten vor dem Senden (Quellenkodierung).
- TLS beispielsweise bietet optional Kompressionsmethoden. Diese werden vor der Verschlüsselung angewandt. (Warum davor?)
- Ein häufig (u. a. von TLS) verwendetes Kompressionsverfahren für Texte ist der Huffman-Code.

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 eines Huffman-Codes

- 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.[1]

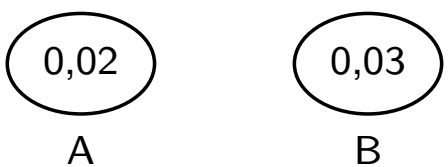
z	$\Pr[X = z]$
A	0,02
B	0,03
C	0,05
D	0,08
E	0,12
F	0,15
G	0,25
H	0,30

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

Konstruktion eines Huffman-Codes

- 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.[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

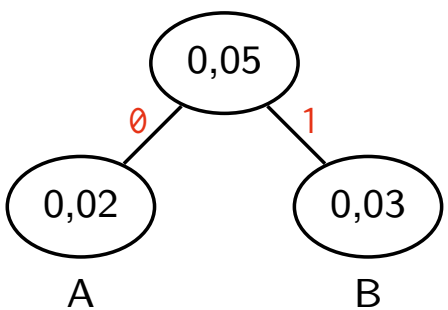


[1] Andernfalls würden die Aussagen zur Optimalität des Huffman-Codes im Allgemeinen nicht mehr zutreffen.

Konstruktion eines Huffman-Codes

- 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.[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

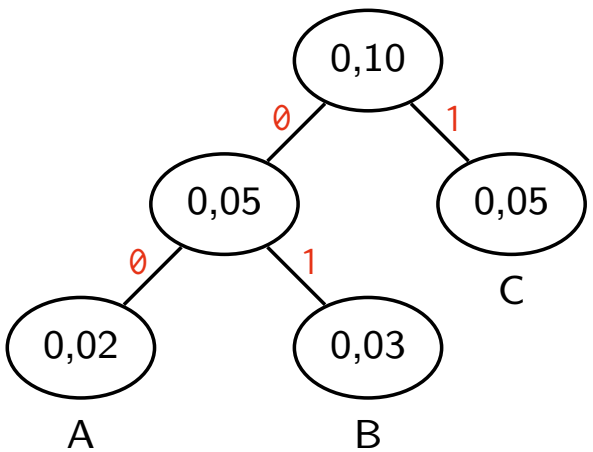


[1] Andernfalls würden die Aussagen zur Optimalität des Huffman-Codes im Allgemeinen nicht mehr zutreffen.

Konstruktion eines Huffman-Codes

- 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.[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

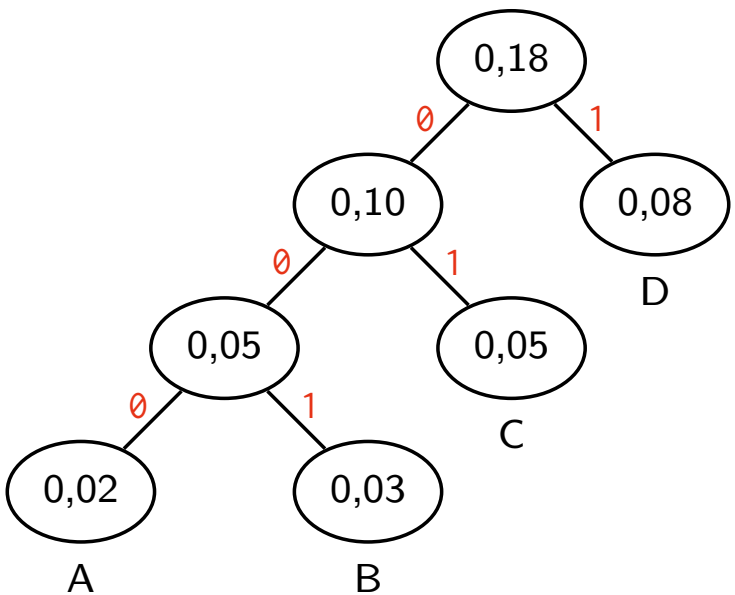


[1] Andernfalls würden die Aussagen zur Optimalität des Huffman-Codes im Allgemeinen nicht mehr zutreffen.

Konstruktion eines Huffman-Codes

- 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.[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

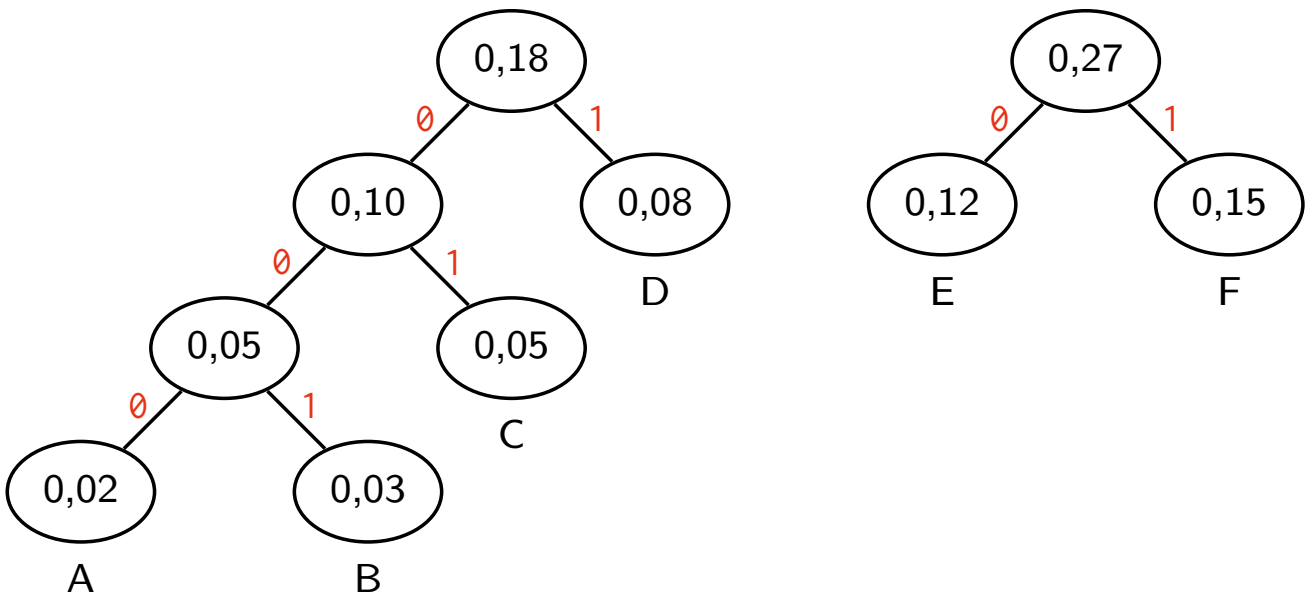


[1] Andernfalls würden die Aussagen zur Optimalität des Huffman-Codes im Allgemeinen nicht mehr zutreffen.

Konstruktion eines Huffman-Codes

- 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.[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

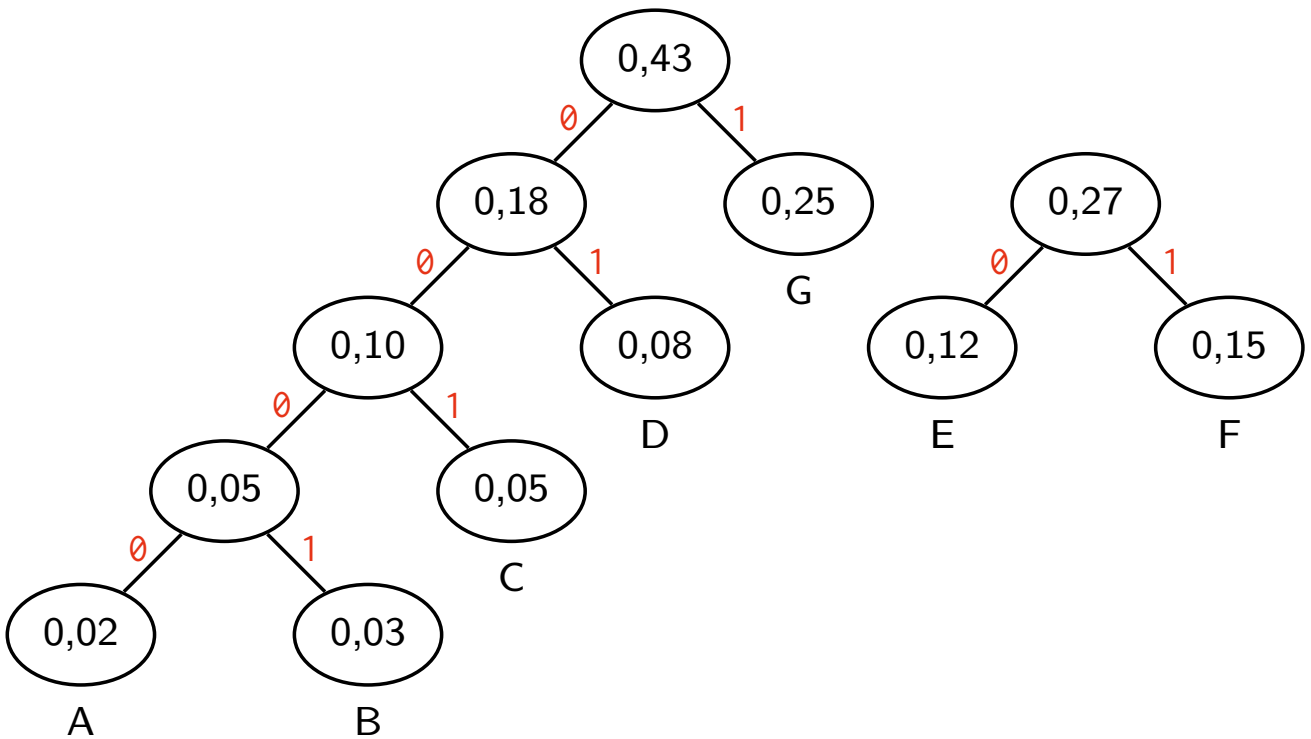


[1] Andernfalls würden die Aussagen zur Optimalität des Huffman-Codes im Allgemeinen nicht mehr zutreffen.

Konstruktion eines Huffman-Codes

- 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.[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

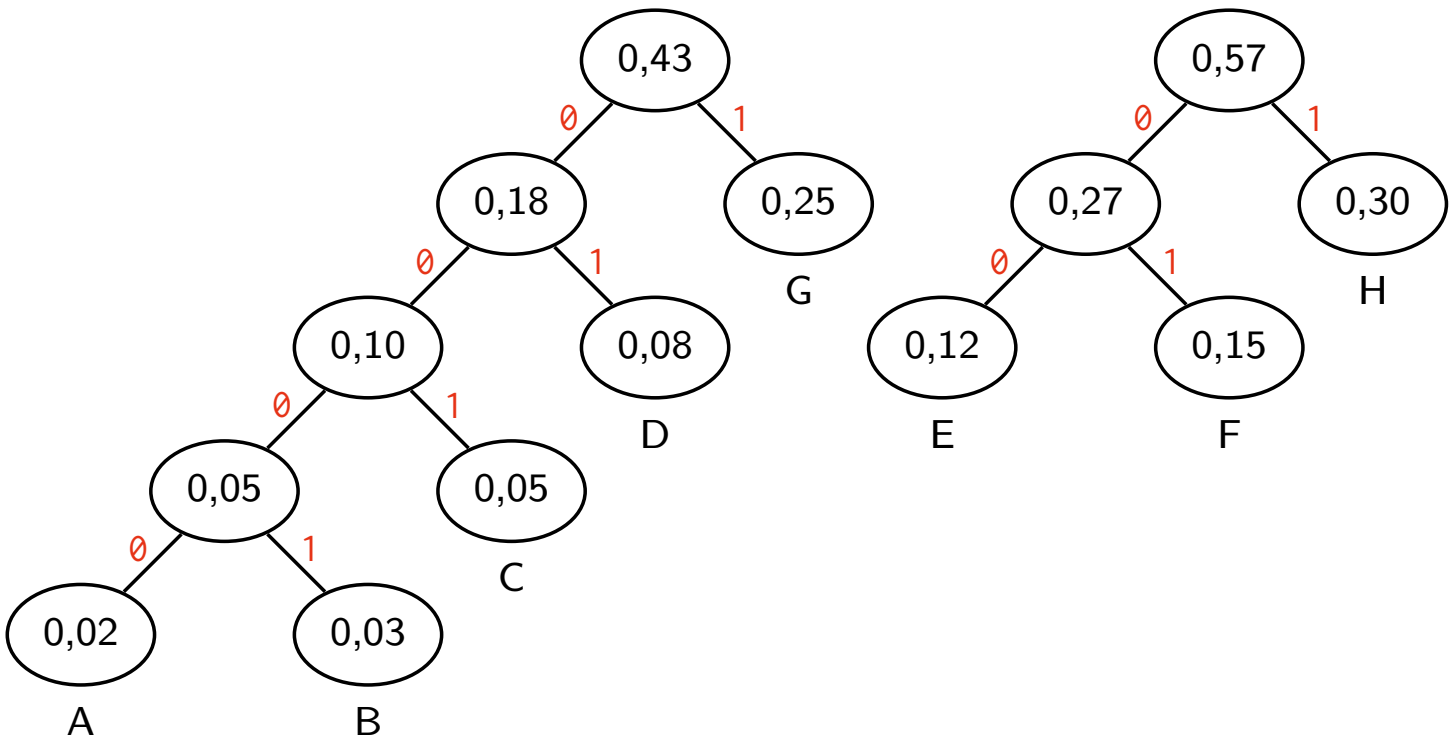


[1] Andernfalls würden die Aussagen zur Optimalität des Huffman-Codes im Allgemeinen nicht mehr zutreffen.

Konstruktion eines Huffman-Codes

- 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.[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

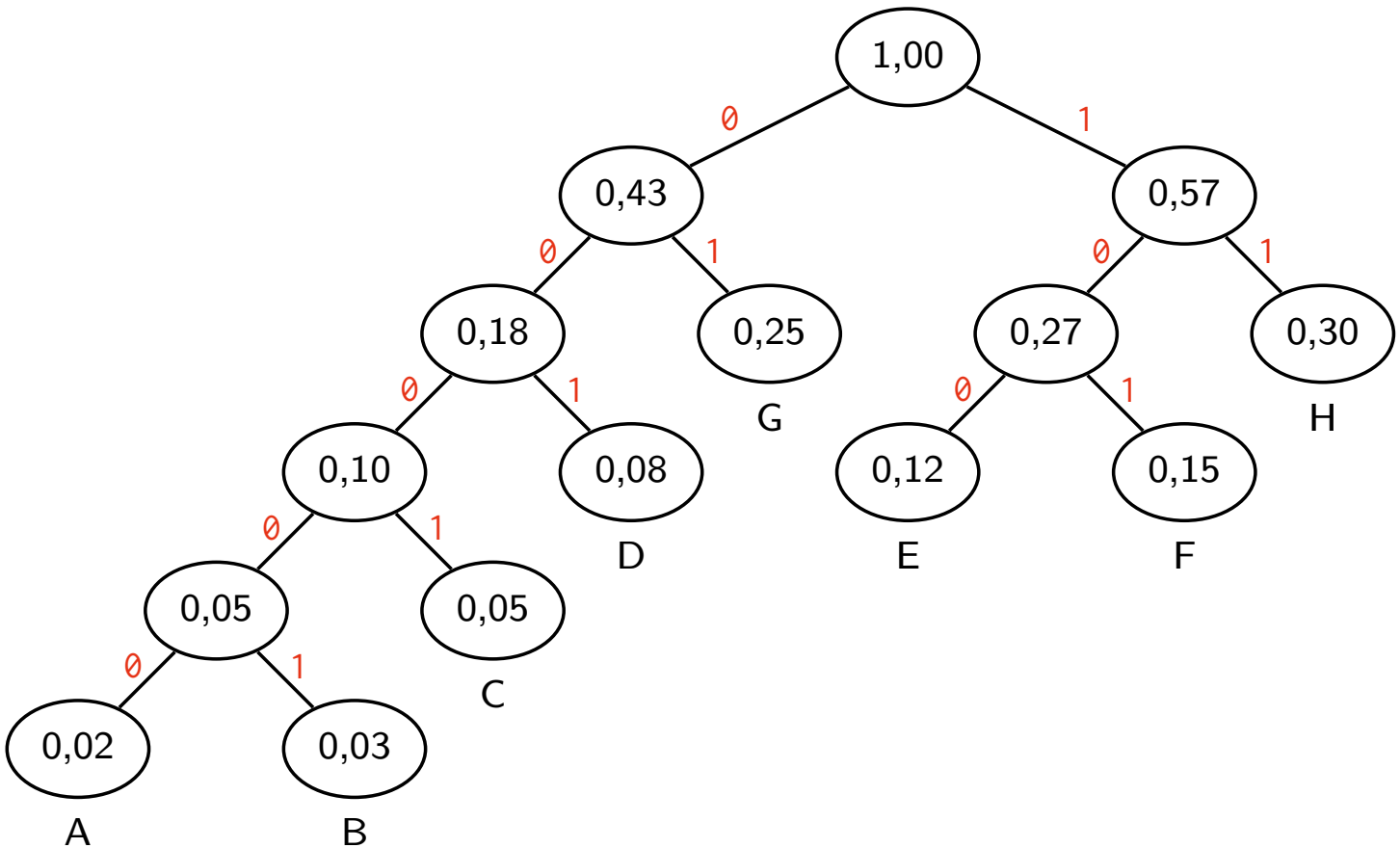


[1] Andernfalls würden die Aussagen zur Optimalität des Huffman-Codes im Allgemeinen nicht mehr zutreffen.

Konstruktion eines Huffman-Codes

- 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.[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

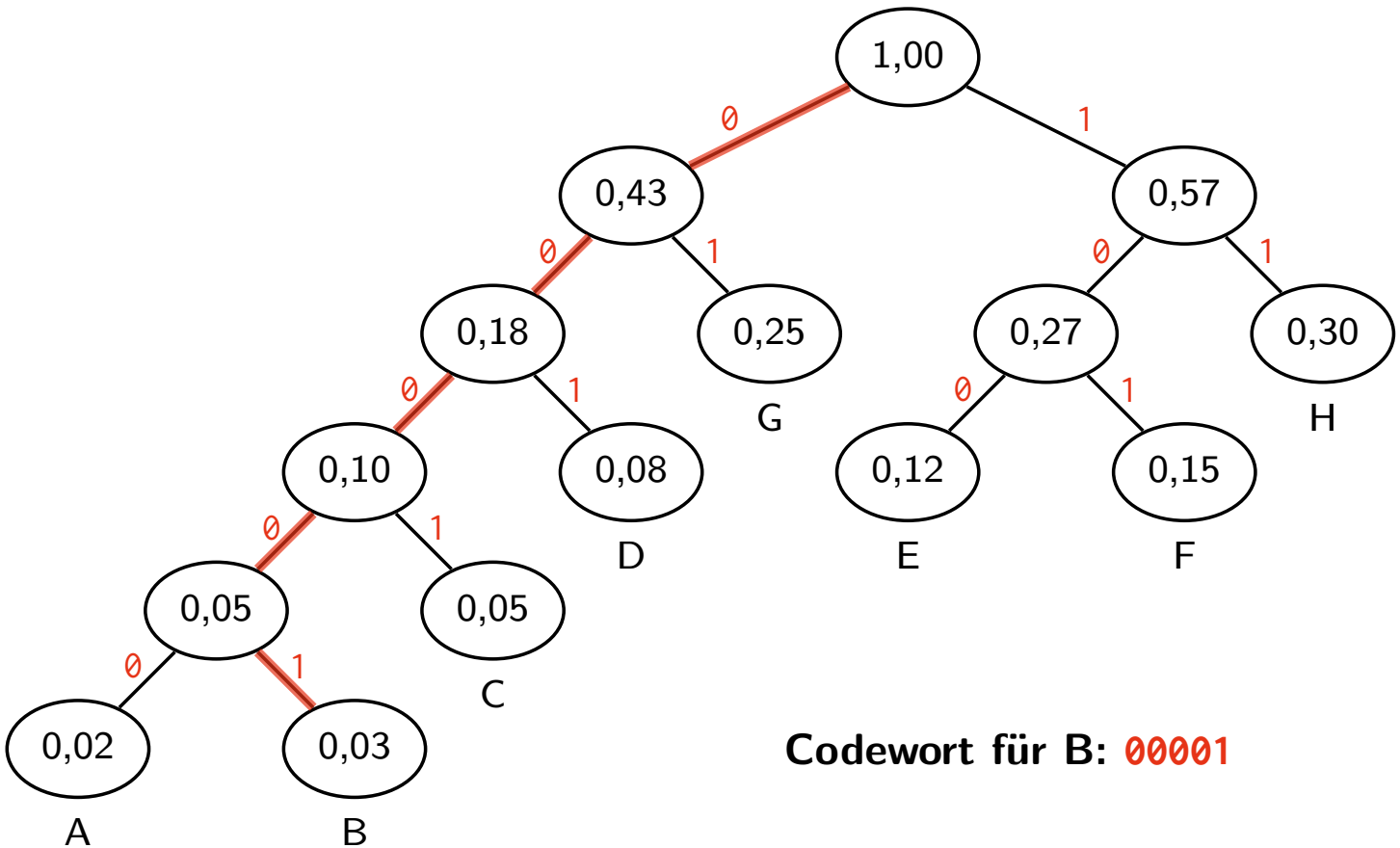


[1] Andernfalls würden die Aussagen zur Optimalität des Huffman-Codes im Allgemeinen nicht mehr zutreffen.

Konstruktion eines Huffman-Codes

- 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.[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



Codewort für B: 00001

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

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	010	3	100
F	0,15	011	3	101
G	0,25	10	2	110
H	0,30	11	2	111

- Uniformer Code:

$$\mathrm{E}[l(z)] = 3{,}0 \quad \text{, da alle Codewörter gleich lang sind}$$

- Huffman-Code:

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

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

Anmerkungen zum Huffman-Code:

- Statische Huffman-Codes sind darauf angewiesen, dass die Auftrittswahrscheinlichkeit der Zeichen den Erwartungen entspricht.
- Zeichenhäufigkeiten können dynamisch bestimmt werden, allerdings muss dem Empfänger dann das verwendete Codebuch mitgeteilt werden.
- Längere Codewörter (z. B. ganze Wörter statt einzelner Zeichen) werden infolge der Komplexität zum Bestimmen des Codebuchs ein Problem.
- 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 Auftrittswahrscheinlichkeit von $i \in \mathcal{A}$ und $c(i)$ die Abbildung auf ein entsprechendes Codewort bezeichnen.

Beispiel 2: Familie der Run-length Codes

Grundlegende Idee

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

Eigenschaften

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

Verwendung

- In den verschiedensten Bildkompressionsverfahren
- Fax
- Analog- und ISDN-Modems

Übersicht

Anwendungsschicht

Die Anwendungsschicht (Application Layer) ist die höchste Schnittstelle[1] zwischen Anwendungen und dem Netzwerk. Protokolle der Anwendungsschicht stellen spezifische Dienste bereit.

Beispiele:

- Domain Name System (DNS)
  Auflösung sog. vollqualifizerter Domänennamen[2] 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 Binärdaten 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.
- Telnet
  Einfaches Protokoll zur interaktiven Kommunikation mit anderen Hosts (vgl. TCP-Chat Client).
- Secure Shell (SSH)
  Verschlüsselte entfernte Anmeldung an einem Host.
- Simple Network Management Protocol (SNMP)
  Dient der überwachung und dem Management von Netzkomponenten.

[1] In vielen Fällen sind Protokolle der Anwendungsschicht Bestandteil der Anwendungen selbst, also innerhalb einer Anwendung implementiert und nicht notwendigerweise Bestandteil des Betriebssystems. Innerhalb gewisser Grenzen trifft dies bereits auf die Schichten 5 und 6 zu.
[2] Umgangssprachlich als „Webadressen“ bezeichnet.

Domain Name System (DNS) [2, 4, 6, 7]

Motivation:

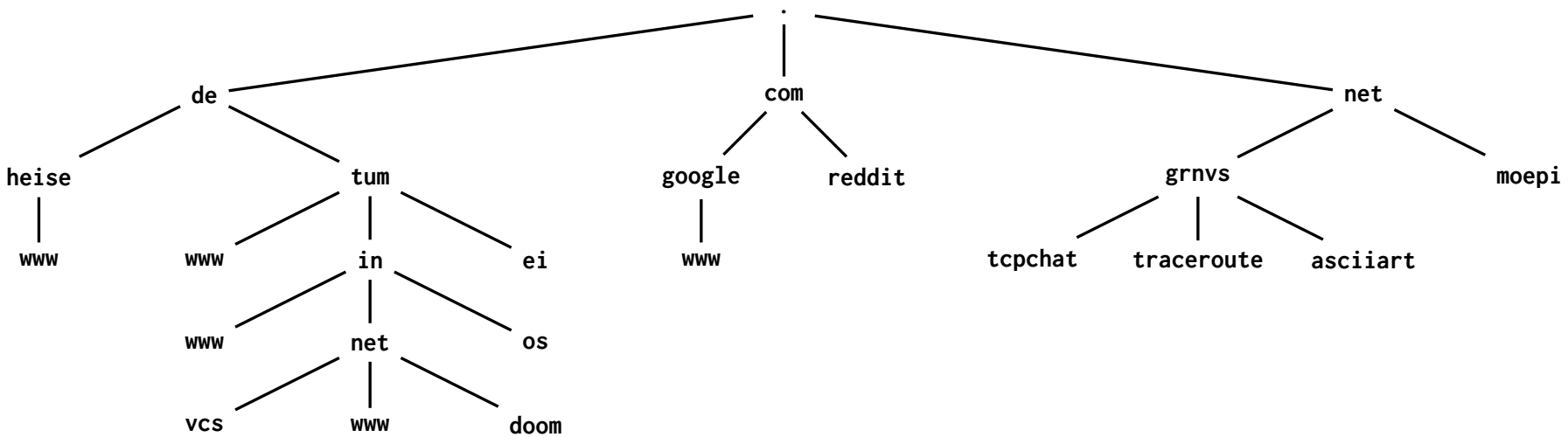
- Möchte ein Nutzer (Mensch) einen Computer adressieren, z. B. beim Aufruf einer Webseite, will er sich gewöhnlich nicht dessen IP-Adresse merken müssen.
- Stattdessen adressiert man das Ziel überlicherweise mittels eines hierarchisch aufgebauten Namens, z. B. www.google.com.

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

1. Der Domain Namespace ist
   - ein hierarchisch aufgebauter Namensraum mit
   - baumartiger Struktur.
2. Nameservers speichern
   - Informationen über den Namensraum,
   - wobei jeder Server nur kleine Ausschnitte des Namensraums kennt.
3. Resolver sind Programme,
   - die durch Anfragen an Nameserver Informationen aus dem Namespace extrahieren und
   - anfragenden Clients bzw. Anwendungen zur Verfügung stellen.

