Vladimir Zhelezarov
Immatrikulationsnummer: .....
.....

Bachelorarbeit

Thema:

Konzept für einen selbsteinrichtenden aktiven Netzteilnehmer mit Serverdiensten ohne Benötigung einer eigenen IP-Adresse für IP-Netzwerke am Beispiel eines Kabelfernsehnetzes

Bachelor-Arbeit, vorgelegt zur Erlangung des Zeugnisses über die Bachelorprüfung im Studiengang Digital Engineering und Angewandte Informatik der AKAD Hochschule Stuttgart – staatlich anerkannt –.

Betreuender Dozent: .....

....., 08.05.2022

Inhaltsverzeichnis

Abbildungsverzeichnis

Programmlistings

Algorithmenverzeichnis

1 Einleitung

Das Internet wird als die weltweite Sammlung von verbundene Netze bezeichnet[1]. Das moderne Leben wäre ohne Internet undenkbar. Zahlreiche Rechner, Laptops und intelligente Geräte brauchen eine Verbindung mit dem weltweiten Netz um unseren Bedürfnisse zu dienen. Allein die Zahl der kleinen intelligenten Geräte, oftmals „Internet of Things“ genannt, wird erwartungsgemäß auf mehr als 25 Milliarden im Jahr 2030 steigen[2]. Es sind nicht nur die kleinen Gadgets, die eine zuverlässige Verbindung brauchen, sondern auch für die Geschäfte kritische Server und Dienste. Die Verluste für größere Firmen aufgrund Netzausfälle können durchaus im Bereich von Millionen Dollar per Stunde reichen[3]. Somit ist es von größter Bedeutung, dass die globale Netzwerke stabil und zuverlässig funktionieren. Für diese korrekte Funktion und Interoperabilität sorgen die Netzwerkregeln - viele davon sind normiert und fest vorgegeben in der Form von Standarte, manche sind nur in der Form von Konventionen. Es ist auf jeden Fall ausgeschlossen, dass ein neues Gerät am Netz teil nimmt, ohne diese Regeln zu beachten. Im besten Fall würde das neue Gerät von den anderen Teilnehmer ignoriert und im schlimmsten Fall würde es Störungen und vielleicht sogar Datenverlust verursachen.

Die Komplexität aller Anforderungen und Bedingungen um am Netzverkehr teilzunehmen kann für den unerfahrenen Benutzer unter Umständen zu viel sein. Um eine hohe Benutzerfreundlichkeit zu leisten, werden viele Software und Hardware-Produkte und Projekte als „Plug-and-Play“ konzipiert und angeboten. Vom Benutzer wird in diesem Fall kaum etwas mehr als der Anschluss oder das Einschalten vom Gerät erwartet. Oftmals arbeiten diese Lösungen nur für sich oder ist deren Einfluss über externen Hardware oder Software gering, gerade um bei dem „Plug-and-Play“ Konzept zu bleiben. Als Beispiele können hier IoT-Geräte, Entertainment-Geräte u.a. sein.

Auf der anderen Seite des Spektrums existieren interessante und nutzbare Projekte, unter deren Ziele auch die Beeinflussung von externen Geräte ist. Typische Beispiele sind Netzwerkfilter, Netzwerk-Server-Dienste, Firewalls, etc. Leider sind diese Projekte oftmals mit einer nicht trivialen Umstellung des kontrollierten Netzsegment verbunden. Dieser Punkt macht sie unattraktiv für viele Benutzer, die entweder komplett mit dem Projekt aufgeben, oder Drittstellen mit der Einführung des Projekts und seiner Wartung beauftragen. Gerade bei Projekte, die

[1] Vgl. Parziale et al. (2006), S.4
[2] Statista, Inc. (2021), Internetquelle
[3] Brown (2019), Internetquelle

viel mit private Benutzerdaten arbeiten, kann das ein empfindliches Thema sein.

Das Ziel dieser Arbeit ist die Vorstellung und die praktische Ausführung eines Konzepts für invasive Einmischungen und aktive Teilnahme am Netz ohne den Bedarf etwas an diesem Netz zu ändern und ohne Bedarf einer eigenen IP-Adresse zu bekommen. Das Konzept soll möglichst flexibel und vielseitig einsetzbar sein. Für die praktische Ausführung wird eine Implementierung vorgestellt, die als Netzeinmischung ein DNS-Filter benutzt. Passende Hardware wird ausgewählt und mit der entwickelten Software versehen. Nach dem Anschluss dieser Hardware werden alle Geräte dahinten vom besagten DNS-Filter geschützt. Es werden keine Änderungen bei den anderen Geräte oder am Router notwendig und die Präsenz der neuen Hardware wird nur nach einer gesonderten Suche bemerkbar sein. Der Benutzer soll nichts von der Funktion des neuen Geräts oder seines Netzwerks verstehen. Die Netzwerkdaten und selbst die Anschlussrichtung werden vom neuen Gerät selber berechnet. Als Einsatzumgebung wird die Arbeit ein Heimnetzwerk als Beispiel nehmen, das über Kabelanschluss am Internet angeschlossen ist.

Die Entwicklung des Konzepts wird zuerst mit einer Feststellung und Überlegungen über den Einsatzbedingungen anfangen. Die verfügbare Lösungen und deren Probleme werden in Bezug auf die gezielte Entwicklung geklärt. Das konkrete Konzept wird im Kontext der Benutzerbedürfnisse stufenweise aufgebaut. Dabei wird ein Maximum an Benutzerfreundlichkeit gesucht im Rahmen der Netzgegebenheiten.

Das Konzept wird auch als einer praktischen Implementierung verwirklicht, um die Ideen in realen Umgebungen zu testen. Besondere Beachtung werden auch alle Dienste verdienen, die für den Benutzer unbewusst die Funktion der Lösung gewährleisten, wie z.B. Aktualisierungen, Dynamic-DNS, etc. Auch wenn das Höchstpriorität die Benutzerfreundlichkeit ist, muss auch an den fortgeschrittene Benutzer gedacht werden. Diesen werden erweiterten Informationen zur Verfügung gestellt. Für die gute Lesbarkeit der Implementierung wird die Software geeignet organisiert und mit genügend Kommentare und Formatierung versehen. Für das Verständnis des Konzepts und ihrer Ausführung wird diese Arbeit adäquate Diagrammen, Flowcharts und textliche Erklärungen anbieten.

Die vorliegende Arbeit ist in vier Hauptteile organisiert. Im ersten Teil werden Netze, deren Aufbau und die dafür zuständige Netzwerkregeln diskutiert, um eine gute Grundlage für die weitere Diskussion anzubieten. Im zweiten Teil, der aus zwei Kapitel besteht, stellen wir die Gegebenheiten in der Nutzerumgebung fest und diskutieren die vorhandene Ansätze, die für ähnliche Probleme benutzt werden. Im nächsten Teil wird das tatsächliche Konzept, als eine geeignete Lösung

für das vorgestellte Problem, entwickelt. Diese Entwicklung folgt der Klärung der bestimmten Voraussetzungen der konkreten Umgebung, sowie auch der Anforderungen, die daran gestellt werden.

Alle Software und zugehörige Dateien sind im Dateienanhang dieser Arbeit zugefügt.

2 Grundlagen

2.1 IP-Netzwerke

Aufbau

Das Internet besteht aus einer enormen Menge an angeschlossenen Geräte. Für deren Organisation und um die Kommunikation zwischen diesen zu steuern werden sie in Teilnetze mit verschiedenen Größen konzipiert und angeschlossen. Größere Teilnetze, die eigenständig von verschiedenen Organisationen verwaltet sind, werden auch autonome Systeme genannt[1]. Diese Netze variieren nach Größe und Mächtigkeit und kooperieren miteinander um Internetverbindung an den teilnehmenden Geräte anzubieten[2]. Es entsteht eine logische Unterordnung, in der größere Netze Dienste für die kleinere anbieten. Die Anbindung eines kleineres Netzes an den weltweiten Internet entsteht durch Anschließen an einem autonomen System, welches aus diesem Grund auch Internetdienstanbieter genannt wird[3]. Die Netze können nach deren Größe in Subkategorien unterteilt werden[4]:

- Die Personalnetze bezeichnen ein Arbeitsplatz und können zum Beispiel die Verbindung eines Rechners mit seinen Peripheriegeräte sein;
- Die Lokalnetze (Local Area Network - LAN) bestehen aus mehrere Teilnehmer in einem Gebäude wie z.B. ein Büro;
- Ein Großstadtnetz (Metropolitan Area Network - MAN), wie der Name vermutet bezieht sich auf die Verbindungen innerhalb einer Stadt;
- Ein Weitverkehrnetzwerk (Wide Area Network - WAN) erstreckt sich über einem größeren geografischen Gebiet, wie etwa ein Land oder sogar ein Kontinent;
- Das Internet verbindet alle autonome Netze miteinander und hat somit eine weltweite Abdeckung.

Das Umfeld für diese Arbeit sind die lokale Netze (LAN). Charakteristisch für diese Netze ist die räumliche Abgrenzung im Rahmen eines einzelnen Gebäude wie ein Haus, ein Büro oder eine Fabrik[5]. Typisch werden in LANs Personalcomputer und elektronische Geräte angeschlossen, damit sie ihre Ressourcen teilen,

[1] Vgl. Mandl (2019), S.31
[2] Ebd.
[3] Vgl. Mandl (2019), S.32
[4] Vgl. Tanenbaum; Wetherall (2014), S.18
[5] Tanenbaum; Wetherall (2014), S.19

oder damit sie Informationen miteinander austauschen[1]. Ein LAN kann auch ein Unternehmensnetzwerk sein, unterliegt aber immer noch den selben Besonderheiten.

Übertragungsmedium

Die tatsächliche Verbindungen in einem LAN können über Kabel, sowie auch drahtlos über WLAN, Bluetooth oder ähnliches passieren. Kabel-gebundene Verbindungen können mithilfe von Kupferkabel oder auch mit optischen Leitungen ausgeführt werden. Die WLAN Technologie ist standardisiert unter dem Namen IEEE 802.11, was auch als WiFi bekannt ist[2]. Typisch für den WLAN-Aufbau ist die Anwesenheit eines zentralen Rechner, oftmals als ein eingebettetes System, der mit Antenne ausgestattet ist - der Wireless Router. Alle Verbindungen innerhalb dieses drahtloses Netzes verlaufen über dem Router und er kümmert sich standardmäßig um Authentifizierung, Sicherheit, Vergabe von korrekten Einstellungen für die verbundene Geräte usw. Die Verbindung mit der Außenwelt (das Internet) verläuft ausschließlich über dem Router. Ganz ähnlich funktioniert es auch bei kabelgebundene LANs, mit dem Unterschied, dass kein Radio-Signal für die Verbindungen benutzt wird. Die typische Verbindungstechnologie, die in solchen Netzen benutzt wird, ist unter dem Namen *Ethernet* standardisiert[3].

Daten-Engpässe

Bei der so gezeigten Unterteilung der Netze entstehen bestimmte Strecken, oder Punkte, die sich um den ganzen Datenverkehr zwischen mehrere Endstellen kümmern. Ein typisches Beispiel dieser Netzstrecken von größter Bedeutung sind die Standleitungen, die Städte, oder sogar Kontinente verbinden - die s.g. Backbones[4]. Die Existenz solcher Strecken und Punkte ist von besonderer Bedeutung für die Zwecke dieser Arbeit, aufgrund der Tatsache, dass dadurch die Netzkommunikation mehreren Teilnehmer durchläuft und dabei eine Beobachtung oder sogar auch Kontrolle der Kommunikation ermöglicht wird.

Die Referenzmodelle

Wenn ein Netzwerkpaket für ein Empfänger innerhalb des Netzwerkes eines Internetanbieters bestimmt ist, wird dieses direkt intern geleitet. Wenn das nicht

[1] Ebd.
[2] Ebd.
[3] Vgl. Tanenbaum; Wetherall (2014), S.20
[4] Vgl. Mandl (2019), S.32

der Fall ist, dann wird das Paket über einem Backbone zu dem nächsten autonomen Netzwerk gesendet. Die Entscheidung, ob und wie das Paket weitergeleitet wird, ist ein Teil der komplexen Regel, die darum sorgen, dass das Internet so funktioniert, wie wir es erwarten. Das Zusammenspiel aller Geräte, die am globalen Internet angeschlossen werden, ist keine triviale Aufgabe. Die Geräte und deren lokale Netze sind unterschiedlich in deren Aufbau und Zwecke. Die Teilnehmer sind an stark heterogene Netze, oftmals geographisch sehr weit voneinander, angeschlossen. Klare Regeln müssen dafür sorgen, dass das Ganze nicht ins Chaos gerät. Solche Regeln stellen die Netzwerkstandards dar. Diese Standards sollen die erwünschte Interoperabilität zwischen den Netzwerkteilnehmer gewährleisten[1]. Aus historischer Sicht, wurden diese nicht immer von einem Gremium verabschiedet, sondern entstehen manchmal in der Praxis als Konventionen, die von den Industrie anerkannt sind[2]. Die drei in der Praxis bedeutendste Standards sind das TCP/IP-Referenzmodell, das OSI-Referenzmodell und das hybride Referenzmodell[3].

Gemeinsam haben alle Standards die Organisation der Netzwerke in Schichten[4]. Diese Schichten bauen aufeinander auf. Jede Schicht hat klar definierte Dienste und Schnittstellen, die der nächsten Schicht angeboten werden. Somit müssen sich höhere Schichten nicht um Einzelheiten der Implementierung bei der unteren Schichten kümmern, wodurch eine Entkoppelung entsteht. Das Konzept ähnelt der Objekt-orientierten Programmierung indem Verbergen von Informationen, Abstraktion und Datenverkapselung gezielt wird[5]. Die Regeln, wie eine Schicht mit der anderen kommuniziert, bilden die Netzwerkprotokolle. Ein Protokoll ist praktisch eine Vereinbarung, wie die Kommunikation durchlaufen soll[6]. Durch die Entkoppelung der Schichten wird erreicht, dass Schichten vom selben Level logisch gesehen direkt miteinander kommunizieren, wobei die physikalische Kommunikation erst alle tiefer liegenden Schichten überqueren muss. Die technische Ausführung dieses Konzept basiert auf dem Zufügen und Entfernen von zusätzlichen Informationen (Header und evtl. auch Trailer), die der Nachricht von jeder Schicht zugefügt oder gegebenenfalls entfernt werden. Jeder Header oder Trailer enthält Information für die entsprechende Schicht, die dadurch den weiteren Umgang mit der Nachricht entscheidet[7]. Eine grafische Darstellung der

[1] Vgl. Tanenbaum; Wetherall (2014), S.76
[2] Ebd.
[3] Baun (2020), S.35
[4] Vgl. Tanenbaum; Wetherall (2014), S.29
[5] Ebd.
[6] Ebd.
[7] Baun (2020), S.41

Schichten-Konzept und der Unterschied zwischen der logischen und physikalischen Kommunikation ist auf der Abbildung 1 auf der nächsten Seite dargestellt.

Betrachten wir im Weiteren die drei Referenzmodelle.

- TCP/IP-Referenzmodell

Das TCP/IP-Referenzmodell wurde ab 1970 vom Department of Defense (DoD) der USA entwickelt[1]. Es ist auch unter dem Namen DoD-Schichtenmodell bekannt. Die Aufgaben der Kommunikation wurden in vier Schichten unterteilt - Netzzugang-, Internet-, Transport- und Anwendungsschicht.

- OSI-Referenzmodell

Das OSI-Referenzmodell ist der einzige formale Standard von den drei Modelle. Es wurde von der Internationalen Standardorganisation (International Standards Organization - ISO) erstellt. Diese Organisation besteht aus den Standardorganisationen der 157 Mitglied-Staaten und ist somit weltweit anerkannt. Das OSI-Referenzmodell wurde im Jahr 1983 erstellt und im Jahr 1995 weiter überarbeitet[2]. Es ähnelt dem TCP/IP-Modell, allerdings hat es sieben Schichten. Die Netzzugangsschicht wird auf zwei weitere Unterschichten geteilt - die Bitübertragungs- und die Sicherungsschicht. Zusätzlich ist bei dem Modell die Anwendungsschicht in drei Schichten aufgeteilt - Sitzungs,- Darstellungs- und Anwendungsschicht.

- Hybrides Referenzmodell

In der Praxis hat sich gezeigt, dass eine zu detaillierte Aufteilung der obersten Schicht unnötig ist, weil die Funktionen dieser Schichten meistens von derselben Anwendung übernommen werden[3]. Zusätzlich haben die Bitübertragungs- und Sicherungsschicht sehr unterschiedliche Funktionen und sind dadurch besser als getrennte Schichten zu betrachten. Durch diesen Überlegungen entsteht das hybride Referenzmodell, das optimaler die Funktionsweise von Computernetzen abbilden soll. Es soll die Vorteile des TCP/IP-Referenzmodells und des OSI-Referenzmodells kombinieren, ohne deren jeweilige Nachteile zu übernehmen[4]. Die weitere Betrachtungen in dieser Arbeit basieren auf dem hybriden Modell. Weil der Fokus in der Arbeit auf den untersten Schichten liegt, überlappen sich die Bezeichnungen mit dem OSI-Modell.

[1] Baun (2020), S.36
[2] Tanenbaum; Wetherall (2014), S.41
[3] Vgl. Tanenbaum; Wetherall (2014), S.47
[4] Baun (2020), S.44

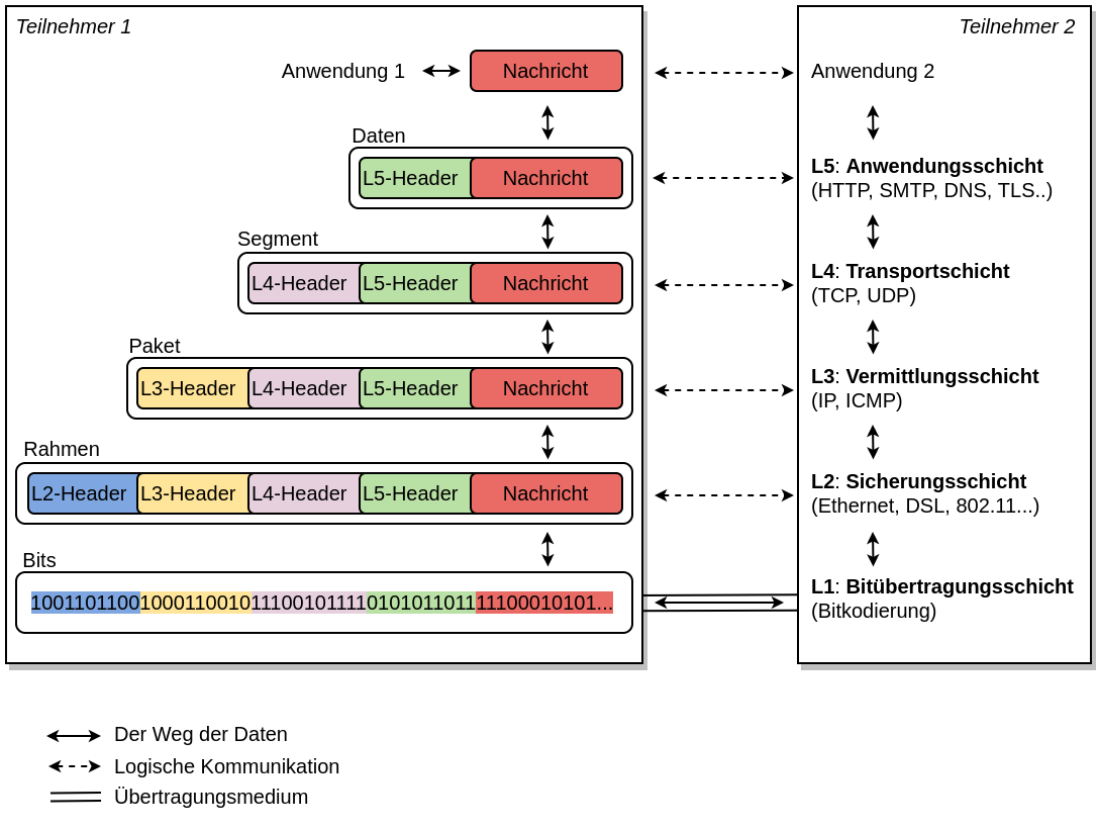

Abbildung 1: Referenzmodell mit Schichten

(In Anlehnung an Baun (2020), S.41 und Tanenbaum; Wetherall (2014), S.48)

Netzwerkschichten

Bis auf die Anwendungsschicht überlappen sich die Schichten des hybriden Referenzmodells mit den des OSI-Modells und sind wie folgt definiert:

Schicht 1: Bitübertragungsschicht Die unterste Schicht im Modell beschäftigt sich mit der physikalischen Übertragung von beliebigen Bitströme zwischen zwei Kommunikationspartner, die mittels einem Übertragungsmedium direkt verbunden sind[1]. Jedes Medium hat andere Charakteristika, die die Übertragungsgeschwindigkeit oder die Fehlerrate beeinflussen[2]. Zusätzlich sind auch wirtschaftliche Kriterien entscheidend ob und für welche Strecke ein bestimmtes Übertragungsmedium eingesetzt wird[3]. Grundsätzlich wird zwischen leitungsgebundene und drahtlose Medien unterschieden[4]. In der Praxis meist-benutzte leitungsgebundene Medien sind elektrische Leiter aus Kupfer, in der Form von

[1] Vgl. Luntovskyy; Gütter (2020), S.55-56
[2] Tanenbaum; Wetherall (2014), S.89
[3] Vgl. Tanenbaum; Wetherall (2014), S.104-105
[4] Baun (2020), S.64

verdrillte Kabel oder als Koaxialkabel, die elektrische Signale übertragen, sowie auch Lichtwellenleiter, wodurch die Information als Lichtimpulse gesendet wird. Drahtlose Medien sind z.B. Funk, Infrarot und Laser[1]. Das Konzept der Netzwerkinfrastruktur in einem Gebäude oder zwischen Gebäuden in einem Campus wird als „strukturierte Verkabelung“ bezeichnet und soll dazu sorgen eine zuverlässige Übertragung möglichst vieler Daten über möglichst weniger Übertragungsmedien erlauben[2].

Die Netzwerkgeräte, die in dieser Schicht arbeiten sind die Repeater und die Hubs, die praktisch Repeater mit mehrere Ports sind[3]. Diese dienen der physikalischen Kopplung zwei Netze und besitzen keine eigene Adressen oder weitere Funktionen. Deren einzige Funktion ist die Verstärkung der elektrischen Signale des Netzes[4]. Aus der Sicht der Netzwerkteilnehmer sind diese Geräte vollständig transparent.