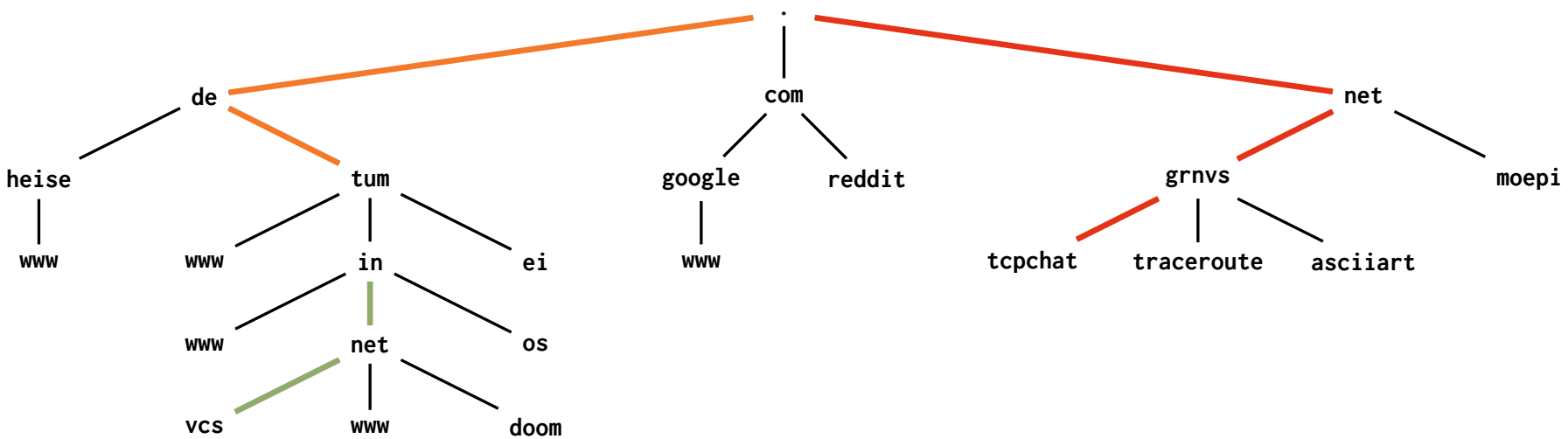
Domain Namespace

Ein kleiner Auszug aus dem Namespace:



Domain Namespace

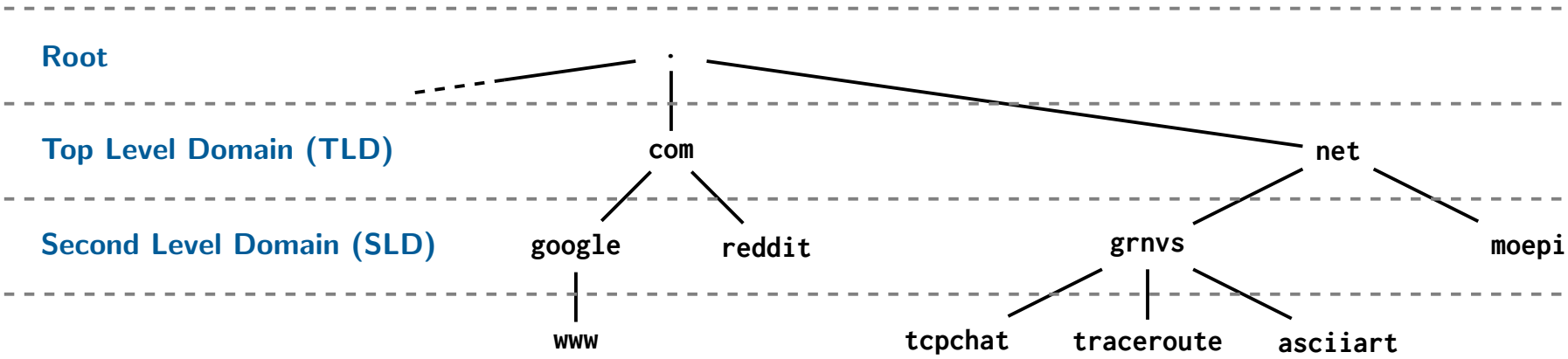
Ein kleiner Auszug aus dem Namespace:



- 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. `vcs.net.in`.
  - Ein FQDN kann als Suffix für einen nicht-qualifizierten Namen verwendet werden, z. B. ergibt `vcs.net.in` zusammen mit dem FQDN `tum.de.` einen neuen FQDN `vcs.net.in.tum.de.`.
  - Ob ein FQDN existiert (z. B. in eine Adresse aufgelöst werden kann), bleibt zunächst offen.

Übliche Bezeichnungen

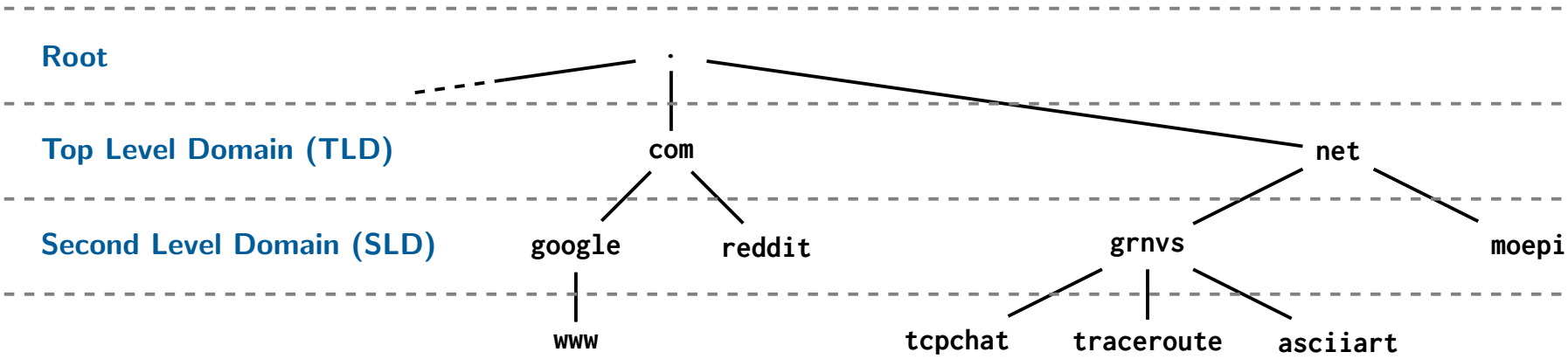
Für die ersten drei Hierarchieebenen im Name Space eigene Bezeichnungen üblich:



Darunter liegende Ebenen werden gelegentlich als Subdomain bezeichnet.

Übliche Bezeichnungen

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



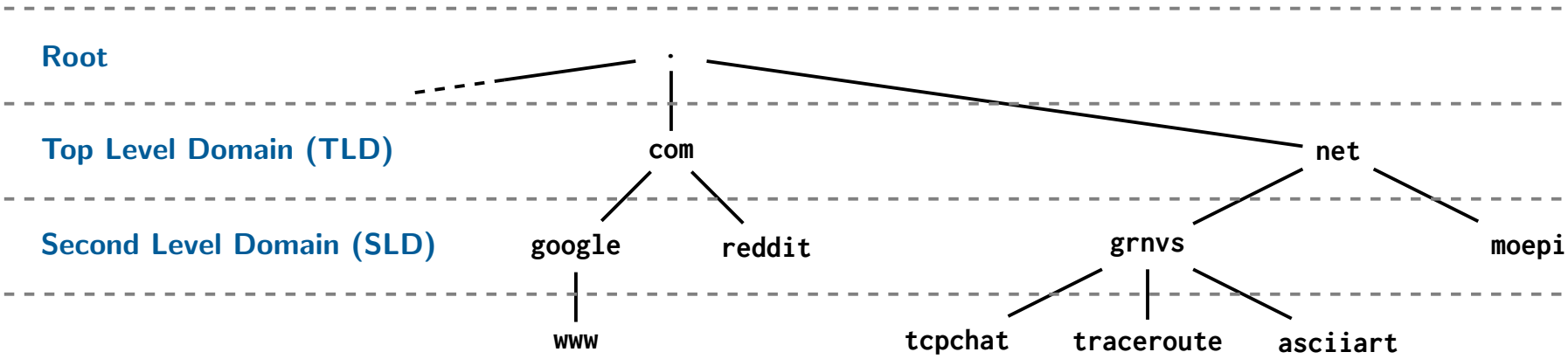
Darunter liegende Ebenen werden gelegentlich als Subdomain bezeichnet.

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.

Übliche Bezeichnungen

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



Darunter liegende Ebenen werden gelegentlich als Subdomain 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.

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.

Nameserver

Der Namespace wird

- in Form einer verteilten Datenbank
- von ein 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 UDP/TCP 53:

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

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). Wird für unterschiedliche Zwecke verwendet und missbraucht.
- 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 wieteren Resource Records für den betreffenden FQDN mehr geben darf.
- Für eine Zone bzw. Domain gibt es üblicherweise mehrere NS bzw. MX Records.

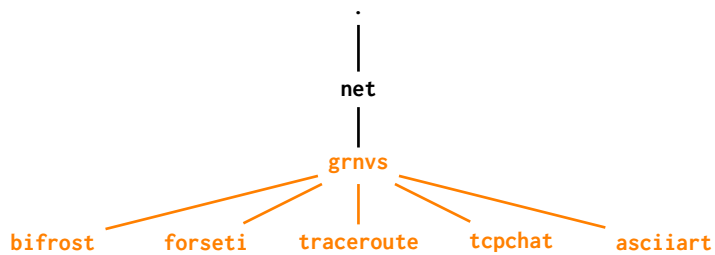
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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  svm502.net.in.tum.de.
```

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



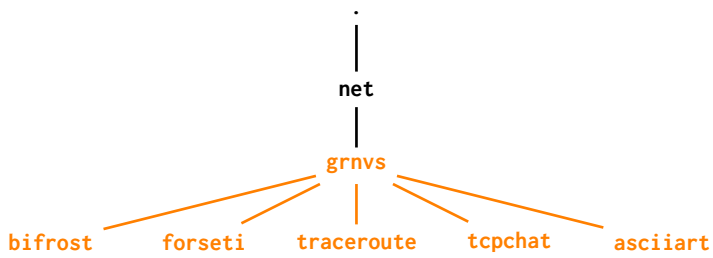
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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  svm502.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zur 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.

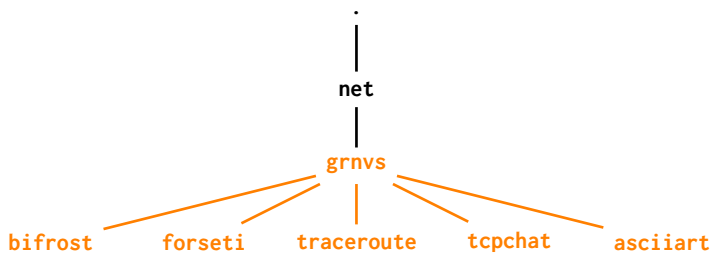
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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  svm502.net.in.tum.de.
```

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



- SOA Record für die Domain `grnvs.net.`[2]

[2] In der Zone File ist `grnvs.net` als realativer Domain Name angegeben (also kein FQDN), da zuvor der Ausgangspunkt zuvor via `$ORIGIN` festgelegt wurde.

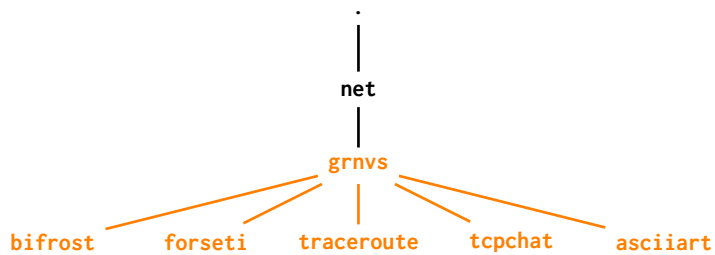
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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  svm502.net.in.tum.de.
```

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



- SOA Record für die Domain `grnvs.net.`[2]
  - `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.

[2] In der Zone File ist `grnvs.net` als realativer Domain Name angegeben (also kein FQDN), da zuvor der Ausgangspunkt zuvor via `$ORIGIN` festgelegt wurde.

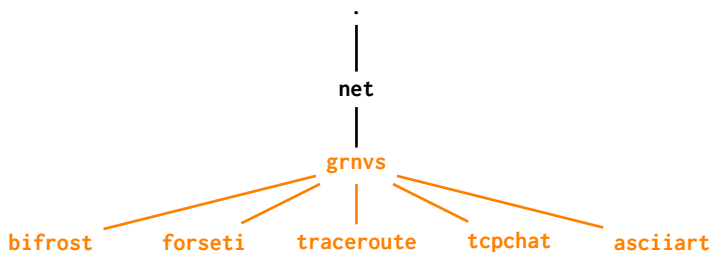
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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  svm502.net.in.tum.de.
```

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



- SOA Record für die Domain `grnvs.net.`[2]
  - `refresh` gibt das Zeitintervall an, in dem Secondaries versuchen, ihre lokale Kopie der Zonendatei mit dem Primary abzugleichen.

[2] In der Zone File ist `grnvs.net` als realativer Domain Name angegeben (also kein FQDN), da zuvor der Ausgangspunkt zuvor via `$ORIGIN` festgelegt wurde.

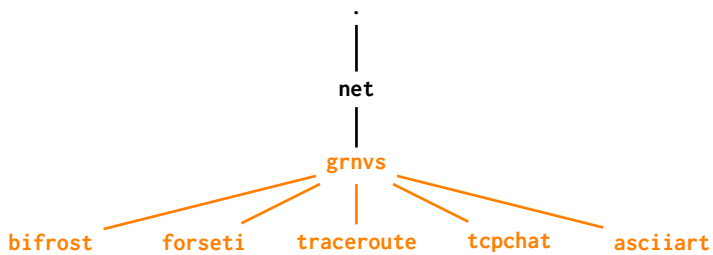
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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      129.187.145.241

$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   svm502.net.in.tum.de.
```

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



- SOA Record für die Domain `grnvs.net.`[2]
  - `refresh` gibt das Zeitintervall an, in dem Secondaries versuchen, ihren Primary zu kontaktieren, falls der vorherige Versuch fehlgeschlagen ist.

[2] In der Zone File ist `grnvs.net` als realativer Domain Name angegeben (also kein FQDN), da zuvor der Ausgangspunkt zuvor via `$ORIGIN` festgelegt wurde.

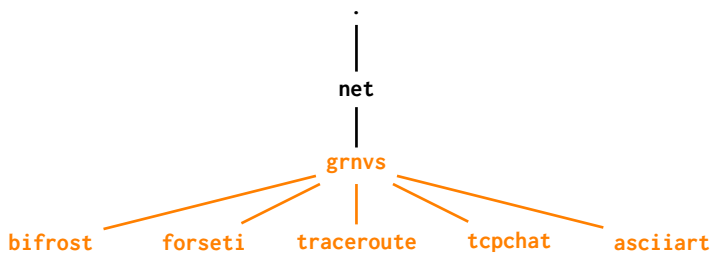
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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      129.187.145.241

$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   svm502.net.in.tum.de.
```

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



- SOA Record für die Domain `grnvs.net.`[2]
  - `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.

[2] In der Zone File ist `grnvs.net` als realativer Domain Name angegeben (also kein FQDN), da zuvor der Ausgangspunkt zuvor via `$ORIGIN` festgelegt wurde.

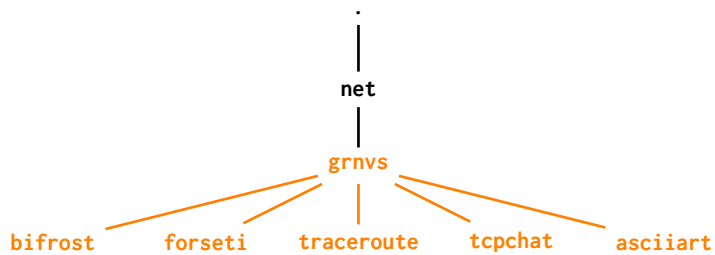
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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  svm502.net.in.tum.de.
```

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



- SOA Record für die Domain `grnvs.net.`[2]
  - `nxdomain`[3] gibt an, wie lange anfragende Resolver in ihrem Cache halten dürfen, dass ein angefragter Resource Record nicht existiert.

[2] In der Zone File ist `grnvs.net` als realativer Domain Name angegeben (also kein FQDN), da zuvor der Ausgangspunkt zuvor via `$ORIGIN` festgelegt wurde.

[3] Dieses Feld hatte früher eine andere Bedeutung.

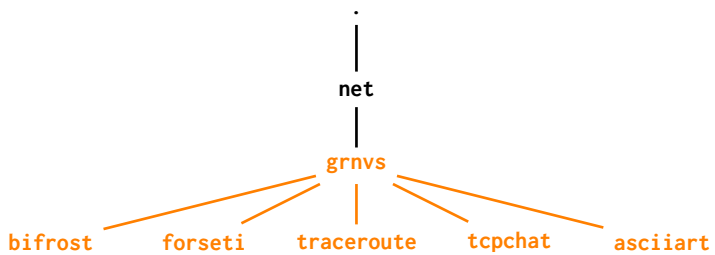
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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  svm502.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zur 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.

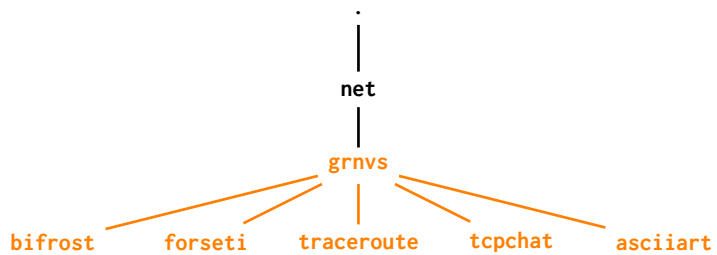
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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  svm502.net.in.tum.de.
```

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



- A Record für den FQDN **grnvs.net.**
- Was geschieht, wenn dieser fehlt, sieht man beim Versuch die beiden FQDNs **in.tum.de.** und **ei.tum.de.** im Browser aufzurufen.

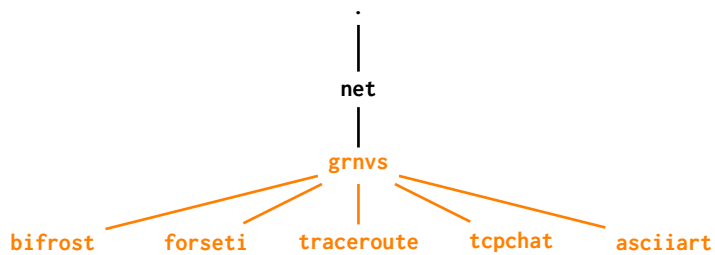
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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  svm502.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zur 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.

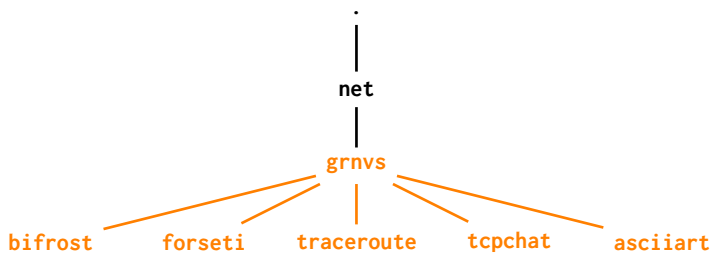
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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 svm502.net.in.tum.de.
```

Relevanter Teil des Namespaces (der zur 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.

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

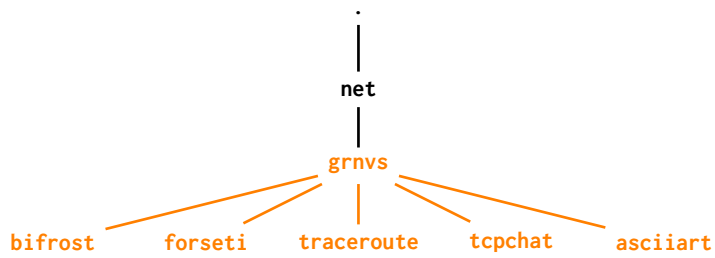
```
$TTL 86400 ; 1 day
grnvs.net. IN  SOA   bifrost.grnvs.net. hostmaster.grnvs.net. (
                         164160 ; serial
                         1800   ; refresh (30 minutes)
                         300    ; retry (5 minutes)
                         604800 ; expire (1 week)
                         1800   ; nxdomain (30 minutes)
                     )
               NS    bifrost.grnvs.net.
               NS    forseti.grnvs.net.
               A     129.187.145.241

$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  svm502.net.in.tum.de.
```

- AAAA Records für zwei Hosts (IPv6).

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



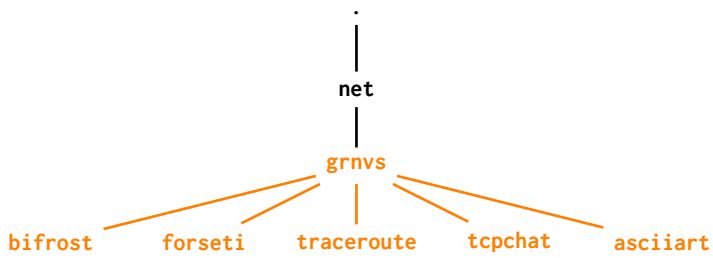
Die Resource Records einer Zone werden auf Nameservern in Form von Zone Files gespeichert:

```
$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     129.187.145.241

$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  svm502.net.in.tum.de.
```

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



- CNAME Record, der auf eine unserer virtuellen Maschinen verweist.
- Das Ziel `svm502.net.in.tum.de.` liegt außerhalb dieser Zone.

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. selbst für keine Zonen selbst autoritativ.

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

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. selbst für keine Zonen selbst autoritativ.

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

- Resolver verfügen über eine statische Liste der 13 Root-Server[1], 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.

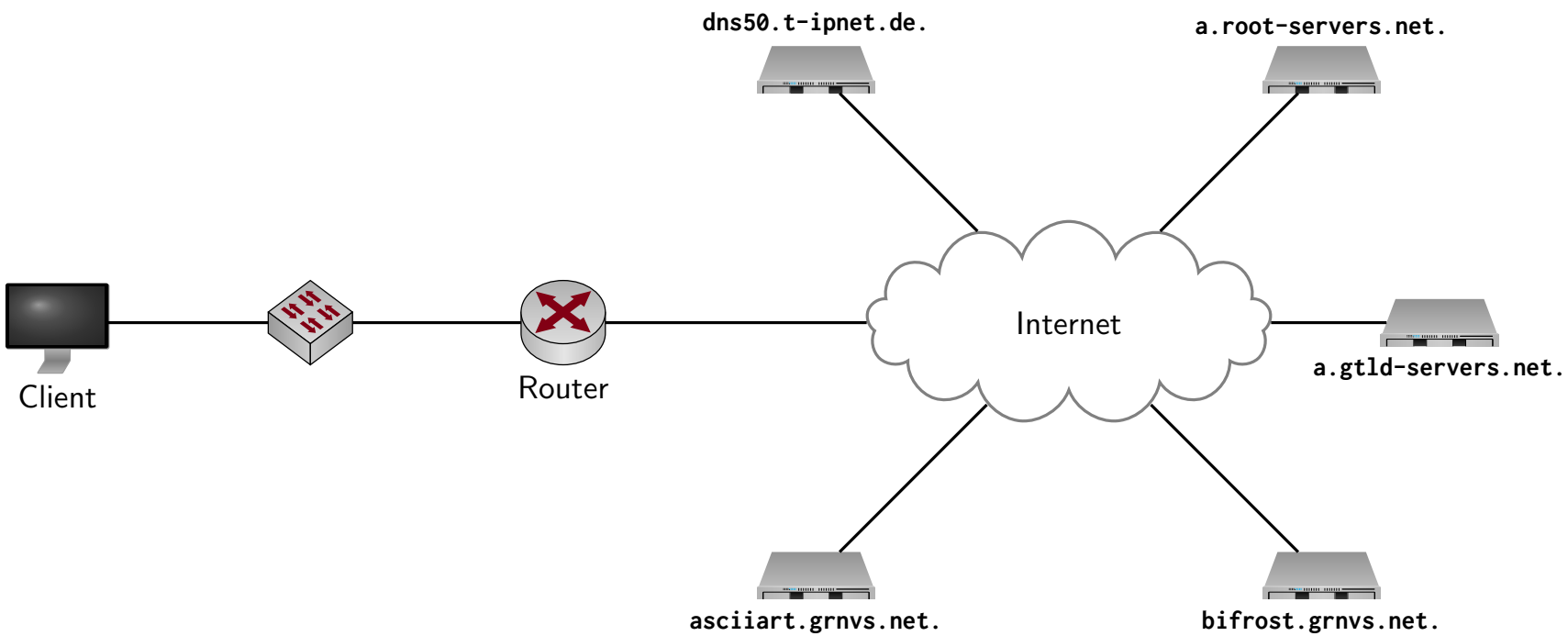
[1] In Wirklichkeit handelt es sich dabei um hunderte Server, welche über die 13 IP-Adressen via Anycast erreichbar sind.

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:    May 23, 2015
;       related version of root zone:   2015052300
;
; 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
;
.
.
.
```

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

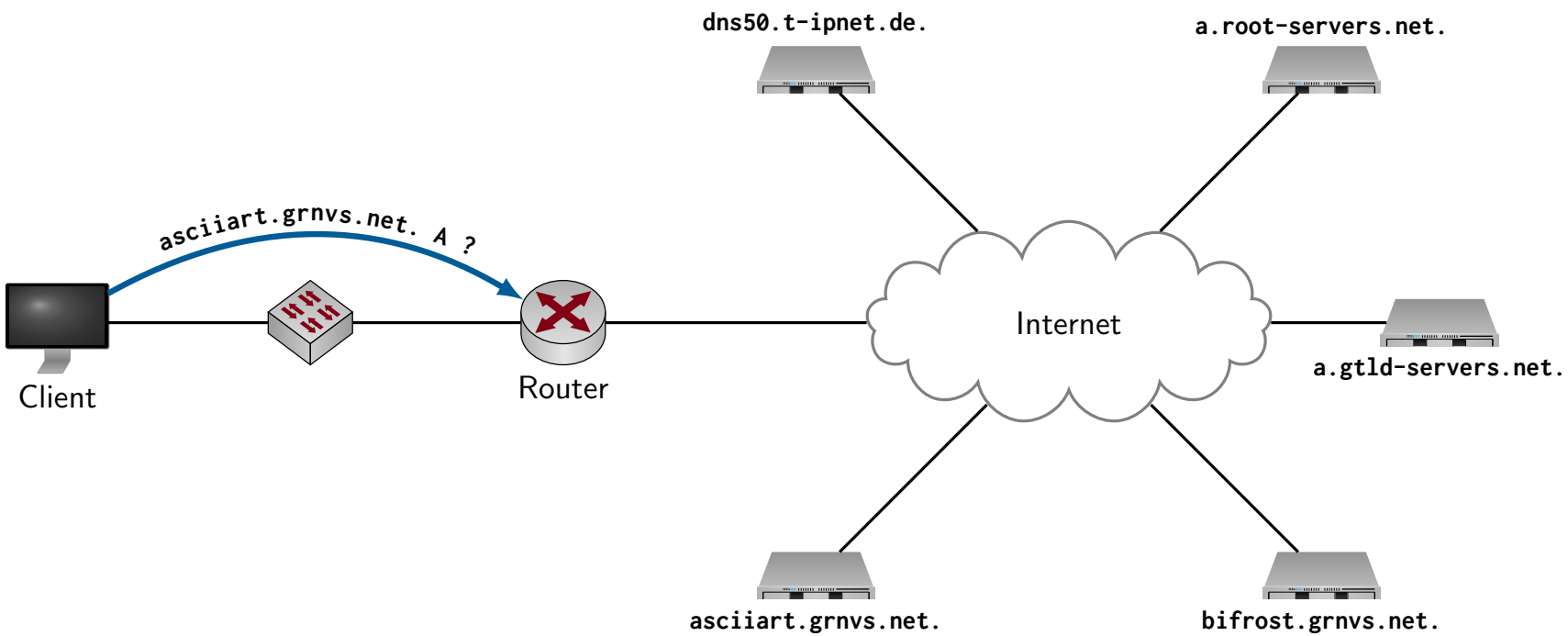
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.[1]
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind rekursiv (recursive 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.