Ein typisches Netz, womit wir uns auch in dieser Arbeit beschäftigen, besteht aus einem LAN mit strukturierter Verkabelung und eventuell ergänzende drahtlose WLANs, sowie eine Internetanbindung, die an sich auch leitungsgebunden oder drahtlos sein kann[5].

Die meist benutze Technologie für LANs ist IEEE 802.3, auch unter dem Namen „*Ethernet*“ bekannt[6]. Die Verkabelung bei Ethernet besteht aus verdrillte Kupferleitungen oder auch aus Lichtwellenleiter und ist für die verschiedene Ethernet-Variationen und unterstützte Geschwindigkeiten entsprechend ausgelegt[7]. Bei den ersten Ethernet-basierten LANs waren die Netzteilnehmer über einem einzelnen Kabel verbunden - das s.g. „klassisches Ethernet“[8]. Moderne Netze benutzen dagegen Switches, die für Punkt-zu-Punkt Verbindungen sorgen. Diese werden weiter unten diskutiert, weil sie auch in der nächsten Schicht arbeiten.

Die in Deutschland benutzte Internetanbindungen für Breitbandzugang sind über dem Telefonfestnetz, über einem Fernsehnetz oder auch über einer dedizierten Lichtwellenleitung[9]. Auch wenn die Lichtwellenleitung die besten Eigenschaften anbietet, weil sie extra für diese Zwecke entwickelt ist, ist die Verlegung neuer Kabel sehr kostenintensiv. Aus diesem Grund werden vorhandene Kommunikationskanäle benutzt, wie typischerweise die Telefonleitungen oder die Koaxialkabel

[1] Ebd.
[2] Baun (2020), S.75
[3] Vgl. Luntovskyy; Gütter (2020), S.240
[4] Ebd.
[5] Luntovskyy; Gütter (2020), S.144
[6] Tanenbaum; Wetherall (2014), S.20
[7] Vgl. Luntovskyy; Gütter (2020), S.156
[8] Tanenbaum; Wetherall (2014), S.21
[9] Vgl. Bundesministerium für Verkehr und digitale Infrastruktur (2021), S.7

der Fernsehanbieter[1].

Die zwei Varianten der Internetanbindung, die die vorhandene Infrastruktur der Telefonleitung oder des Fernsehnetzes benutzen, übertragen ihre Daten über analoge Trägerfrequenzen. Dafür werden über einem Modem die digitale Signale auf diese Trägerfrequenzen im Hochfrequenz aufmoduliert. Auf der Gegenseite der Verbindung kann ein anderes Modem die ursprüngliche Signale durch Demodulation zurückgewinnen[2]. Die Leistung der Telefonleitung ist durch die physikalische Gegebenheiten der verdrillte Kabel sehr begrenzt und ist zusätzlich stark von der Entfernung zum anderen Ende der Kommunikation - das Modem des Netzbetreibers - abhängig[3]. Als Vorteil kann die Punkt-zu-Punkt Verbindung genannt werden, die mehr oder weniger eine stabile Datenrate garantiert. Die Internetanbindung über dem Fernsehnetz bietet dagegen wesentlich mehr maximale Leistung an, die durch die bessere technische Eigenschaften des Koaxialkabels gegeben ist. Andererseits ist diese Leitung schwierig zu garantieren, weil das Kommunikationsmedium geteilt ist und somit die Leistung stark von der Anzahl der zur Zeit aktive Benutzer beeinflusst ist[4].

Die Lichtwellenleitung ist die moderne Lösung für Datenübertragung. Einige Netzbetreiber bieten eine Internetanbindung über Lichtwellenleitung an, die direkt am Haus angeschlossen wird[5]. Die Lichtwellenleiter bieten wesentlich mehr Bandbreite an, sie sind sehr schwierig abzuhören und sind auch unempfindlich gegen Störungen. Nachteilig ist, dass die benötigte Hardware teurer ist[6].

Drahtlose Internetanbindungen haben zur Zeit vergleichsweise weniger Bedeutung[7] und werden hier nicht weiter diskutiert. Drahtlose lokale Netze - die WLANs - haben dagegen eine sehr große Verbreitung[8], werden aber hier nicht weiter diskutiert, weil sie für die Zwecke dieser Arbeit irrelevant sind.

Schicht 2: Sicherungsschicht Die Bitübertragungsschicht alleine kann nicht eine sichere und korrekte Übertragung aller Daten gewährleisten. Störungen bei der Übertragung, Synchronisierung zwischen den Teilnehmer und weitere Probleme können zu einer Verfälschung der Information führen. Die Vermeidung, die Entdeckung und die Behebung dieser Probleme sind die Aufgaben der Siche-

[1]Vgl. Tanenbaum; Wetherall (2014), S.144-145
[2]Baun (2020), S.79
[3]Vgl. Tanenbaum; Wetherall (2014), S.148
[4]Vgl. Tanenbaum; Wetherall (2014), S.185-186
[5]Vgl. Bundesministerium für Verkehr und digitale Infrastruktur (2021), S.7
[6]Vgl. Tanenbaum; Wetherall (2014), S.105
[7]Bundesnetzagentur (2021), S.68
[8]Vgl. Luntovskyy; Gütter (2020), S.153

rungsschicht[1]:

- Die Sicherungsschicht bietet klar definiert Schnittstellen und Dienste für die obere Schicht (die Vermittlungsschicht) an;

- Sie kontrolliert und korrigiert eventuelle Kommunikationsfehler, die bei der Übertragung entstehen;

- Sie reguliert den Datenfluss um eine korrekte Kommunikation zwischen Teilnehmer mit verschiedene Geschwindigkeiten zu erlauben.

Grundsätzlich können Netzwerkverbindungen auf zwei Kategorien geteilt werden - solche die Punk-zu-Punkt Verbindungen benutzen und solche, die über Broadcast kommunizieren[2]. Typische Vertreter der Broadcast-Variante sind klassisches Ethernet oder das WLAN. In solche Netze sind mehrere Teilnehmer in der Lage, zum beliebigen Punkt das selbe Medium zu benutzen. Dadurch ist es wichtig eine Kontrolle über dieser Benutzung zu haben, um potenzielle Kollisionen zu vermeiden. Diese Zugriffskontrolle ist auch ein Teil der Aufgaben der Sicherungsschicht[3].

IEEE (Institute of Electrical and Electronics Engineers) unterscheidet die Zugangskontrolle als eine zusätzliche Unterschicht - die Schicht der Medienzugriffskontrolle (MAC - Media Access Control). Die Mechanismen der Fehlerkontrolle werden auch in einer separaten Unterschicht betrachtet - die Schicht der logischen Verbindungen (LLC - logical link control)[4].

Es existiert eine Vielfalt von Verfahren und Algorithmen, die sich darum kümmern, eine Kollision-freie Kommunikation in einem geteilten Medium zu gewährleisten. Diese werden in zwei Gruppen geteilt - deterministische Zugriffssteuerung und stochastische Zugriffssteuerung. Die deterministische Steuerung basiert auf klar definierte Regeln wer und wann das Medium benutzen darf. Bei der stochastischen Steuerung entscheidet dagegen jeder Teilnehmer selber wann er das Medium benutzen will. Kollisionen werden erkannt und durch geeignete Algorithmen umgegangen[5]. Moderne Ethernet LANs sind vollständig über Switches verbunden und dadurch frei von Kollisionen[6]. Solche geswitchte Netze sind die Zielumgebung dieser Arbeit, deshalb betrachten wir nicht weiter andere Netze oder deren Algorithmen für Kollisionsvermeidung.

Die schon erwähnte Switches sind Geräte, die auf der Sicherungsschicht und unter Umstände auch auf der nächsten Schicht - die Vermittlungsschicht, ar-

[1]Vgl. Tanenbaum; Wetherall (2014), S.193-194
[2]Vgl. Tanenbaum; Wetherall (2014), S.257
[3]Ebd.
[4]Microsoft Inc. (2021), Internetquelle
[5]Luntovskyy; Gütter (2020), S.27ff
[6]Vgl. Baun (2020), S.108

beiten. Dementsprechend werden diese Layer-2 und Layer-3 Switches genannt[1]. Neben den Funktionen eines Repeaters - Verstärkung der elektrischen Signale des Netzwerks - untersuchen die Switches auch die einzelne Pakete auf Prüfsummen und Korrektheit[2]. Switches achten auf die Hardware-Adressen in den Daten und leiten diese nur an ihren Ports weiter, wo der gezielte Empfänger angeschlossen ist[3]. Die einfachste Form eines Layer-2 Switches ist auch unter dem Namen „*Bridge*“ bekannt und dient typischerweise dazu, zwei Netzsegmente transparent zu verbinden[4]. Layer-3 Switches treffen zusätzlich Routing-Entscheidungen und können das lokale Netz in logische Unterbereiche trennen[5].

Alle Netzwerkgeräte haben eindeutige Hardware-Adressen (oder auch „MAC-Adressen“), die es erlauben die Geräte anzusprechen sowie auch Netzwerk-Pakete mit einer Sender-Adresse zu versehen. Diese Adressen sind Teil des Ethernet-Standards und werden von IEEE weltweit kontrolliert vergeben[6]. Die Adressen sind 48-Bit lang, indem die ersten 24 Bit von IEEE den Hardware-Herstellern zugeordnet werden. Die korrekte, eindeutige Einstellung der restlichen 24 Bits an jedem produzierten Netzwerkgerät ist Aufgabe des Herstellers. Es wird gezielt, dass weltweit keine zwei Netzwerkgeräte die gleiche Hardware-Adresse haben[7]. Wie wir weiter unten sehen ist es aber möglich diese Adressen per Software zu ändern, was für das Projekt von besonderen Bedeutung ist. Es muss beachtet werden, dass das Vorhandensein zwei gleichen Hardware-Adressen im selben Netz zu einer Kollision führen würde, weil die Netzteilnehmer nicht mehr eindeutig adressiert werden können. Außerhalb der Kommunikation zwischen direkt miteinander verbundenen Geräten spielen die Hardware-Adressen keine Rolle, weil dafür die Dienste der höheren Schichten benutzt werden[8].

Schicht 3: Vermittlungsschicht Die Aufgabe von der Vermittlungsschicht ist das Routing von Pakete vom Sender zum Empfänger[9]. Es ist nicht immer der Fall, dass Sender und Empfänger im selben Netz sind. Dafür ist es notwendig optimale Wege für die Kommunikation zu finden - eine nicht unbedingt leichte und eindeutige Aufgabe. Es kann z.B. nach dem schnellsten Weg gesucht wer-

[1] Baun (2020), S.22
[2] Ebd.
[3] Vgl. Tanenbaum; Wetherall (2014), S.289
[4] Vgl. Luntovskyy; Gütter (2020), S.242
[5] Vgl. Baun (2020), S.142
[6] Vgl. Tanenbaum; Wetherall (2014), S.282-283
[7] Ebd.
[8] Vgl. Tanenbaum; Wetherall (2014), S.355
[9] Tanenbaum; Wetherall (2014), S.362

den, oder nach dem billigsten[1]. Diese Entscheidungen werden von den Geräte getroffen, die in dieser Schicht arbeiten - die Router - basierend auf die Wünsche der Netzbetreiber sowie auch nach komplexe Algorithmen. Die Dienste der Vermittlungsschicht kommen in zwei Formen vor - verbindungsorientiert und verbindungslos[2]. Bei der verbindungsorientierten Kommunikation wird eine Route vor dem Anfang der tatsächlichen Kommunikation ausgerechnet und dann für die Dauer der Kommunikation erhalten. Die verbindungslose Kommunikation ist mit einer Berechnung der Route für jedes einzelnes Paket verbunden. Dabei können die Routen für die einzelne Pakete derselben Kommunikation unterschiedlich sein. Diese Variante ist robuster gegen Hardware- oder Streckenausfall, dafür muss aber die Routenberechnung bei jedem Paket neu stattfinden.

Die Vermittlungsschicht abstrahiert die untere Ebenen und somit können Netze verbunden werden, die unterschiedlich aufgebaut und nicht direkt miteinander kompatibel sind. Dieses Ziel wird durch die Benutzung von Adressen der Vermittlungsschicht erreicht[3]. Das Protokoll, dass diese Adressen beschreibt, ist das „Internet-Protokoll“, gekürzt IP. IP ist das wichtigste Protokoll der Vermittlungsschicht und kann als Basis des Internets betrachtet werden[4]. Das Protokoll sieht vor, dass jedes Gerät, das an einem Netzwerk angeschlossen ist, eine eigene IP-Adresse haben muss[5]. Aktuell existieren und werden benutzt zwei Versionen des Protokolls - die IP Version 4 und die IP Version 6. Fast ausschließlich wird immer noch die ältere Version des Protokoll benutzt[6], mit einer kompletten Umstellung auf IP Version 6 kann aber sicherlich gerechnet werden[7].

Der bedeutendste Unterschied zwischen den beiden IP-Versionen ist die Länge der unterstützen IP-Adressen - bei der Version 4 sind die Adressen 32 Bit[8] und bei Version 6 - 128 Bit[9]. Bei dem rasanten Wachstum der Zahl von Internet-fähigen Geräte in den letzten Jahre ist es dazu gekommen, dass keine IPv4 Adressen mehr verfügbar sind. Die neuere Version vom Protokoll - IPv6 - bietet neben der Möglichkeit von wesentlich mehr Adressen auch weitere Verbesserungen an, wie etwa eine leichtere Verarbeitung der Pakete aufgrund der vereinfachten Header sowie auch eine bessere Unterstützung für erweiterte Optionen[10].

[1] Tanenbaum; Wetherall (2014), S.431
[2] Tanenbaum; Wetherall (2014), S.358-362
[3] Vgl. Tanenbaum; Wetherall (2014), S.427
[4] Mandl (2019), S.41
[5] Mandl (2019), S.43
[6] Tanenbaum; Wetherall (2014), S.456
[7] Vgl. Tanenbaum; Wetherall (2014), S.465
[8] Tanenbaum; Wetherall (2014), S.439
[9] Tanenbaum; Wetherall (2014), S.456
[10] Tanenbaum; Wetherall (2014), S.457

Die Vergabe von IP-Adressen wird weltweit zentral von der Organisation IANA (Internet Assigned Number Authority) kontrolliert, die geografisch bedingt IP-Adressräume an regionale Organisationen vergibt - die Regional Internet Registries (RIR)[1]. Die Local Internet Registries (LIR) bekommen Adressräume von RIR und kümmern sich dann um die direkte Vergabe von Adressen an Endkunden. Diese Regelung hat das Ziel, das jedes Internet-verbundenes Gerät eine eindeutige Adresse hat. Das ähnelt der Regelung bei der Adressen der Sicherungsschicht, welche aber strenger ist. Wenn für die MAC-Adressen wichtig ist, dass sie immer eindeutig sind, besteht die Möglichkeit für Geräte, die nicht direkt mit dem Internet verbunden sind, eine nicht weltweit eindeutige IP-Adresse zu benutzen. Diese Adressen sind die s.g. private Adressen und sie dürfen nicht von Geräte, die direkt mit dem Internet verbunden sind, benutzt werden[2].

Das IP-Protokoll arbeitet verbindungslos, also es wird keine Garantie gegeben, dass die Pakete ordentlich, zeitnah oder überhaupt ankommen[3]. Es existieren Protokolle der Vermittlungsschicht, die verbindungsorientiert arbeiten. Ein typisches und bekanntes Beispiel ist das MPLS (MultiProtocol Label Switching) Protokoll[4]. Das Grundprinzip bei diesem Protokoll ist das Zufügen von zusätzliche Header (Labels), die leicht vom nächsten Router analysiert werden können[5]. Somit werden die Pakete im Rahmen der Router-zu-Router Kommunikation schnell verarbeitet. Dieses Protokoll hat wenige Bedeutung für unsere Ziele und der Fokus im Rest der Arbeit fällt auf dem IP-Protokoll.

Für die Arbeit und Weiterleitung von Pakete speichern Netzwerkgeräte die schon berechnete Routen in den s.g. Routing-Tabellen[6]. Es wäre sehr unpraktisch und besonders aufwendig, wenn jeder Router alle mögliche Routen kennen und speichern muss. Ein Teil der Lösung dieses Problems ist, dass die Router hierarchisch arbeiten und somit die Routing-Aufgaben untereinander teilen[7]. Ein anderes Aspekt ist die Aufteilung der Netze in Subnetze. Dazu werden die IP-Adressen in einem Netzwerk- und einem Hostanteil aufgeteilt[8]. Zusätzlich zu der IP-Adresse wird dazu auch die s.g. Netzmaske mitgeteilt. Praktisch gesehen ist das eine Bitmaske, die durch logisches *AND* mit der IP-Adresse summiert wird[9]. Aus dem Ergebnis versteht der Router die Größe des Subnetzes und kann alle

[1] Mandl (2019), S.42
[2] Tanenbaum; Wetherall (2014), S.452
[3] Vgl. Mandl (2019), S.41
[4] Vgl. Tanenbaum; Wetherall (2014), S.471
[5] Ebd.
[6] Mandl (2019), S.13
[7] Tanenbaum; Wetherall (2014), S.378-379
[8] Luntovskyy; Gütter (2020), S.77
[9] Tanenbaum; Wetherall (2014), S.445

dafür bestimmte IP-Adressen gleich verwalten[1].

Die Private Adressen sind, wie der Name vermutet, nur für lokale Netze bestimmt. Pakete mit solcher Adressen dürfen nicht über den Grenzen des lokalen Netzes weitergeleitet werden[2]. Dabei dürfen mehrere Geräte mit der selben privaten IP-Adressen existieren, solange sie nicht am selben Netz angeschlossen sind. Die Bereiche der privaten Adressen sind wie folgt[3]

- 10.0.0.0 mit einer Netzmaske von 255.0.0.0. Das ergibt den Bereich 10.x.x.x;
- 172.16.0.0 mit einer Netzmaske von 255.240.0.0. Das ergibt den Bereich 172.16.x.x – 172.31.x.x und
- 192.168.0.0 mit einer Netzmaske von 255.255.0.0. Das ergibt den Bereich 192.168.x.x.

Zusätzlich sind auch die folgende spezielle IP-Adressen vorgesehen[4]:

- 0.0.0.0 - bedeutet der Host selbst;
- 0.0...Host - spricht den Host *Host* im selben Netz an;
- 255.255.255.255 - für Broadcast im lokalen Netz;
- Netz.255.255.255 - für Broadcast im fremden Netz *Netz* und
- 127.... - reserviert für Loopback (Schleifenschaltung).

Für die korrekte Funktion der Internet-basierten Netzwerken sind einige zusätzliche Hilfsprotokolle und Ansätze von besonderen Bedeutung[5]:

ARP

Das Routing basiert auf die IP-Adressen und liefert Netzwerkpakete weltweit. Die gelieferte Pakete müssen dafür die ganze Strecke zum Ziel durchlaufen - von einem Netzwerkgerät zum nächsten. Das Durchlaufen der einzelnen Teilstrecken, die aus direkt miteinander verbundene Netzwerkgeräte bestehen, passiert durch Dienste aus der Sicherungsschicht. Die Geräte der Sicherungsschicht verstehen aber keine IP-Adressen und können deshalb nur aufgrund dieser Information die Pakete nicht zum nächsten Rechner liefern[6]. Es ist ein Mittel benötigt, um die

[1] Mandl (2019), S.42
[2] Mandl (2019), S.47
[3] Ebd.
[4] Tanenbaum; Wetherall (2014), S.451
[5] Mandl (2019), S.105
[6] Vgl. Tanenbaum; Wetherall (2014), S.467

IP-Adressen in MAC-Adressen zu übersetzen. Dieses Mittel ist das Hilfsprotokoll mit dem Namen ARP (Address Resolution Protocol), standardisiert in RFC826[1]. Der Standard sieht vor, dass jeder Netzwerkteilnehmer eine Tabelle/Cache pflegt, wo die entsprechende MAC-Adressen für die IP-Adressen der direkt erreichbaren Netzwerkgeräte gespeichert sind[2]. Wenn eine Adresse benötigt wird, die nicht in der Tabelle auffindbar ist, sendet der Host eine Broadcast-Nachricht im Ethernet, die auch die IP-Adresse des gesuchten Geräts beinhaltet. Wenn ein Gerät sich angesprochen füllt, antwortet er auf die Nachricht und somit teilt seine MAC-Adresse mit[3]. Die Geräte, die die Nachrichten mitbekommen, können die Information auch nutzen, indem sie ihre eigene ARP-Tabellen aktualisieren[4]. Der entsprechende Dienst für IPv6 Netze ist das NDP (Neighbor Discovery Protocol), dessen Funktionalität identisch ist[5].

NAT

Der Mangel an IPv4 Adressen hat dazu geführt, dass neue Ansätze dringend gefragt sind um das Problem umzugehen, bis die neue IPv6-Adressen überall in Benutzung kommen. Eine mögliche Option ist die Vergabe von den s.g. „dynamischen“ IP Adressen. Diese werden nur benutzt, solange der Netzwerkteilnehmer online ist und danach wieder zur Verfügung für andere Benutzer gestellt[6]. Auch wenn dieser Ansatz etwas Flexibilität für private Kunden anbietet, ist er weniger für Geschäftskunden geeignet, die dauerhaft online sein möchten.

Ein anderer, mehr robuster Ansatz ist die Gruppierung mehreren Kundenrechner aus dem selben Netz hinter einer einzigen IP-Adresse, die von außen sichtbar ist[7]. Weil die beteiligte Rechner für ihre Netzwerkfunktionalitäten immer noch eine eigene IP-Adresse benötigen, werden solche aus dem Privatadressenraum zugewiesen. Diese Vorgehensweise benötigt ein Netzwerkgerät, der an der Grenze des Lokalnetzes angeschlossen wird und die Übersetzung zwischen lokale, private Adressen und die äußere IP-Adresse übernimmt. Aus der Funktion kommt auch der Name des Ansatzes - NAT (Network Address Translation). NAT unterliegt dem Standard RFC3022. Für eine einfache Übersetzung, die nur einen Host betrifft, werden im NAT-Gerät nur die IP-Adressen umgeschrieben. Diese Form bezeichnet der Standard als „*Basic NAT*“[8]. In der Praxis ist sie von weniger

[1] Ebd.
[2] Network Working Group (1982), S.1
[3] Vgl. Baun (2020), S.138
[4] Vgl. Tanenbaum; Wetherall (2014), S.469
[5] Baun (2020), S.109
[6] Tanenbaum; Wetherall (2014), S.452
[7] Ebd.
[8] Network Working Group (2001), S.3-4

Bedeutung, weil sie nicht mit mehrere Rechner umgehen kann. Dafür wird eine andere Form von NAT vorgesehen - NAPT (Network Adsress Port Translation)[1]. Das NAPT-Gerät pflegt zusätzlich in seinem Speicher auch Port-Nummer. Die Port-Nummer, zusammen mit den IP-Adressen lösen das Problem der Adressierung hinter der NAT. Die Ports sind ein Konzept aus der nächsthöheren Schicht und der Zugriff darauf ist ein Verstoß gegen dem Schichtenmodell, wird allerdings benutzt als die aktuell einzige praktische Lösung für den IP-Adressenmangel[2].

ICMP

Das Internet Control Message Protocol (ICMP) dient der Beobachtung und Kontrolle von Verbindungen. Die Router benutzen auch das Protokoll um miteinander zu kommunizieren[3].

DHCP

Die schon betrachtete Protokolle setzen voraus, dass die Hosts ihre IP-Adressen und Netzwerkdaten kennen. Grundsätzlich existieren zwei Möglichkeiten für ein netzwerkfähiges Gerät diese Daten zu bekommen[4]:

- Die IP-Adresse und alle relevante Information können auf dem Host gespeichert werden. Das benötigt manuelle Einstellung und ist nicht immer praktisch;
- Die andere, mehr flexible Variante ist die Benutzung des DHCP Protokolls. DHCP (Dynamic Host Configuration Protocol) setzt das Vorhandensein eines DHCP-Servers im Lokalnetz voraus, der auf Anfrage der Teilnehmer mit den richtigen Einstellungen antwortet. Die Netzwerkgeräte können diese übernehmen und dann in der IP-Kommunikation teilnehmen.

Schicht 4: Transportschicht Das Ziel der Netzwerkkommunikation ist der Datenaustausch zwischen Anwendungen. Diese Brauchen dazu eine Schnittstelle zu den unteren Schichten des Netzwerkmodells. Auch wenn eine direkte Benutzung der Vermittlungsschicht theoretisch möglich ist, bringt sie folgende Probleme mit sich[5]:

[1] Network Working Group (2001), S.5-7
[2] Tanenbaum; Wetherall (2014), S.454-455
[3] Tanenbaum; Wetherall (2014), S.465
[4] Tanenbaum; Wetherall (2014), S.470
[5] Tanenbaum; Wetherall (2014), S.496-497

- Die Vermittlungsschicht beinhaltet alle Router und Netzwerkgeräte, die auf dem Weg zum Ziel passiert werden. Somit haben die Anwendungen keine Kontrolle und können dadurch keine Garantie haben, dass die Pakete korrekt und überhaupt geliefert werden;

- Anderes Problem entsteht bei Änderungen in der Vermittlungsschicht - es ist viel einfacher und sicherer für die Anwendungsprogrammierer, wenn sie gegen einer fest-definierten Schnittstelle ihre Software schreiben, die die Besonderheiten der unterliegenden Schicht abstrahiert.

Somit ist die Aufgabe der Transportschicht eine Bereitstellung von Protokolle für Ende-zu-Ende Interprozesskommunikation[1].

Ein zusätzliches Aspekt der Interprozesskommunikation, was wir bisher nicht betrachtet haben, ist dass zu jedem Zeitpunkt mehrere Anwendungen eine Kommunikation aufbauen oder weiterführen wollen. Wenn diese Kommunikation nur auf der IP-Adresse der Maschine basiert, ist die Anzahl an Prozesse auf der Anzahl von den zugewiesenen IP-Adressen begrenzt. Dieses Problem wird in der Transportschicht gelöst, indem der Kommunikation zusätzliche Information zugefügt wird. Als Schnittstelle für die Anwendungen bietet sie die sogenannte *Sockets* an[2]. Ein Socket ist ein Tupel aus der IP-Adresse der Maschine und einer Portnummer. Der Port repräsentiert einen Speicherbereich für die Nachrichtenübergabe[3].

Die beiden am häufigsten verwendeten Transportprotokolle sind UDP und TCP[4]:

- UDP (User Datagram Protocol) arbeitet verbindungslos und ist praktisch eine Erweiterung für das IP-Protokoll mit der Möglichkeit mehrere Prozesse parallel zu bedienen[5]. UDP ist bestens für Anwendungen gedacht, die keine genaue Kontrolle über Timing, Fehlerkontrolle und Netzwerkpaketfluss benötigen[6]. UDP wird typischerweise bei der Sendung kleiner Mengen von Daten benutzt, oder auch bei Echtzeitanwendungen, wo der Verlust einzelnen Pakete nicht so wichtig ist[7]. UDP unterliegt dem RFC 768 Standard;

- TCP (Transmission Control Protocol) ist das Hauptprotokoll für die Sendung von Daten im Internet[8]. Es arbeitet verbindungsorientiert und somit

[1] Baun (2020), S.183
[2] Luntovskyy; Gütter (2020), S.87
[3] Ebd.
[4] Baun (2020), S.184
[5] Luntovskyy; Gütter (2020), S.88
[6] Tanenbaum; Wetherall (2014), S.543
[7] Parziale et al. (2006), S.146
[8] Tanenbaum; Wetherall (2014), S.543

garantiert eine sichere und korrekte Lieferung der Pakete. Das Protokoll benutzt Ports für die Angabe der angesprochenen Anwendung. TCP ist wesentlich komplexer als UDP und beinhaltet zusätzliche Informationen, wie z.B. eine Sequenznummer, die für den ordentlichen Wiederaufbau der angekommenen Pakete benutzt wird, oder eine Bestätigungsnummer, wodurch den Empfang der Pakete bei dem Sender bestätigt wird[1]. TCP besitzt auch viele andere Funktionen, die wir hier nicht weiter diskutieren. TCP unterliegt RFC 793 und weitere Standards.

Schicht 5: Anwendungsschicht Diese Schicht, wie der Name vermutet, beinhaltet die Anwendungen. Typische Beispiele von Protokolle und Anwendungen dieser Schicht sind die folgende:

- Das Domain Name System (DNS) ist ein Protokoll für die Übersetzung von symbolischen Namen (Domainnamen) zu IP-Adressen[2]. Die DNS-Information ist hierarchisch organisiert, in separate Teile gegliedert und im gesamten Internet auf Nameservern verteilt[3]. Das DNS-Protokoll benutzt UDP auf Port 53[4];

- Das Hypertext Transfer Protocol (HTTP) ist ein Protokoll für die Übertragung von Webseiten-Daten, meistens durch die Benutzung von TCP als Transport-Protokoll[5];

- Die Transport Layer Security (TLS), aktuell in der Version 1.3 ist ein in RFC 8446 standardisiertes Protokoll, das einen sicheren Tunnel zwischen zwei Kommunikationsteilnehmer herstellt, mit dem Ziel Authentifizierung, Vertraulichkeit und Integrität der Daten anzubieten[6];

- Die Secure Shell (SSH), standardisiert in RFC 4251, ermöglicht den Zugriff und die Steuerung von entfernten Rechner über einem verschlüsselten Tunnel[7];

- Virtual Private Network (VPN) ist ein Ansatz für die logische Verbindung entfernten Netzwerke über einem privaten, verschlüsselten Tunnel[8]. Aus

[1]Tanenbaum; Wetherall (2014), S.558
[2]Mandl (2019), S.207
[3]Ebd.
[4]Mandl (2019), S.209
[5]Vgl. Mandl (2019), S.216-217
[6]Internet Engineering Task Force (2018), S.5
[7]Group (2006), S.2
[8]Parziale et al. (2006), S.862

technischer Sicht kann das VPN auf den Schichten 2, 3 oder 4 aufgebaut werden[1].

2.2 Modifikation der Felder eines Netzwerkpakets

Ein mittels Wireshark[2] abgefangenes und analysiertes Paket aus einem Realwelt-Netzwerk ist auf Abbildung 2 auf der nächsten Seite dargestellt. Dabei handelt sich um einen Kleinteil einer TLS-Kommunikation. Weil die tatsächliche Daten, die transportiert werden, verschlüsselt sind, können wir nur raten, zu welcher Anwendung diese Kommunikation gehört. Dafür sind aber alle andere Elemente der Kommunikation wie Hardware- und IP-Adressen, Ports und Felder leicht lesbar. Diese sind genau die Felder, die für unsere Ziele wichtig sind. Auf der Abbildung sind sie ausgelassen und durch XXX../YYY.. ersetzt. Das sind die Felder, die vom jeden Netzwerkgerät auf dem Weg zum Ziel gelesen werden, um Entscheidungen über der weiteren Lieferung des Pakets zu treffen. Für die korrekte Lieferung der Pakete ist es auch manchmal notwendig, dass diese Paketfelder geändert werden. Das Paradenbeispiel ist der schon im vorigen Kapitel vorgestellten Ansatz von NAT. Das NAT-Gerät ändert die Felder IP Adresse und Port Nummer. Eine detaillierte Darstellung eines NAT-Netzwerks unter Benutzung konkreter Werte ist auf Abbildung 3 auf Seite 22 zu sehen.

Eine willkürliche Änderung der Felder würde sicherlich nicht zu einem sinnvollen Ergebnis führen. Andersherum, mit etwas Kenntnis über der Funktion des Netzwerks und den allgemeinen Regeln, können wir gezielt Pakete ändern und nach unseren Wünsche steuern. Diese werden auch von den Zwischenstationen anerkannt und geliefert, weil sie den selben Regeln folgen.

[1] Baun (2020), S.225-226
[2] Sharpe et al. (o.J.), Internetquelle

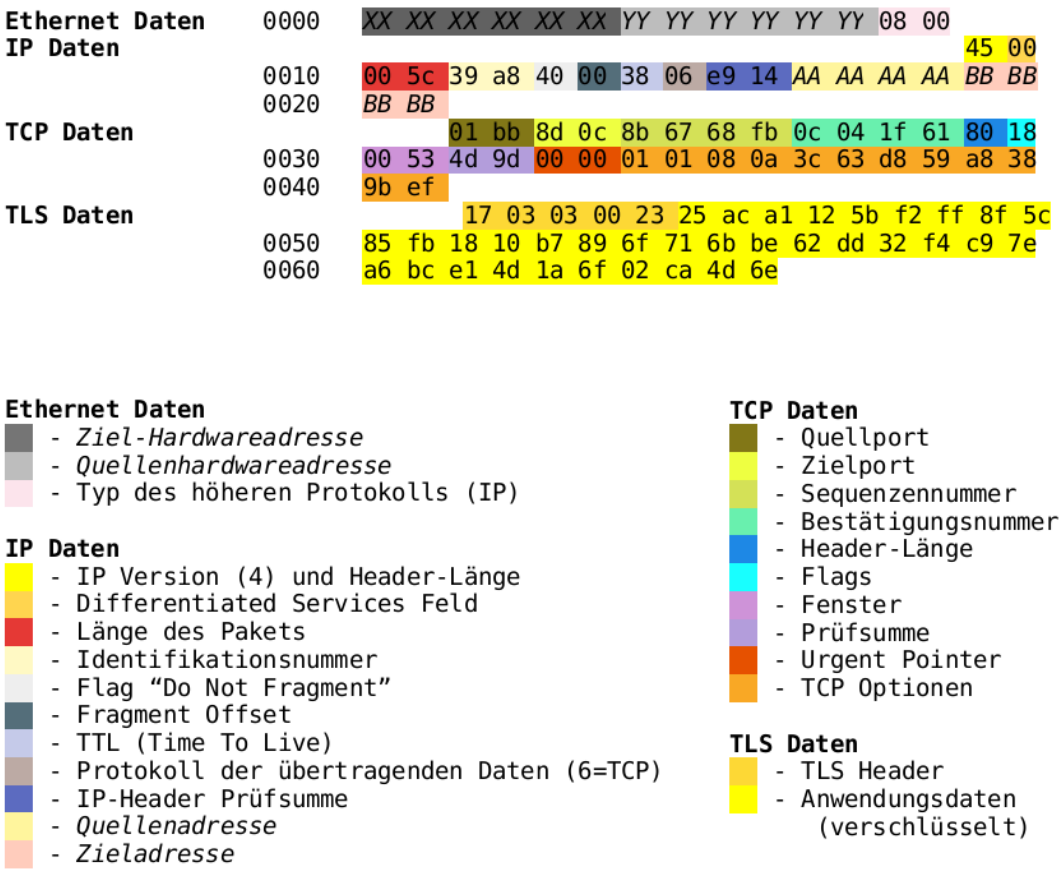

Abbildung 2: Die Zusammensetzung eines TLS-Pakets

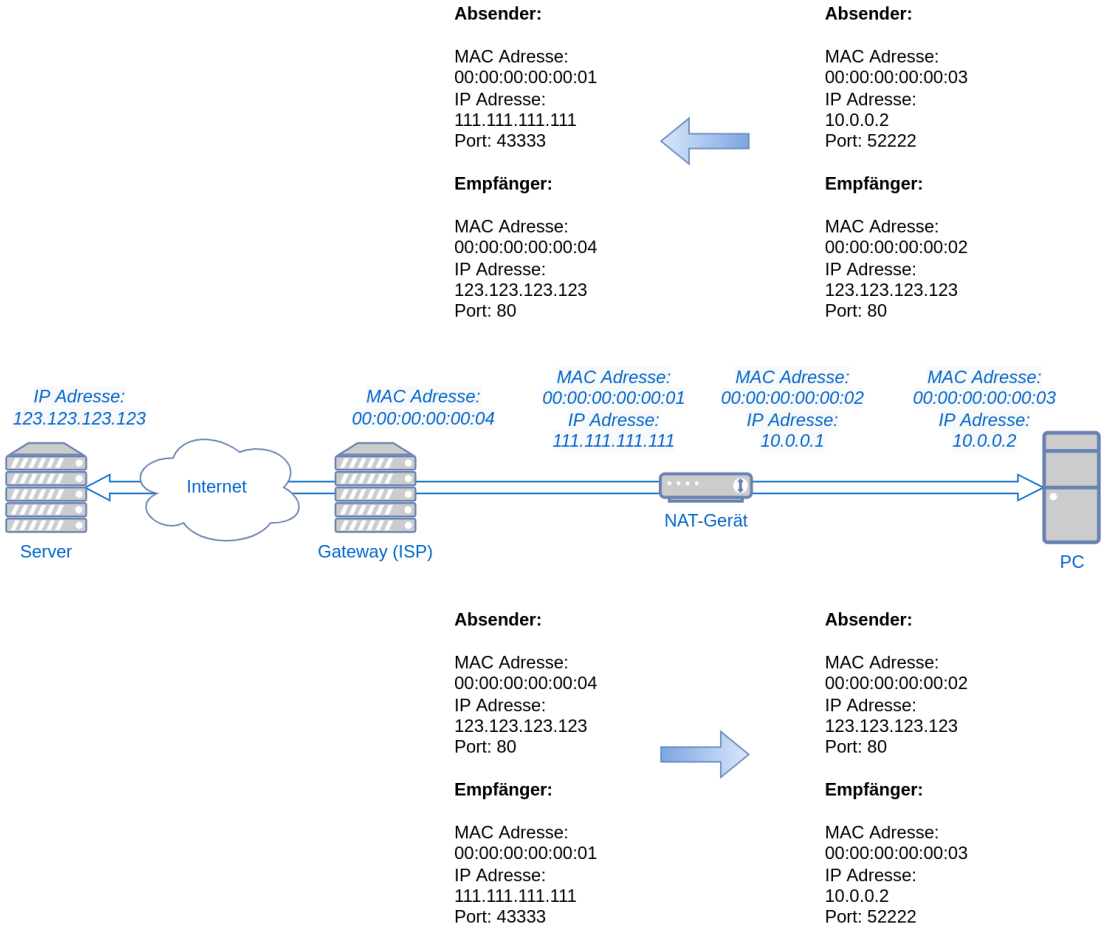

Abbildung 3: Modifikation der Paketfelder durch NAT

Linux-basierte Betriebssysteme bieten umfangreiche Möglichkeiten für die Arbeit mit Netzwerkpakete an. Für ein besseres Verständnis wie das Ganze funktioniert betrachten wir zuerst den Weg eines Netzwerkpakets durch den Kernel und die Einrichtungen, die dabei überquert werden.

Der Weg eines Netzwerkpakets durch den Linux-Kernel

Der Verlauf einer Netzwerkkommunikation im Linux-Kernel folgt den selben Standards, die alle andere Netzwerkteilnehmer folgen müssen, damit die Kommunikation überhaupt möglich ist. Das Schichtenmodell, beschrieben im vorigen Kapitel, wird von jedem Paket passiert - von der Anwendung bis zum Kabel und dann umgekehrt beim Empfänger. Der Verlauf an Pakete über dem Linux-Kernel können wir besser beschreiben, wenn wir dieser in einem Kontext stellen. Betrachten wir dafür den allgemeinen Ablauf einer typischen Kommunikation zwischen einem Browser und einem Web-Server[1]:

[1] Vgl. Benvenuti (2006), S.271-274

1. Der Browser bekommt die Web-Adresse aus der Benutzereingabe und übersetzt sie in einer IP-Adresse mittels DNS. Die Entscheidung, wie diese Adresse zu erreichen ist, ist eine der Aufgaben der Vermittlungsschicht. Aus der Sicht des Browsers kann eine (virtuelle) Verbindung zu der Web-Server-Software in der Anwendungsschicht aufgebaut werden. Praktisch wird die Kommunikation über dem ganzen Netzwerk-Stack zwei Mal verlaufen - einmal beim Sender und einmal beim Empfänger;

2. Die Kommunikation zwischen der Anwendungsschicht und der Transportschicht erfolgt über den s.g. *sockets*[1]. Sockets erfüllen die Funktion einer Schnittstelle zwischen den Anwendungen und den Kernel-Netzwerkstack[2];

3. In der Transportschicht werden bei Bedarf die Daten in Segmente aufgeteilt und jedem Segment wird ein Header zugefügt. Ein Teil des Headers sind die Port-Adressen vom Sender und Empfänger. Wie schon diskutiert, können verschiedene Anwendungen gleichzeitig auf einem Rechner laufen und in Netzwerkkommunikation teilnehmen. Die Ports beinhalten genau die Information welche Anwendung gemeint ist;

4. Über den Daten in der Transportschicht wird nur die Empfänger-Anwendung angegeben. Die Information über dem Rechner, wo diese Anwendung läuft und wie dieser erreicht werden kann ist im Aufgabenbereich der Vermittlungsschicht. Über komplexe Routing-Algorithmen werden Entscheidungen über der Strecke getroffen, über die die Pakete ihr Ziel erreichen sollen sowie auch welches Gerät als nächstes auf dieser ausgerechneten Strecke liegt. Weil die direkte Übergabe von Information über dem Kommunikationsmedium noch eine Stufe tiefer verläuft, wird mittels des ARP-Protokolls die korrekte Sicherungsschicht-Adresse (die MAC-Adresse) abgefragt;

5. Mit dem Wissen über der MAC-Adresse des nächsten Geräts werden die Daten über dem Medium über einem Schicht-2 Protokoll wie z.B. Ethernet gesendet. Der Code für die direkte Abbildung von Daten in elektrischen oder anderen Signale auf dem Medium ist Hardware-spezifisch und befindet sich in den Treiber[3];

6. Auf dem Weg zum Ziel werden bei jedem Netzwerkgerät die Schicht-2 Felder gelesen und entfernt. Über dem Inhalt des Schicht-3 Headers entscheidet das

[1] Chimata (2005), S.3
[2] https://www.man7.org/linux/man-pages/man7/socket.7.html, Internetquelle
[3] Ebd.

Gerät ob es selber als Empfänger gemeint ist, oder muss das Paket weiterleiten. Im Fall einer Weiterleitung werden wieder Routing-Entscheidungen durch Routing-Tabellen getroffen. Dann wird das Paket mit einem neuen Schicht-2 Header versehen und zum nächsten Gerät gesendet;

7. Bei dem endgültigen Empfänger verläuft das Paket den Netzwerkstack wieder, in der umgekehrten Richtung. Ein ankommendes Paket löst eine Kernel-Unterbrechung aus und landet schließlich in einer neuen *sk_buff* Struktur[1]. Der Schicht-2 Header wird entfernt. Die Funktion *netif receive skb* entscheidet über dem Typ des Pakets aufgrund der Information im Schicht-3 Header und leitet es weiter zu der Vermittlungsschicht[2]. Dort entfernt die Funktion *ip_rcv* den entsprechenden Header, prüft das Paket auf Fehler und eventuell defragmentiert es. Die Entscheidung, ob das Paket für eine lokale Anwendung gemeint ist, oder weitergeleitet werden muss, trifft die Funktion *ip rcv finish*[3]. Im Fall eines lokalen Empfängers entscheidet *ip_local_deliver_finish* welche Funktion aus der Transportschicht aufgerufen wird - z.B. für TCP die Funktion *tcp_v4_cv* - und übergibt ihr die weitere Kontrolle;

8. Die Anwendung kann die Daten aus der Transportschicht mittels der *sockets*-Schnittstelle bekommen.

Für die Ziele dieser Arbeit müssen wir direkt Pakete auf ihren Weg durch den Kernel manipulieren. Für dieses Zweck brauchen wir ein Mittel um auf die Pakete während ihrer Bearbeitung zuzugreifen. Das Netfilter-Projekt erlaubt uns genau das. Der Code vom Netfilter ist ein Teil des Kernels und erlaubt neben einer direkten Manipulation von Pakete auch Filterung, NAT, Loggen und Zugriff auf Pakete aus der Anwendungsebene[4]. Von besonderen Bedeutung für uns ist das Angebot von *hooks*. *Hooks* stellen ein Framework im Kernel dar, das uns erlaubt, Callback-Funktionen in verschiedenen Stellen direkt im Netzwerkstack zu definieren. Diese werden für jedes Paket aufgerufen, das die gemeinte Stelle durchquert[5]. Ein *Hook* bietet somit die Möglichkeit an, in einer bestimmten Phase der Bearbeitung von Netzwerkpakete auf diese in deren momentanen Zustand zuzugreifen und gegebenenfalls sie zu ändern.

[1] Chimata (2005), S.12
[2] Chimata (2005), S.13
[3] Chimata (2005), S.14
[4] https://www.netfilter.org/, Internetquelle
[5] Ebd.

Eine Darstellung des Netzwerkkommunikationsverlaufs und die Stellen der Netfilter-*Hooks* sind auf Abbildung 4 zu sehen.