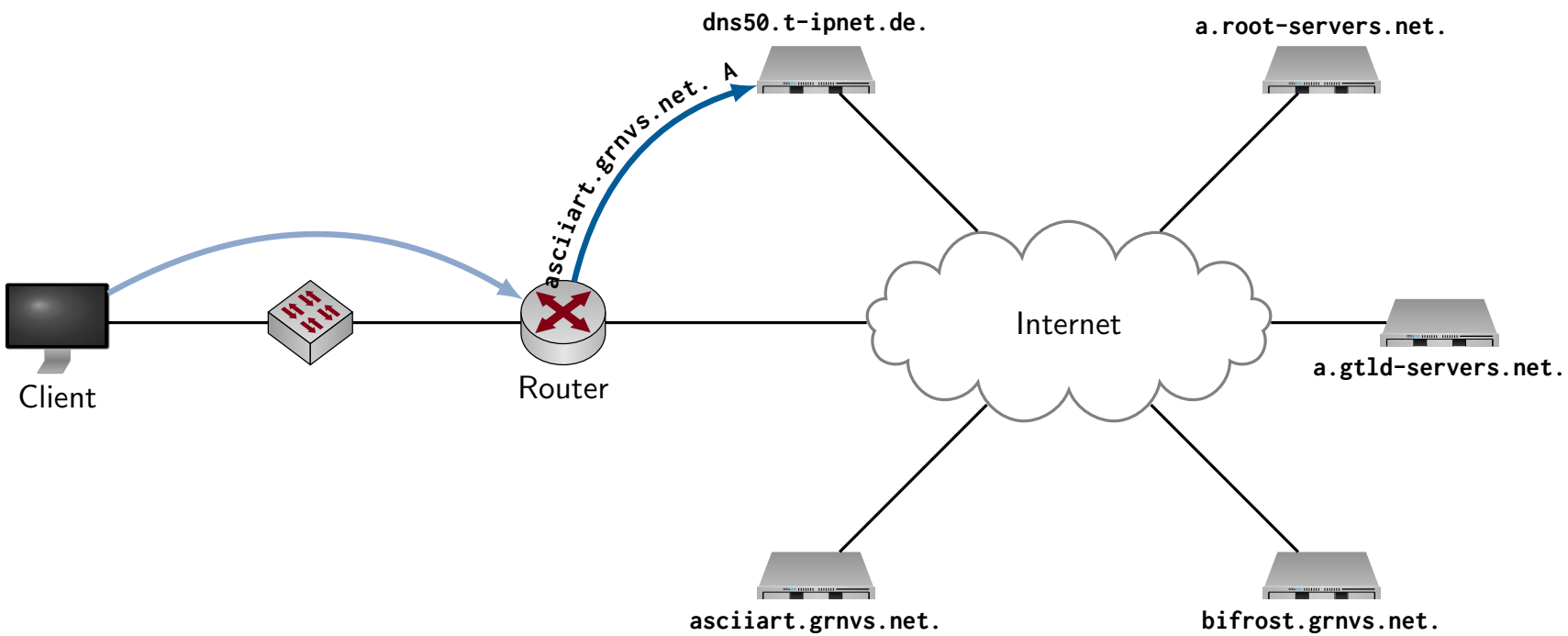
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.[1]
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind rekursiv (recursive 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.

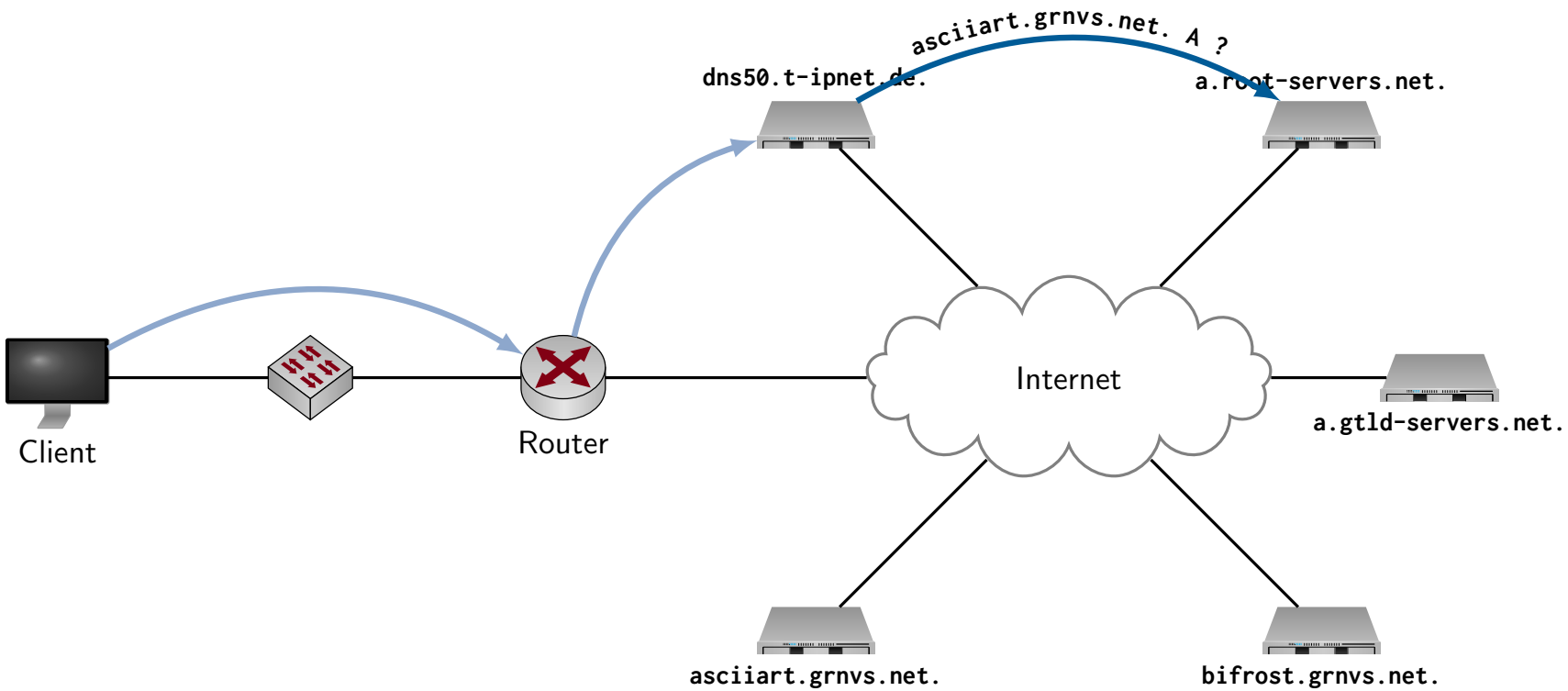
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.[1]
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind rekursiv (recursive 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.

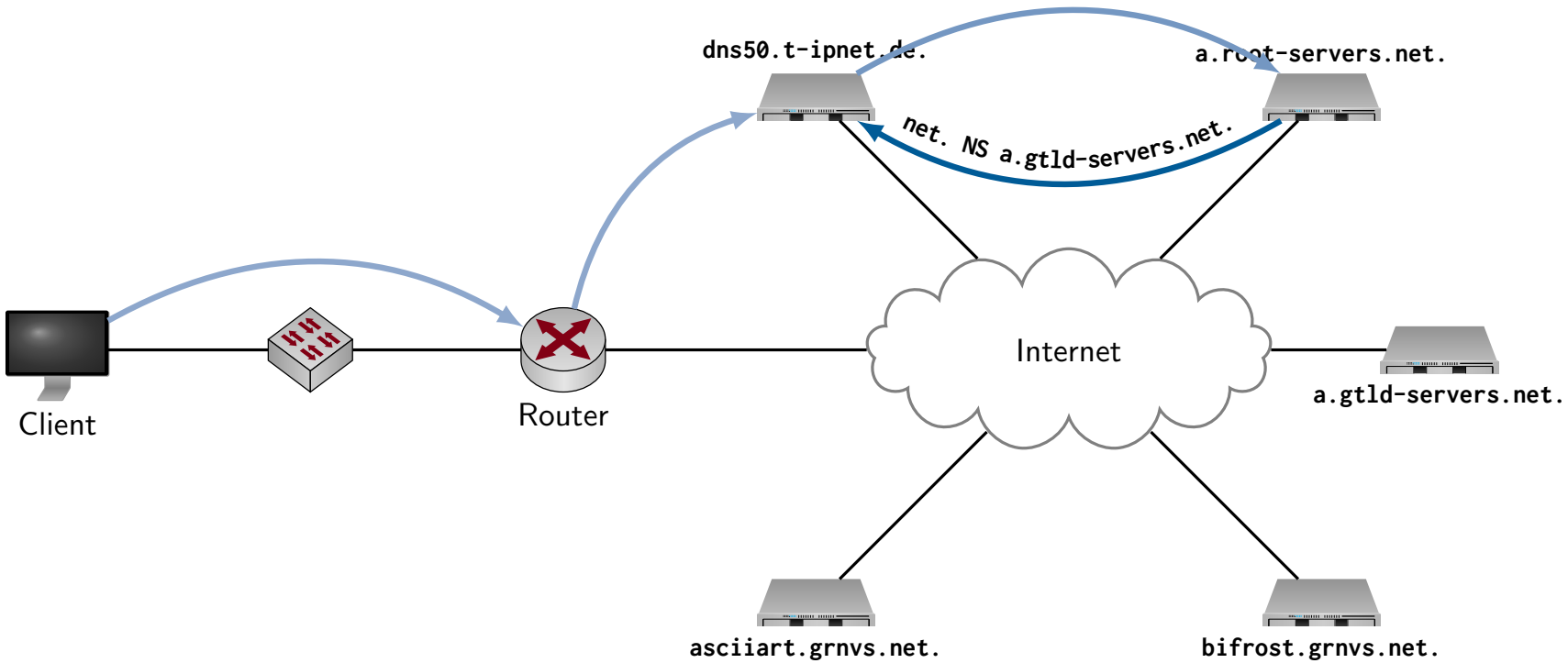
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.[1]
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind rekursiv (recursive 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.

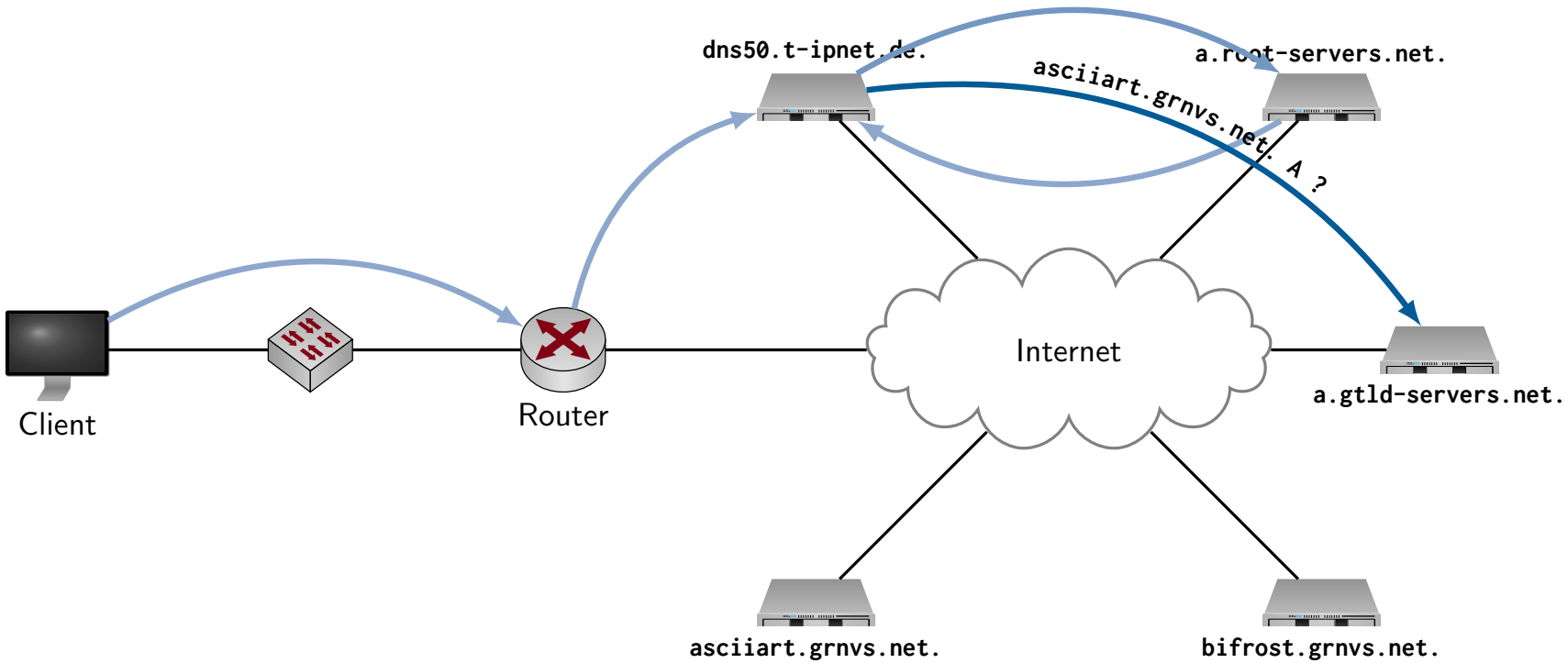
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.[1]
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind rekursiv (recursive 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.

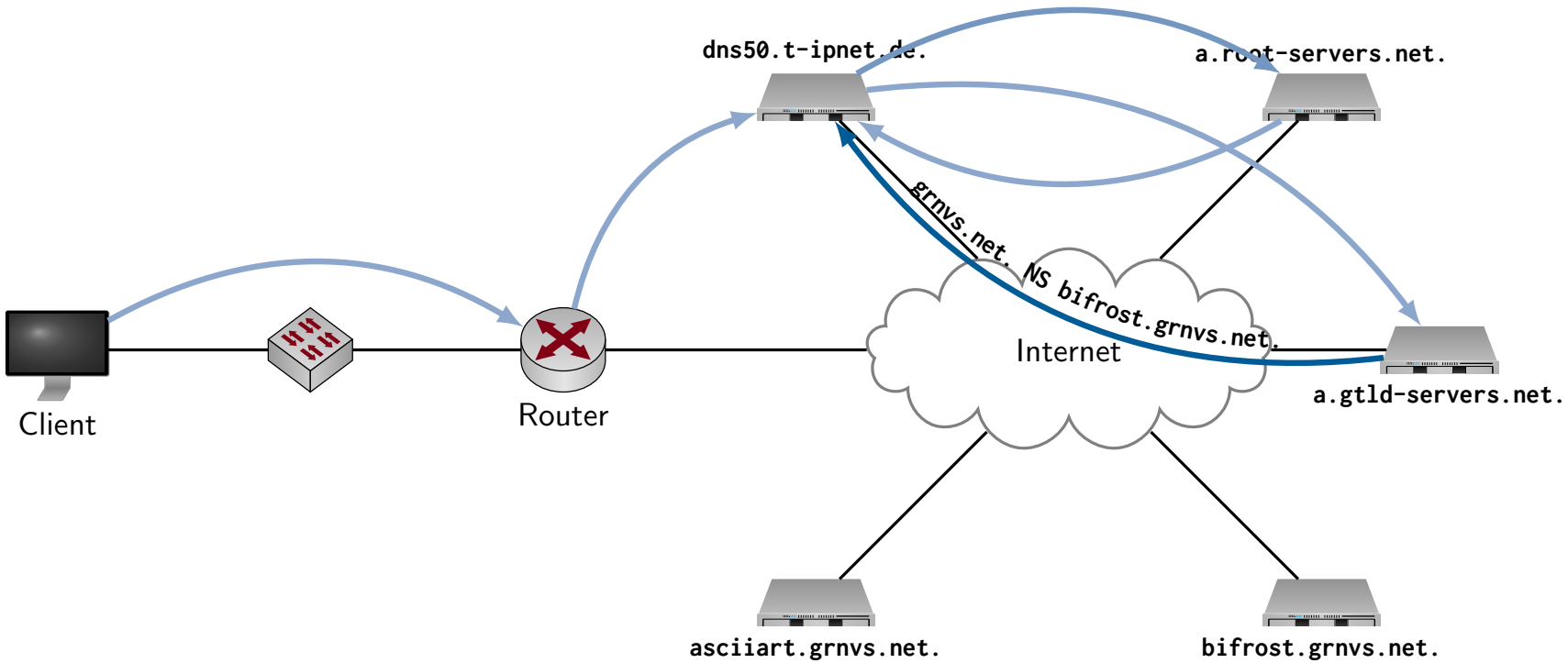
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.[1]
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind rekursiv (recursive 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.

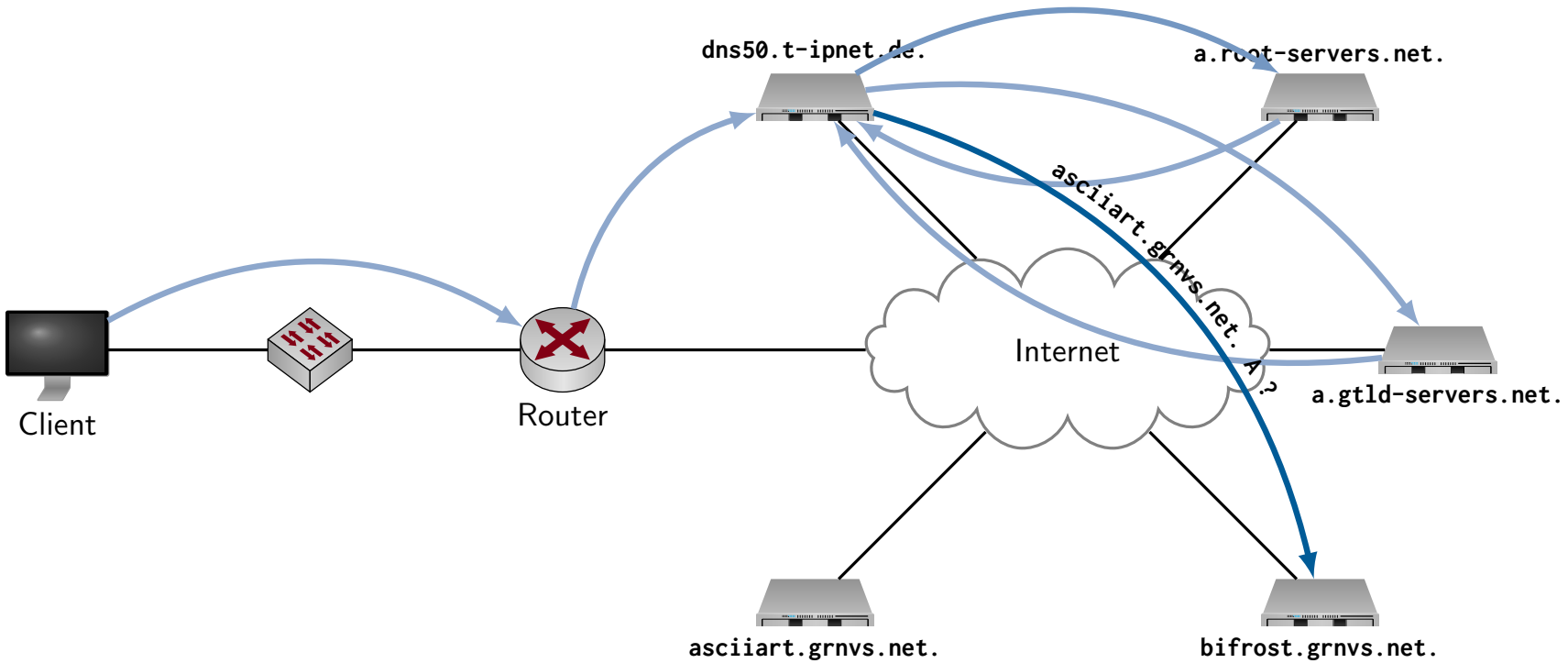
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.[1]
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind rekursiv (recursive 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.

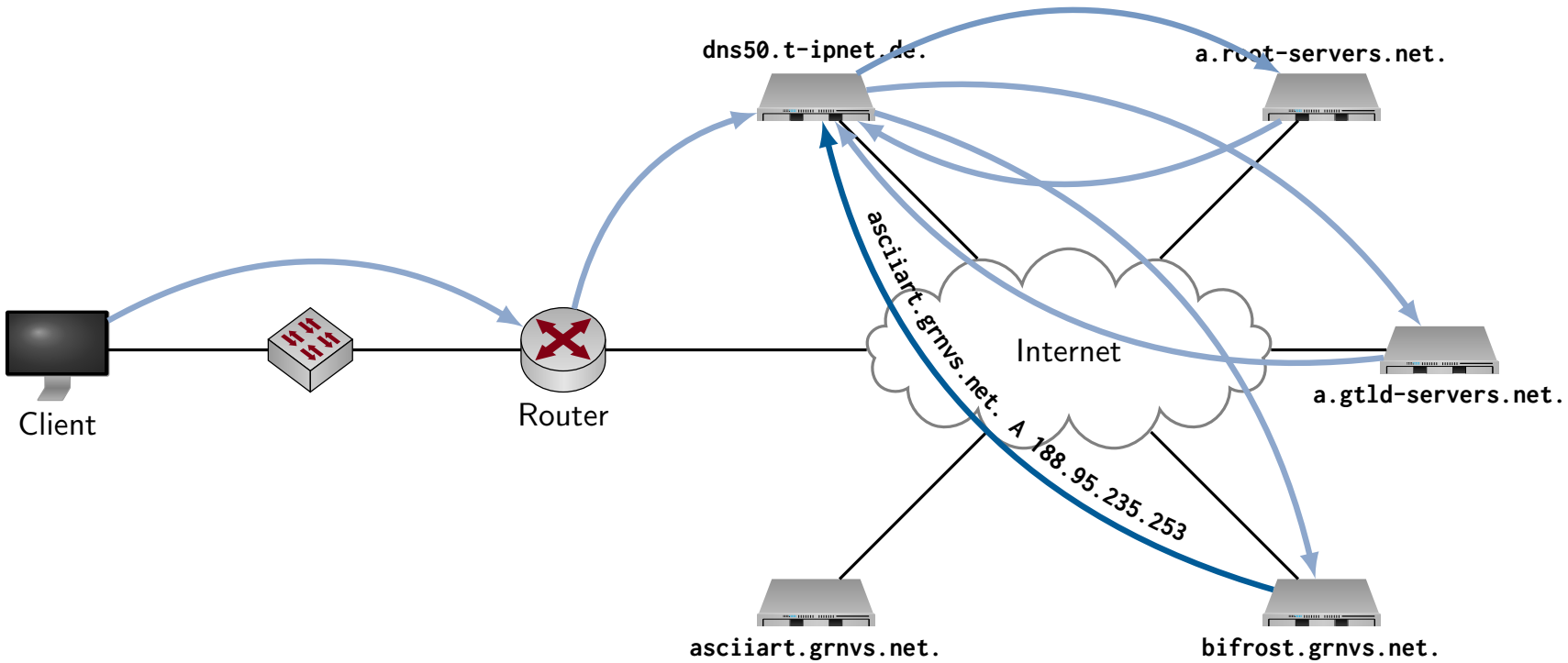
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.[1]
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind rekursiv (recursive 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.

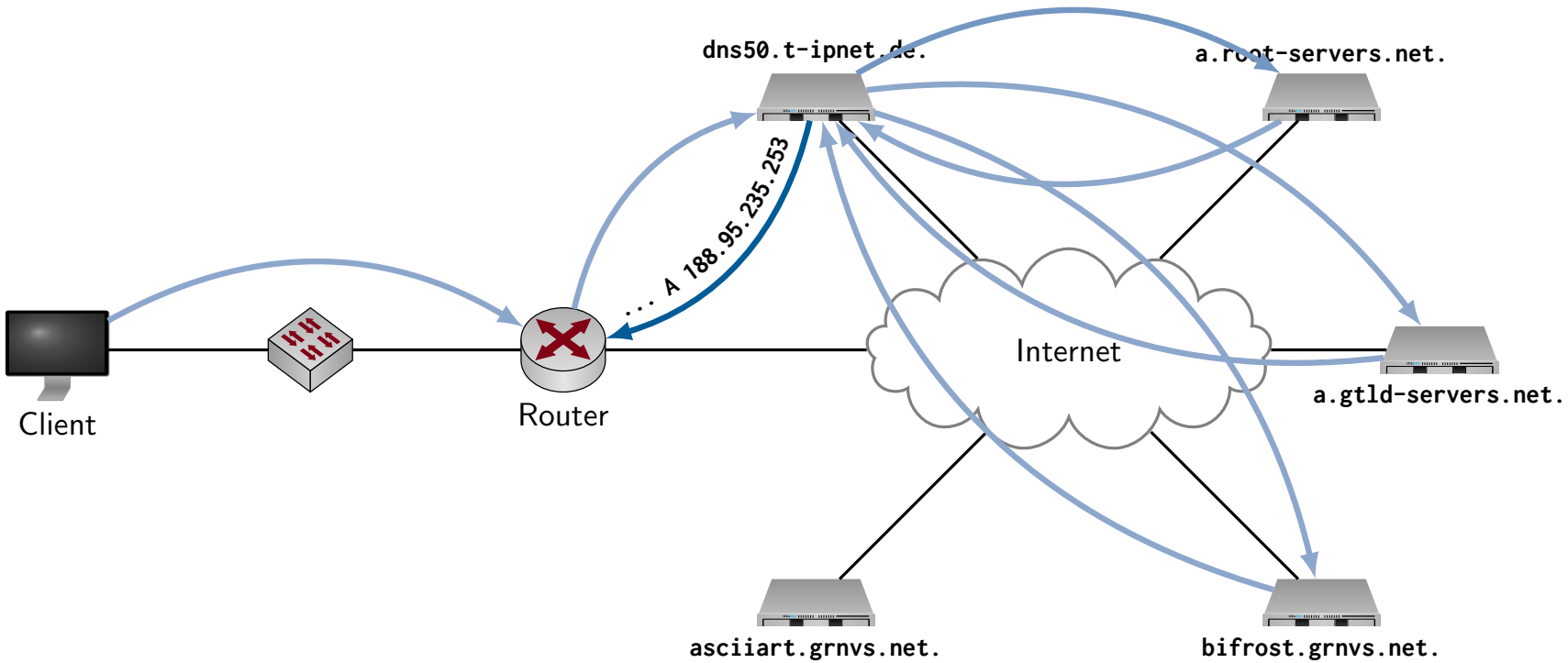
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.[1]
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind rekursiv (recursive 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.