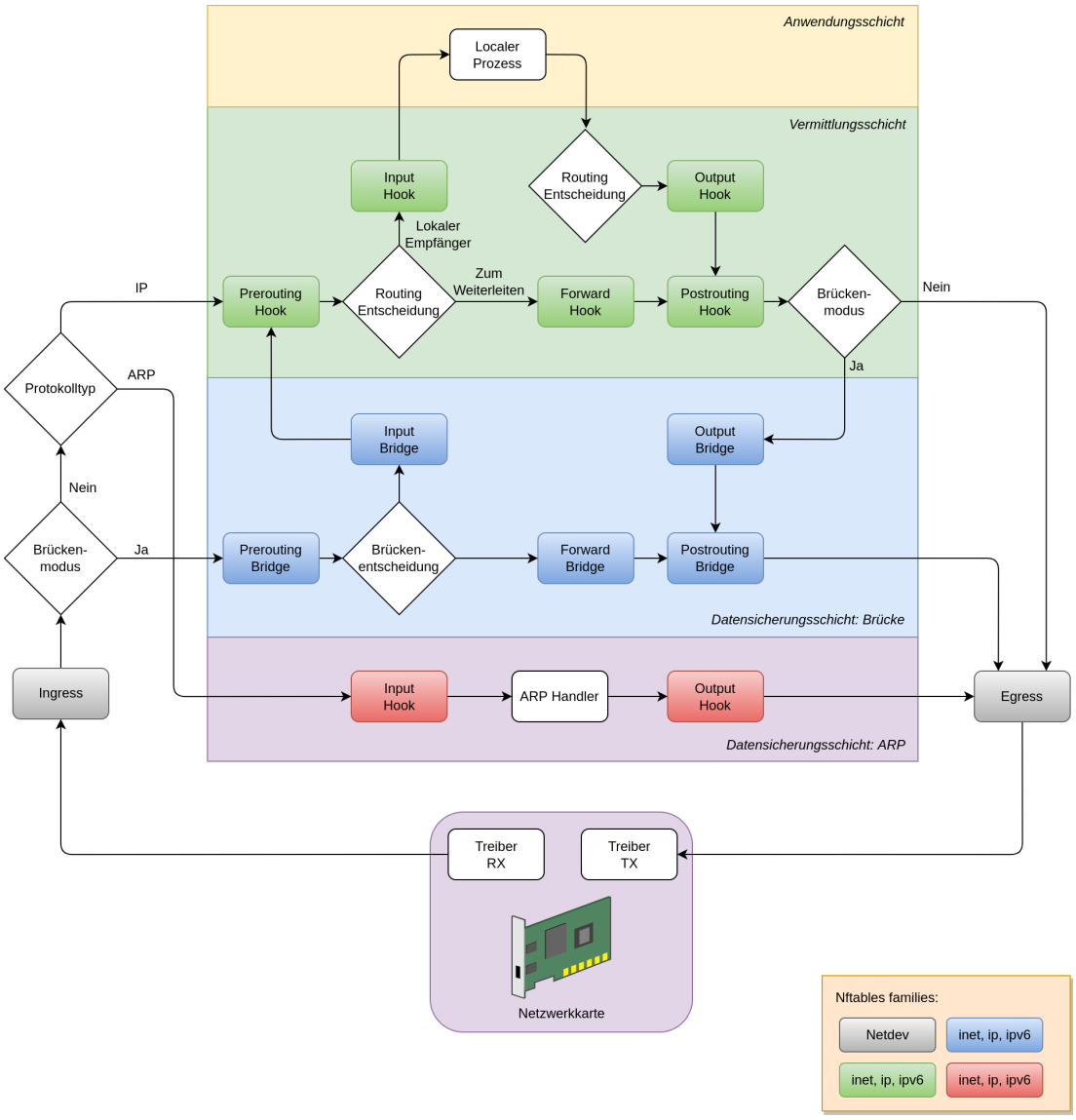

Abbildung 4: Netzwerkfluss und Netfilter-Hooks im Linux-Kernel

(Modifiziert von: https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks, Zugriff am 08.05.2022)

Kernel-Einrichtungen und Software für die Arbeit mit Netzwerkpakete

iproute2

Für die Arbeit mit dem Netzwerk aus der Anwendungsebene in Linux ist aktuell *iproute2* angeboten. Das ist eine Sammlung aus Software zur Steuerung von TCP/IP-Netzwerke und Kontrolle des Netzwerkverkehrs (traffic control)[1].

[1]https://wiki.linuxfoundation.org/networking/iproute2, Zugriff am 08.05.2022

Die wichtigste Werkzeuge dabei sind *ip* und *tc* - entsprechend für die Arbeit mit TCP/IP und für die Verkehrskontrolle. Verkehrskontrolle ist von weniger Bedeutung für diese Arbeit, deshalb fokussieren wir uns auf *ip*.

Die Form eines *ip*-Aufrufs ist[1]:

```
1  ip [ OPTIONS ] OBJECT { COMMAND | help }
```

ip ist sehr umfangreich und durch Auswahl des OBJECTs kann aus einer Vielfalt von Themen gewählt werden. Diese beinhalten[2]:

```
1  ip { link | address | addrlabel | route | rule | neigh | ntable | tunnel |
2      tuntap | maddress | mroute | mrule | monitor | xfrm | netns | l2tp |
3      tcp_metrics | token | macsec | vrf | mptcp }
```

Für unsere Ziele sind folgende OBJECTs interessant:

- *ip link*[3] - Abfragen und Konfiguration von der Netzwerkkarte. Typische Anwendungen sind z.B. Hoch- und Herunterfahren der Netzwerkkarte, Abfrage und Änderung ihrer MAC-Adresse sowie auch die Erstellung und Einstellung von virtuellen Netzwerkgeräte. Diese werden im nächsten Paragraph weiter betrachtet;
- *ip address*[4] - gibt IPv4- und IPv6-Adressen und Einzelheiten darüber aus, fügt neue Adressen hinzu oder löscht schon vorhandene Adressen;
- *ip route*[5] - zeigt die Routing-Tabellen im Kernel an und manipuliert diese;
- *ip neigh*[6] - gibt Informationen über den ARP-Tabellen im Kernel oder ändert diese;
- *ip netns*[7] - ist das Werkzeug für die Arbeit mit virtuellen Netzwerk-Adressenräume. Diese werden weiter im nächsten Paragraph kommentiert.

Standardmäßig wird Information über dem angegeben OBJECT ausgegeben, oder - nach Eingabe eines Befehls als COMMAND - werden auch entsprechend Änderungen am OBJECT durchgeführt.

[1] https://www.man7.org/linux/man-pages/man8/ip.8.html, Zugriff am 08.05.2022
[2] Ebd.
[3] https://www.man7.org/linux/man-pages/man8/ip-link.8.html, Zugriff am 08.05.2022
[4] https://www.man7.org/linux/man-pages/man8/ip-address.8.html, Zugriff am 08.05.2022
[5] https://www.man7.org/linux/man-pages/man8/ip-route.8.html, Zugriff am 08.05.2022
[6] https://www.man7.org/linux/man-pages/man8/ip-neighbour.8.html, Zugriff am 08.05.2022
[7] https://www.man7.org/linux/man-pages/man8/ip-netns.8.html, Zugriff am 08.05.2022

Linux-Namensräume

Ein Namensraum abstrahiert globale Systemressourcen. Die Prozessen innerhalb des Namensraums bekommen den Eindruck, dass sie ihre eigene isolierte Instanz der globalen Ressource haben. Änderungen an der globalen Ressource sind für andere Prozesse im selben Namensraum sichtbar, für die Prozesse aus den anderen Namensräume aber nicht[1]. Traditionell gehört jedes Prozess im Linux zu demselben Namensraum, der unter dem *init*-Prozess ausgebaut wird. Seit der Kernel-Version 2.6 kann jedes Prozess einen separaten Namensraum haben[2]. Ein Namensraum kann sich auch nur auf Netzwerkressourcen beziehen. In diesem Fall werden die Netzwerkgeräte, die IP-Stacks, die Routing-Tabellen, die Firewall-Regeln und andere Ressourcen in der Abstraktion zusammengefasst[3]. Zwei verschiedene Netzwerk-Namensräume können über virtuellen Netzwerkgeräte verbunden werden und dadurch miteinander über dem (virtuellen) Netzwerk kommunizieren[4].

nftables / nft

Das Netfilter-Projekt haben wir schon diskutiert, bis auf der praktischen Benutzung von Filterung und Hooks. Dafür wird das Framework *nftables* angeboten[5]:

- *nftables* ist verfügbar als einen Teil vom Linux-Kernel seit der Version 3.13;
- Die Arbeit mit *nftables* aus der Anwendungsebene erfolgt über dem Werkzeug *nft*;
- *nftables* hat auch eine Kompatibilitätsschicht für die Arbeit mit älteren Versionen der Anwendungssoftware (*iptables*);
- *nftables* ist sehr flexibel bei der Definition von Regeln, die auch in Strukturen organisiert werden können.

Das Tool *nft* ist sehr leistungsfähig und für die Arbeit damit kann es hilfreich sein, zuerst die Grundkonzepte zu klären:

- *Skript*: *nft* kann selbstständig aus dem Terminal benutzt werden, meistens werden aber Dateien/Skripten mit den Regeln erstellt, die danach von *nft* gelesen und ausgeführt werden[6];

[1] https://www.man7.org/linux/man-pages/man7/namespaces.7.html, Zugriff am 08.05.2022
[2] Bovet; Cesati (2005), S.484
[3] https://www.man7.org/linux/man-pages/man7/network_namespaces.7.html, Zugriff am 08.05.2022
[4] Ebd.
[5] https://wiki.nftables.org/wiki-nftables/index.php/What_is_nftables%3F, Zugriff am 08.05.2022
[6] https://wiki.nftables.org/wiki-nftables/index.php/Scripting, Zugriff am 08.05.2022

- *Tabelle*: Tabellen sind die Top-Level Strukturen innerhalb eines *nftables*-Regelsatzes; Sie enthalten Ketten, Sets, Maps, Flusstabellen und zustandsbehaftete Objekte. Jede Tabelle gehört zu genau einer Familie. Dadurch erfordert ein Regelsatz mindestens eine Tabelle für jede Familie, die gefiltert werden muss[1];

- *Familie*: netfilter erlaubt die Arbeit in mehreren Netzwerkschichten. Die Schichten werden unter dem Konzept von Familie abstrahiert. Diese sind *ip*, *ip6*, *inet*, *arp*, *bridge* und *netdev*[2]. Graphisch sind diese in der Abbildung 4 auf Seite 25 dargestellt.

  - *ip* und *ipv6* beziehen sich entsprechend auf die Arbeit mit IPv4 und IPv6-Pakete;
  - *inet* ist eine Sammelfamilie für IPv4 und IPv6-Pakete;
  - *arp* erfasst und bearbeitet den ARP-Verkehr;
  - *bridge* ist für die Arbeit mit Paketen, die Brücken passieren;
  - *netdev* arbeitet auf der untersten Schicht, direkt mit der Netzwerkkarte.

- *Kette*: Die Regeln gehören zu Ketten, die zu einem konkreten Hook angeschlossen werden. Der Name der Kette ist frei wählbar[3];

- *Regel*: Die Regeln entscheiden wie die Pakete behandelt werden, die die Kriterien den Regeln entsprechen - z.B. Akzeptieren oder Verwerfen[4];

Somit ist der grundsätzliche Ablauf für die Arbeit mit *nft* wie folgt:

1. Wir klären ab, was für Pakete unser Ziel sind sowie auch in welcher Stelle im Kernel sie erfasst werden sollen;

2. Wir definieren mit Regeln, was mit diesen Pakete passieren muss;

3. Diese Regeln sammeln wir in einer Kette, die zu einem Hook angeschlossen wird;

[1] https://wiki.nftables.org/wiki-nftables/index.php/Configuring_tables, Zugriff am 08.05.2022
[2] https://wiki.nftables.org/wiki-nftables/index.php/Nftables_families, Zugriff am 08.05.2022
[3] https://wiki.nftables.org/wiki-nftables/index.php/Configuring_chains, Zugriff am 08.05.2022
[4] https://wiki.nftables.org/wiki-nftables/index.php/Simple_rule_management, Zugriff am 08.05.2022

4. Alle Ketten sammeln wir in einer Tabelle. Je nach Bedarf für die Arbeit mit mehrere Familien, erstellen wir auch weitere Tabellen;

5. Alle Tabellen stellen wir in einer Datei zusammen, die von *nft* ausgeführt wird.

iproute2 und *nft*ables, zusammen mit den Standardtools, die Teil einer typischen Linux-Installation sind, bilden den Werkzeugsatz für das in dieser Arbeit vorgestellte Konzept. Betrachten wir jetzt die Umgebung und die Umstände, wo die Entwicklung des Konzepts und der darauf folgender Einsatz stattfinden werden.

3 Das Kabelfernsehnetz

In Deutschland ist DSL immer noch die dominierende Variante für Internetanbindung über einem Festnetz[1]. Die Anbindung über das Fernsehnetz ist vergleichsweise neuere Technologie, deren Anteil aber schnell wächst[2]. Für die Strecken mit erhöhtem Datenverkehr werden Glasfaserkabel gelegt und die Endstrecken zum Kunde sind über Koaxialkabel ausgeführt. Die Technologie bezeichnet man als HFC (Hybrid Fiber Coax)[3]. Einige Vor- und Nachteile im Vergleich zwischen den Internetanbindungen haben wir schon im letzten Kapitel diskutiert. Grundsätzlich stellt HFC wesentlich mehr Bandbreite zur Verfügung, allerdings ist diese nicht garantiert und hängt von der Anzahl der zur Zeit aktiven Benutzer ab[4].

Die Endgeräte, die die Daten für die Sendung über dem Kabel vorbereiten, oder diese aus dem Kabel empfangen - die Kabelmodems - haben wir auch schon erwähnt. Betrachten wir jetzt das Protokoll, das von den Modems benutzt wird und worauf die Kommunikation basiert:

DOCSIS (Data Over Cable Service Interface Specification) ist ein Standard, der eine bidirektionale Sendung von einem IP-basierten Datenverkehr zwischen den Head-Ends des Kabelnetzanbieters und den Kundengeräte über Koaxialkabel- oder HFC-Netze regelt[5]. Unter Head-Ends versteht sich die Hardware bei dem Kabelnetzanbieter, die die Audio- und Video-Signale über Broadcast ins Netz einspeist[6]. Die Hardware, die den Head-Ends zugefügt wird, um die Arbeit mit Datenpakete und Netzwerkkommunikation zu ermöglichen, bezeichnet der Stan-

[1] Vgl. Bundesnetzagentur (2021), S.54
[2] Vgl. Bundesnetzagentur (2021), S.56-57
[3] Tanenbaum; Wetherall (2014), S.180
[4] Tanenbaum; Wetherall (2014), S.180
[5] Cable Television Laboratories, Inc. (2015), S.21
[6] Cable Television Laboratories, Inc. (2015), S.33

dard als CMTS (Cable Modem Termination System)[1]. Der grundsätzliche Aufbau ist auf Abbildung 5 dargestellt.

DOCSIS ist aktuell in der Version 3.1, die auch abwärtskompatibel zu DOCSIS 3.0 und ältere Versionen ist[2].

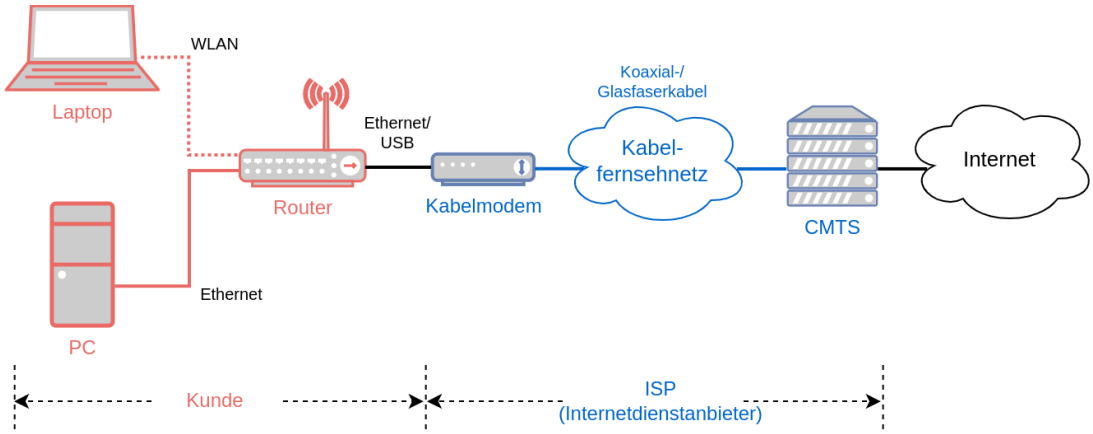

Abbildung 5: Internetanbindung über dem Kabelfernsehnetz

(In Anlehnung an Cable Television Laboratories, Inc. (2015), S.21 und Tanenbaum; Wetherall (2014), S.183)

Das Kabelmodem und der Kundenrouter müssen nicht unbedingt zwei separate Geräte sein, sondern werden oftmals von den Anbieter als ein einziges Gerät zur Verfügung gestellt, das beide Funktionen übernimmt, eventuell mit der WLAN-Funktion extra zubuchbar[3]. Wenn der Benutzer andere Ansprüche auf den Router hat, oder selber sein Netzwerk konfigurieren und kontrollieren möchte, hat er das Recht einen eigenen Router anzuschließen. Dieses Recht ist vom Gesetzgeber geregelt[4] [5].

Grundsätzlich haben die Geräte im Heimnetz keine öffentliche IP-Adressen, sondern benutzen solche aus dem Privatadressenraum[6]. Das bedeutet praktisch, dass der Heimnetzrouter auch NAT-fähig ist. Die Verteilung der IP-Adressen erfolgt, wie schon gesehen, typischerweise per DHCP. Die Verbindung zwischen dem Router und dem Kabelmodem, vorausgesetzt dass die beide separate Geräte sind, erfolgt standardmäßig mittels Ethernet-Kabel[7] und die standardmäßige Ethernet-Protokolle[8].

[1] Cable Television Laboratories, Inc. (2015), S.31
[2] Cable Television Laboratories, Inc. (2015), S.49
[3] Vodafone Deutschland GmbH (2022), Internetquelle
[4] Deutscher Bundestag (2015), S.13042
[5] Deutscher Bundestag (2021), S.1897
[6] Baun (2020), S.177
[7] Tanenbaum; Wetherall (2014), S.183
[8] Cable Television Laboratories, Inc. (2017), S.10

Betrachten wir jetzt die andere Seite des Kabelmodems. Hier werden mehrere komplexe Ansätze nebeneinander benutzt, um die Bandbreite des Kabelnetzes zwischen den Benutzer zu teilen[1]. Wenn das Modem hochfährt, scannt es die Downstream-Frequenzen nach spezielle Pakete, die das Head-End periodisch sendet. Wenn so ein Paket erkannt ist, meldet sich das Modem über der Upstream-Frequenz an, wird eventuell vom Head-End anerkannt und bekommt Einstellungen, womit es dann arbeiten kann[2].

DOCSIS 3.1 definiert eine Base Line Privacy Plus (BPI+) Architektur, die eine Ende-zu-Ende Kommunikation zwischen dem Kabelmodems und dem CMTS vorsieht, mit Unterstützung von Authentifizierung und einen sicheren Schlüsselaustausch[3]. Für das Abfangen und die Analyse von einem DOCSIS-basierten Datenverkehr werden am Markt komplexe, kostenintensive Produkte angeboten[4].

4 Bedarf nach Netzwerkdienste im lokalen Netzwerk

Es ist nicht zwingend erforderlich, dass im lokalen Netzwerk Dienste angeboten werden. Ein typisches Heimnetz kann durchaus mit einem DHCP-Server auskommen, der vom Router angeboten ist. Aufgrund des technischen Progress werden aber von den Benutzer immer mehr Funktionen und Dienste erwartet, die oftmals von kommerziellen Anbieter zur Verfügung gestellt werden. Manche dieser Dienste verbessern vorhandene Gegebenheiten, manche können auch etwas neues anbieten. Mit etwas technischem Verstand können aber Dienste im lokalen Netzwerk betrieben werden, die die gleiche Funktionen ausführen, mit dem extra Vorteil, dass die Kontrolle der eigenen Daten nicht in fremden Händen verlassen werden muss.

4.1 DNS

Die frei lesbare und änderbare Felder aus den unteren Schichten - wie MAC- und IP-Adressen, haben wir schon kommentiert. Änderungen an diesen Felder würden zu einer Umleitung der Pakete oder im schlimmsten Fall zu deren Verlust führen. Das bedeutet aber noch keinen Zugriff auf die Benutzerdaten, solange diese gesichert in der Anwendungsschicht transportiert werden. Die moderne Kommu-

[1] Tanenbaum; Wetherall (2014), S.183
[2] Ebd.
[3] Cable Television Laboratories, Inc. (2016), S.23
[4] Excentis (o.J.), Internetquelle

nikationen über dem Internet erfolgt fast ausschließlich verschlüsselt[1] und somit ist eine Absicherung gegen Abhören der Nutzdaten gegeben. Allerdings ist das nicht genug, um die Kommunikation als sicher zu erklären. Ein Protokoll, dass in weiter Benutzung ist und auch kritisch für die Funktion von Internet ist, ist das DNS Protokoll, das wir schon kurz vorgestellt haben. Betrachten wir jetzt das Protokoll im Einzelnen und warum es die sonst sichere Kommunikationen gefährden kann.

Der DNS-Namensraum ist hierarchisch geordnet. Der Namensraum ist in Domains aufgeteilt, die wiederum in Sub-Domains aufgeteilt werden. Somit entsteht eine Baum-ähnliche Struktur, deren Blätter die konkrete Hosts entsprechen[2]. Die Top-Level Domains können allgemein, wie z.B. *.com* oder Länder-spezifisch, wie z.B. *.de* sein[3]. Diese werden direkt von ICANN gesteuert. Alle andere Domains können kommerziell erworben werden, meistens durch die Vermittlung eines ISPs[4]. Die Root-DNS-Server sind auf 13 Domains zu finden - von *a.root-servers.net* bis *m.root-servers.net* und sind mehrfach weltweit verteilt, aufgrund deren kritischen Bedeutung[5]. Sie sind für die Top-Level Domains zuständig.

Die Information für eine Domain ist ein Datensatz, der im einfachsten Fall eine IP-Adresse enthaltet, aber auch mehrere andere Informationen haben kann, wie etwa Parameter, Adresse des Mail-Servers, Alias, Text-Beschreibung u.a.[6] Manche Felder können mehrere Einträge haben - z.B. es kann sein, dass ein Host über mehrere IP-Adressen erreicht werden kann[7].

Die DNS-Namensauflösung erfolgt in mehrere Anfragen zu den zuständigen DNS-Server mit dem Endziel eine IP-Adresse für die gesuchte Domain zu finden[8]. Der Bedarf nach mehrere Anfragen kommt aus der Tatsache, dass kein DNS-Server die Information über alle Domains hat. Um die Suche zu optimieren werden Anfragen in einem Cache gespeichert und mit einem Zeitstempel versehen. Der Stempel ermöglicht das Verwerfen alter, potentiell falsche DNS-Einträge[9]. Das Prozess fängt mit der Top-Level Domain an und dann weiter in der Hierarchie, bis der zuständige DNS-Server gefunden wird, von wem die konkrete IP-Adresse bekommen wird, oder auch, bis die Information irgendwo auf dem Weg in einem

[1] Vgl. Sandvine Corporation (2022), S.13
[2] Mandl (2019), S.124
[3] Tanenbaum; Wetherall (2014), S.613
[4] Mandl (2019), S.125
[5] Tanenbaum; Wetherall (2014), S.621
[6] Tanenbaum; Wetherall (2014), S.616-617
[7] Ebd.
[8] Tanenbaum; Wetherall (2014), S.620
[9] Tanenbaum; Wetherall (2014), S.622

Cache gefunden wird[1].

Wie man sich schon vorstellen kann verlaufen die Anfragen bei der DNS-Namensauflösung viele verschiedene und potentiell auch lange Strecken im Internet. Wenn das DNS-Protokoll in seiner aktuellen Form im 1987 entwickelt wurde, gab es keine Bedenken über Sicherheit oder Authentifizierung der Teilnehmer bei der Namensauflösung - in den beiden RFCs, die das DNS-Protokoll definieren - RFC1034 und RFC1035 - gibt es keine Anmerkung darüber[2] [3]. DNS benutzt unverschlüsselte und nicht unterzeichnete Anfragen über UDP oder TCP und den Port 53[4]. Damit ist das Abfangen und die Änderung der DNS-Kommunikation trivial. Eine Vielfalt von Szenarien werden dadurch denkbar[5]. Die beide Angriffe, die direkt den Benutzer gefährden sind:

- Falsche DNS-Daten können zu dem DNS-Server geschickt werden um seinen Cache mit der gewünschten Information zu füllen;
- Die DNS-Kommunikation kann auch direkt zu einem anderen DNS-Server umgeleitet werden.

Betrachten wir die Konsequenzen dieser Möglichkeiten:

- Der Benutzer kann zu einem vom Angreifer kontrollierten Server geführt werden, der z.B. die gesuchte Webseite imitiert um Login-Daten zu sammeln. Dank SSL-Zertifikaten, die dafür sorgen, dass die Authentizität eines Servers bestätigt werden kann ist dieser Angriff von weniger Bedeutung[6]. Das setzt natürlich voraus, dass der Server ein solcher Zertifikat hat. Glücklicherweise ist das der Fall bei den meisten moderne Web-Server[7]. Das Problem bleibt aber bestehen, wenn separate Domains mit eigenem Zertifikat benutzt werden, deren Namen ähnlich der gesuchten Domain sind;
- Der Benutzer kann umgeleitet werden. Diese Variante hat zweiseitige Bedeutung. Einerseits kann der Benutzer auf einer schädlichen Domain landen mit den damit verbunden Konsequenzen. Es kann aber auch sein, dass die Weiterleitung gewünscht ist, um z.B. genau die schädliche Domains umzugehen. Das letzte wird DNS-Filtering genannt und kann von kommerziellen Anbieter erworben werden, oder auch vom Benutzer selbst konfiguriert

[1] Ebd.
[2] Network Working Group (1987a)
[3] Network Working Group (1987b)
[4] Network Working Group (1987b), S.31
[5] https://www.cloudflare.com/learning/dns/dns-security/, Zugriff am 08.05.2022
[6] https://www.cloudflare.com/learning/ssl/what-is-an-ssl-certificate/, Zugriff am 08.05.2022
[7] Internet Security Research Group (2022), Internetquelle

werden[1]. Eine andere Benutzung dieser Technologie ist in der Form von Kindersicherung - Domains, die Inhalt ungeeignet für Kinder bereitstellen, werden gefiltert und können nicht geladen werden[2];

- Dem Benutzer kann den Zugriff auf eine Domain gesperrt werden indem er auf seine Anfrage z.B. die Antwort 0.0.0.0 („nicht spezifiziert“) bekommt[3]. Das kann als eine Variante von DNS-Filtering gesehen werden und kann sowie erwünscht oder auch nicht erwünscht erfolgen;

- Auch wenn nichts in der Anfrage geändert wird, kann diese trivial gelesen werden. Somit kann der Benutzer verfolgt werden. Zusätzlich ist das Problem noch schlimmer, weil dieses Abhören nicht vom Benutzer festgestellt werden kann, weil es vollständig passiv erfolgen kann.

Die Sicherheitsprobleme mit DNS haben dazu geführt, dass neue Ansätze oder Erweiterungen zu DNS entwickelt werden, um diese Probleme umzugehen. Ein solcher Ansatz ist DNSSEC (DNS Security), dass eine Authentifizierung des DNS-Server vorsieht[4]. Die Anfragen sind aber immer noch unverschlüsselt und es besteht immer noch die Möglichkeit von DNS-Filtering (erwünscht oder nicht). Zusätzlich ist DNSSEC immer noch wenig verbreitet und wegen der Abwärtskompatibilität kann einfach ignoriert werden[5].

Betrachten wir jetzt eine DNS-Anfrage aus einer realen Benutzerumgebung, indem wir über der Software *dig*[6] die Domain *www.akad.de* abfragen. Das Parameter „*+trace*“ erlaubt uns den ganzen Ablauf der Anfrage zu sehen. Über dem „*@*“ Parameter können wir den DNS-Server spezifizieren. In der Ausgabe[7] können wir den erwarteten Ablauf verfolgen:

- Der lokale DNS-Server, welcher IP-Adresse vom ISP gegeben ist[8], wird zuerst abgefragt und antwortet mit einer Liste der Root-DNS-Server (Zeile 3-19);

- Aus der Liste wird ein Server willkürlich ausgewählt - in unserem Fall *e.root-servers.net* - und nach die Top-Level Domain *.de* gefragt. Davon bekommen

[1] https://www.cloudflare.com/learning/access-management/what-is-dns-filtering/, Zugriff am 08.05.2022
[2] Prince (2020), Internetquelle
[3] https://docs.pi-hole.net/ftldns/blockingmode/, Zugriff am 08.05.2022
[4] https://www.cloudflare.com/learning/dns/dns-security/, Zugriff am 08.05.2022
[5] Sullivan (2014), Internetquelle
[6] https://manpages.debian.org/bullseye/bind9-dnsutils/dig.1.en.html, Zugriff am 08.05.2022
[7] Im Anhang auf Seite 88
[8] in dem betrachteten Beispiel: 83.169.185.161

wir die Liste mit Server, die für die Deutschland-spezifische Domains zuständig sind (Zeile 21-29);

- Ähnlich wählt die Software einen *.de* Server - im angezeigten Beispiel *s.de.net* - und fragt nach den zuständigen Server für *akad.de.* Die Antwort beinhaltet drei Server (Zeile 31-38);

- Aus den drei hat die Software *ns01.agenturserver.co* gewählt, fragt nach *www.akad.de* und schließlich bekommt davon die endgültige Antwort[1], die neben der gesuchten IP-Adresse, auch den Typ des Eintrages hat (*A*: Hostadresse) und die Zeit, für die diese Antwort gültig ist (*TTL*/time to live: 10800 Sekunden).

Interessante Beobachtung ist das Vorhandensein von DNSSEC-Einträge für alle Anfragen, bis auf den letzten Server. Das bedeutet schließlich, dass wir, basierend auf DNSSEC, die Authentizität der Root-Server und der .de-Server überprüfen können, aber nicht des akad.de-Servers und somit ist diese Antwort nicht unbedingt verlässlich. Wie schon erwähnt ist das nicht so schlimm, weil wenn wir die Webseite besuchen, kann unser Browser das SSL-Zertifikat verifizieren, das in diesem Fall von *DigiCert* ausgestellt ist[2].

Die detaillierte Ausgabe von *dig* und die Angaben über der Authentifizierung sind im Anhang auf Seite 88 dargestellt.

Angesicht der Besonderheiten von DNS sind grundsätzlich zwei Ansätze für die Sicherung der Kommunikation denkbar:

- Ausnutzung des freien Zugangs zu der DNS-Kommunikation durch Filterung von ungewollte oder schädliche Domains. Das ist das Prinzip der DNS-Filter[3]. Diese basieren auf Blocklisten, die an sich auf Domains, auf IP-Adressen oder auch auf beides beziehen können[4]. Wenn ein DNS-Filter benutzt wird, führt eine Aktivierung von schädlichen Links zu keine Schäden, weil die Seiten nicht geladen werden, wenn die Anwendung keine (sinnvolle) IP-Adresse aus der DNS-Abfrage bekommt. Das setzt natürlich voraus, dass die schädliche Domains schon in der Liste sind;

- Verschlüsselung der DNS-Kommunikation. Zwei Ansätze, die zur Zeit entwickelt werden - DNS über TLS (DoT) und DNS über HTTPS (DoH) -

[1] *„www.akad.de. 10800 IN A 188.94.253.154“*

[2] Oder wir landen an einer ähnlich-aussehenden Domain, die eigenes, echtes Zertifikat hat und uns doch täuscht.

[3] https://www.cloudflare.com/learning/access-management/what-is-dns-filtering/, Zugriff am 08.05.2022

[4] Ebd.

benutzen TLS für den verschlüsselten Transport von DNS[1]. Der Hauptunterschied bei den beiden Ansätze ist der benutze Port - DoT benutzt 853 und DoH - 443[2]. Die Benutzung vom Port 443 bei DoH macht den Verkehr sehr schwer unterscheidbar von einem normalen HTTPS-Verkehr. Der Kritikpunkt bei solcher Lösungen ist, dass das Vertrauen vom ISP zu dem DoH/DoT-Anbieter geschoben wird. Bei einer Standardinstallation kann aber nicht nur der ISP unsere DNS-Kommunikation lesen und filtern, sondern auch jeder Rechner, der auf dem Weg der DNS-Anfrage liegt. DoT ist in RFC7858, und DoH - in RFC8484 standardisiert;

- Ein noch neueren Ansatz, der mit dem Kritikpunkt beim DoH/DoT umgehen möchte ist „Oblivious DoH“ - ein gemeinsam von Cloudflare, Apple, und Fastly entwickeltes Protokoll, das IP-Adressen von DNS-Anfragen trennt, sodass kein Rechner gleichzeitig beides sehen kann[3]. Das Grundprinzip ist die Benutzung eines Proxy-Servers, der von einem dritten Anbieter kontrolliert wird. Die Anfragen werden mit dem Public-Key vom DoH-Anbieter verschlüsselt und über dem Proxy geleitet. Somit kann der Proxy-Server nur die IP-Adresse vom Client sehen und der DoH-Resolver - nur die Anfrage. Ein potentieller Angreifer kann nur dann die DNS-Kommunikation verfolgen, wenn er gleich beide - den Proxy-Server und den Resolver kontrolliert[4]. Der DoH-Anbieter muss nicht unbedingt auch DNS-Resolver sein. Cloudflare kombiniert aber beide Funktionen für eine bessere Leistung[5].

Die Verschlüsselung der DNS-Kommunikation ist ein sehr umstrittenes Thema[6]. Ein Kritikpunkt - der Bedarf an Vertrauen zum DoH/DoT-Anbieter - haben wir schon betrachtet. Zusätzliche Probleme, die die Gegner der DNS-Verschlüsselung sehen, sind[7]:

- Der Benutzer kann ein falsches Sicherheitsgefühl bekommen. Es existieren viele andere Wege, ihn zu verfolgen;
- Firewalls, lokale DNS-Filter und andere Software, die mit DNS arbeiten, können nicht mehr korrekt funktionieren. Hierfür ist die Benutzung von DoT bevorzugt, aufgrund des separaten Ports und der Unterscheidung vom HTTPS-Verkehr.

[1] https://www.cloudflare.com/learning/dns/dns-over-tls/, Zugriff am 08.05.2022
[2] Ebd.
[3] Verma; Singanamalla (2020), Internetquelle
[4] Ebd.
[5] Ebd.
[6] Whittaker (2019), Internetquelle
[7] Cimpanu (2019), Internetquelle

4.2 Netzwerk-Dienste im lokalen Netzwerk

Ein DNS-Server ist durchaus nicht die einzige Möglichkeit für einen Server im Heimnetz. Der Bedarf an andere Dienste wird oftmals von Internet-basierte kommerzielle Cloud-Lösungen gedeckt, es besteht aber auch die Möglichkeit, diese lokal zu betrieben. Somit bleibt die Kontrolle über den eigenen Daten bei dem Benutzer, verbunden mit dem Nachteil, dass er sich selber um Konfigurationen und Wartung kümmern muss. Für die Ziele dieser Arbeit können wir die Netzwerkdienste in zwei Gruppen aufteilen:

- Netzwerkdienste, die transparent für das Netzwerk arbeiten können. Das Netzwerkgerät muss nur angeschlossen werden und es sind keine Änderungen des Netzwerks notwendig. Das Gerät hat keinen Bedarf an IP-Adresse, weil es nur in den unteren Schichten arbeitet. Typische Beispiele dafür sind Firewalls und Netzwerkfilter. Gesehen aus der Netzwerksicht funktionieren die Geräte als transparente Brücken. Diese Variante hat die einfachste Inbetriebnahme, ist aber stark an der Funktionalität begrenzt, weil keine Schichten über der Sicherungsschicht angesprochen werden können;

- Netzwerkdienste, die in den höheren Schichten arbeiten. Für die korrekte Arbeit dieser Geräte ist eine eigene IP-Adresse zwingend notwendig. Je nach dem, wie und wo das Gerät angeschlossen wird, muss eventuell das Netzwerk umkonfiguriert werden. Als Beispiele können die verschiedenste Server-Dienste genannt werden - wie etwa E-Mail-Server, Dateiserver, Web-Server, SSH-Server usw., oder einfach jede Anwendung, die aktiv am Netzwerkverkehr teilnimmt. Für die typische Stern-Konfiguration vom Heimnetz, wo auch die Einstellungen über DHCP gegeben werden, ist keine weitere Konfiguration oder Umstellung notwendig. Die einzige Voraussetzung wird das Vorhandensein eines DHCP-Clients auf dem neu-angeschlossenen Gerät sein.

Wie haben schon Engpässe diskutiert. Für unsere Ziele werden diese besonders attraktiv, weil an diesen Stellen einen Zugriff auf dem gesamten Netzwerkverkehr möglich ist. Für die Arbeit mit allen Netzwerkschichten wird eine eigene IP-Adresse notwendig sein. Zusätzlich wird das Folgegerät beeinflusst, indem seine Netzwerkeinstellungen nicht mehr funktionieren können. Das Problem ist auf Abbildung 6 auf der nächsten Seite dargestellt, für den Fall eines Anschlusses nach dem Heimnetzrouter in Richtung Betreiber. Das Netz wird unterbrochen und es entsteht ein neues Netz. Das neu-angeschlossene Gerät muss alle Pakete, die nicht für lokale Bearbeitung gemeint sind, weiterleiten. Eine NAT auf

dem neu-angeschlossenen Gerät wird notwendig sein, oder alternativ zusätzliche Software, die sich um DHCP und ARP zwischen den Subnetze kümmert[1].

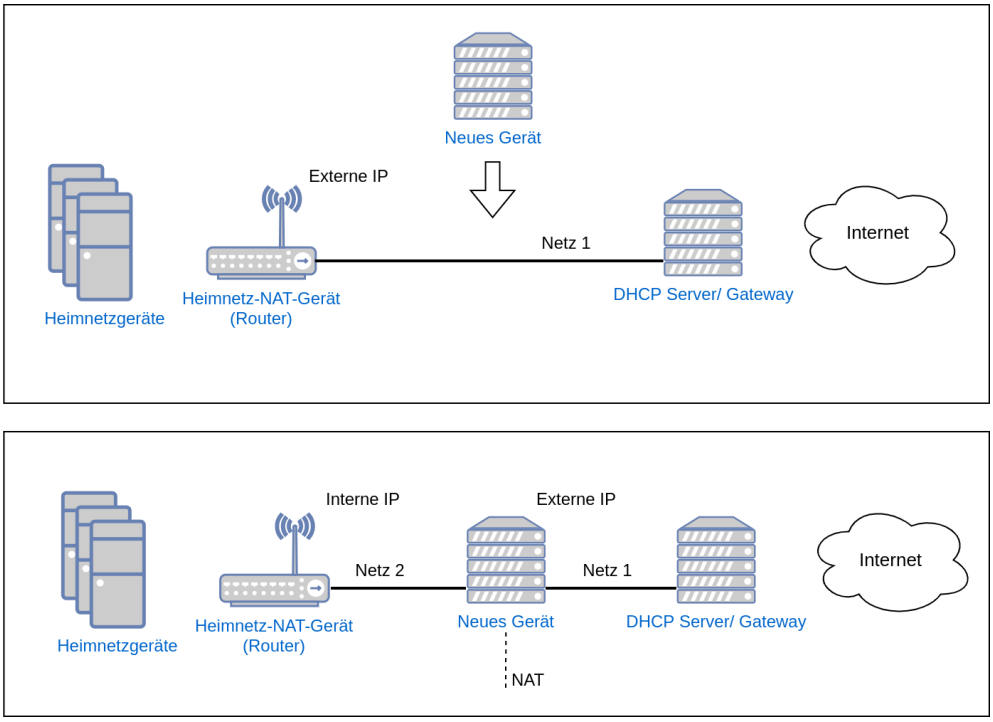

Abbildung 6: NAT in einem Engpass

4.3 Erreichbarkeit von Dienste aus dem Internet

Port-Weiterleitung / NAT-Traversal

Wie schon diskutiert ist NAT sehr verbreitet und praktisch der Standard bei Heimnetze[2]. Das ist mit Vor- und Nachteile verbunden. Einer der Vorteile - die Lösung des Problems mit dem Mangel an IP-Adressen - haben wir schon betrachtet. Anderer Vorteil ist der Schutz des Heimnetzes, indem die NAT nur Verbindungen erlaubt, die aus dem Lokalnetz stammen[3]. Dieser Vorteil kann aber zu einem Nachteil werden, wenn Dienste innerhalb des Lokalnetzes aus dem Internet erreichbar sein sollen. Es sind grundsätzlich zwei Ansätze denkbar, um dieses Problem zu lösen:

- Das NAT-Gerät kann manuell so konfiguriert werden, dass bestimmter Port immer zu einer bestimmten Kombination Rechner/Port im Lokalnetz übersetzt wird - die s.g. Port-Weiterleitung[4];

[1] https://wiki.debian.org/BridgeNetworkConnectionsProxyArp, Zugriff am 08.05.2022
[2] Vgl. Tanenbaum; Wetherall (2014), S.455
[3] Vgl. Network Working Group (2000), S.9
[4] Ebd.

- Die NAT kann auch umgegangen werden. Diese Ansätze sind unter dem Namen NAT-Traversal bekannt[1]. Das kann in der Form eines extra Rechners im Internet in der Rolle eines Vermittlers sein, oder auch in der Form von extra Software bei den beiden Kommunikationspartner, die die NAT umgehen möchten[2].

  Dynamic-DNS

Ein anderes Problem mit der Erreichbarkeit von Server im lokalen Netz ist das Problem der Adressierung. Wie schon im vorigen Kapitel diskutiert, werden standardmäßig für die Kommunikation im Internet Domains benutzt, die mittels DNS in IP-Adressen übersetzt werden. Leider ist bei Heimnetze die externe IP-Adresse, die vom ISP zugewiesen wird, nicht zuverlässig für die Registrierung einer Domain, weil sie meistens über DHCP zugewiesen ist und dadurch oftmals geändert[3].

Die Lösung dieses Problems besteht in der Installation einer Software im Lokalnetz, die regelmäßig die externe IP-Adresse prüft und diese bei einem Internet-basierten Server meldet, der auch die Domain-Einträge steuert. Eine DNS-Abfrage wird schließlich von diesem Server beantwortet und beinhaltet die aktuelle IP-Adresse, worüber das NAT-Gerät erreicht werden kann[4].

5 Konzept eines aktiven Teilnehmers ohne eigener IP-Adresse

Das Ziel dieser Arbeit ist die Vorstellung des Konzepts für einen aktiven Netzwerkteilnehmer ohne eigener IP-Adresse. So ein Konzept kann vielseitig eingesetzt werden. Es sind Szenarien denkbar, die eine unaufdringliche oder unauffällige Einmischung am Netz als Ziel haben, zum Beispiel um das Netz im Standardzustand zu testen, um den Verkehr eines spezifischen Teilnehmers abzufangen oder ähnliches. Allerdings, um den Umfang in Grenzen zu halten, werden wir uns auf ein bestimmtes Szenario fokussieren und zwar in der schon vorgestellten Umgebung des Heimnetzes. Als Zielgruppe kommt der Gelegenheitsbenutzer in Frage, der eine Lösung braucht, die „einfach“ funktioniert und nach dem Anschließen keine weitere Eingriffe benötigt.

[1] Tanenbaum; Wetherall (2014), S.454
[2] Vgl. Anderson (2020), Internetquelle
[3] https://www.cloudflare.com/learning/dns/glossary/dynamic-dns/, Zugriff am 08.05.2022
[4] Ebd.

Für die weitere Diskussion im Bezug auf den aktiven Netzwerkteilnehmer ohne eigener IP-Adresse werden die Begriffe „IP-lose Box“, „IP-loses Gerät“, „IP-loser Teilnehmer“ sowie einfach „die Box“ gebraucht.

5.1 Voraussetzungen und Annahmen

Anschlussposition:

Für den IP-losen Teilnehmer ist es irrelevant, wo er sich im Netz befindet, solange es nicht das letzte Gerät an der Abzweigung ist - es muss also immer wenigstens ein anderer Netzwerkteilnehmer dahinter angeschlossen sein. Für den Kontext unserer Beispielanwendung - der DNS-Filter - können die beste Ergebnisse an dem Engpass des Netzes erreicht werden, aus den Gründen, die schon diskutiert wurden. Bei dem typischen Heimnetz, das auf der Abbildung 5 auf Seite 30 angezeigt ist, sind folgende Stellen denkbar:

1. Nach dem Kabelmodem. In diesem Fall profitieren wir vom Zugang zu dem gesamten Netzverkehr, allerdings müssen wir dafür am DOCSIS-Protokoll teilnehmen. Das ist auch möglich, benötigt aber einen wesentlich komplizierten Ansatz, darunter extra Software, spezialisierte Hardware und zusätzlich Absprache mit dem Betreiber, der unser neu-angeschlossenes Gerät anerkennen muss, damit wir an der verschlüsselten Kommunikation teilnehmen können[1];

2. Zwischen dem Kabelmodem und dem Kundenrouter. Diese Stelle bietet die selbe Möglichkeiten und den selben unbegrenzten Zugang, zusammen mit dem Vorteil, dass die Kommunikation über Ethernet abläuft und somit mit Standard-Hardware und Software abfangbar. Leider ist diese Stelle nicht unbedingt immer vorhanden, wenn der Router und das Modem in einem Gerät kombiniert sind;

3. Vor dem Router im Heimnetz. Diese Stelle ist immer vorhanden und auch technisch leicht auszuführen. Der Nachteil dabei ist, dass das neu-angeschlossene Gerät nicht mehr Zugang zu dem kompletten Verkehr hat. Je nach dem, was das Ziel des Anschlusses ist, kann das ein Problem sein. Für unsere Beispielanwendung als DNS-Filter ist das durchaus problematisch, weil manche Netzwerkteilnehmer den neuen DNS-Server ignorieren können indem sie ihre eigene Einstellungen benutzen. Dieses Problem kann über Firewall-Regeln am Router umgegangen werden[2]. Ein anderes potentielles Problem

[1] Siehe dazu Kapitel 3 auf Seite 29
[2] Vgl. Prytuluk (2022), Internetquelle

dieser Konstellation ist, dass das neue Gerät hinter NAT steht und somit ohne weiteres nicht aus dem Internet erreichbar ist. Wenn keine solche Verbindung erwünscht ist, kommt die zusätzliche Isolation durch der NAT als Vorteil.