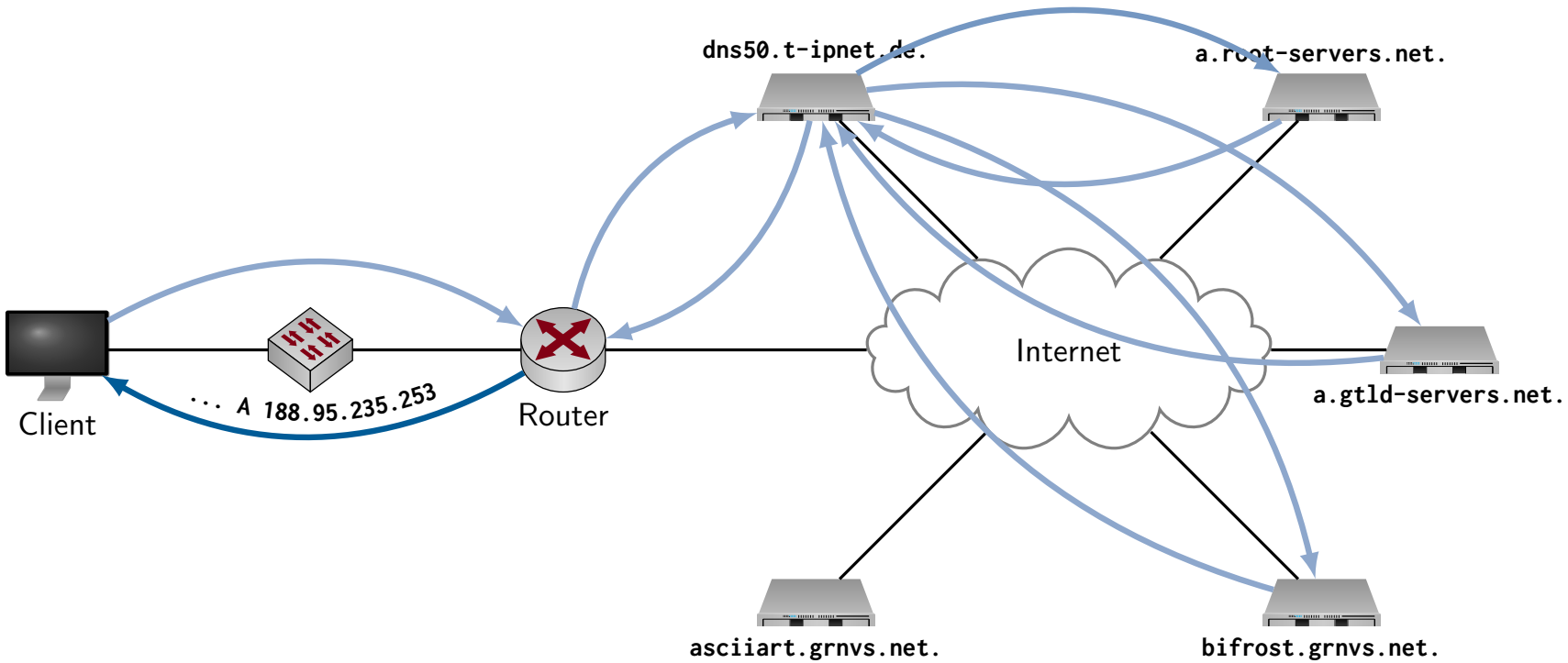
- Der Router arbeitet zwar als Resolver, leitet sämtliche Anfragen aber an einen Resolver des Providers weiter. Dessen IP-Adresse sei dem Router bekannt.[1]
- Der Client verwendet den Router als Resolver. Dessen IP-Adresse ist bekannt.
- DNS-Anfragen des Clients an den Router sind rekursiv (recursive 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 kein A Record mit der gesuchten IP-Adresse, sondern lediglich einen CNAME Record erhalten.
   - Das liegt daran, dass laut der Zone File auf Folie 38 kein A Record sondern nur ein CNAME Record für **`asciiart.grnvs.net.`** existiert (das ist nur dem Beispiel geschuldet).
   - In Wirklichkeit hätte der Resolver im Anschluss mittels weiterer Anfragen einen autoritativen Nameserver für **`svm502.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.

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:

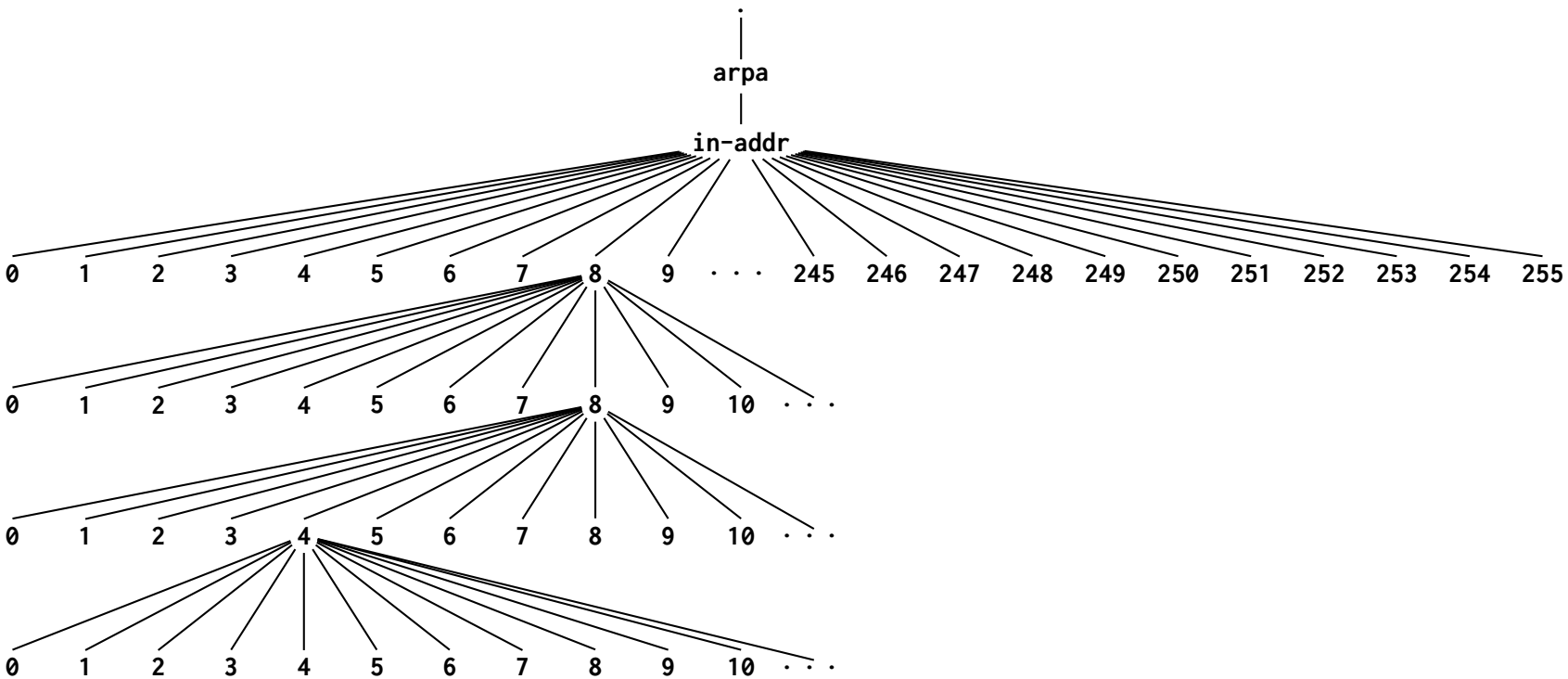
- in-addr.arpa. für IPv4
- ip6.arpa. für IPv6

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:

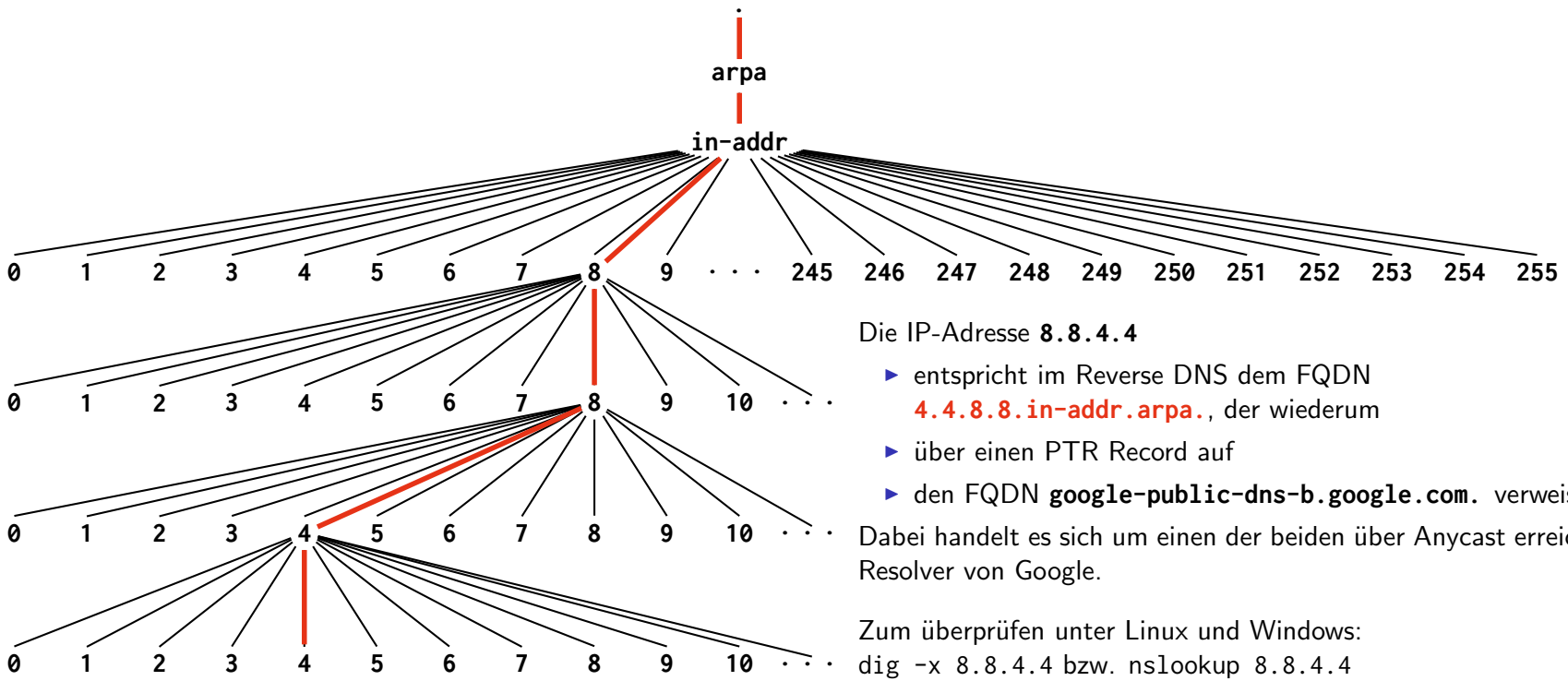


Reverse DNS

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:



Die IP-Adresse **8.8.4.4**

- entspricht im Reverse DNS dem FQDN `4.4.8.8.in-addr.arpa.`, der wiederum
- über einen PTR Record auf
- den FQDN **google-public-dns-b.google.com.** verweist.

Dabei handelt es sich um einen der beiden über Anycast erreichbaren Resolver von Google.

Zum überprüfen unter Linux und Windows:
`dig -x 8.8.4.4` bzw. `nslookup 8.8.4.4`

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 (letztere 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 (letztere 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 die bestimmten Resource adressieren, z. B. eine bestimmte Datei.

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

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

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 die bestimmten Resource adressieren, z. B. eine bestimmte Datei.

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 Name[2], 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>`. Mehrere 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 genausogut via Rundfunk bekanntgeben könnten...

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

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 die bestimmten Resource adressieren, z. B. eine bestimmte Datei.

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 Name[2], 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>. Mehrere 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/ne23puw/exam/IN0010-20150612-0000.pdf`
- `https://www.mymail.alsoinsecure/mybox?user=student&password=nolongeryourscret`

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

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

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 gefragt 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 `http://in.tum.de` und `http://ei.tum.de`.
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 gefragt 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 `http://in.tum.de` und `http://ei.tum.de`.
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 nichts[1] 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 voraussgesetzt, da man andernfalls unter den gängigen Betriebssystemen keinen listening Socket $<$ 1024 anlegen kann.

HyperText Transfer Protocol (HTTP)

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)?

HyperText Transfer Protocol (HTTP)

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[1] angesehen werden, da die einzelnen Anfragen voneinander unabhängig sind.

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 Webseiten 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 entscheiden, welche Daten ausgeliefert werden sollen
[2] In HTTP in falscher Schreibweise als Referer[sic]

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 Webseiten über den Referrer [2])
     - Den User-Agent, also die verwendete Client-Software
2. Response (vom Server zum Client), enthält
   - eine Status-Line Status (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 entscheiden, welche Daten ausgeliefert werden sollen
[2] In HTTP in falscher Schreibweise als Referer[sic]

HTTP Methods und Response Codes

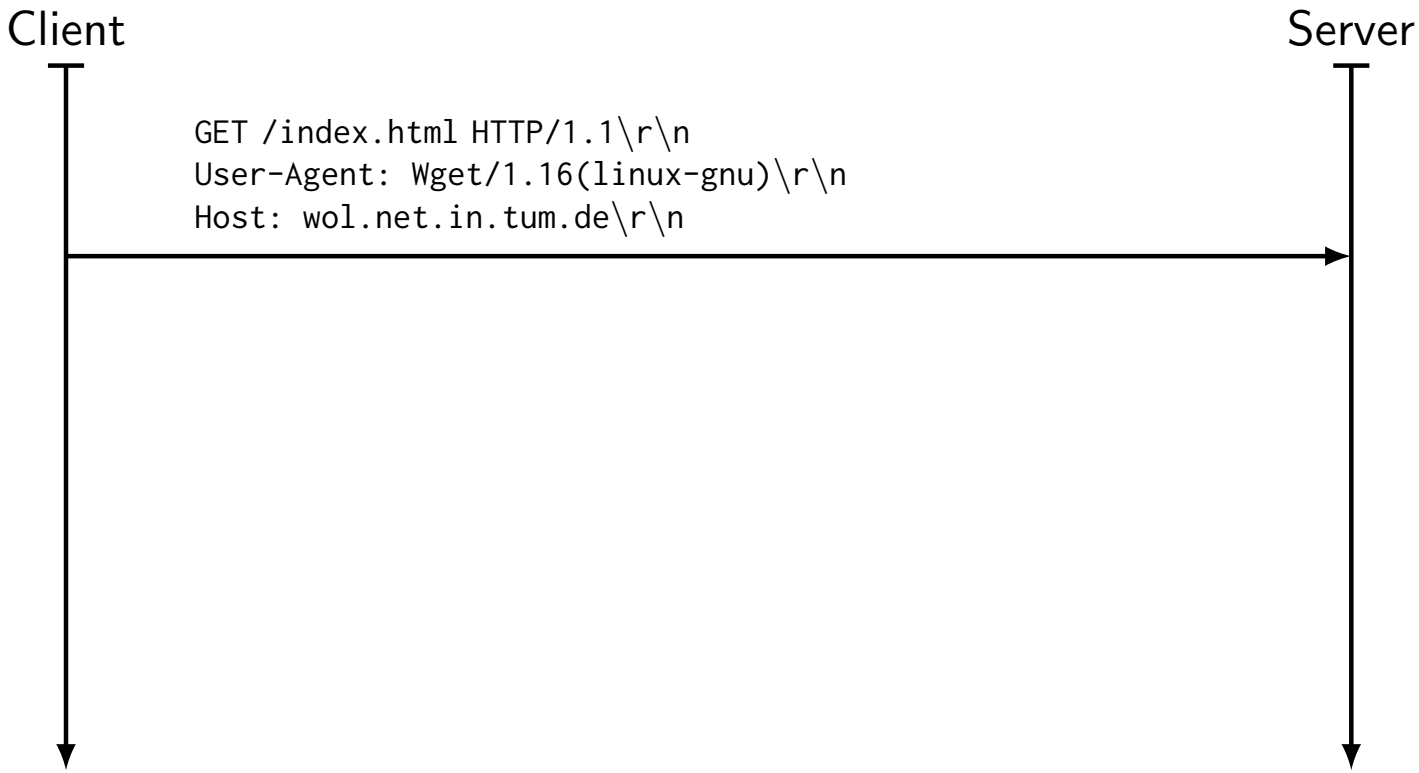
Clients verwenden eine übersichtliche Anzahl unterschiedlicher Kommandos (Methods):

- `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 einer als Header bezeichnet wird)
- `PUT` – Übertragung eines Objekts vom Client zum Server, welches ggf. ein bereits existierendes Objekt überschreibt
- `POST` – Übertraung 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
- `LINK` / `UNLINK` – Herstellung bzw. Entfernung einer Beziehung zwischen zwei unterschiedlichen Objekten

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

- `200` – OK
- `400` – Bad Request
- `401` – Unauthorized
- `403` – Forbidden
- `404` – Not Found
- etc.

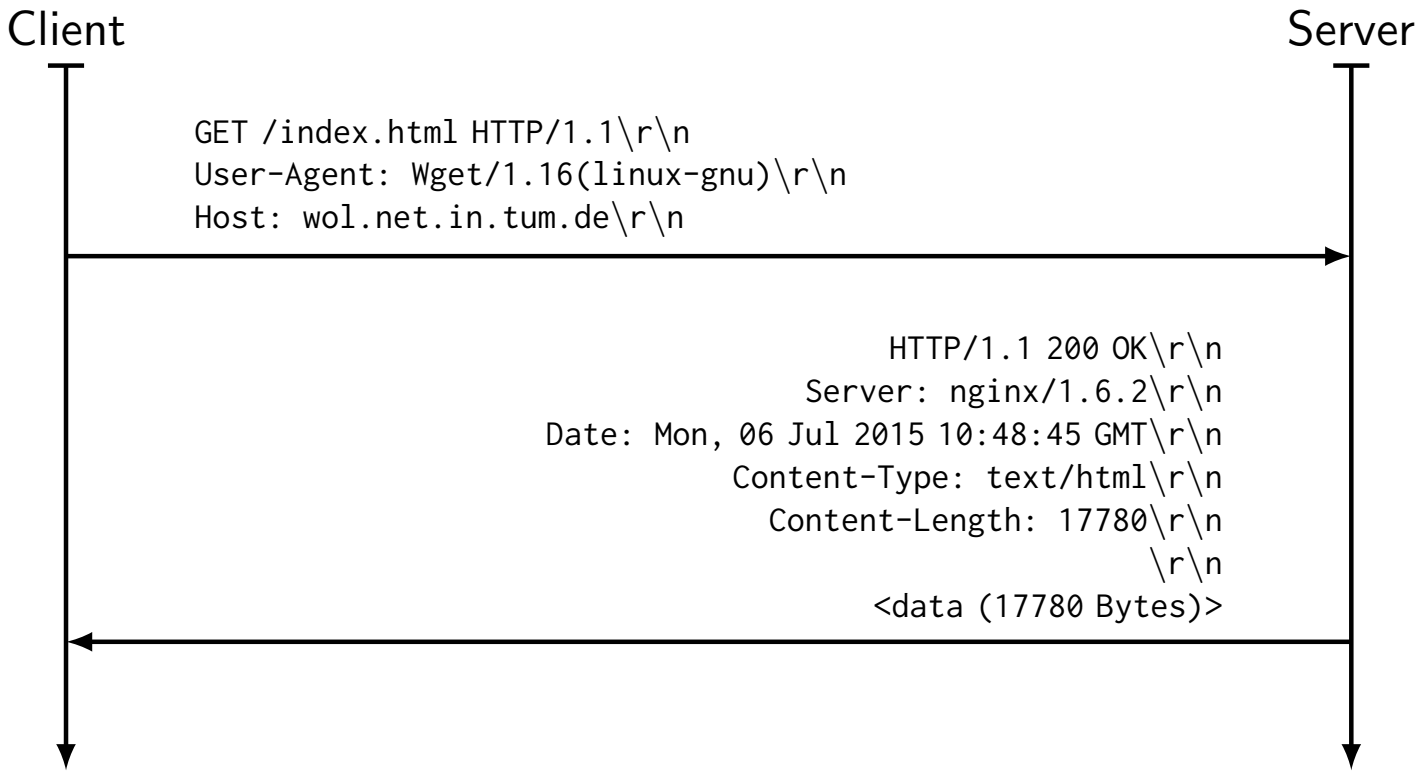
Beispiel: Zugriff auf `http://vcs.net.in.tum.de/svn/grnvss15/pub`



- 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://vcs.net.in.tum.de/svn/grnvss15/pub`



- Der Server antwortet mit dem entsprechenden Status Code.
- Zusätlizch werden verschiedene Optionen angegeben[2], 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.

Verschlüsselung bei HTTP

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

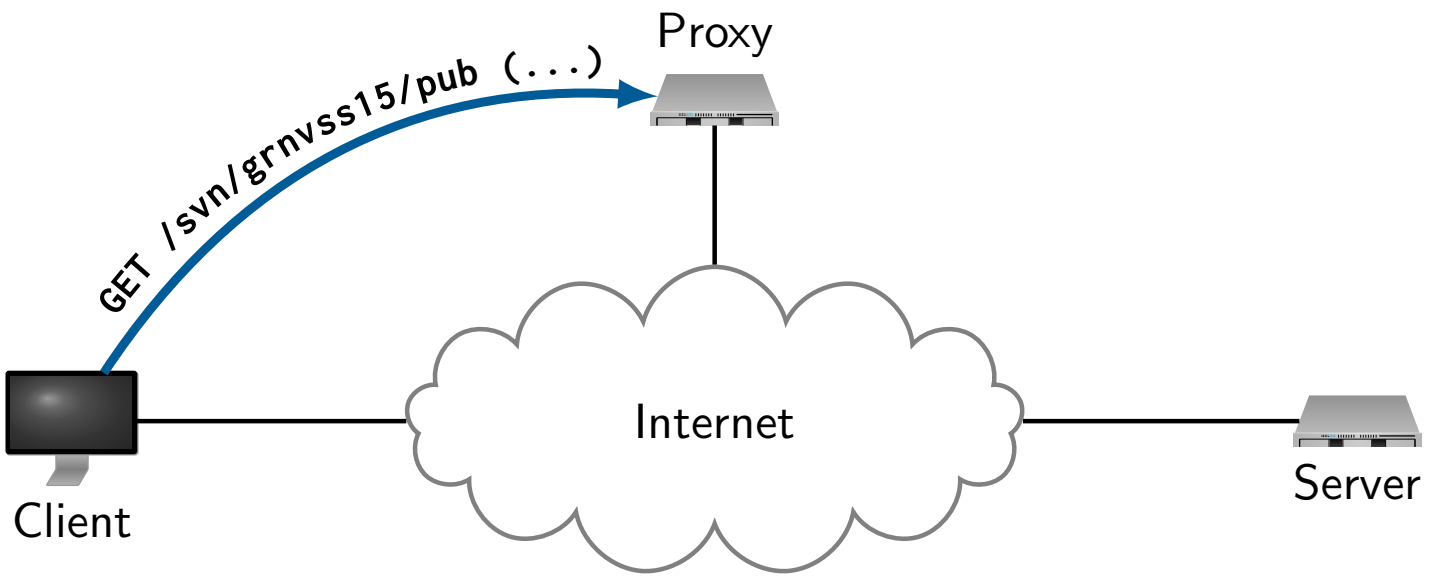
Hyper Text Transfer Protokol Secure (HTTPS)

- 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.

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 verwedet:

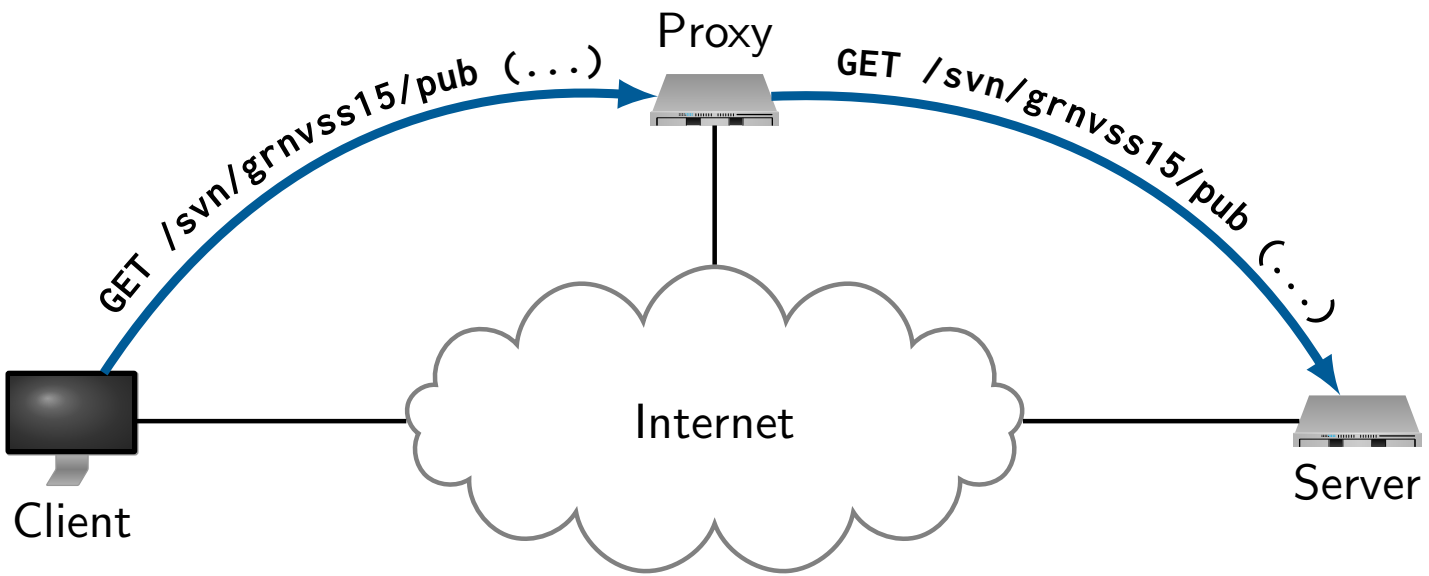
- Anstelle den Zielserver direkt zu kontaktieren, werden HTTP-Anfragen an einen HTTP-Proxy geschickt.
- Dieser nimmt die Anfragen entgegen, baut seinerseits eine neue Verbindung zum eigentlich Zielserver (oder einem weiteren Proxy) auf, und stellt die Anfrage anstelle des Clients.
- Die entsprechende Antwort wird dem 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.



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 verwedet:

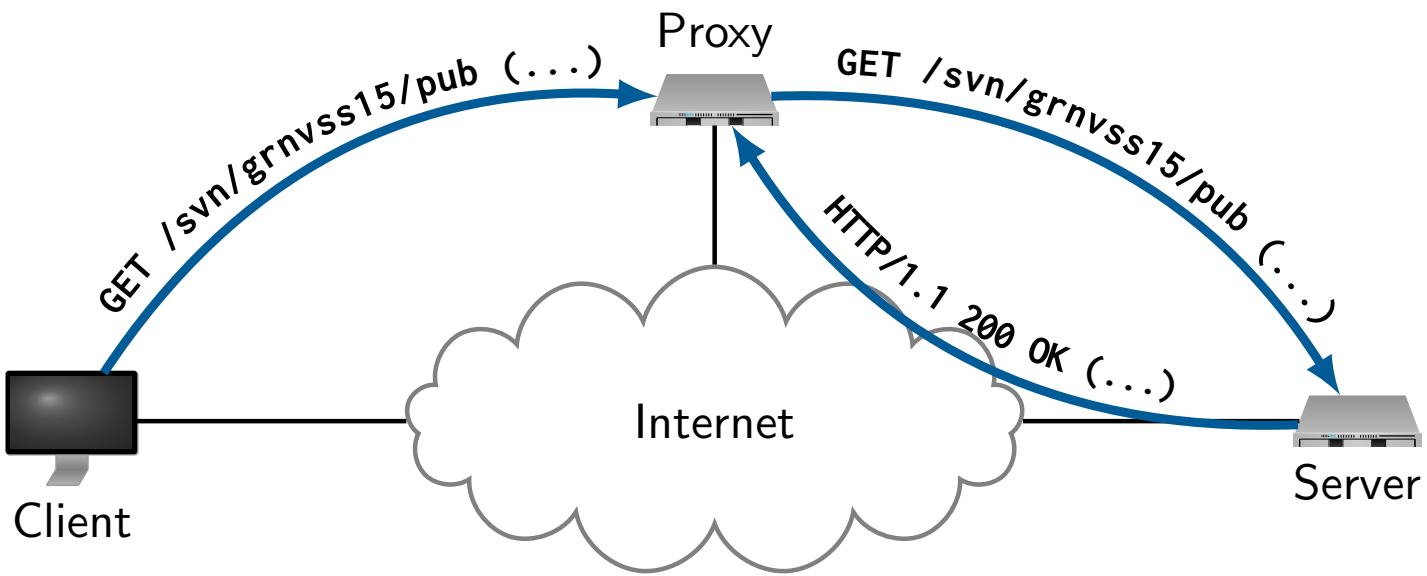
- Anstelle den Zielserver direkt zu kontaktieren, werden HTTP-Anfragen an einen HTTP-Proxy geschickt.
- Dieser nimmt die Anfragen entgegen, baut seinerseits eine neue Verbindung zum eigentlich Zielserver (oder einem weiteren Proxy) auf, und stellt die Anfrage anstelle des Clients.
- Die entsprechende Antwort wird dem 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.