Für die Ziele dieser Arbeit werden wir annehmen, dass der Router und das Modem zwei separate Geräte sind. Wenn das nicht der Fall ist, ändert es nichts am Konzept, es wird nur etwas mehr vom Benutzer gefordert, wenn er den Verkehr des gesamten Netzes abfangen möchte. Er würde entweder Zugang zu seinem Router bekommen und die Firewall-Regeln einstellen, oder einen zusätzlichen privaten Router mit vollem Zugang besorgen, womit die benötigte Anschlussstelle geschaffen wird. Auf weitere Diskussion in dieser Richtung wird in dieser Arbeit verzichtet.

Externe IP-Adresse des Routers:

Ein anderes potenzielles Problem würde sich ergeben, wenn die IP-Adresse des Routers nicht aus dem Internet erreicht werden kann (weil es z.B. hinter einer extra NAT vom ISP ist):

Die externe IP-Adresse, die die Dynamic-DNS-Software feststellt, wird von einem anderen Gerät sein. Dieses kann dann aus dem Internet erreicht werden, allerdings haben wir dort keine Möglichkeit Port-Weiterleitung einzustellen und somit sind die Dienste unseres Heimnetzes unerreichbar. Lösungen dieses Problems sind denkbar, allerdings benötigen diese extra Schritte, wie etwa die Installation zusätzlicher Software für NAT-Traversal[1]. Um das vorgestellte Projekt nicht weiter zu komplizieren, nehmen wir an, dass die dem Router zugewiesene externe IP-Adresse aus dem Internet erreichbar ist.

5.2 Anforderungen

Die zahlreiche Möglichkeiten, private und Datenschutz-freundliche Dienste und Server im Heimnetz zu betrieben, ist vom wenigen Gebrauch für den Nutzer, wenn er die Software nicht installieren und konfigurieren kann. Wir streben nach einer Kombination aus maximalen Benutzerfreundlichkeit für die Gelegenheitsbenutzer bei bestehender Möglichkeiten für die erfahrene Benutzer, erweiterte Funktionen durchzuführen. Dabei definieren wir die Anforderungen an der IP-losen Anwendung wie folgt:

[1] Vgl. Anderson (2020), Internetquelle

1. Das Gerät muss als Plug-and-Play funktionieren können. Auch wenn der Benutzer überhaupt nichts mehr nach dem Anschließen macht, soll eine Grundfunktion vorhanden sein. Für das hier vorgestellte Szenario wird das die Funktion eines DNS-Filters sein. Erweiterte Funktionen sollen immer noch für fortgeschrittene Benutzer zugänglich sein;

2. Es dürfen keine Annahmen über die IP-Einstellungen des konkreten Heimnetzes gemacht werden. Diese müssen vom neuen Gerät selbst und automatisch erkannt werden;

3. Es soll ein Mittel vorhanden sein, um das Gerät automatisch zu aktualisieren, ohne Bedarf von Benutzer-Einmischung. Für die fortgeschrittene Benutzer sollen Logs mit detaillierter Information verfügbar sein;

4. Durch geeignete Mittel sollen einfachen Angaben über der Funktion und dem Zustand des Gerätes ausgegeben werden. Diese sollen auch für unerfahrene Benutzer leicht zugänglich und verständlich sein;

5. In dem Fall eines Problems muss das ganze System leicht im Ausgangszustand versetzt werden kann. Als Ausgangszustand wird in der vorgestellten Lösung der Brückenmodus sein.

5.3 Konzeptentwicklung

Netzkonfiguration

Wir haben schon diskutiert, wie die Netzwerkteilnehmer allgemeine Regeln beachten, um eine Kommunikation zwischen entfernte Partner zu ermöglichen. Geräte, die sich nicht an den Netzwerkregeln halten, werden im besten Fall ignoriert oder können im schlimmsten Fall Störungen im Netz verursachen. Dabei gibt es einen wichtigen Punkt, denn wir in dieser Arbeit ausnutzen wollen - das Verhalten eines Netzwerkteilnehmers wird von den anderen Geräte beurteilt und solange die empfangene Daten die Regeln beachten, werden sie auch weitergeleitet und verarbeitet. Für uns bedeutet das, dass unser Gerät nicht unbedingt konventionelles Verhalten haben muss, solange das von keinem anderen Teilnehmer als problematisch wahrgenommen wird.

Eine Voraussetzung für die Funktion des neuen Geräts ist die Positionierung zwischen zwei andere, aktiv am Netzverkehr teilnehmende Geräte und zwar am besten an den Grenzen des Heimnetzwerks. Die Netzwerkregeln besagen, dass jedes Gerät, das an IP-Kommunikation teilnehmen möchte eine eigene IP-Adresse haben muss. Wir haben schon diskutiert, wie aus diesem Prinzip in der Situation

zwei Netze entstehen würden (siehe Abbildung 6 auf Seite 38). Betrachten wir jetzt die selbe Situation aus der Sicht der beiden äußeren Geräte, bevor das neue Gerät inzwischen angeschlossen wird.

Der Heimnetzrouter leitet das ganze Netzverkehr, das nicht für das Lokalnetz bestimmt ist, an dem Gateway weiter. Dazu benötigt der Router seine IP-Adresse und seine MAC-Adresse. In dem typischen Szenario, was wir betrachten, bekommt der Router selber seine Netzkonfigurationsdaten über DHCP aus dem Gateway, der einen DHCP-Server betreibt. Die Kommunikation verläuft auf der Netzwerkebene (i.e. bis auf die Transportschicht) und basiert auf IP-Adressen, MAC-Adressen und Netzmasken. Dadurch, aus der Sicht des Routers, wenn das nächst-angeschlossenes Gerät die Daten vom Gateway hat, dann ist das das Gateway. Andersherum gilt das für die Gateway-Seite.

Definieren wir dann das Grundprinzip des IP-losen Gerätes:

Das IP-losen Gerät benutzt die selbe Netzdaten wie das Gateway, wenn gesehen aus der Seite des Heimnetzrouters und die Netzdaten vom Heimnetzrouter, wenn gesehen aus der Seite des Gateways.

Daraus folgt, dass das neue Gerät für die andere Teilnehmer praktisch nicht existiert und sie haben keinen Bedarf ihre Konfigurationen zu ändern.

Aktive Teilnahme am Netzverkehr

Die beschriebene Netzkonfiguration entspricht die Konfiguration einer Netzbrücke und ist somit nichts neues oder interessantes. In dieser Arbeit zielen wir zusätzlich eine aktive Teilnahme am Netzverkehr. Das ist für eine Netzbrücke unmöglich, weil sie keine eigene IP-Adresse hat. Die IP-lose Box hat auch keine eigene IP-Adresse, kann aber eine fremde benutzen und zwar die Adresse des nächst-angeschlossenen Geräts - der Heimnetzrouter. Betrachten wir im Einzelnen, wie das möglich wird.

Wenn ein Paket bei der IP-losen Box ankommt, ist das Paket sicherlich nicht direkt für sie bestimmt, weil sie im Netz quasi nicht existiert. Sie hat aber voller Zugang auf das Paket und kann dieses analysieren:

- Wenn das Paket für die Box bestimmt ist[1], kann sie darauf reagieren, indem sie eine Antwort mit den Daten des nächst-angeschlossenen Gerätes antwortet - z.B. bei einer Antwort an Anfrage aus dem Internet, wird sie die Daten vom Heimnetzrouter benutzen;
- Wenn das Paket für jemand anders bestimmt ist, hat sie dann zwei Optionen:

[1] Wie sie das feststellt, wird weiter unten diskutiert.

– Das Paket kann einfach weitergeleitet werden, ohne dabei etwas zu ändern;
– Das Paket kann verworfen werden - als Teil der Funktion des Netzfilters. Die strategische Position an der Grenze des Netzwerks ist für solche Aufgaben sehr gut geeignet.

Anschaulich ist das Prinzip auf Abbildung 7 vorgestellt.

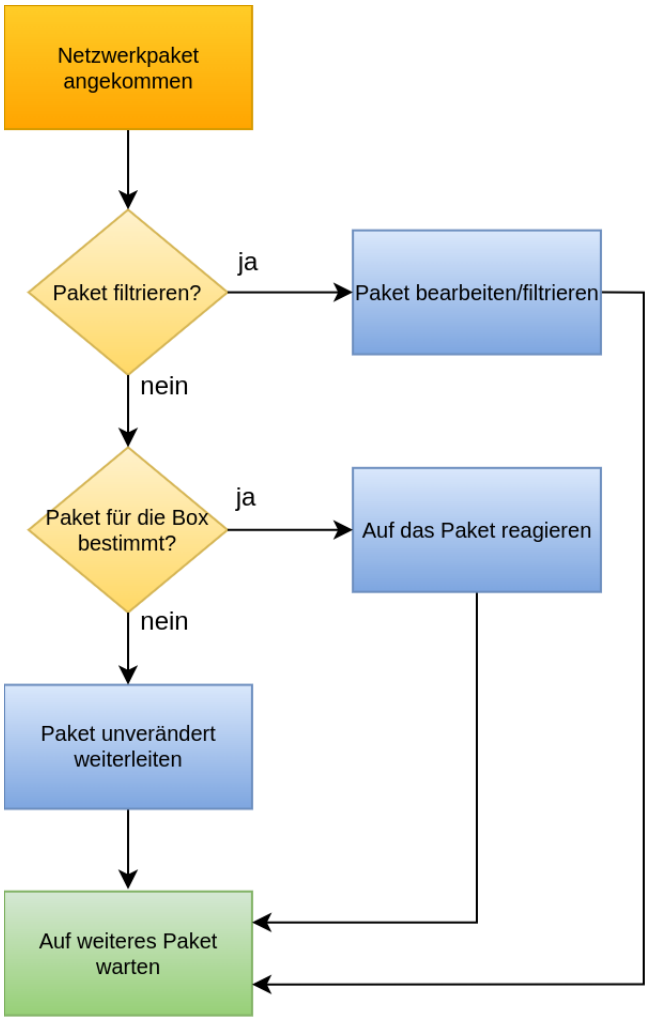

Abbildung 7: IP-lose Box: IP-Funktion

Ansprechen der IP-losen Box

Das vorgestellte Prinzip führt zu einer Situation, die einer NAT ähnelt - es existieren mehrere Geräte hinter einer einzigen IP-Adresse. Dafür können wir die gewährte Ansätze der NAT-Netzwerke auch hier benutzen. Die vom internen Netz

startende Kommunikation wird nach außen durchgelassen. Die zwei mögliche Szenarien für Pakete, die von außen kommen, sind die Initiierung einer Kommunikation von außen und die Weiterführung einer Kommunikation, die von einem internen Gerät gestartet wurde:

- Bei dem ersten Szenario passiert die Adressierung eines internen Gerätes hinter der NAT durch das Mapping eines Ports. Wenn eine Anfrage an diesem Port ankommt, wird sie an dem gemappten Gerät direkt weitergeleitet;
- Wenn die Kommunikation vom einem internen Gerät gestartet ist, pflegt das NAT-Gerät seine Tabelle mit Information darüber. Wenn die Antwort kommt, kann aufgrund dieser Tabelle entschieden werden, welches Gerät hinter der NAT gemeint ist.

Die IP-lose Box kann die selbe Prinzipien anwenden. Die Dienste, die auf der Box laufen, können auch durch gesonderte Port-Mappings aus dem Internet ansprechbar sein. Ähnlich kann die Box aus der Seite des Heimnetzes erreicht werden. Allerdings, wie wir später in der Implementierung sehen werden ist das mit Komplikationen verbunden. Eine mögliche Lösung ist die Benutzung einer realen IP-Adresse aus dem Internet, die nach bestimmte Kriterien für die Benutzer des Heimnetzes uninteressant ist. Die Clients können ihre Anfrage an dieser IP-Adresse ganz normal starten wie bei einer Kommunikation mit einem Internet-Host. Die Anfrage wird dann aber von der IP-losen Box abgefangen. Die Kommunikation über dieser Adresse erfolgt tatsächlich mit der Box.

Die Lösung für das Ansprechen aus der Internet-Seite ist leider nicht ohne Probleme. Das Gerät hinter der IP-lose Box weißt nichts über ihre Existenz und noch weniger über ihre Konfiguration und kann unter Umstände versuchen den selben Port zu benutzen, der für die Erreichbarkeit des IP-losen Gerätes benutzt ist. Eine mögliche Lösung wäre die Benutzung anderer Methoden für die Adressierung, wie z.B. Port-Knocking[1]. Andere Lösungen sind auch denkbar und hängen zusätzlich von der Implementierung ab.

Funktion in der Sicherungsschicht

Daten der Sicherungsschicht werden grundsätzlich nicht im nächsten Subnetz weitergeleitet, mit der Ausnahme einer ARP-Proxy[2]. Für die IP-lose Box wollen wir die beide Ansätze kombinieren:

[1] https://wiki.nftables.org/wiki-nftables/index.php/Port_knocking_example, Zugriff am 08.05.2022

[2] https://wiki.debian.org/BridgeNetworkConnectionsProxyArp, Zugriff am 08.05.2022

- ARP-Verkehr muss weitergeleitet werden;
- Vor der Weiterleitung muss die Information abgelesen werden, um den ARP-Cache der Box zu pflegen;
- Die Box muss auch eigene Anfragen schicken können. Diese werden jeweils mit der MAC-Adresse des nächst-angeschlossenen Gerätes in der entgegenstehenden Richtung versehen und geschickt.

Ein Seiteneffekt dieses Ablaufs ist, dass alle ARP-Antworten weitergeleitet werden, auch wenn sie auf Anfrage der IP-losen Box gekommen sind. Dies kann auch als Vorteil gesehen werden, indem die Empfänger ihren Cache häufiger aktualisieren. Grafisch ist das ARP-Konzept auf Abbildung 8 vorgestellt.

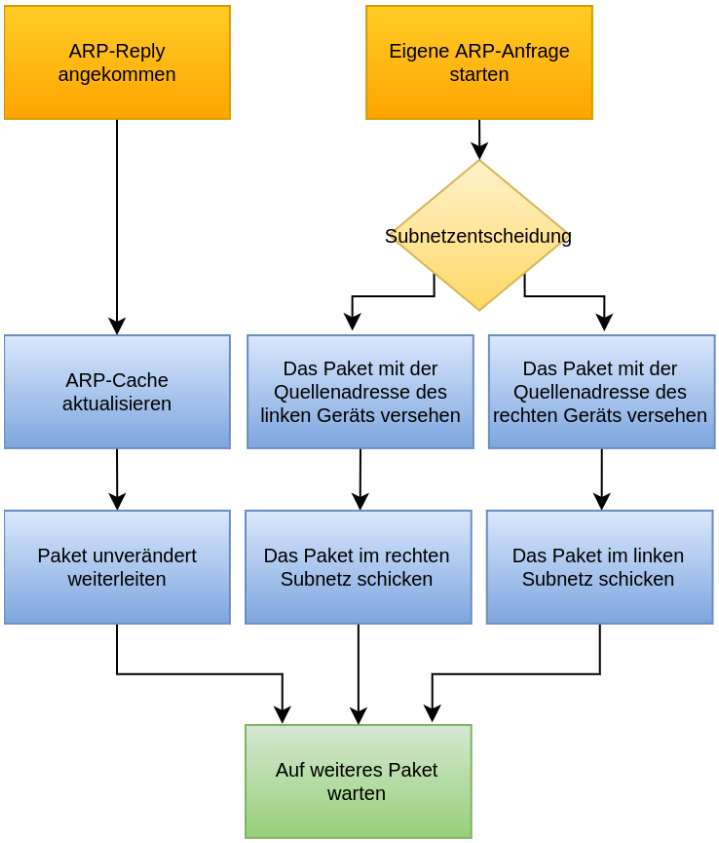

Abbildung 8: IP-lose Box: ARP-Funktion

Datenbeschaffung

In der bisherigen Diskussion sind wir davon ausgegangen, dass die Netzdaten der beiden Nachbarn-Geräte bekannt sind. Das ist aber nicht der Fall bei dem Anschluss und wir dürfen es auch nicht vom Benutzer verlangen, wie schon in

den Anforderungen definiert. Die benötigte Daten müssen von der Box selbst beschaffen werden.

Eine reibungslose Funktion mit minimaler Unterbrechung des Netzes kann erreicht werden, wenn die IP-lose Box direkt nach dem Anschließen sich als eine Netzbrücke verhält - alle Kommunikation werden unverändert weitergeleitet - IP sowie auch ARP. Dabei kann die Box den Verkehr beachten und die benötigte Information herausfinden. Betrachten wir die mögliche Pakete, die über der Brücke passieren können:

- In den Empfänger-Felder sind zwei Kombinationen denkbar:
  - Die MAC-Adresse und die IP-Adresse vom Router, wenn Pakete für den Router gemeint sind;
  - Die MAC-Adresse vom Gateway und eine externe IP-Adresse, wenn die Pakete für ein externes Gerät gemeint sind[1].
- Die Kombinationen für die andere Richtung sind genau umgekehrt, nur mit den Felder für Absender.

Mit diesen Beobachtungen können wir aus wenigen abgehörten Pakete die benötigte Information extrahieren:

- Es werden ausschließlich zwei MAC-Adressen beobachtet. Die MAC-Adresse mit der konstanten IP-Adresse gehört dem Heimnetzrouter und die andere - dem Gateway;
- Mit der so gewonnenen MAC-Adresse vom Gateway, können wir eine umgekehrte ARP Anfrage starten, womit wir seine IP-Adresse gewinnen. Eine zweite, mehr passive Option, ist die weitere Beobachtung des Netzes bis ein ARP-Reply vom Gateway ankommt.

Starten und Zurücksetzen

Die IP-lose Box kann in Brückenmodus starten. Wenn genug Netzverkehr beobachtet ist und die benötigte Daten gewonnen sind, kann eine Umschaltung zu einem aktiven Betrieb erfolgen. Als Fallback-Lösung kann die Box immer zurück in Brückenmodus versetzt werden.

Weitere Anforderungen

[1] Den Fall, dass Pakete für den Gateway gemeint sind, betrachten wir nicht. Das ändert aber nichts am Prinzip.

Die Erfüllung der weiteren Anforderungen, darunter die verschiedene Arte von Meldungen, die Rücksetzung und die Aktualisierungen werden der Implementierung überlassen.

6 Prototypische Implementierung

Betrachten wir jetzt eine mögliche Implementierung in dem schon beschriebenen Anwendungsszenario - Anschluss an der Grenze des Heimnetzes zwischen dem Heimnetzrouter und dem Betreibergerät (Modem/Gateway/etc.) mit dem Betrieb eines DNS-Filters. Die Box wird über SSH erreichbar sein - vom internen (Heimnetz) sowie auch von externen Netz (Internet). Die Standardeinstellung beim Anschluss wird der Brückenmodus sein, der nachfolgend zu einem vollen Betrieb umgeschaltet wird.

Die einfachste Variante des Netzwerkmodells aus der Abbildung 6 ist, wenn ein DHCP-Server am Gateway und ein DHCP-Client am Heimnetzrouter vorhanden sind. Zwischen den beiden Geräten kann ein neues Gerät mit zwei Netzwerkkarten angeschlossen werden, das am Netz zum Router ein DHCP-Server anbietet und am Netz zum Gateway selber durch DHCP eine eigene Adresse bekommt. Zwischen den Netzwerkkarten kann *ip_forward*[1] eingeschaltet werden und die Kommunikation zwischen den beiden Netzsegmenten kann über eine NAT erfolgen. Das neu-angeschlossene Gerät kann dabei auch selber aktiv am Netz teilnehmen und sogar bei Bedarf die MAC-Adresse vom Heimnetzrouter an der externen Netzwerkkarte kopieren. Dabei sind keine Änderungen an beiden vorhandenen Geräte notwendig. Dieser einfacher Ansatz fällt aber komplett durch, wenn statische Einstellungen benutzt werden, oder wenn keine Änderung an der IP-Adresse vom Heimnetzrouter erwünscht ist. Aus diesen Gründen verfolgen wir einen etwas komplizierten Ansatz, der auch wesentlich flexibler bei der Anwendung ist.