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 verwedet:

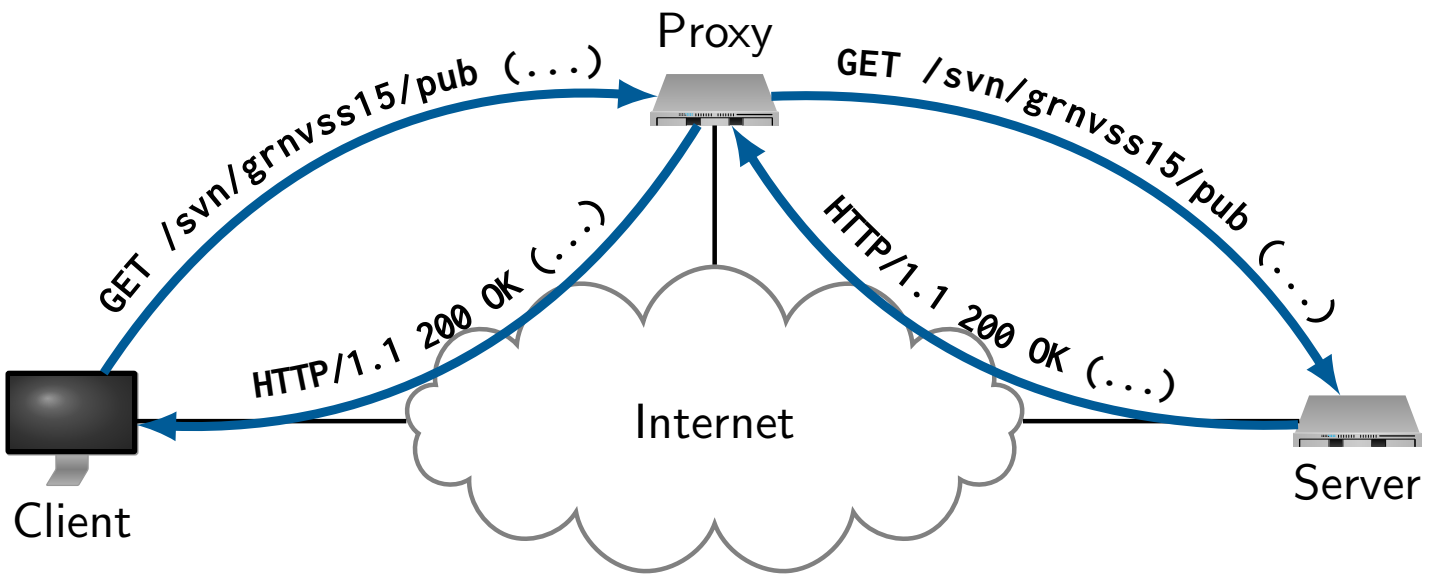
- Anstelle den Zielserver direkt zu kontaktieren, werden HTTP-Anfragen an einen HTTP-Proxy geschickt.
- Dieser nimmt die Anfragen entgegen, baut seinerseits eine neue Verbindung zum eigentlich Zielserver (oder einem weiteren Proxy) auf, und stellt die Anfrage anstelle des Clients.
- Die entsprechende Antwort wird dem 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.

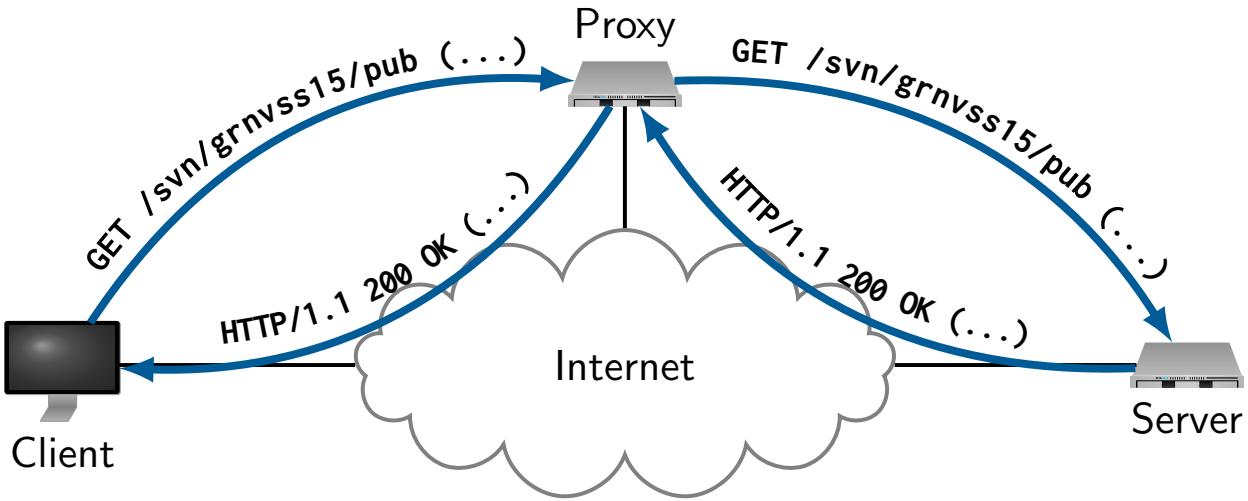


HTTP Proxy

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

- 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 dem 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.





- Der Client selbst bleibt hinter dem Proxy (weitgehend) verborgen, d. h. der Webserver sieht lediglich den Proxy, nicht aber die Clients.[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 unbetroffen 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)

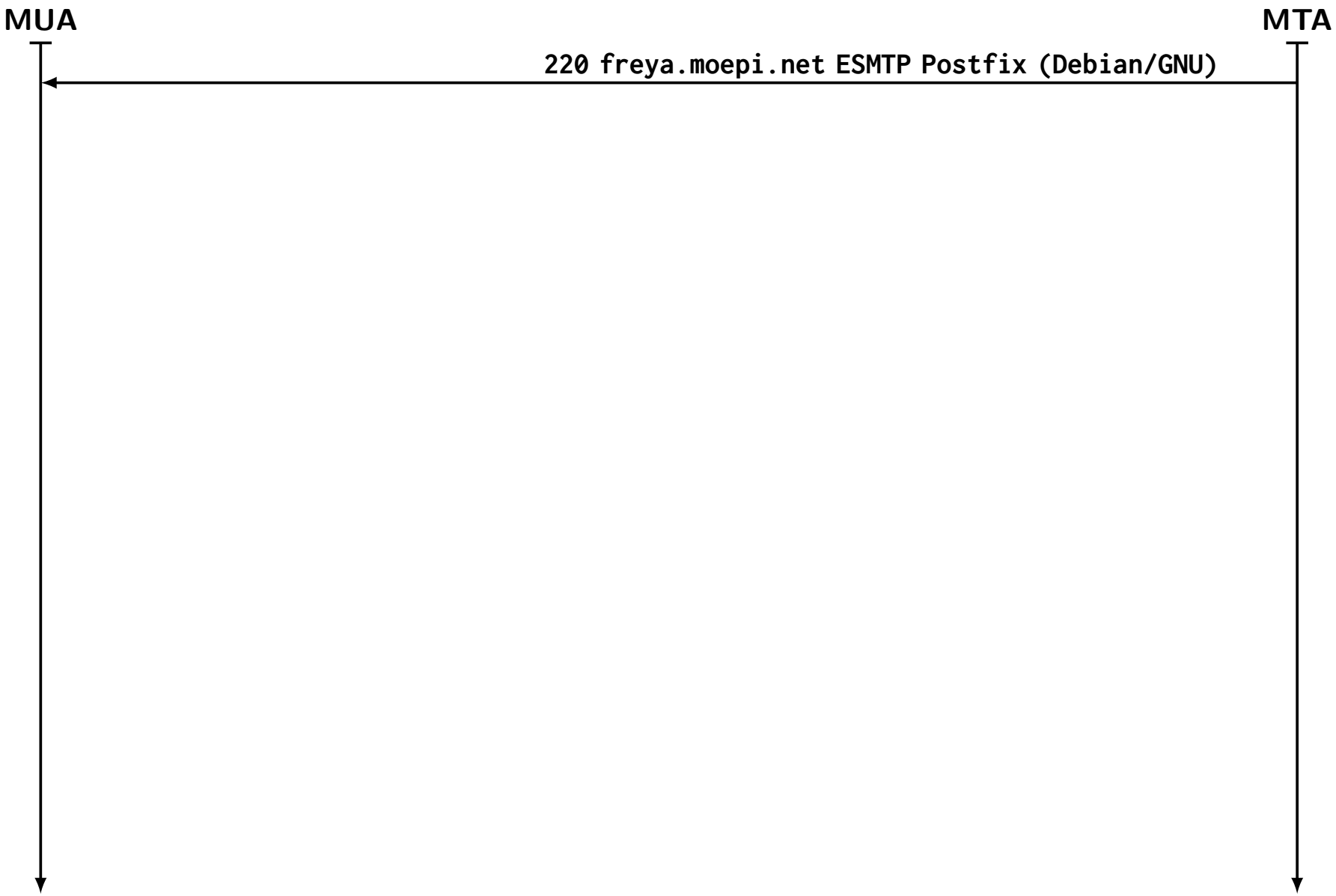
Das Simple Mail Transfer Protocol (SMTP) ist ein textbasiertes Protokoll und dient

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

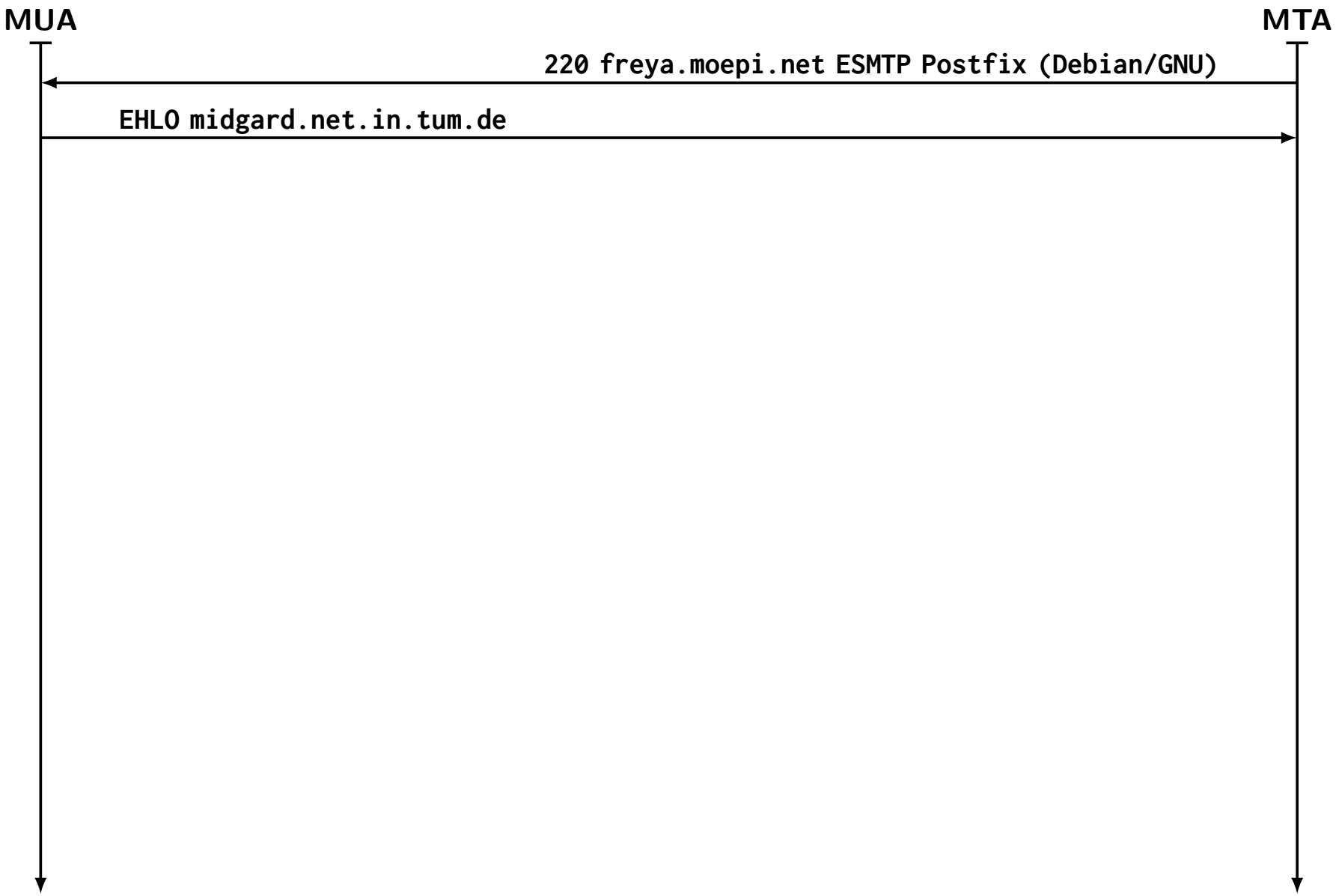
In Empfangsrichtung werden i. d R.