Der Ansatz basiert auf einer Trennung der beiden Netzwerkschnittstellen durch Virtualisierung voneinander. Das würde uns erlauben Routen und Regeln zu benutzen, die sich sonst widersprechen werden. Im Prinzip wollen wir zwei separate Netzwerk-Hosts hinter derselben IP-Adresse verstecken. Dieses Verstecken erinnert an NAT und das ist genau, was wir benutzen - dabei zweifach um die gleichzeitige Benutzung der beiden gleichen Adressen zu erlauben. Die Virtualisierung erfolgt über Linux-Namensräume, die miteinander über einem Paar von virtuellen Netzwerkschnittstellen verbunden sind. Als gut dafür geeignet zeigen

[1] https://www.kernel.org/doc/html/latest/networking/ip-sysctl.html, Zugriff am 08.05.2022

sich die VETH-Geräte („Virtual Ethernet“), die ein Teil des Linux-Kernels sind[1]. Die beide Namensräume bekommen jeweils eine Seite des Paars zugewiesen und können darüber kommunizieren. Die Protokolle wie ARP und teilweise DHCP, die auf untere Schichten arbeiten, werden wir gesondert behandeln. Ein Überblick der Implementierung zeigt die Abbildung 9

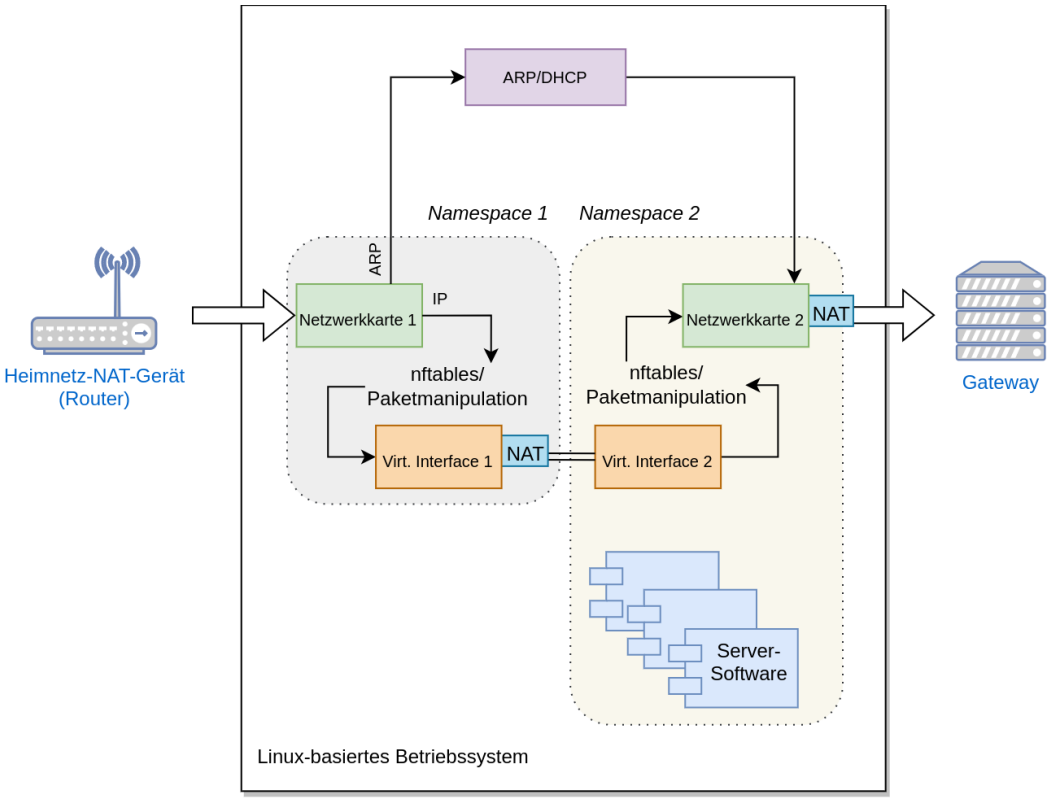

Abbildung 9: IP-lose Box: Überblick der internen Funktion

Die Steuerung der Box werden Skripten übernehmen, die automatisch beim Hochfahren gestartet werden und auf bestimmte Ereignisse oder Benutzerbefehle reagieren.

Fangen wir zuerst mit der Hardware an.

6.1 Hardware, Betriebssystem und Software

Hardware

Die Auswahl der Hardware muss einen Kompromiss zwischen Leistung, Preis und Verfügbarkeit machen. Die Kriterien würden natürlich auch für jeden konkreten Einsatz variieren, abhängig von dem Typ und der Anzahl an Dienste, die

[1]Vgl. Liu (2018), Internetquelle

auf der Box laufen werden, sowie auch von der Größe und den Bedürfnisse des Netzwerks dahinter. Es müssen ausführliche Tests mit verschiedene Hardware-, Software- und Netz-Konfigurationen ausgeführt werden, um die optimale Parameter für jeden Einsatz zu finden. Solche optimale Parameter zu finden ist außerhalb des Themengebiets dieser Arbeit. Für die hier gezielte Demonstration des vorgestellten Konzepts würde uns ein Prototyp reichen, der leicht verfügbar und schnell programmierbar ist, auch wenn nicht mit optimalen Hardware-Eigenschaften.

Aus diesen Gründen benutzen wir einen Raspberry-Pi 3B - Einplatinencomputer, der mit Linux betrieben wird und über GPIO-Pins verfügt[1]. Für die Funktion als IP-lose Box werden zwei Netzwerkkarten erforderlich und der Pi hat nur eine. Die Funktion der zweiten Karte übernimmt ein handelsüblicher USB-zu-Ethernet Adapter.

Betriebssystem

Als Betriebssystem installieren wir das für den Raspberry-Pi empfohlene, Debian-basierte „*Raspberry-Pi OS*“[2]. Für die weitere Arbeit mit dem System helfen uns neben der offiziellen Raspberry-Pi-Dokumentation auch die Handbücher und Referenzen von Debian[3]. Für den Standardterminal - *bash*, benutzen wir das offizielle bash-Handbuch[4].

Für die Installation des Betriebssystems installieren wir die „leichte“ Variante von Raspberry-Pi OS, weil wir kein Bedarf von grafischen Anwendungen haben. Die zurzeit aktuelle Version ist „January 28th 2022“, basierend auf Debian 11. Nach der Anleitung[5], prüfen wir die Checksumme und übertragen wir die Datei auf einer SD-Karte:

Listing 1: Raspberry-Pi: Übertragung der Installationsdatei

```
1 $ sha256 -c 2022-01-28-raspios-bullseye-armhf-lite.zip.sha256
2     2022-01-28-raspios-bullseye-armhf-lite.zip: OK
3 $ unzip 2022-01-28-raspios-bullseye-armhf-lite.zip
4 $ sudo dd bs=4M if=2022-01-28-raspios-bullseye-armhf-lite.img of=/dev/sdf conv
    =fsync status=progress
```

Um die Arbeit mit dem Pi zu vereinfachen, erlauben wir SSH-Zugriff direkt nach dem Starten. Zusätzlich wollen wir sicherstellen, dass der Pi im Notfall

[1] https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#raspberry-pi-3-model-b, Zugriff am 08.05.2022

[2] https://www.raspberrypi.com/documentation/computers/os.html, (Zugriff am 08.05.2022)

[3] Beaupré et al. (2022), Internetquelle

[4] Ramey; Fox (2020)

[5] https://www.raspberrypi.com/documentation/computers/getting-started.html#installing-images-on-linux, (Zugriff am 08.05.2022)

über einem zweiten Weg erreichbar ist. Dafür aktivieren wir den Zugang über der seriellen Konsole:

1. SSH aktivieren, durch Erstellung einer leeren Datei namens „*ssh*“ im „*/boot*“ Ordner[1];
2. Änderungen in */boot/config.txt* für die Einschaltung des seriellen Zugangs[2]*:*

Listing 2: Raspberry-Pi: Einschalten von UART

```
1 # Enable UART
2 enable_uart=1
```

„*console=serial0*“ war schon im „*/boot/cmdline.txt*“ von unserem Pi eingestellt, deshalb gab es keinen Bedarf an weiteren Änderungen in dieser Datei. Eine serielle Konsole ist über den GPIO-Pins 14 (TX), 15 (RX) und der daneben stehende GND verfügbar[3]. Die Nummer der GPIO-Schnittstellen entsprechen nicht der physikalischen Ordnung der Pins und können aus der Abbildung 18 auf Seite 90 im Anhang entnommen werden. Bei dem Anschluss ist es wichtig auf die korrekte Spannung zu achten - die GPIO-Pins arbeiten mit 3,3 Volt.

Für die Arbeit mit der seriellen Konsole benutzten wir einen handelsüblichen USB-zu-Seriell Adapter und die Software *cutecom*[4]. Um Zugriff für den normalen Benutzer am Entwicklungsrechner zu erlauben, muss er zu der *„dialout“* Gruppe zugefügt werden[5]:

Listing 3: Benutzerrechte für die Arbeit mit der seriellen Konsole

```
1 $ sudo usermod -a -G dialout ${USER}
```

Neustart und Standardeinstellungen

Mit diesen Einstellungen kann der Pi schon angeschlossen und gestartet werden. Die IP-Einstellungen kann er automatisch über DHCP bekommen. Wie auf Abbildung 6 auf Seite 38 dargestellt und im früheren Kapitel diskutiert, würde der

[1] Vgl. https://www.raspberrypi.com/documentation/computers/configuration.html#setting-up-a-headless-raspberry-pi, (Zugriff am 08.05.2022)

[2] https://www.raspberrypi.com/documentation/computers/config_txt.html#enable_uart, (Zugriff am 08.05.2022)

[3] https://www.raspberrypi.com/documentation/computers/os.html#gpio-and-the-40-pin-header, (Zugriff am 08.05.2022)

[4] https://packages.debian.org/bullseye/cutecom, (Zugriff am 08.05.2022)

[5] https://www.raspberrypi.com/documentation/computers/configuration.html#changing-your-username, (Zugriff am 08.05.2022)

direkte Anschluss am Engpass zu einer Unterbrechung des Netzzugangs für alle Geräte hinter der Box führen, weil der Pi im Startzustand keine NAT anbietet. Um solche Störungen zu minimieren, schließen wir die Box zuerst im Heimnetz an, bis alle Software heruntergeladen wird und die Konfiguration vollendet ist.

Wenn der Pi hochgefahren ist, führen wir die Standardmaßnahmen durch, um den Zugang zu sichern[1] - Änderung des Passworts, Einstellungen von „*sudo*“, „*ssh*“, System Aktualisierungen usw. Zusätzlich installieren wir zusätzliche Tools, die bei der Entwicklung helfen. Die Überwachung des Netzverkehrs wird ausschließlich über *tcpdump* erfolgen.

Listing 4: IP-lose Box: Installation von Software

```
1  sudo apt install vim tmux mc git nethogs tcpdump unattended-upgrades gawk \
2     dnsutils # ... und weitere
```

Automatische Aktualisierungen

Der „*unattended-upgrades*“ Skript kann sich um die automatische Aktualisierung des Rechners kümmern[2]. Hinweise für die Konfiguration entnehmen wir aus der Datei */etc/apt/apt.conf.d/50unattended-upgrades*. Nach der gewünschten Änderungen in der Datei testen wir die korrekte Funktion mit *sudo unattended-upgrade -d*.

Listing 5: Raspberry Pi: Quellen für automatische Softwareaktualisierungen

```
1  pi@raspberrypi:~ $ sudo apt-cache policy
2  Package files:
3   100 /var/lib/dpkg/status
4       release a=now
5   500 http://archive.raspberrypi.org/debian bullseye/main armhf Packages
6       release o=Raspberry Pi Foundation,a=stable,n=bullseye,l=Raspberry Pi
7  Foundation,c=main,b=armhf
8       origin archive.raspberrypi.org
9   500 http://raspbian.raspberrypi.org/raspbian bullseye/rpi armhf Packages
10      release o=Raspbian,a=stable,n=bullseye,l=Raspbian,c=rpi,b=armhf
11      origin raspbian.raspberrypi.org
12  500 http://raspbian.raspberrypi.org/raspbian bullseye/non-free armhf Packages
13      release o=Raspbian,a=stable,n=bullseye,l=Raspbian,c=non-free,b=armhf
14      origin raspbian.raspberrypi.org
15  500 http://raspbian.raspberrypi.org/raspbian bullseye/contrib armhf Packages
16      release o=Raspbian,a=stable,n=bullseye,l=Raspbian,c=contrib,b=armhf
17      origin raspbian.raspberrypi.org
```

[1] https://www.raspberrypi.com/documentation/computers/configuration.html#securing-your-raspberry-pi, (Zugriff am 08.05.2022)

[2] https://wiki.debian.org/UnattendedUpgrades, (Zugriff am 08.05.2022)

```
18   500 http://raspbian.raspberrypi.org/raspbian bullseye/main armhf Packages
19       release o=Raspbian,a=stable,n=bullseye,l=Raspbian,c=main,b=armhf
20       origin raspbian.raspberrypi.org
21   Pinned packages:
```

Die Datei */etc/apt/apt.conf.d/50unattended-upgrades* sieht nach den Änderungen so aus (unbenutzte Segmente gekürzt):

Listing 6: Raspberry Pi: Einstellung von automatischen Softwareaktualisierungen

```
1   Unattended-Upgrade::Origins-Pattern {
2           "o=Raspberry Pi Foundation,a=stable,n=bullseye,l=Raspberry Pi Foundation,c=main";
3           "o=Raspbian,a=stable,n=bullseye,l=Raspbian,c=rpi";
4           "o=Raspbian,a=stable,n=bullseye,l=Raspbian,c=non-free";
5           "o=Raspbian,a=stable,n=bullseye,l=Raspbian,c=contrib";
6           "o=Raspbian,a=stable,n=bullseye,l=Raspbian,c=main";
7   };
8
9   // Python regular expressions, matching packages to exclude from upgrading
10  Unattended-Upgrade::Package-Blacklist {
11  };
12
13  // Remove unused automatically installed kernel-related packages
14  // (kernel images, kernel headers and kernel version locked tools).
15  Unattended-Upgrade::Remove-Unused-Kernel-Packages "true";
16
17  // Do automatic removal of newly unused dependencies after the upgrade
18  Unattended-Upgrade::Remove-New-Unused-Dependencies "true";
19
20  // Do automatic removal of unused packages after the upgrade
21  // (equivalent to apt-get autoremove)
22  Unattended-Upgrade::Remove-Unused-Dependencies "true";
23
24  // If automatic reboot is enabled and needed, reboot at the specific
25  // time instead of immediately
26  // Default: "now"
27  Unattended-Upgrade::Automatic-Reboot-Time "02:00";
```

Ausschaltung von Dienste

Gewisse Dienste, die Teil der Standardinstallation sind, können für unsere Ziele störend wirken. Darunter zählen:

- nftables.service - dient zur Steuerung der Firewall[1];
- dhcpcd.service - dient zur automatischen Übernahme von Netzwerkeinstellungen über DHCP[2].

[1] https://manpages.debian.org/bullseye/nftables/nftables.8.en.html, (Zugriff am 08.05.2022)
[2] https://manpages.debian.org/bullseye/dhcpcd5/dhcpcd.8.en.html, (Zugriff am 08.05.2022)

Beide schalten wir über *systemctl* aus, damit sie auch nicht beim Neustart wieder gestartet werden[1].

Routing

Die Box wird Routing-Funktionen ausführen. Dafür muss sie auch Pakete zwischen den Netzwerksschnittstellen weiterleiten. Dafür ist der *ip_forward* Kernel-Parameter zuständig[2]. Die Einstellung würde auch dauerhaft bleiben, wenn sie in der Konfigurationsdatei */etc/sysctl.conf* eingetragen wird. Die Einstellung kann mittels „*sudo sysctl -p /etc/sysctl.conf*“ sofort übernommen werden[3].

Mit diesen Vorbereitungen kann schon die Entwicklung der Steuerungsskripten beginnen.

6.2 Brückenmodus und Netzwerkverkehrsanalyse

Die sicherste Einstellung um Störungen am Netz zu vermeiden ist der Brückenmodus - bei diesem werden die beide Netzwerkschnittstellen zusammen geschaltet und alle Netzwerkschichten werden transparent durchgelassen[4]. Aktuell ist das Softwarepaket *iproute2* dazu zuständig[5]. In dem beigefügten Steuerungsskript ist die Funktion „*set_bridge*“ ab Zeile 270 für die Umstellung in Brückenmodus zuständig.

Dieser wenig spektakulärer Betrieb ist auf Abbildung 10 angezeigt. Der Brückenmodus ist aber für uns von besonderen Bedeutung, weil er uns erlaubt, Netzwerkverkehr vollständig passiv zu beobachten - d.h. ohne jeglicher Einmischung am Netz. Durch diese Beobachtung können wir Daten sammeln, deren Analyse uns die Parameter liefern wird, die für die erweiterte Funktionen der Box notwendig sind. Alle eigene Netzwerk-Funktionen der Box sind in dieser Phase ausgeschaltet.

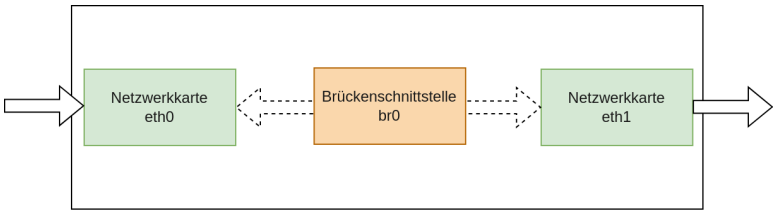

Abbildung 10: IP-lose Box: Brückenmodus

[1] https://manpages.debian.org/bullseye/systemctl/systemctl.1.en.html, (Zugriff am 08.05.2022)
[2] Benvenuti (2006), S.967
[3] https://www.man7.org/linux/man-pages/man5/sysctl.conf.5.html, (Zugriff am 08.05.2022)
[4] https://wiki.linuxfoundation.org/networking/bridge, (Zugriff am 08.05.2022)
[5] Ebd.

Die Analyse kann zuerst mit Feststellung der benötigten Daten anfangen:

1. Die Box muss feststellen in welcher Richtung sie funktionieren soll - d.h. welche Netzwerkkarte am internen Netz, und welche - am externen Netz angeschlossen ist. Wir werden weiter unten sehen, warum das eine wichtige Rolle spielt;

2. Die Box muss die beide für uns wichtigen Netzwerkteilnehmer erkennen - den Heimnetzrouter und das ISP-Gateway. Von beiden sind folgende Daten notwendig:

   (a) MAC-Adressen

   (b) IP-Adressen

   (c) Netzmaske

   (d) Hostname[1]

   (e) DNS-Server[2]

Betrachten wir eine Verallgemeinerung des Netzverkehrs, der durch die Box passieren kann. Die weitere Überlegungen basieren auf Abbildung 11.

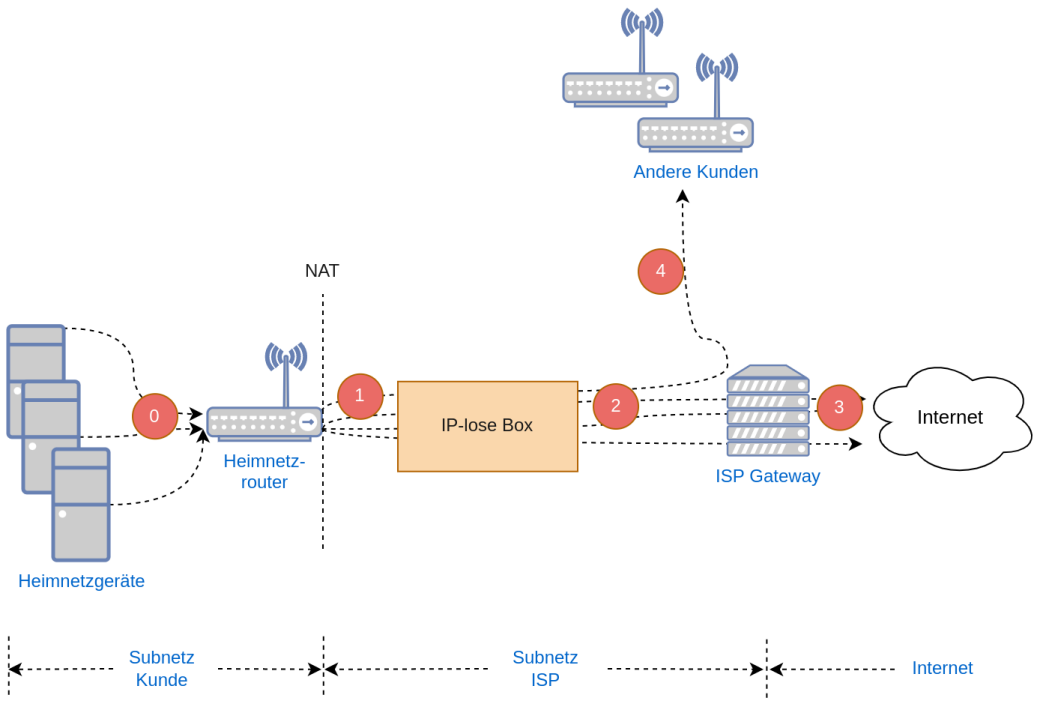

Abbildung 11: IP-lose Box: Netzverkehrsanalyse

[1]Dieser wird eine Rolle in DHCP spielen, ansonsten nicht dringend wichtig.
[2]Wie oben.

Der Verkehr „0“ stammt aus dem Heimnetz und aufgrund der NAT am Router wird im Verkehr „1“ umgewandelt. Der Verkehr „2“ besteht aus den Kommunikationen „3“ und „4“. Für den Ansatz an der Grenze des Heimnetzes ist die Kommunikation „4“ weniger wahrscheinlich, werden wir aber mitbetrachten, um das Konzept flexibel zu halten[1].

Für die aus der Box beobachtete Pakete können folgende Schlussfolgerungen gezogen werden:

- Netzwerkverkehr „1“ mit Richtung „2“:
  - Source-IP-Adresse: immer die selbe (die externe Adresse vom Router);
  - Destination-IP-Adresse: unterschiedlich, je nach Empfänger;
  - Source-MAC-Adresse: immer die selbe (die externe Netzwerkkarte vom Router);
  - Destination-MAC-Adresse: eventuell unterschiedlich, je nach dem ob die Kommunikation „4“ stattfindet. Wir machen die Annahme, dass der Anteil der Kommunikation „3“ wesentlich mehr als der Anteil von „4“ ist.
- Netzwerkverkehr „2“ mit Richtung „1“: Gespiegelte Version von „1“ zu „2“

Wenn wir die Netzwerkkarte zum Heimnetzrouter als die “linke“ und die Netzwerkkarte zum externen Netz als die „rechte“ bezeichnen gilt auch folgendes, wenn wir die Richtung des Verkehrs in Absicht nehmen:

- Alle Netzwerkpakete, die in die „linke“ Karte eingehen, haben als Source-MAC-Adresse die Adresse vom Router;
- Alle Netzwerkpakete, die aus der „rechten“ Karte auskommen, haben als Source-MAC-Adresse die Adresse vom Router.

IP-Adressen; MAC-Adressen; Richtung

Mit diesen Überlegungen ergeben sich die Algorithmen für die Feststellung aller Daten. Mithilfe vom Algorithmus 1 auf der nächsten Seite werden die IP-Adressen und die MAC-Adressen vom Router und vom Gateway berechnet. Der Algorithmus startet mit einer der beiden Netzwerkkarten und sammelt ein- und ausgehende Daten, die danach analysiert werden. Mit diesen Daten arbeitet auch

[1] Zum Beispiel für einen Einsatz innerhalb des Heimnetzes

der Algorithmus 2 auf der nächsten Seite, der die Richtung der im Algorithmus 1 ausgewählten Netzwerkkarte feststellt.