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

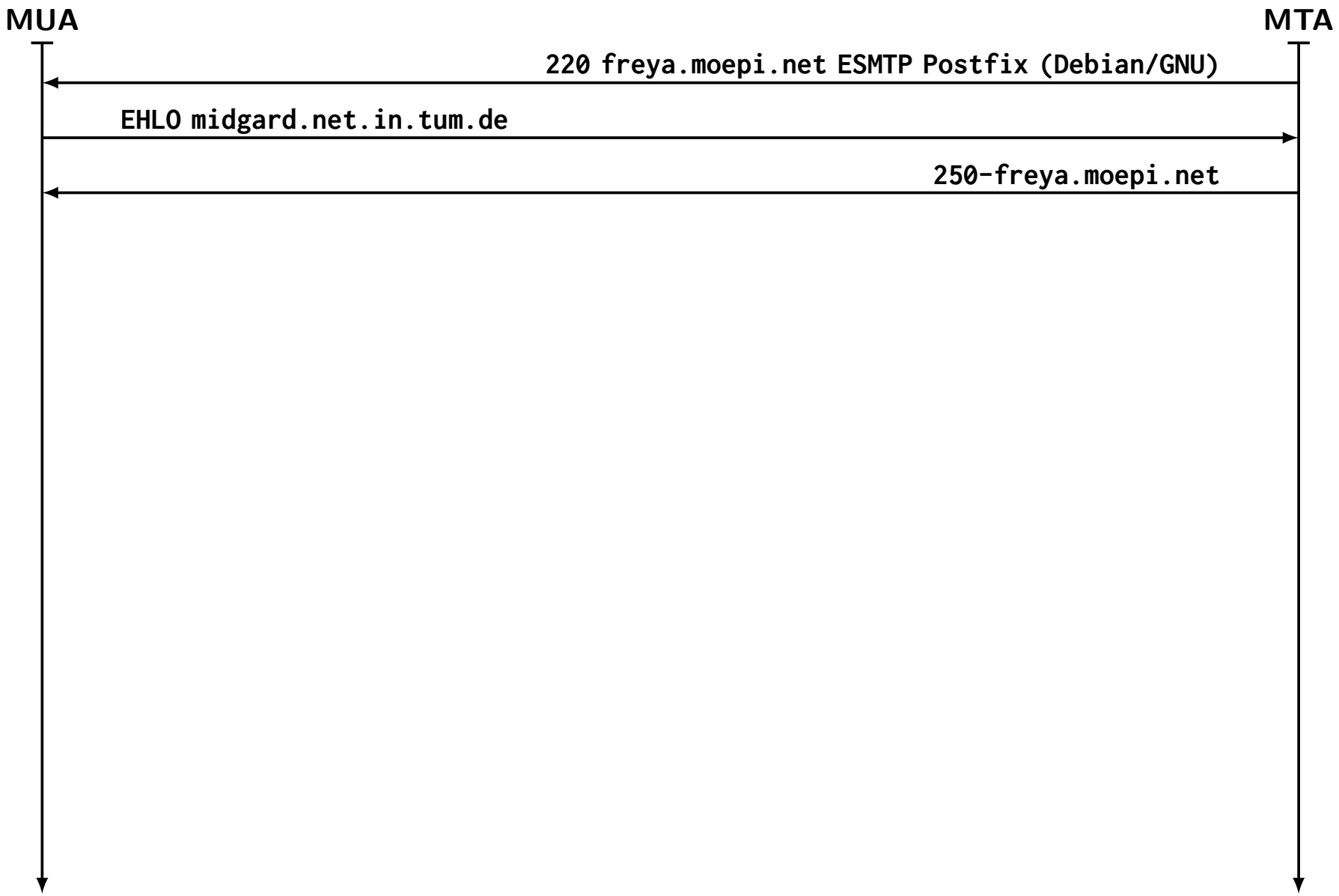
Beispiel: SMTP



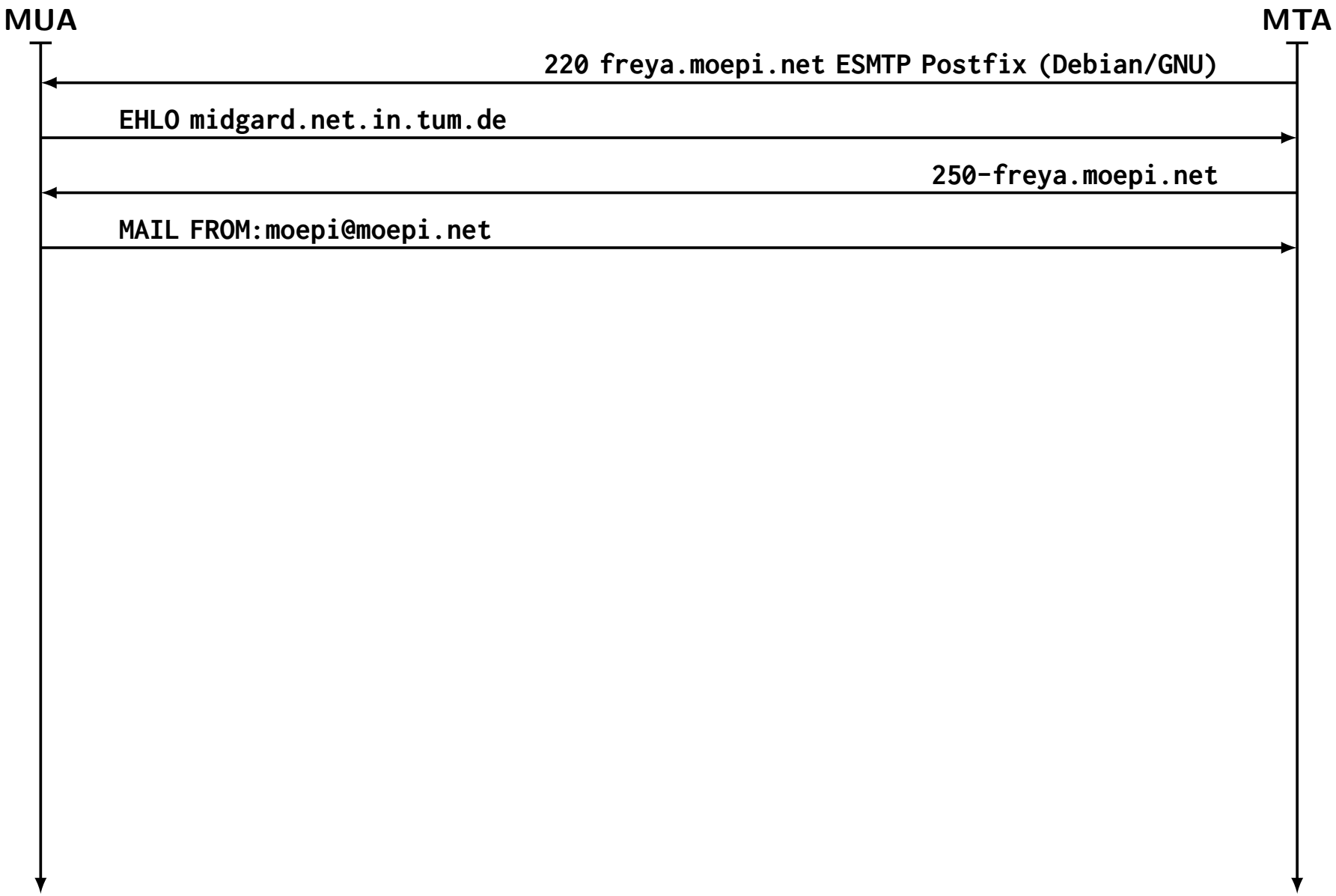
Beispiel: SMTP



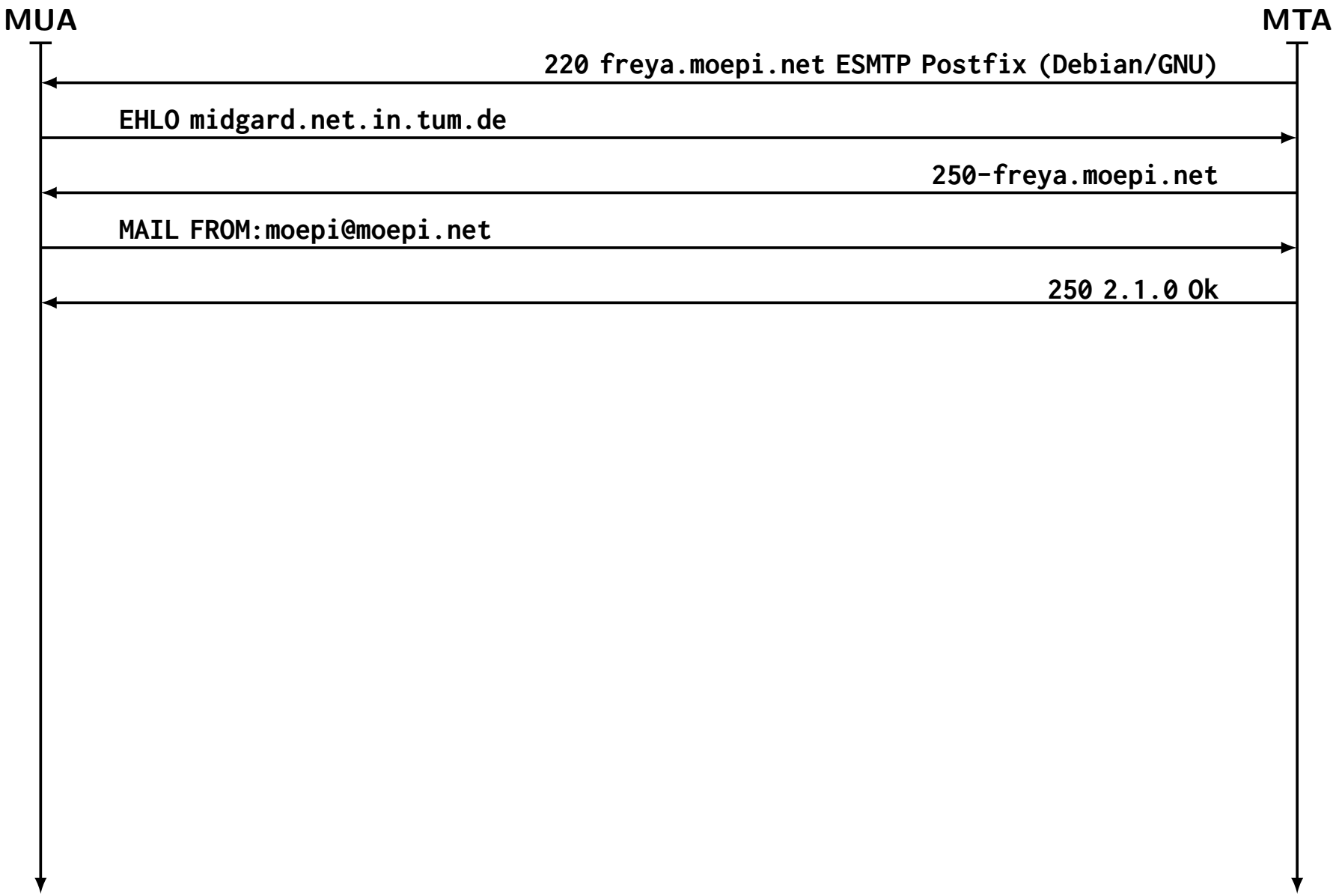
Beispiel: SMTP



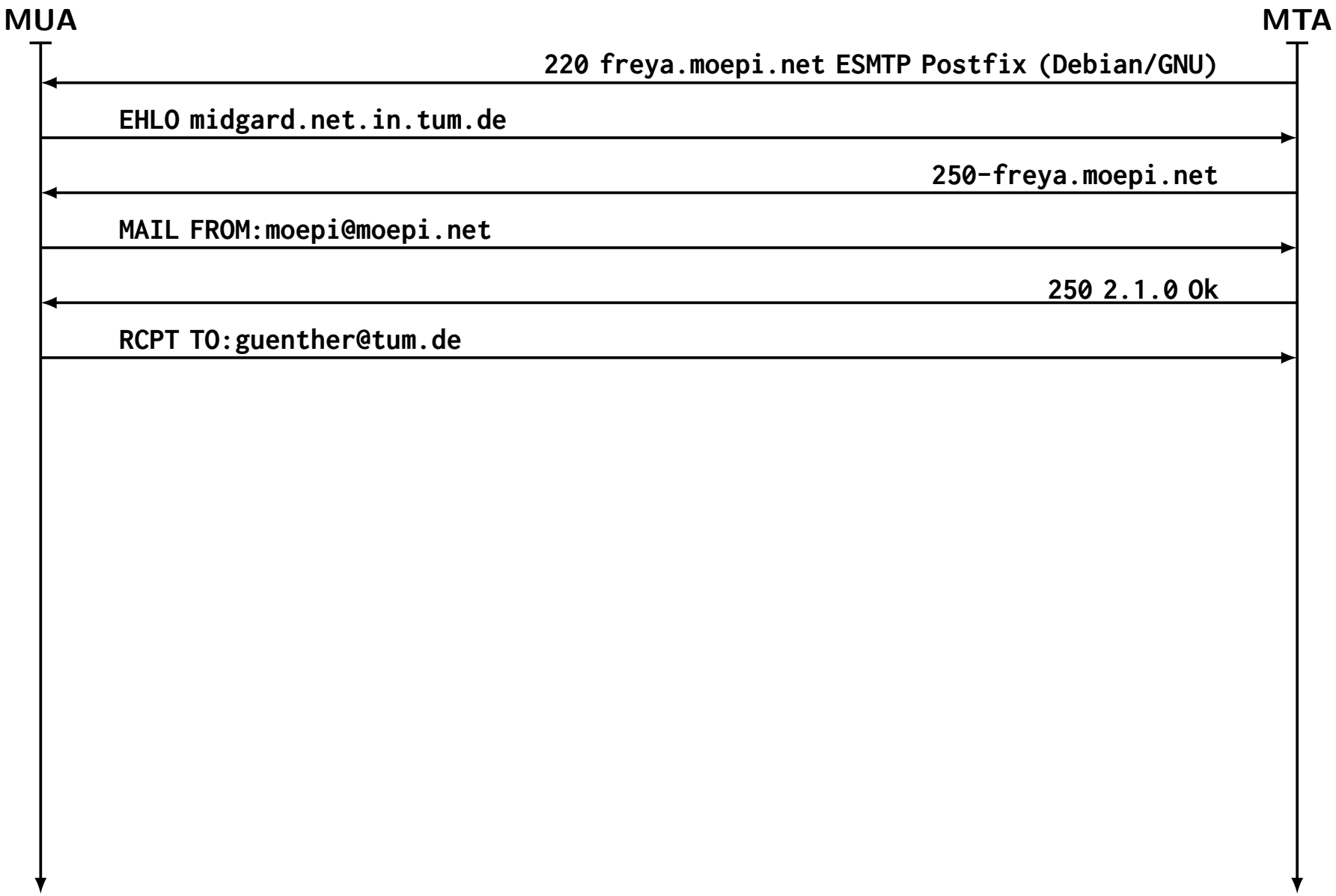
Beispiel: SMTP



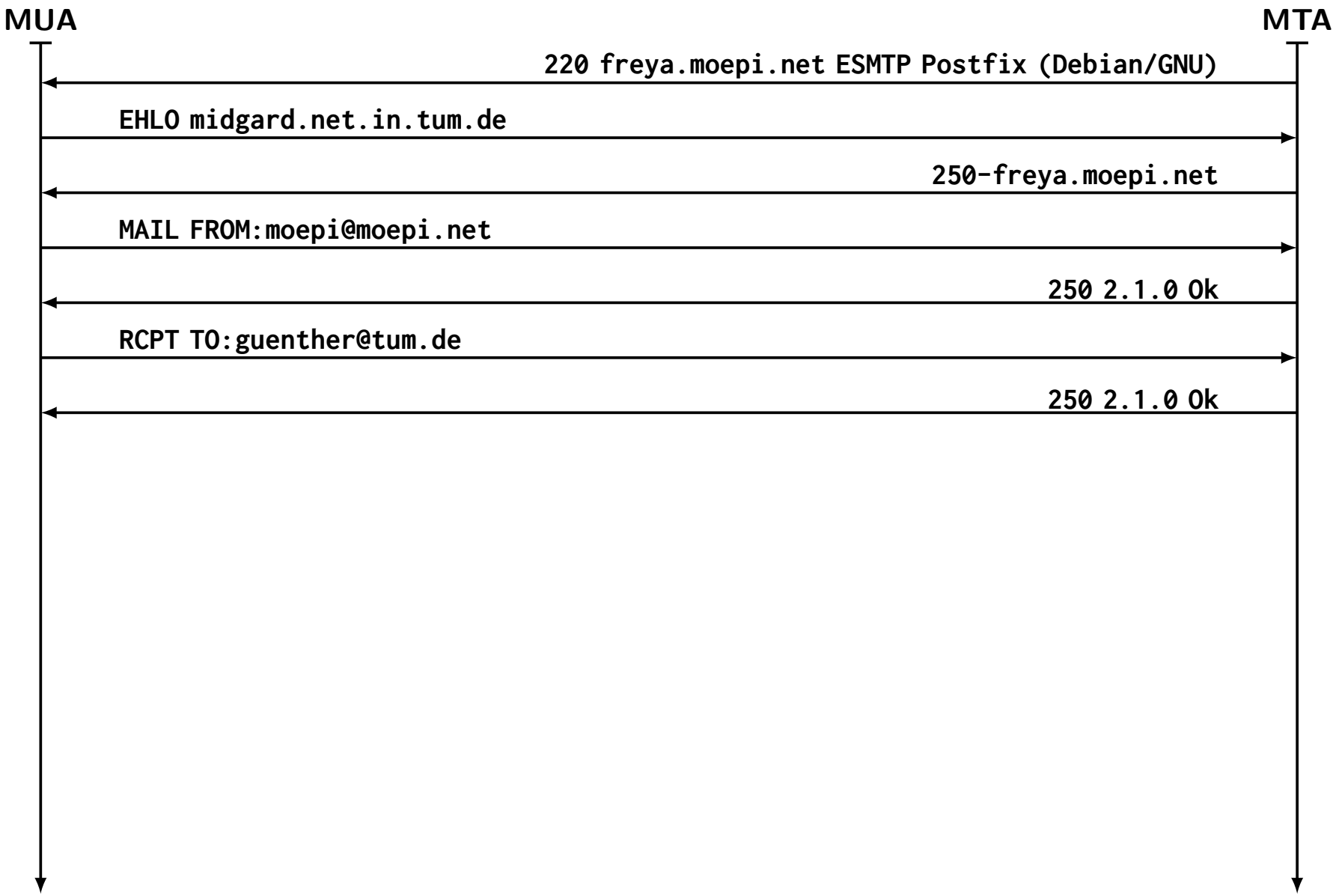
Beispiel: SMTP



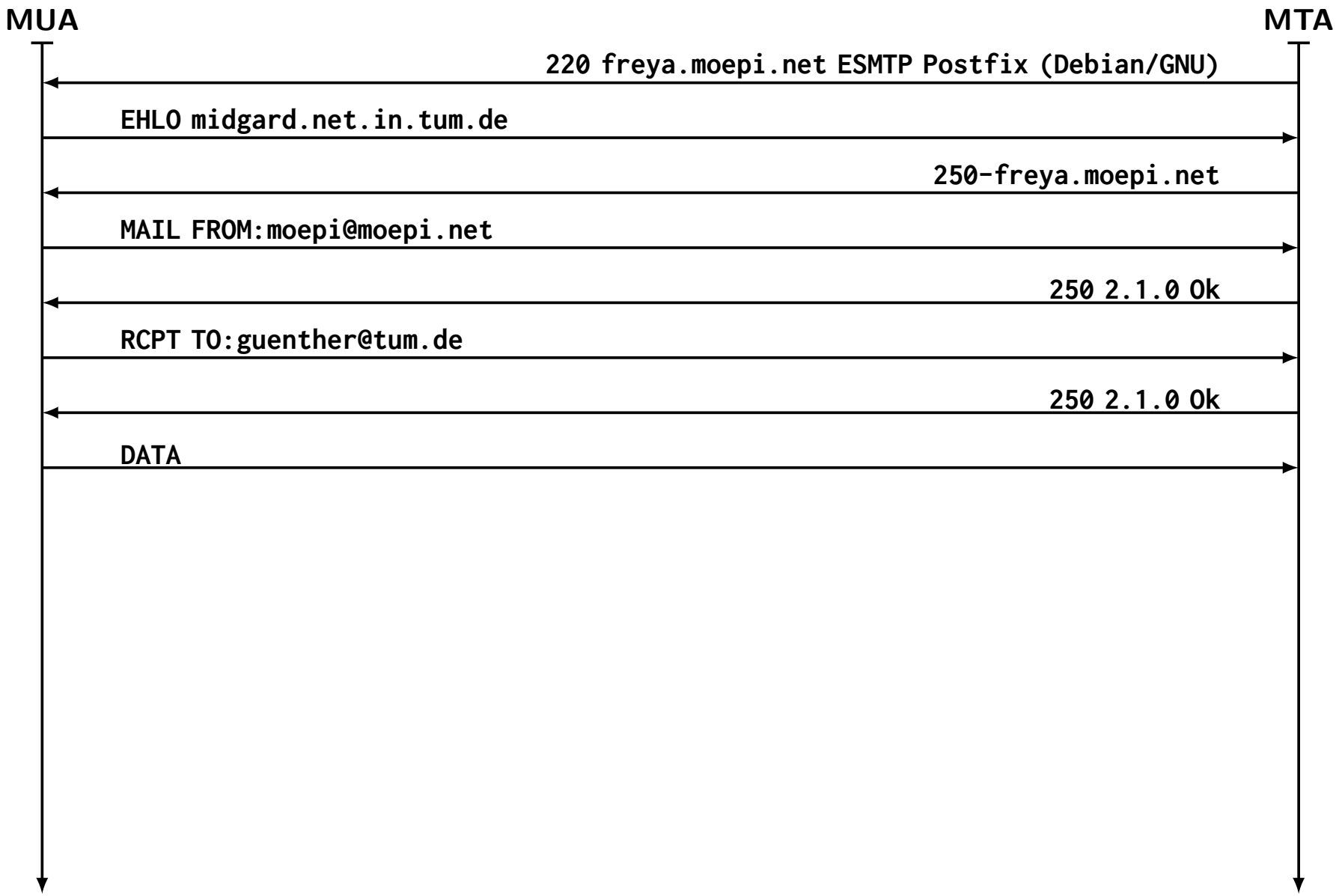
Beispiel: SMTP



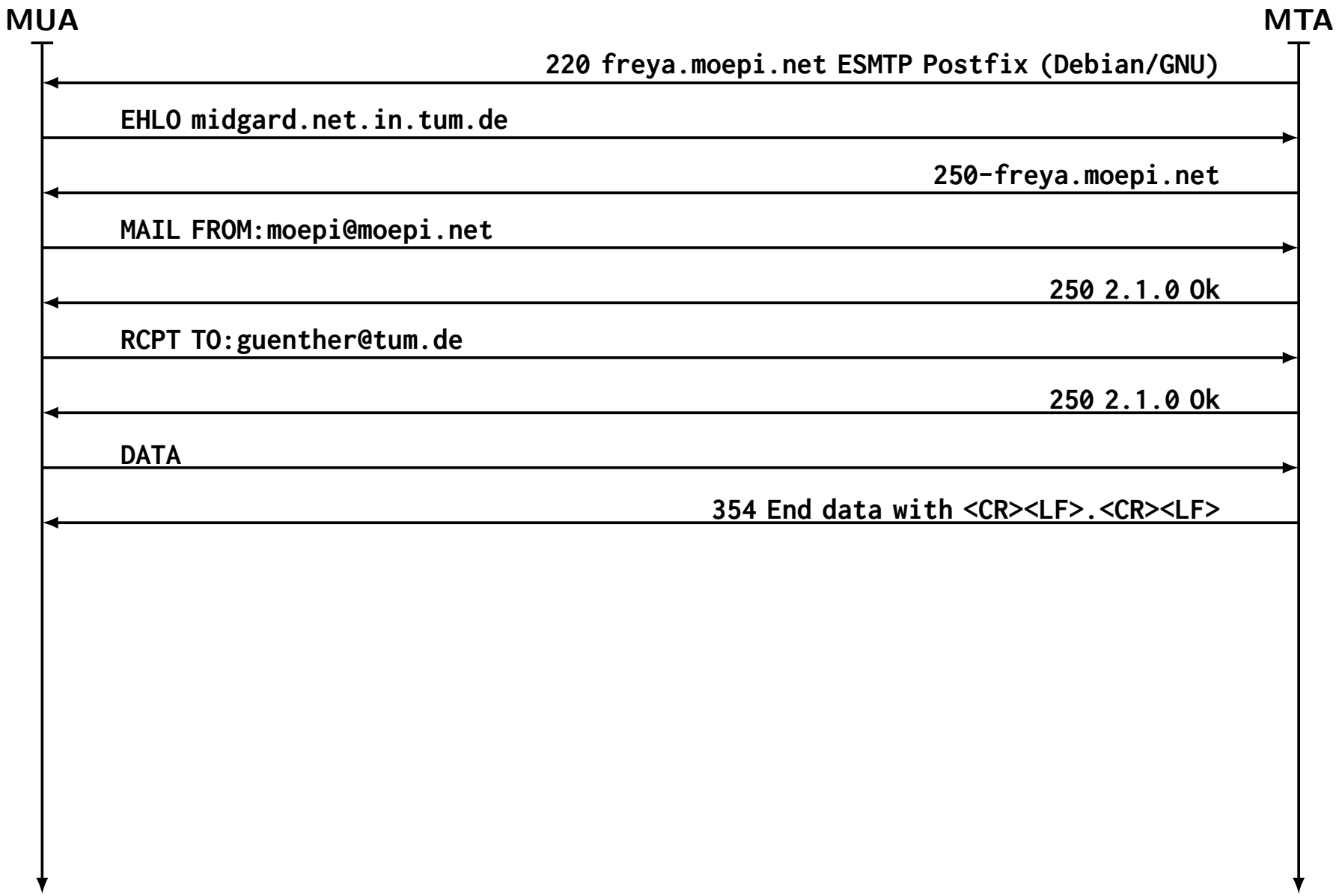
Beispiel: SMTP



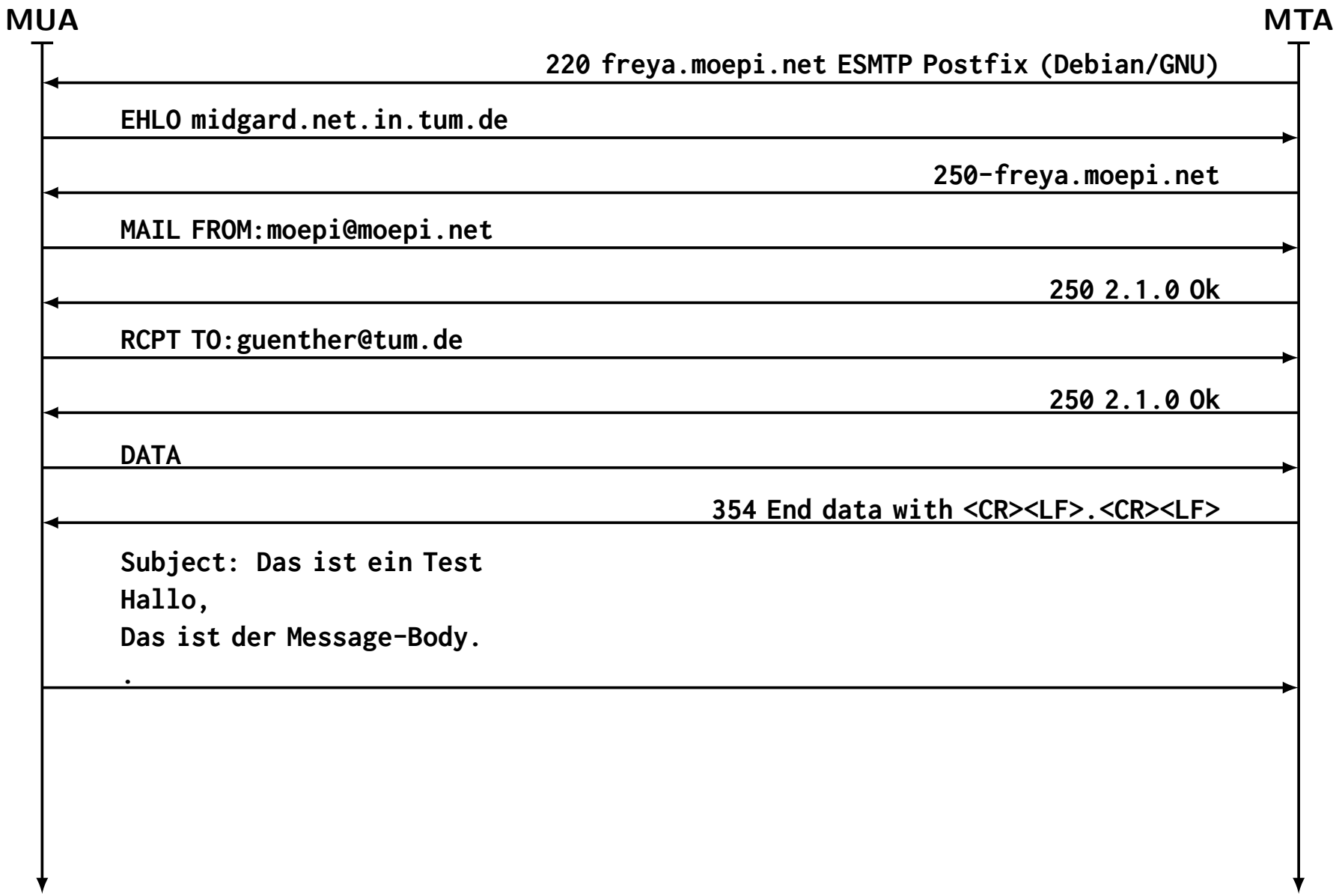
Beispiel: SMTP



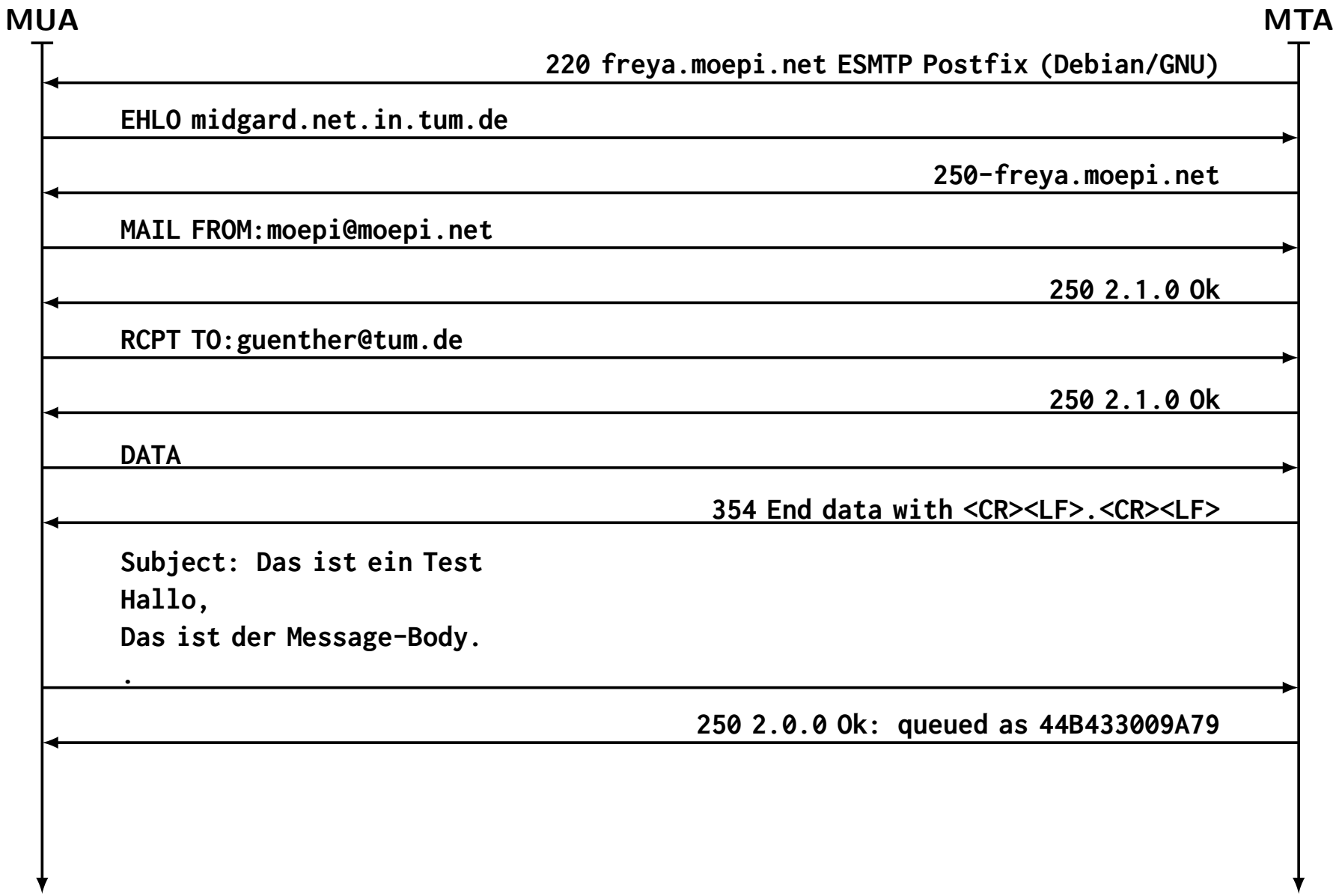
Beispiel: SMTP



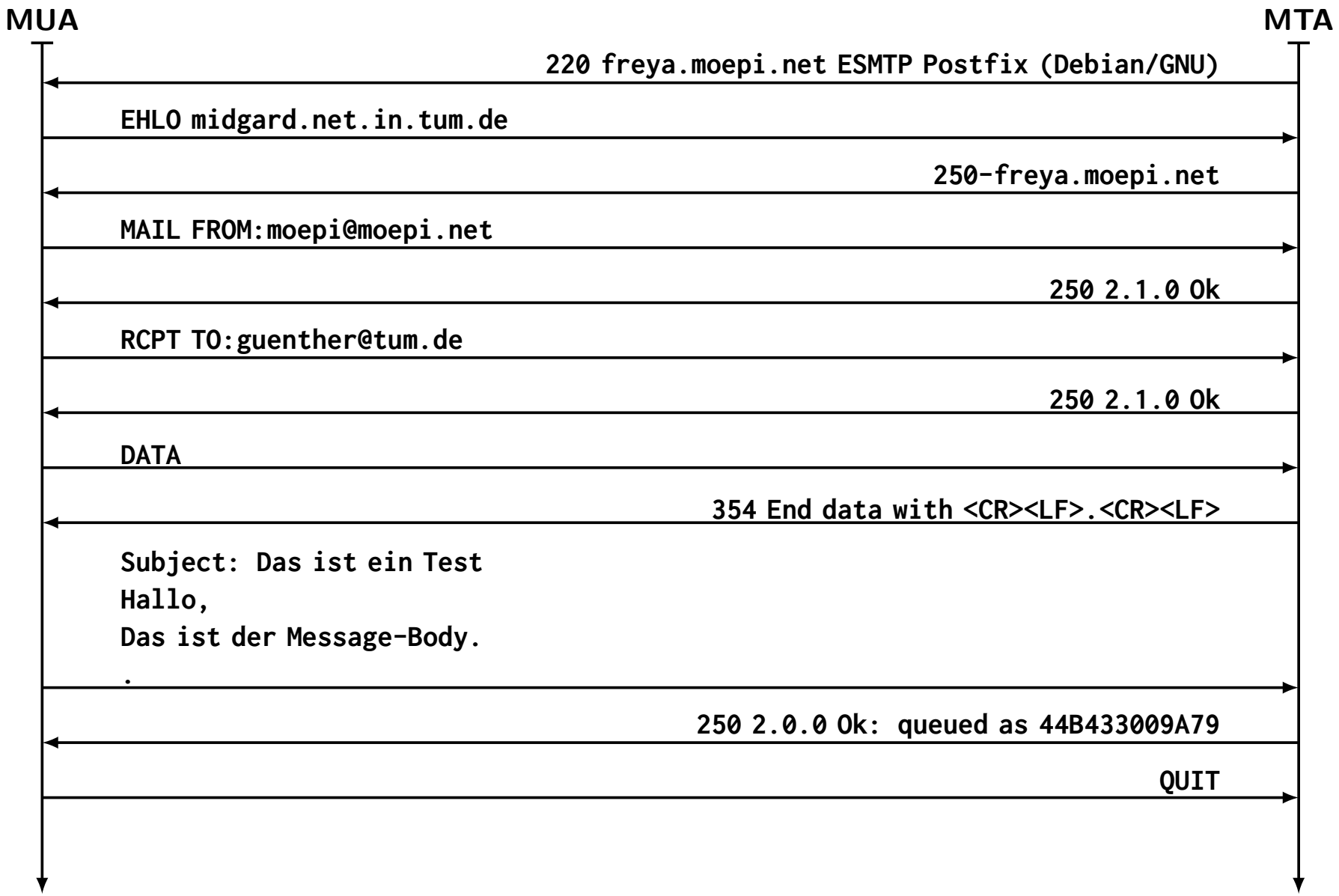
Beispiel: SMTP



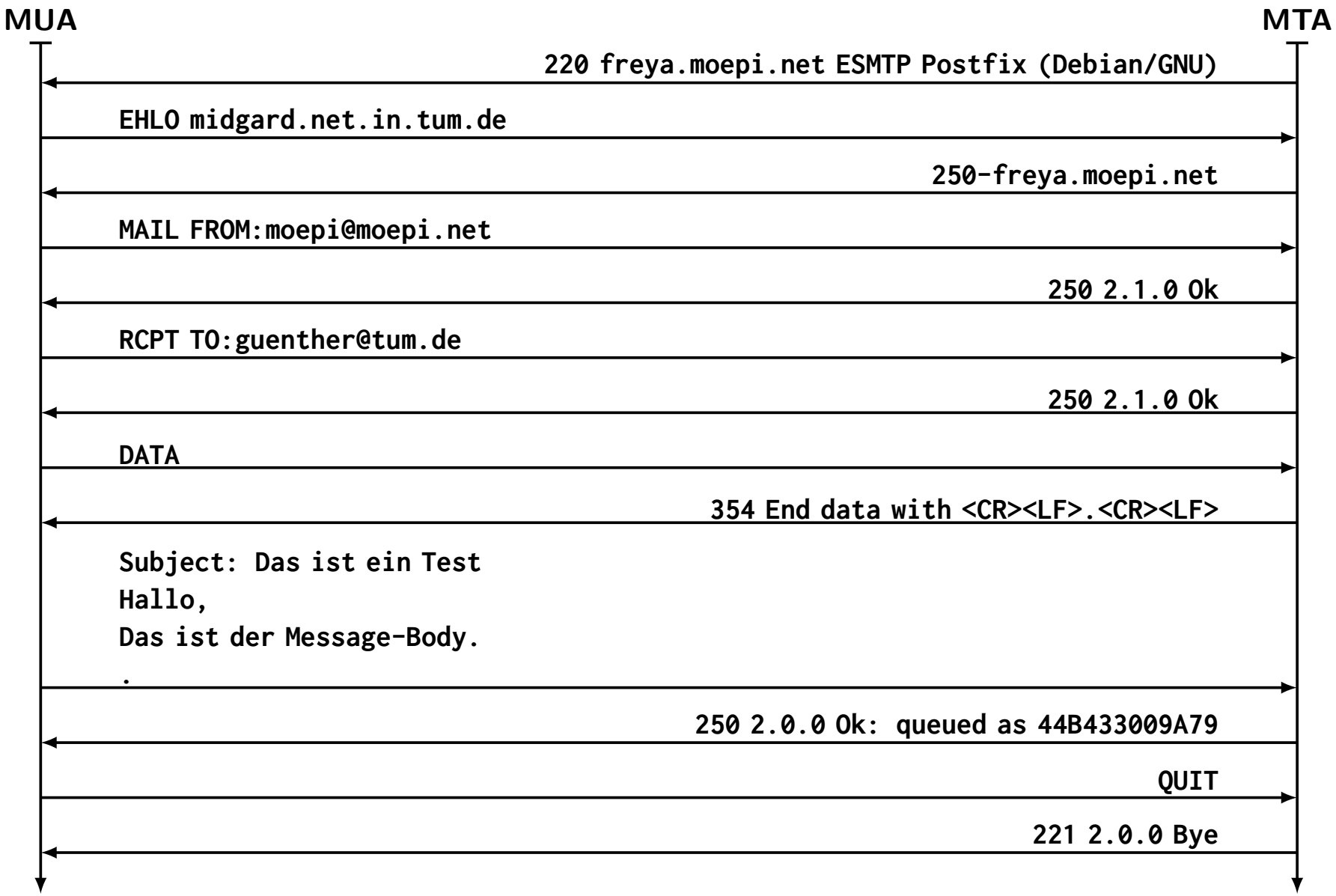
Beispiel: SMTP



Beispiel: SMTP



Beispiel: SMTP



Beispiel: SMTP (Fortsetzung) Nach Erhalt der Email versucht der MTA diese zuzustellen:

- Die Empfängeradresse `guenther@tum.de` enthält der 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 Zeitabst]änden 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 Nachrichtig 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 der 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 Zeitabst]änden 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 Nachrichtig 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 Syncronisation einer Mailbox zwischen dem Server und einem oder mehreren Mailclients.

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 postforw2.mail.lrz.de (lxmhs62.srv.lrz.de [IPv6:2001:4ca0:0:116::a9c:63e])
  by forwout2.mail.lrz.de (Postfix) with ESMTP id 3mM8Qy1SDPz8q
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:34 +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 (...)
```

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 postforw2.mail.lrz.de (lxmhs62.srv.lrz.de [IPv6:2001:4ca0:0:116::a9c:63e])
  by forwout2.mail.lrz.de (Postfix) with ESMTP id 3mM8Qy1SDPz8q
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:34 +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 (...)
```

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 postforw2.mail.lrz.de (lxmhs62.srv.lrz.de [IPv6:2001:4ca0:0:116::a9c:63e])
  by forwout2.mail.lrz.de (Postfix) with ESMTP id 3mM8Qy1SDPz8q
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:34 +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 (...)
```

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 postforw2.mail.lrz.de (lxmhs62.srv.lrz.de [IPv6:2001:4ca0:0:116::a9c:63e])
  by forwout2.mail.lrz.de (Postfix) with ESMTP id 3mM8Qy1SDPz8q
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:34 +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 (...)
```

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 postforw2.mail.lrz.de (lxmhs62.srv.lrz.de [IPv6:2001:4ca0:0:116::a9c:63e])
  by forwout2.mail.lrz.de (Postfix) with ESMTP id 3mM8Qy1SDPz8q
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:34 +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 (...)
```

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 postforw2.mail.lrz.de (lxmhs62.srv.lrz.de [IPv6:2001:4ca0:0:116::a9c:63e])
  by forwout2.mail.lrz.de (Postfix) with ESMTP id 3mM8Qy1SDPz8q
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:34 +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 (...)
```

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 postforw2.mail.lrz.de (lxmhs62.srv.lrz.de [IPv6:2001:4ca0:0:116::a9c:63e])
  by forwout2.mail.lrz.de (Postfix) with ESMTP id 3mM8Qy1SDPz8q
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:34 +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 (...)
```

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 postforw2.mail.lrz.de (lxmhs62.srv.lrz.de [IPv6:2001:4ca0:0:116::a9c:63e])
  by forwout2.mail.lrz.de (Postfix) with ESMTP id 3mM8Qy1SDPz8q
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:34 +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 (...)
```

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 postforw2.mail.lrz.de (lxmhs62.srv.lrz.de [IPv6:2001:4ca0:0:116::a9c:63e])
  by forwout2.mail.lrz.de (Postfix) with ESMTP id 3mM8Qy1SDPz8q
  for <moepi@moepi.net>; Wed,  1 Jul 2015 19:18:34 +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 (...)
```

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 Versandt von Spam missbraucht wird und auf Blacklists landet.
- Authentifizierung zwischen MTAs ist unüblich da technisch nicht umsetzbar.[1]
- Sowohl SMTP als auch POP3 und 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 dynamisches IP-Adressen, welche meist keine PTR-Records besitzen und infolge dessen häufig von anderen MTAs zurückgewiesen werden (Spamschutz).

File Transfer Protocol (FTP) [8]

Das File Transfer Protocol (FTP) ist ein weiteres Protokoll zum Transfer von Daten (Text wie Binaerdaten). 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 Daten.
- Der Kontrollkanal bleibt über mehrere Datentransfers hinweg bestehen, d. h. FTP ist statefull.
- FTP erfordert grundsätzlich eine Art von Authentifizierung (anonymer Zugang mittels Benutzername `anonymous` und beliebigem Passwort sofern konfiguriert)

File Transfer Protocol (FTP) [8]

Das File Transfer Protocol (FTP) ist ein weiteres Protokoll zum Transfer von Daten (Text wie Binaerdaten). Unterschiede zu HTTP:

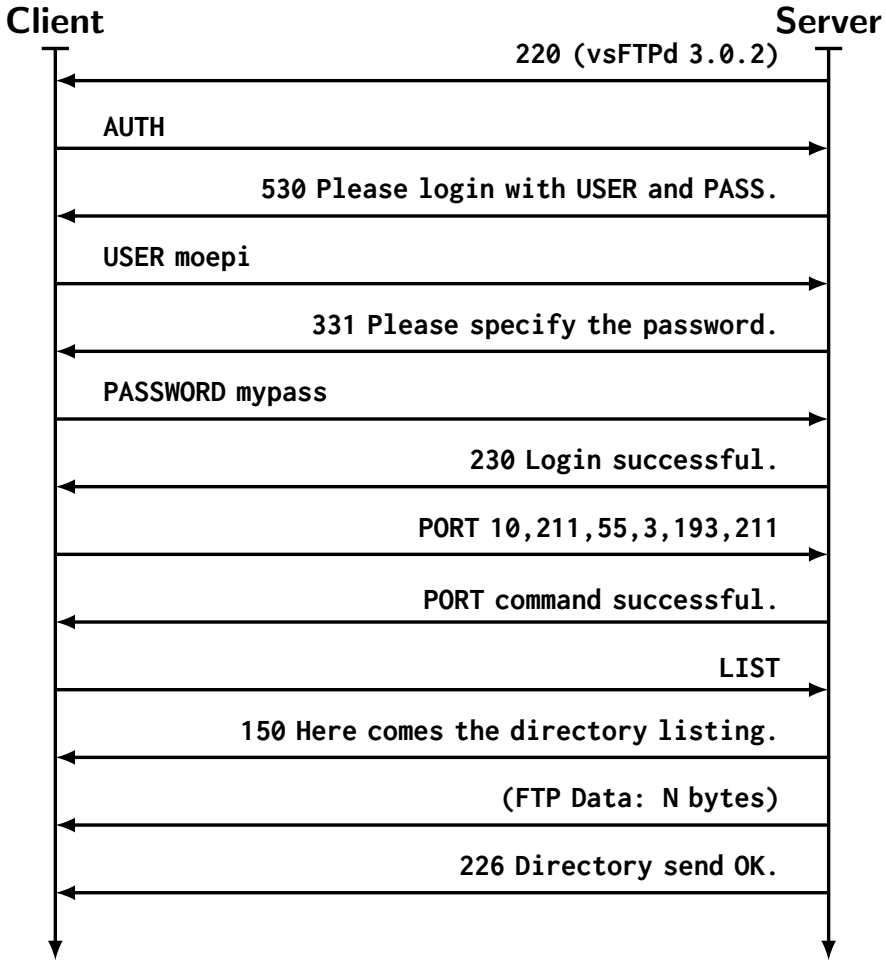
- FTP nutzt zwei getrennte TCP-Verbindungen:
  1. Kontrollkanal zur Übermittlung von Befehlen und Statuscodes zwischen Client und Server.
  2. Datenkanal zur Übertragung der eigentlichen Daten Daten.
- Der Kontrollkanal bleibt über mehrere Datentransfers hinweg bestehen, d. h. FTP ist statefull.
- FTP 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:

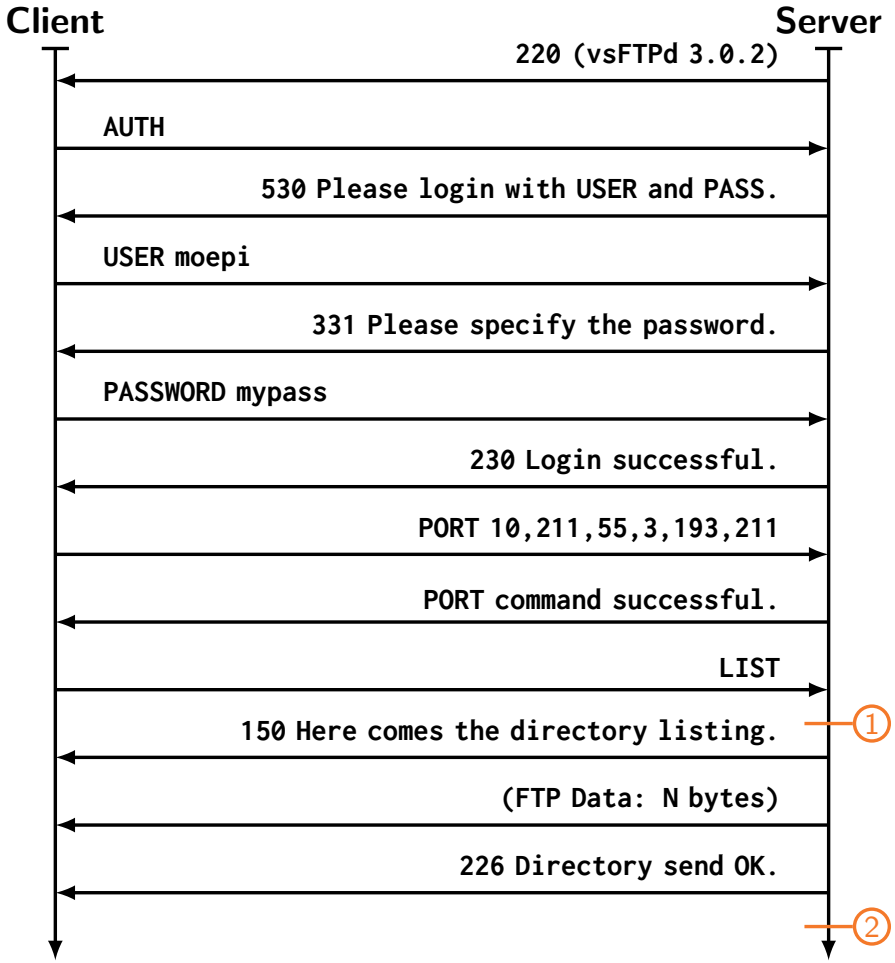
FTP active mode



FTP passive mode

Beispiel:

FTP active mode

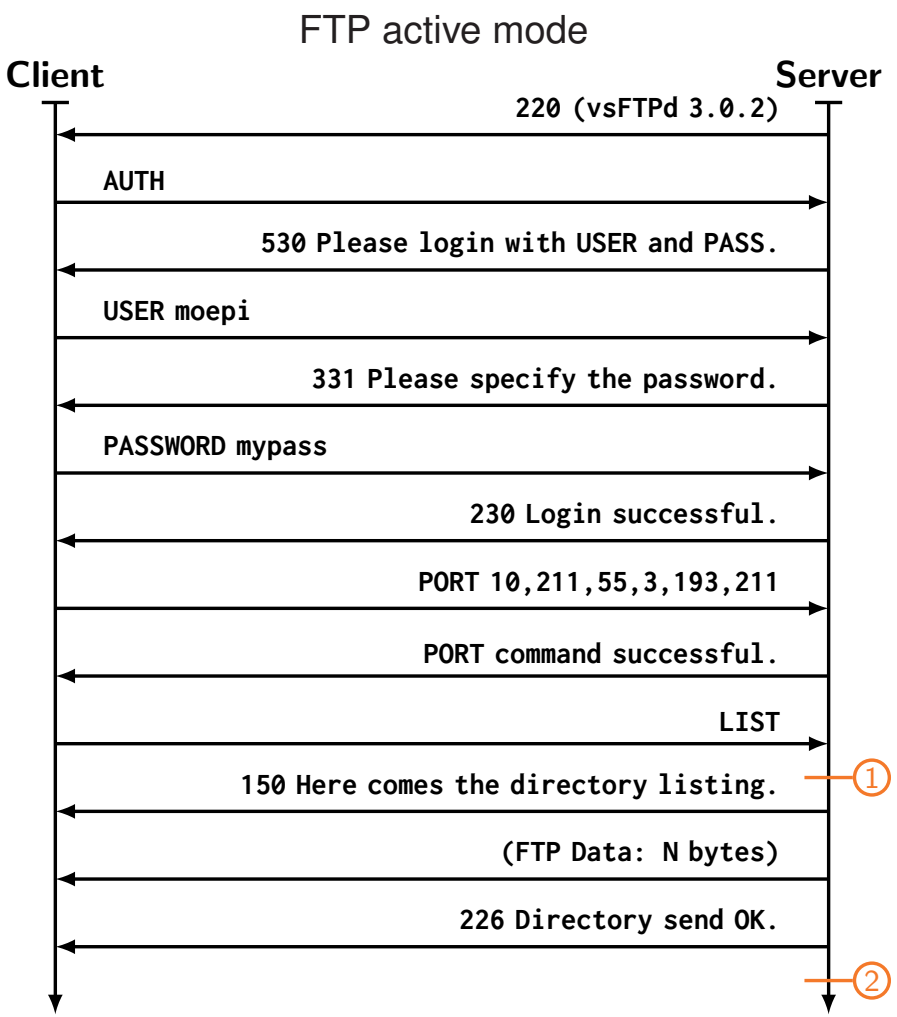


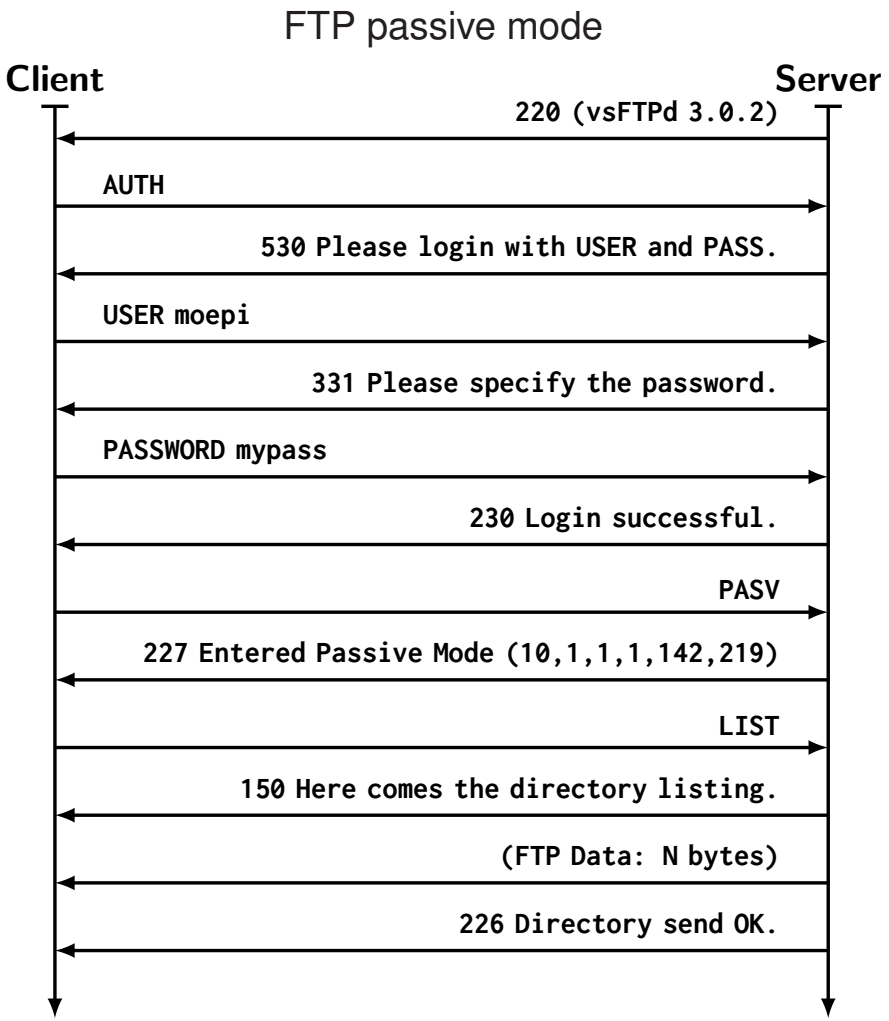
① Aufbau des Datenkanals vom Server an 10.211.55.3:49619

② Abbau des Datenkanals

FTP passive mode

Beispiel:

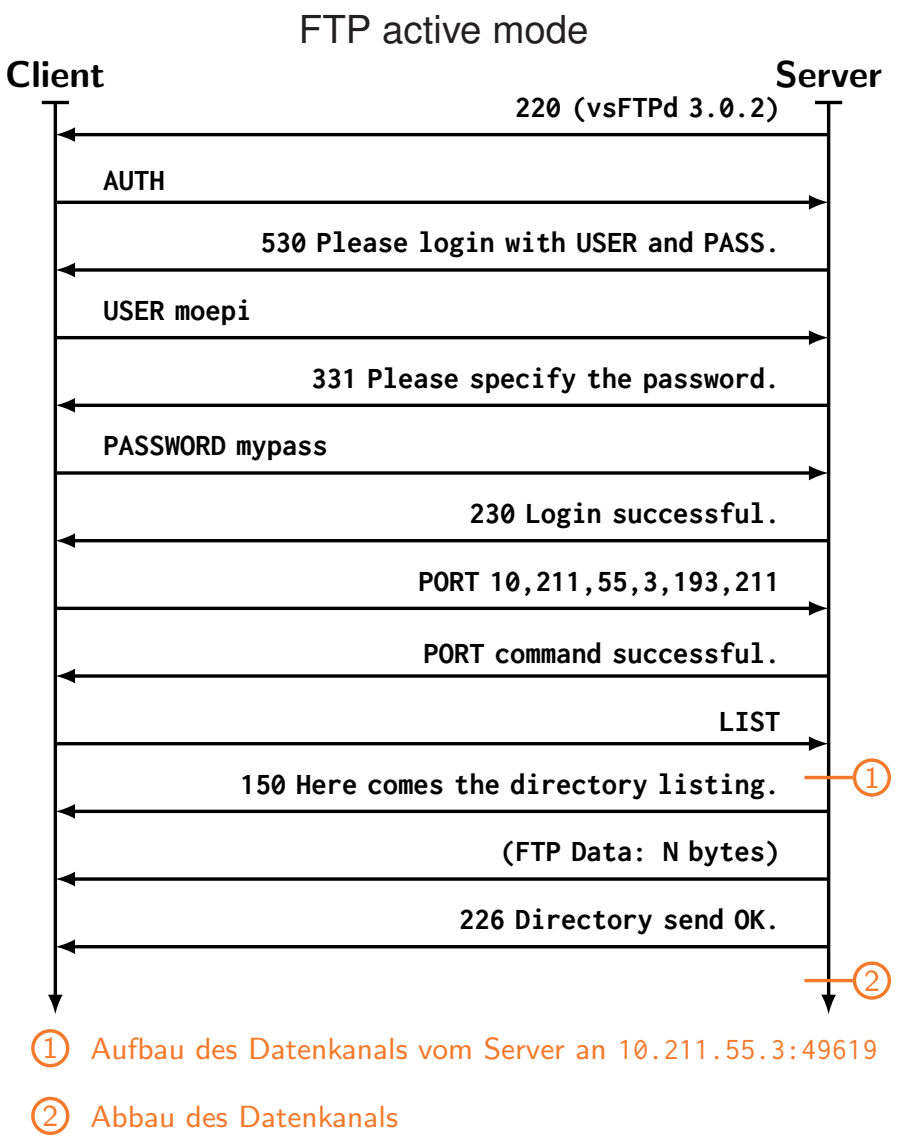




① Aufbau des Datenkanals vom Server an `10.211.55.3:49619`

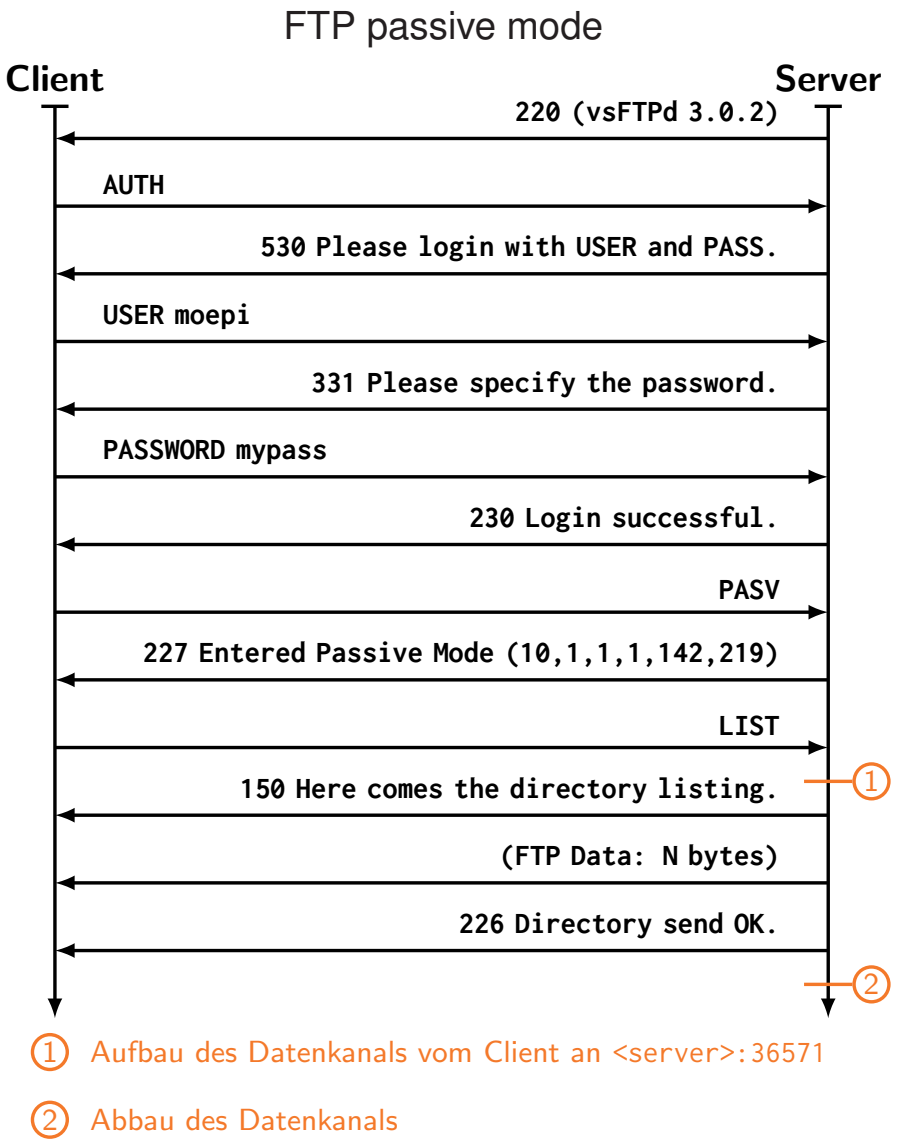
② Abbau des Datenkanals

Beispiel:



① Aufbau des Datenkanals vom Server an 10.211.55.3:49619

② Abbau des Datenkanals

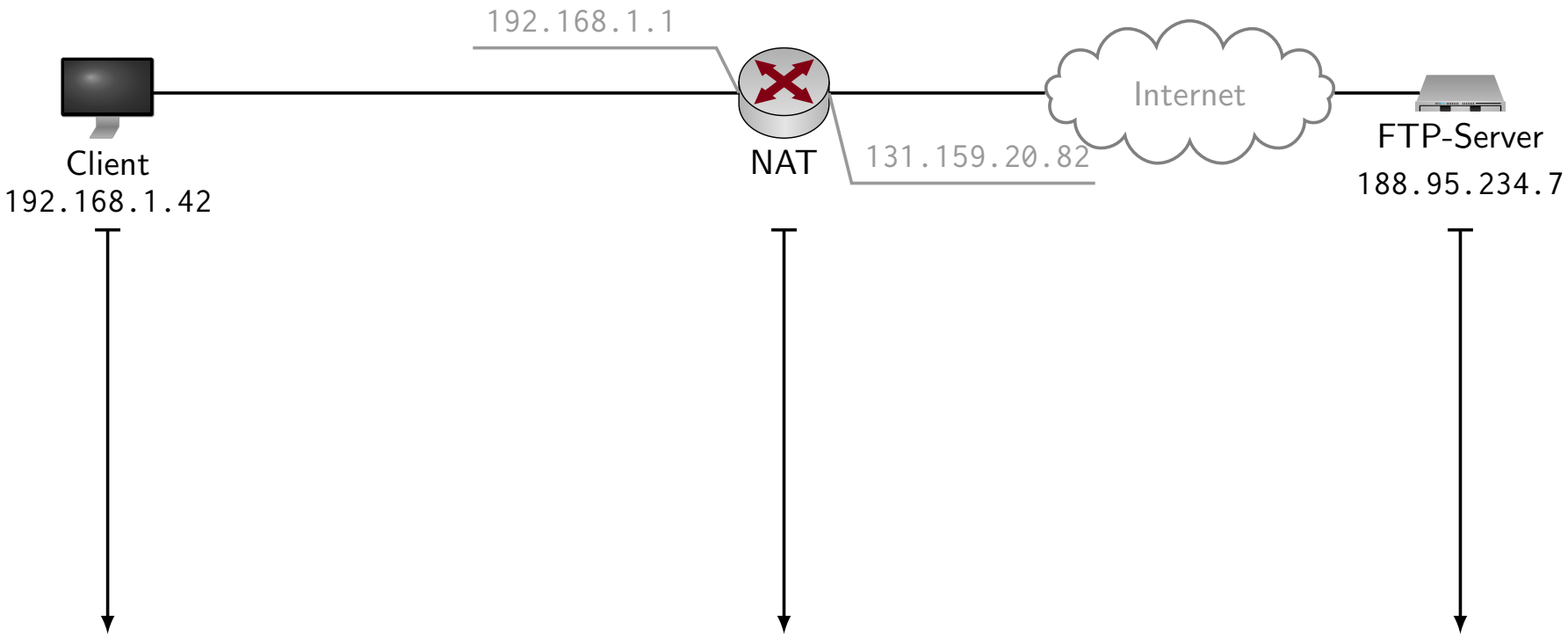


① Aufbau des Datenkanals vom Client an <server>:36571

② Abbau des Datenkanals

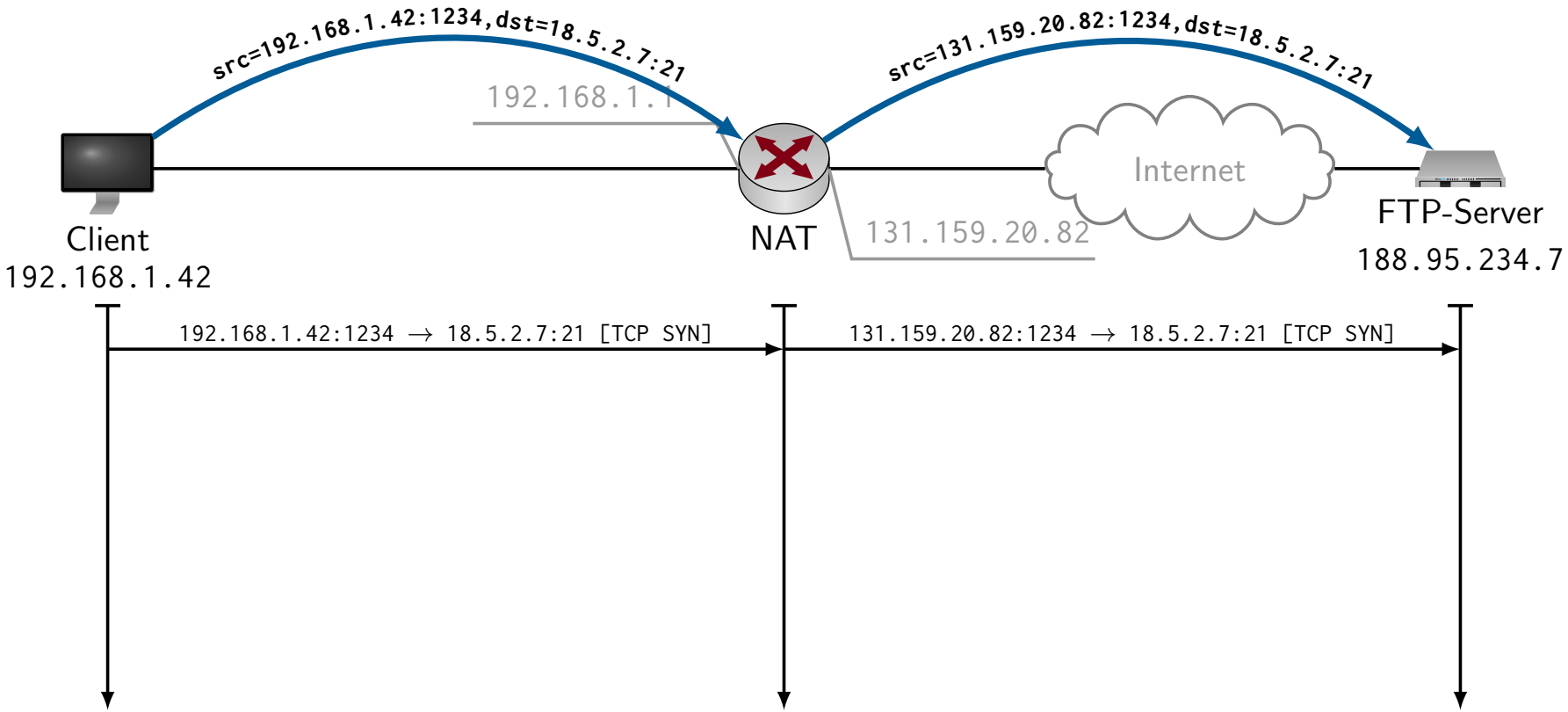
FTP und NAT

Was ist das Problem mit FTP active mode und NAT?



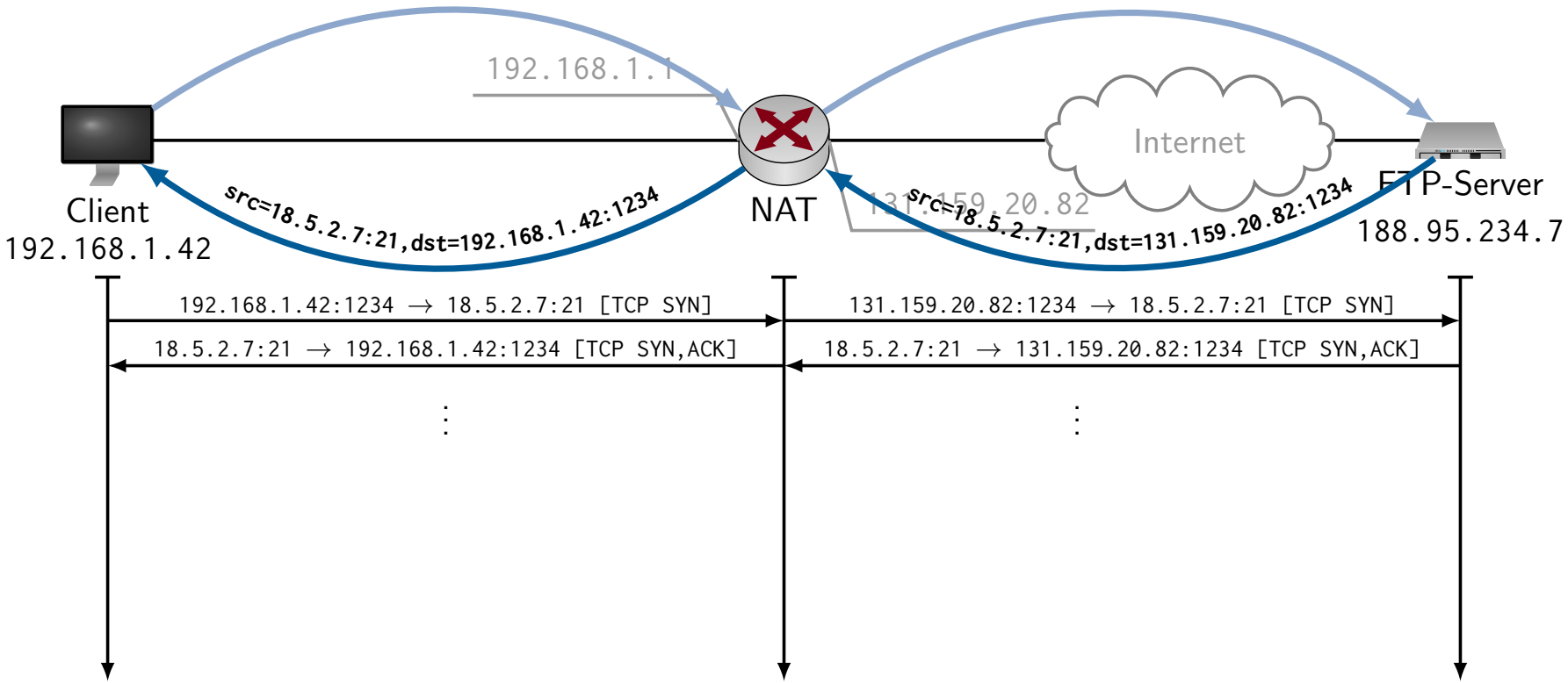
FTP und NAT

Was ist das Problem mit FTP active mode und NAT?



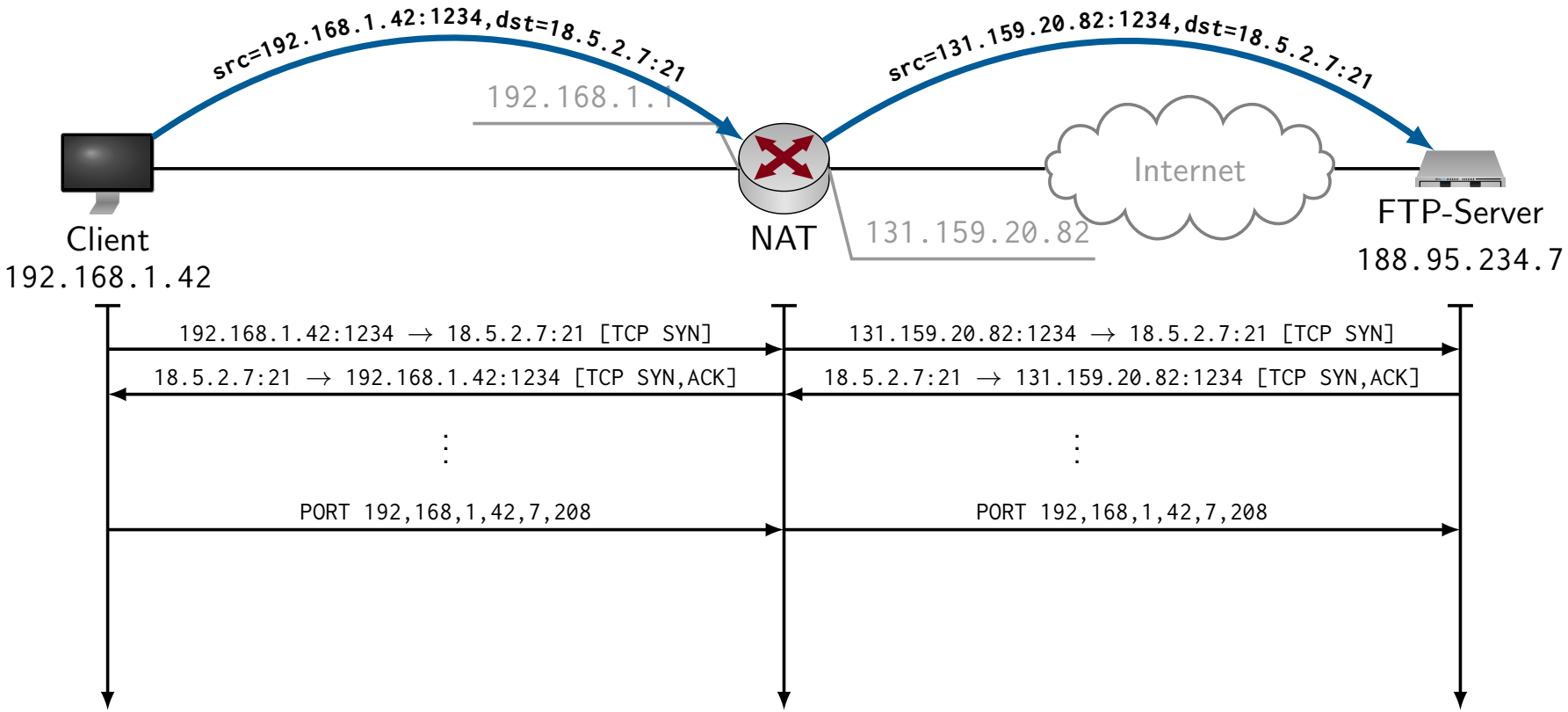
FTP und NAT

Was ist das Problem mit FTP active mode und NAT?



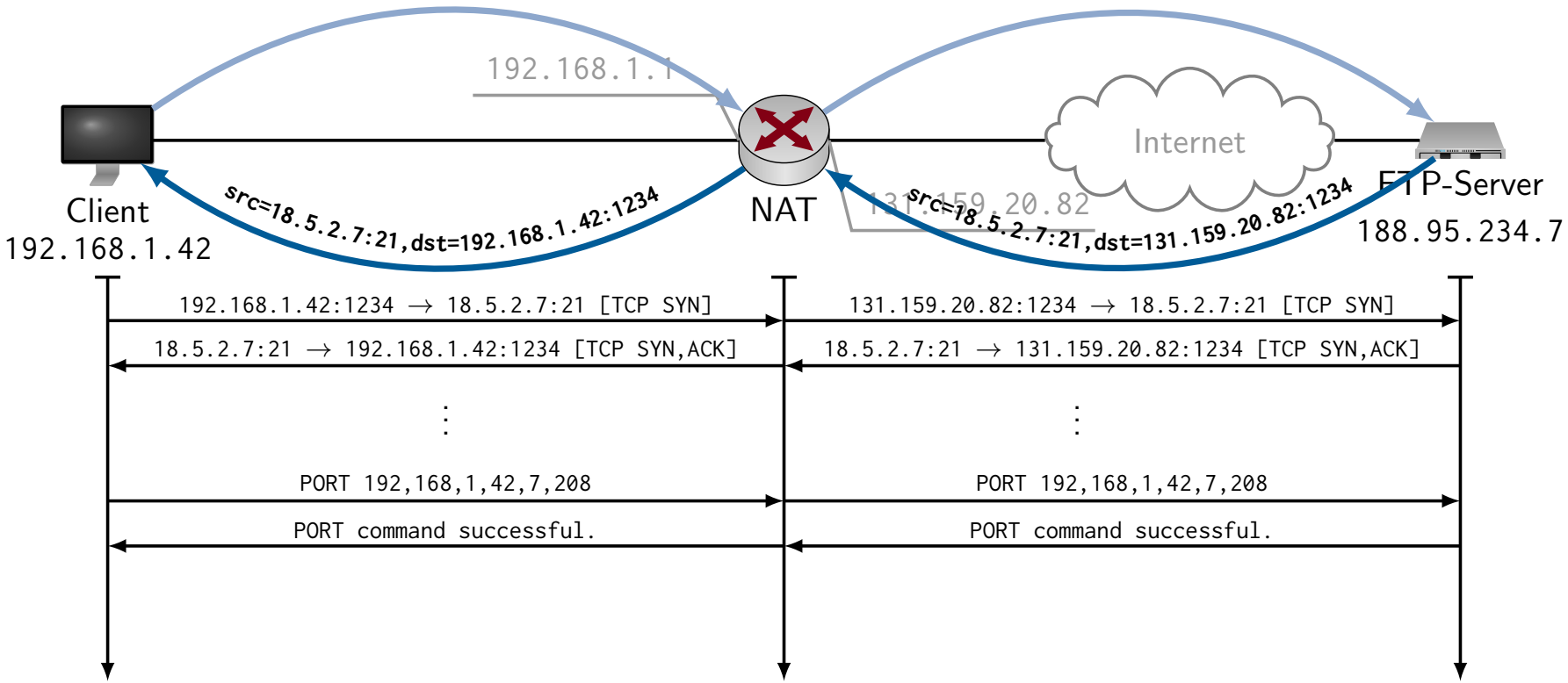
FTP und NAT

Was ist das Problem mit FTP active mode und NAT?



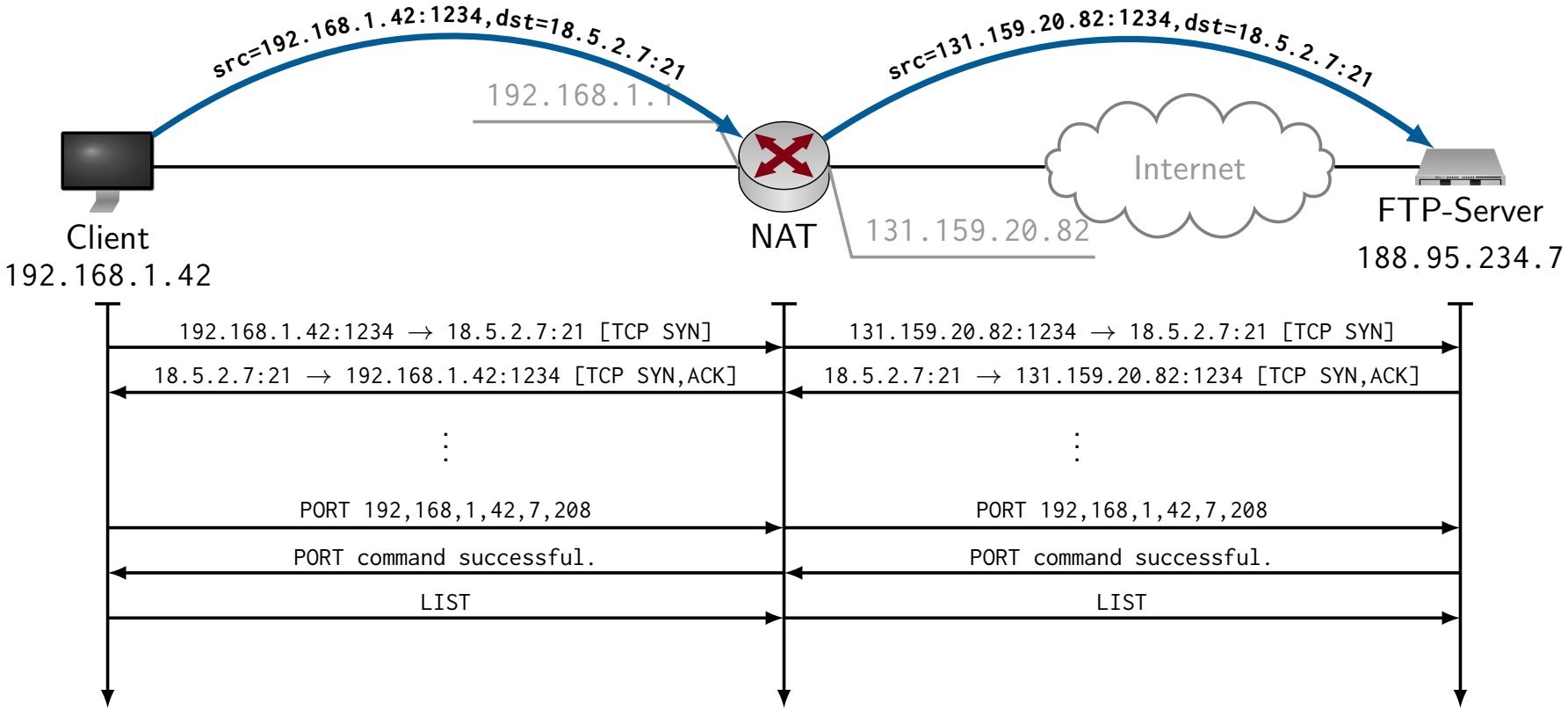
FTP und NAT

Was ist das Problem mit FTP active mode und NAT?



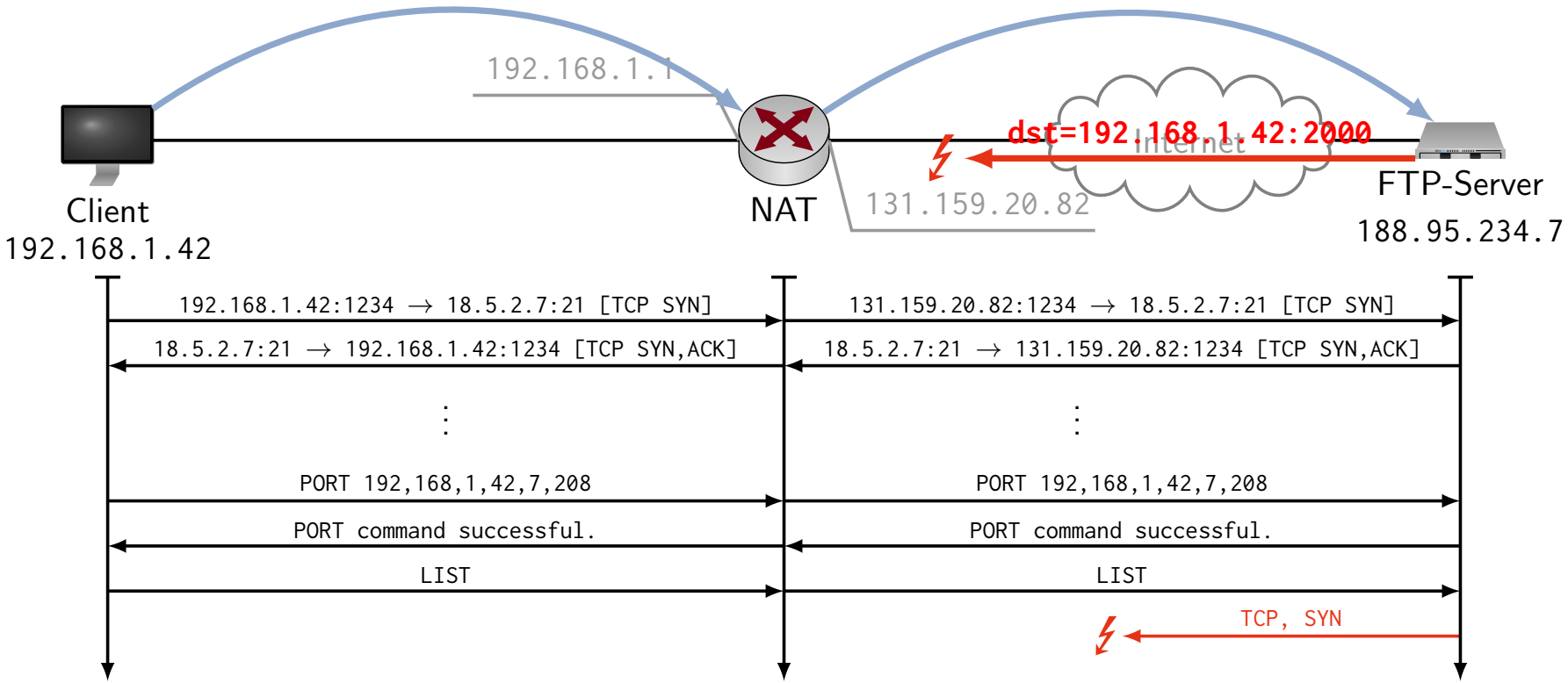
FTP und NAT

Was ist das Problem mit FTP active mode und NAT?



FTP und NAT

Was ist das Problem mit FTP active mode und NAT?

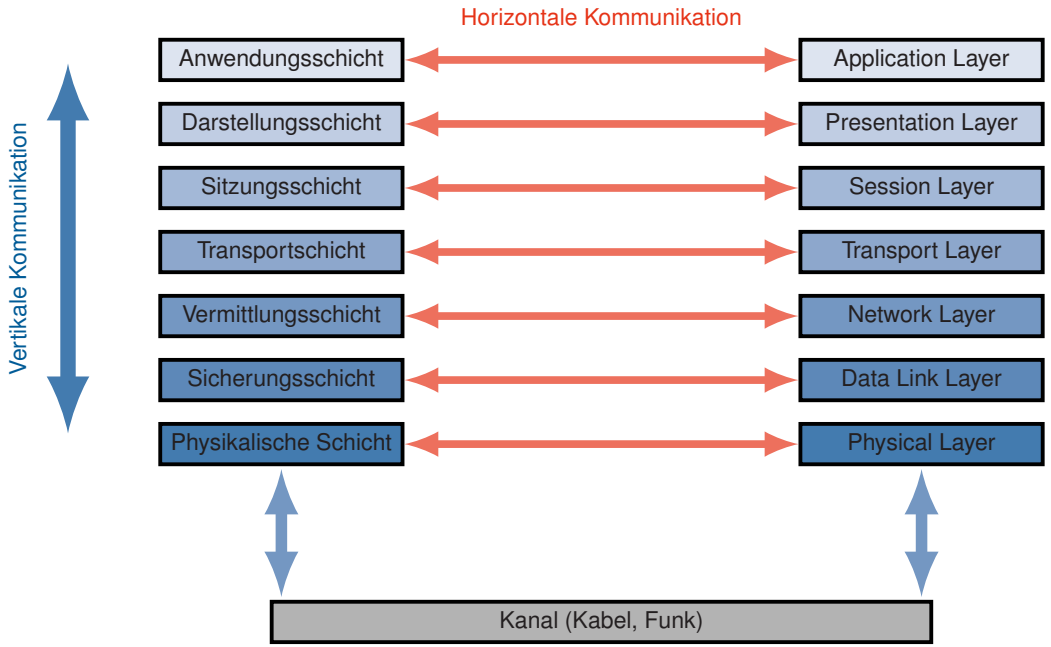


- Der Server versucht infolge des `LIST`-Kommandos eine Verbindung zu `192.168.1.42:2000` – wie zuvor über das `PORT`-Kommando ausgehandelt – aufzubauen.
- Das schlägt allein schon wegen der privaten IP-Adresse fehl.
- Selbst Wenn die Adresse öffenltich erreichbar wäre, hätte das NAT keinen passenden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

Lösungen:

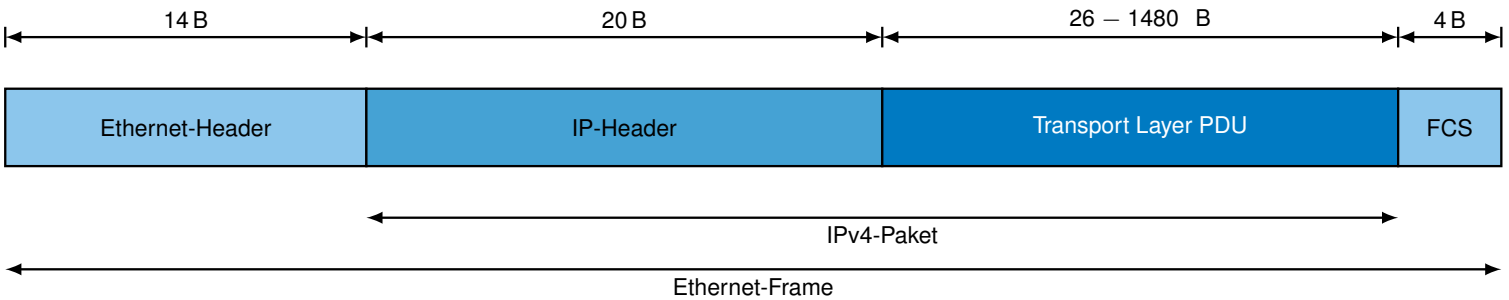
1. Die NAT-Implementierung wird so erweitert, dass sie FTP unerstützt.
   - NAT müsste die die L7-PDU auf das Vorhandensein von `PORT` prüfen und bei Auftreten sowohl
   - die private IP-Adresse durch eine öffentlich gültige ersetzen als auch
   - einen Eintrag in der NAT-Tabelle für den entsprechenden 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 Portnummern erlauben.

Zusammenfassung: ISO/OSI Modell



- Das ISO/OSI Modell unterteilt den Kommunikationsvorgang in 7 Schichten
- Es spezifiziert, welche Dienste in den verschiedenen Schichten zu erbringen sind
- Es wird jedoch keine Implementation vorgegeben
- Die n-te Schicht des Quellsystems kommuniziert nie direkt mit der n-ten Schicht des Ziels
- Kommunikation erfolgt stets erst vertikal im Stack, dann horizontal im Kanal und abschließend wieder vertikal
- Das Internet ist über einen TCP/IP-Stack realisiert, welcher weniger Schichten umfasst

Zusammenfassung: ISO/OSI Modell



- Den Nutzdaten wird auf jeder Ebene ein entsprechender Header vorangestellt
- Die tatsächlich transportierten Daten beinhalten damit für jede aktive Schicht einen Header

Zusammenfassung: Schichten

Physikalische Schicht
Der von der ersten Schicht angebotene Dienst ist die Punkt-zu-Punkt-Übermittlung von reinen Bits in Form von physikalisch messbaren Signalen.

- Generierung von Signales aus Bits (Impulsformung), welche auf Zielseite wiedererkannt werden müssen (Detektion)
- Auftragen der Signale auf eine Trägerwelle (Modulation)
- Als Trägermedium werden i. d. R. elektromagnetische Wellen genutzt
- Teilweise ausgleichen der inhärenten Unzuverlässigkeit des Kanals mithilfe einer Kanalkodierung

Sicherungsschicht
Die Sicherungsschicht übernimmt die Abstraktion eines physischen Kanals auf eine logische Direktverbindung.

- Erkennen von Übertragungsfehlern, nach Möglichkeit Korrektur
- Verhindern einer Überforderung des Zielsystems bei der Kommunikation (Flusskontrolle)
- Regelung des Medienzugriffs auf ein gemeinsam genutztes Kommunikationssystem

Zusammenfassung: Schichten

Vermittlungsschicht
Die Vermittlungsschicht verbindet Systeme über beliebig viele Direktverbindungen hinweg.

- Ermöglichung von Kommunikation über Direktverbindungen und Subnetze
- Bereitstellung einer eindeutigen und logischen Adressierung von Hosts
- Bestimmung möglichst optimaler Vermittlungspfade zwischen kommunizierenden Hosts (Routing)

Transportschicht
Die Transportschicht realisiert eine Ende-zu-Ende-Kommunikation zwischen Prozessen auf unterschiedlichen Hosts.

- Flusskontrolle zur Verhinderung von Überlast beim Empfänger
- Staukontrolle zur Vermeidung von Überlastsituationen im Netz
- Multiplexing von Datenströmen verschiedener Anwendungen bzw. ihrer Instanzen
- Bereitstellung verbindungsloser bzw. -orientierter Transportmechanismen

Zusammenfassung: Schichten

Sitzungsschicht
Die Sitzungsschicht erlaubt die Etablierung eines gemeinsamen Kommunikationszustandes, der mehrere Verbindungen auf der Transportschicht umfassen kann.

- Stellt Synchronisationspunkte für die Kommunikation zur Verfügung
- Ermöglicht damit das Suspendieren und Wiederaufnehmen von Kommunikationen
- Realisiert einen Koordinationsmechanismus für Kommunikationspartner

Darstellungsschicht
Die Darstellungsschicht ermöglicht die Bereitstellung eines abstrakten Formats zur Repräsentation der übertragenen Daten.

- Abstraktion von anwendungsspezifischer Syntax
- Bereitstellung von Datenkompression
- Ver- und Entschlüsselung der Kommunikationsdaten
- Umkodierung der Kommunikationsdaten

Zusammenfassung: Schichten

Anwendungsschicht
Die Anwendungsschicht beinhaltet alle Protokolle, welche direkt mit Anwendungen interagieren und deren Datentransport realisieren.

- Stellen einen Dienst für den User zur Verfügung
- Besitzen kein gemeinsames Dienste-Interface, da ihr Einsatzzweck sehr unterschiedlich ist

Die durch das ISO/OSI Modell realisierten Abstraktionen ermöglichen es Prozessen auf verschiedenen Systemen, transparent über ein Netz zu kommunizieren, ohne sich mit dem eigentlichen Übermittlungsvorgang auseinandersetzen zu müssen.
Die Schichtarchitektur selbst erlaubt, dass die Technologien einzelner Segmente ausgetauscht werden können, ohne dass Änderungen am restlichen Stack vorgenommen werden müssen (z. B. kabelgebundene Kommunikation vs. WLAN).
Zusätzlich ist es möglich, Adapter zur Verfügung zu stellen, welche die gleichzeitige Nutzung verschiedener Technologien während ein und demselben Datenaustausch ermöglichen (z. B. WLAN Access Point).

Bibliography I

[1] The JSON Data Interchange Format, 2013. http://www.ecma-international.org/publications/standards/Ecma-404.htm.

[2] R. Braden. Requirements for Internet Hosts – Communication Layers, 1989. https://tools.ietf.org/html/rfc1122.

[3] T. Bray. The JavaScript Object Notation (JSON) Data Interchange Format, 2014. https://tools.ietf.org/html/rfc7159.

[4] R. Elz and R. Bush. Clarifications to the DNS Specifications, 1997. https://tools.ietf.org/html/rfc2181.

[5] S. Josefsson. The Base16, Base32, and base64 data encodings, 2006. https://tools.ietf.org/html/rfc4648.

[6] P. Moackapetris. Domain Names – Concepts and Facilities, 1987. https://tools.ietf.org/html/rfc1034.

[7] P. Moackapetris. Domain Names – Implementation and Specification, 1987. https://tools.ietf.org/html/rfc1035.

[8] J. Postel and J. Reynolds. File Transfer Protocol (FTP), 1985. https://tools.ietf.org/html/rfc959.