Algorithmus 1 Identifizierung von Nachbargeräte

1. Genug Netzwerkverkehr an beliebiger Netzwerkkarte beobachten. Die Karte vermerken. Dabei ein- und ausgehende Informationen separat sammeln. Ein Kompromiss zwischen Zuverlässigkeit der Berechnungen und die Geschwindigkeit, womit die Daten dafür gesammelt werden, lässt sich empirisch festlegen;

2. Aus alle gesammelten Daten die teilnehmenden MAC-Adressen extrahieren;

3. Die MAC-Adressen aus dem letzten Schritt nach Häufigkeit sortieren;

4. Die zwei meist-benutze MAC-Adressen sind die Adressen vom Router und vom Gateway;

5. Jetzt eine Sammlung aus alle getroffenen Kombinationen in der Form: MAC-Adresse / IP-Adresse ausrechnen;

6. Von den beiden in Schritt 4. berechneten MAC-Adressen gehört eine Adresse immer zu derselben IP-Adresse - so lässt sich der Heimnetzrouter identifizieren;

7. Die andere MAC-Adresse aus Schritt 4. muss dafür mindestens zwei IP-Adressen aus der Sammlung von 5. entsprechen, damit der Algorithmus eindeutig entscheiden kann. Sie gehört dem Gateway. Wenn das nicht eindeutig ist - mit 1. neu starten und dabei eventuell die Menge an gesammelte Daten erhöhen;

8. Auf eine ARP-Reply warten, die die MAC-Adresse vom Gateway als Source beinhaltet. In der ARP-Reply die IP-Adresse vom Gateway ablesen. Anstatt Warten kann eine ARP-Kommunikation durch ein kurzes Ausziehen und Einschließen des Kabels ausgelöst werden.

Algorithmus 2 Identifizierung der Richtung des Anschlusses

1. Die Dateien benutzen (jeweils eine Datei pro Richtung), die im Schritt 1. vom Algorithmus 1 auf der vorherigen Seite gesammelt sind;
2. In beiden Dateien alle Zeilen löschen, die als Source-MAC-Adresse die Mac-Adresse vom Router haben;
3. Die Datei, die leer geblieben ist, beachten;
4. Wenn die leere Datei die „IN“ Datei ist, ist die Netzwerkkarte aus Schritt 1. vom Algorithmus 1 auf der vorherigen Seite die „linke“ Karte - sie ist also in der Richtung vom Heimnetz angeschlossen;
5. Wenn die leere Datei die „OUT“ Datei ist, ist die Netzwerkkarte aus Schritt 1. vom Algorithmus 1 auf der vorherigen Seite die „rechte“ Karte - sie ist also in der Richtung vom externen Netz angeschlossen;
6. Die zweite Netzwerkkarte ist in der anderen Richtung angeschlossen. Beide Ergebnisse vermerken;
7. Wenn beide Dateien leer sind, oder keine Datei leer ist, kann keine Entscheidung getroffen werden. Messungen und beide Algorithmen wiederholen.

Werkzeuge

Um die Daten zu sammeln, benutzen wir *tcpdump* mit ausgewählten Parameter, die die korrekte Netzwerkkarte bestimmen und eine einfache Datenausgabe einstellen, sowie auch mit dem Parameter „*-Q*“, der uns erlaubt die Richtung des Netzverkehrs zu spezifizieren. Für die Bearbeitung der Felder und weitere Filterung der Ergebnisse benutzen wir die gewöhnliche Linux-Werkzeuge, die Teil der Standardinstallation sind: *awk*, *grep*, *sed*, *cut* usw.

Im Schritt 8. von Algorithmus 1 auf der vorherigen Seite wollen wir nur die ARP-Antworten sammeln und zwar diese, die aus dem Gateway kommen. Das Gateway beschreiben wir durch seine MAC-Adresse mit der Option „*ether src*“. *tcpdump* hat aber keine voreingestellte Option um ARP-Antworten zu filtern. Dafür kommt die ARP-Spezifikation zur Hilfe. Im Standard RFC826 sind das Format und alle Felder beschrieben. „*Protocol Type*“ ist ein 16-Bit Feld (=2 Oktetten) und der Code für „*Reply*“ ist 2[1]. Das Feld fängt ab dem Oktett 6 im Protokoll an[2], also würde das in der Sprache von *tcpdump* so aussehen[3]:

[1] Network Working Group (1982), S.3

[2] Wenn wir die Längen alle unteren Felder summieren: 16 + 16 + 8 + 8 = 48 Bits = 6 Oktett (Siehe dazu den RFC, S.3)

[3] https://www.tcpdump.org/manpages/pcap-filter.7.html, Zugriff am 08.05.2022

Finde solche Pakete, wo ab dem Oktett 6 die nächsten zwei Oktetten gleich „Reply“ (2) sind: `arp[6:2] == 2`

Ein Ablauf von beiden Algorithmen mit echten Netzwerkdaten ist im Dateien-Anhang zu dieser Arbeit hinzugefügt.

In dem Steuerungsskript führt die Funktion „*parse_network_data*“ ab Zeile 326 die Netzwerkanalyse aus.

Netzmaske

Soweit sind die ersten benötigte Daten (1., 2.a, 2.b aus Seite 55) ausgerechnet. Betrachten wir die Netzmaske:

Die Netzmaske ist zum Routing dringend notwendig und bestimmt, was mit jedem Paket passieren soll - wird er im Subnetz, über einer festen Route oder über dem default-Gateway geschickt[1]. Ohne Kenntnisse über der korrekten Netzmasken kann die Box nicht funktionieren. Als Problem ergibt sich die Tatsache, dass die Netzmaske kein Teil der normalen Kommunikation zwischen Endgeräte ist. Betrachten wir die folgende zwei Möglichkeiten:

- Das Endgerät vom internen Netz (der Heimnetzrouter) bekommt seine Netzwerkdaten über DHCP. In diesem Fall müssen wir einfach die DHCP-Meldungen abfangen und ablesen. Mehr dazu weiter unten;

- Das Endgerät hat statische Einstellungen oder bekommt sie auf einen anderen Weg, wovon wir nicht wissen, oder aus anderen Gründen nicht abfangen können. Eine Option dabei wäre die Inter-Router-Kommunikation abzuhören, die genug Information beinhalten soll[2]. Wir können uns aber diese Komplikationen sparen indem wir betrachten, dass wir schon ein Gerät kennen, das sicherlich selber die notwendige Netzmaske kennt - das ISP-Gateway. Wenn wir die Abbildung 55 beachten, spielt die Netzmaske soweit eine Rolle, damit wir wissen, wie groß das Netzsegment vom Betreiber ist, wo wir angeschlossen sind. Anders ausgedrückt - um an der Kommunikation „4“ teilzunehmen. Wir stellen dafür eine „enge“ Netzmaske ein, bei der der ganze Verkehr über dem Gateway verläuft, darunter auch der Verkehr für das lokale Subnetz. Einige Tests mit dieser Konstellation haben keine Probleme ergeben, mit der einzigen Ausnahme der SSH-Kommunikation, wo vermutlich dieser Umweg von der Software als potenzieller Angriff wahrgenommen wird. Weitere Tests und Forschung in dieser Richtung sind notwendig, werden aber hier nicht weiter diskutiert.

[1] Vgl. Benvenuti (2006), S.789
[2] Vgl. Tanenbaum; Wetherall (2014), S.374

Der Hostname und die Angaben über den DNS-Server betrachten wir weiter unten bei der Diskussion über DHCP. In einem Betrieb mit statischen Netzwerkeinstellungen interessieren uns diese nicht - die IP-lose Box wird einen eigenen DNS-Server benutzen und den Hostname müssen wir nicht unbedingt mitteilen.

Die Daten über der Netzmaske, dem Hostname und dem DNS-Server werden im Steuerungsskript für die Funktionalität im Vollbetrieb benutzt. Mehr dazu im nächsten Kapitel.

6.3 Vollbetrieb

Mit allen gesammelten Daten und die daraus gewonnene Information über den Netzwerkeigenschaften kann die Box in den Vollbetrieb umschalten. Im Idealfall wird der einzige Unterschied zwischen dem Vollbetrieb und dem Brückenmodus die Möglichkeit für die Box selber am Netzverkehr teilzunehmen. Für die andere Teilnehmer bleibt sie transparent[1].

Den Grundansatz haben wir schon diskutiert - Benutzung von Namensräume für eine Trennung der beiden Netzwerkschnittstellen und zweifache NAT für den Umgang mit IP-Konflikte.

Um die Diskussion zu vereinfachen, nehmen wir an, dass die Netzwerkkarte zum internen Netz „eth0“ und die Netzwerkkarte zum externen Netz - „eth1“ ist. Wie schon im vorigen Kapitel gesehen, kann die Netzwerkanalyse die korrekte Richtung ausrechnen und die richtige Netzwerkkarte für die weitere Steuerung benutzen. Wir nehmen auch an, dass der Heimnetzrouter sein Netzverkehr an der Box weiterleitet. Wie genau er davon mittels ARP überzeugt wird, wird im Unterkapitel für ARP diskutiert. Auch aus Darstellungsgründen nehmen wir an, dass das Subnetz, wo die Box ist, 77.77.77.0/24 ist. Für die Funktion der Box spielen die IP-Adressen und Netzmasken sonst keine Rolle. Der Router, nehmen wir an, hat die externe IP-Adresse 77.77.77.50 und seine externe Netzwerkkarte hat die MAC-Adresse aa:aa:aa:aa:aa:aa. Das Gateway hat IP-Adresse 77.77.77.1 und MAC-Adresse bb:bb:bb:bb:bb:bb.

Wir betrachten den Heimnetzrouter als die Quelle des Verkehrs, wobei das wahrscheinlich nicht korrekt ist. Allerdings, aufgrund der NAT am Router, ist der aus dem Heimnetz stammende Verkehr für unsere Ziele nicht unterscheidbar.

Die Umstellung auf Vollbetrieb fängt mir der Erstellung eines neuen Namensraums an - namens *ns0*. Die Netzwerkkarte zum internen Netz wird dem neuen Namensraum zugewiesen. Die Netzwerkkarte zum externen Netz bleibt wie sie ist - im Namensraum *default*. Ein *veth*-Paar virtuelle Netzwerkschnittstellen wird

[1] Mit der Ausnahme wenn die Box selber adressiert wird.

erzeugt und in den beiden Namensräume verteilt, um die Verbindung zwischen diesen zu sichern. Eine Kombination aus Netzwerkrouten, IP-Einstellungen und *nft*-Regeln wird für das gezielte Verhalten sorgen. An diesem Punkt ist zu beachten, dass alle diese Einstellungen schon separat eingegeben werden müssen - jeweils für jeden Namensraum.

Serverdienste, DNS-Filter und weiteres werden wir im Namensraum „*default*“ behalten. Das hat zwei Gründen:

- Vermeidung von unnötigen Komplizierungen im Skript;
- Vermeidung von Probleme mit der Software, weil nicht alle Software mit Namensräume umgehen kann. Als Beispiel ist der für die Box geplante DNS-Filter *pihole* zu nennen.

Betrachten wir zuerst, was mit dem IP-Verkehr passiert, wenn er durch die Box verläuft.

Vollbetrieb: IP

Die weitere Überlegungen basieren auf der Abbildung 12 auf Seite 63.

Die mögliche Varianten für IP-basierten Netzverkehr, der durch die Box verläuft, sind wie folgt:

Kommunikationsszenarien

1. Quelle: Heimnetz => Ziel: Internet;
2. Ziel: Heimnetz <= Quelle: Internet (ähnlich: Quelle: Subnetz);
3. Quelle: IP-lose Box => Ziel: Internet (ähnlich: Ziel: Subnetz);
4. Ziel: IP-lose Box <= Quelle: Internet (ähnlich: Quelle: Subnetz);
5. Quelle: Heimnetz => Ziel: IP-lose Box[1].

In den Grundlagen haben wir schon die Probleme diskutiert, die entstehen, wenn Geräte hinter NAT erreicht werden müssen. Für den Punkt 2. setzt das voraus, dass Ports am Router geöffnet sind. Für das in dieser Arbeit dargestelltes Szenario ist das nicht unbedingt der Fall. Die Box kann in diesem Fall als Schutz für das ganze Heimnetz dienen, indem sie durch Firewall-Regeln einfach alle neue

[1]Die umgekehrte Variante - von der Box zum Heimnetz betrachten wir nicht, weil sie wenig relevant ist. Sie kann aber ähnlich zu dieser Variante durchgeführt werden.

Verbindungen zum Heimnetz von draußen blockiert. Auf der Abbildung 12 auf der nächsten Seite sind das die Verbindungen „5“ zu „0“.

Die Erreichbarkeit der Box von außen ist auch relativ einfach zu sichern. Der Namensraum „*default*“ steuert die Netzwerkkarte zum externen Netz und dort läuft auch die Server-Software. Mit dem Beispiel eines *SSH*-Zugangs würde das bedeuten, dass wenn die Box den korrekten Port in der Firewall erlaubt, ist sie schon von außen erreichbar. Die externe Netzwerkkarte bildet die Netzwerkkarte vom Router ab. Aus der Sicht der externen Geräte müssen sie eine *SSH*-Verbindung zum „Router“ starten und werden auf die Box gelangen. Auf der Abbildung 12 auf der nächsten Seite ist das die Verbindung „5“ zu „0“ auf Port *SSH*. Die Kommunikation wird aber tatsächlich nur mit der Box im Namensraum „*default*“ stattfinden.

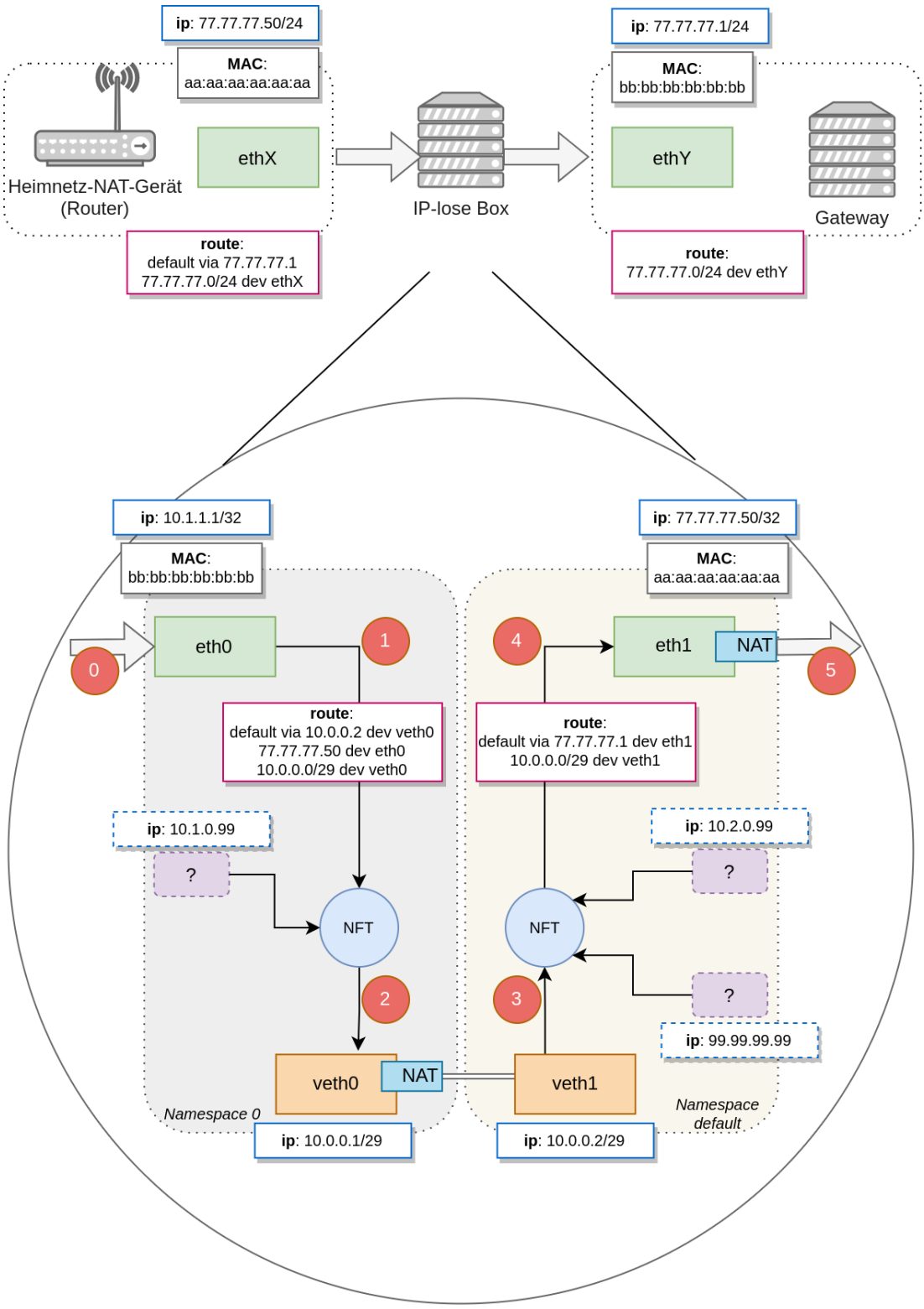

Abbildung 12: IP-lose Box: Behandlung von IP-Pakete

Verbindung mit dem Internet aus dem Heimnetz

Die Verbindungen, die vermutlich am häufigsten gesehen werden, sind vom Punkt 1. der Szenarien - Verbindungen, die aus dem Heimnetz stammen und das Internet als Ziel haben. Damit diese ungehindert passieren, machen wir folgende Einstellungen:

- Die „linke“ Netzwerkkarte - *eth0* auf der Abbildung 12 auf der vorherigen Seite bekommt eine private IP-Adresse, z.B. 10.1.1.1 mit Netzmaske 32. Eine statische Route wird dafür sorgen, dass der Heimnetzrouter aus dieser Schnittstelle erreichbar ist;
- Als Standardroute (default gateway) wird im Namensraum *ns0* die (virtuelle) Netzwerkkarte *veth1* eingestellt;
- Die „rechte“ Netzwerkkarte (*eth1*) bekommt die Adresse vom Heimnetzrouter und die Netzmaske 32[1];
- Das virtuelle Paar bekommt entsprechend 10.0.0.1/29 und 10.0.0.2/29;
- Standardroute im Namensraum „*default*“ wird die Adresse vom Betreiber-Gateway sein.

Betrachten wir dann, was mit der Kommunikation passiert:

- Ein Paket aus dem Heimnetz kommt an *eth0* an („0“ auf der Abbildung 12 auf der vorherigen Seite);
- Das Paket wird von der Box angenommen und sie fragt die Routingtabelle ab, weil das Paket nicht für die lokale Bearbeitung ist („1“);
- Es werden an diesem Punkt keine Änderungen von nft durchgeführt und das Paket wird zum Standardgateway weitergeleitet („2“, „3“);
- Im Namensraum „*default*“ wird auch festgestellt, dass das Paket nicht für die lokale Bearbeitung ist. Nach der Routingtabelle wird das Paket zum Standardgateway weitergeleitet, was in diesem Fall das Betreiber-Gateway über *eth1* bedeutet;
- Die IP-Adresse des Pakets wird zweimal geändert - bei dem Passieren von „2“ zu „3“ aufgrund der NAT in *veth0* und auch von „4“ auf „5“ aufgrund der NAT in *eth1*. Im Endeffekt hat das Paket bei „5“ die selbe IP-Adresse wie in „0“.

[1]Siehe dazu die Diskussion auf Seite 6.2 auf Seite 59

- Auf dem Rückweg wird die Antwort von der Firewall als „*related*“ zugelassen und in einer umgekehrten Reihenfolge bearbeitet. Die Box funktioniert transparent für die Kommunikation.

Die zweifache NAT erscheint zu diesem Punkt überflüssig zu sein und die Funktion der Box unterscheidet sich wenig von einem Brückenmodus. Diese werden sich aber bei den komplexeren Szenarien als unerlässlich zeigen.

Verbindung mit dem Internet aus der Box

Aus der Sicht des externen Netzes existiert nur die IP-Adresse des Heimnetzrouters (77.77.77.50 in unserem Beispiel auf der Abbildung). Hinter dieser Adresse versteckt sich aber auch die Box selber, die die gleiche Adresse benutzt. Wie das möglich ist, kann eine Verfolgung der IP-Felder zeigen, wenn sie durch die Box passieren:

- Das Paket mit Quellenadresse 77.77.77.50 kommt an *eth0* an („0“ auf der Abbildung 12 auf Seite 63);
- Weil das Paket nicht für lokale Bearbeitung ist und nicht gefiltert werden muss, wird es über *veth0* unverändert zum *veth1* geleitet („1“ und „2“);
- Beim Ankunft in *veth1* hat das Paket schon die Quellenadresse 10.0.0.1, aufgrund die NAT auf *veth0* („3“);
- Hier ist auch keine Änderung durch *nft* notwendig, also wird das Paket zum Standardgateway weitergeleitet - 77.77.77.1 über *eth1* („4“);
- Nach dem Verlassen der Box („5“) hat das Paket schon wieder die Adresse 77.77.77.50, aufgrund der NAT auf *eth1*;
- Die beide NATs behalten in ihre Tabellen die notwendige Information, um die Antworten auf dem Rückweg korrekt zuzuweisen.

Somit ist es möglich, dass die Box auch die Adresse 77.77.77.50 für ihre eigene Kommunikation mit dem Internet benutzt. Es entsteht kein Konflikt, weil der Verkehr aus der „echten“ 77.77.77.50 Adresse im Namensraum *default* schon in 10.0.0.1 umgewandelt ist. Für die konkrete Zuordnung der Antworten kümmert sich die NAT.

Verbindung mit der Box aus dem Heimnetz

Die Verbindung mit der Box aus dem Heimnetz ist etwas komplizierter. Folgende Überlegungen sind dabei wichtig:

- Die Box muss über irgendwelche IP-Adresse adressiert werden kann, damit die Verbindung über dem Heimnetzrouter überhaupt passiert;
- Diese Adresse darf keine Adresse aus einem privaten Adressenraum sein, sonst würde der Heimnetzrouter die Verbindung nicht weiterleiten;
- Eine Unterscheidung muss gemacht werden, zwischen dem echten Besitzer der ausgewählten IP-Adresse und der Box.

Um die Komplexität der Implementierung in Grenzen zu halten und die Heimnetzgeräte nicht mit extra Anforderungen oder Software zu belasten, machen wir den Kompromiss, dass wir eine IP-Adresse aus dem Internet „opfern“. Wir wählen also eine IP-Adresse, die nur für den Zugang zur Box benutzt wird. Das hat die Folge, dass das Gerät, das über dieser Adresse verfügt, nicht mehr aus dem Heimnetz erreichbar wird. Wenn wir die Menge an IP-Adressen beachten und die Tatsache, dass viele davon eine extrem geringe Wahrscheinlichkeit haben, aus dem Heimnetz angesprochen zu werden[1], erscheint das nicht so problematisch.

Eine andere Option ist die Auswahl einer IP-Adresse, für die bekannt ist, dass sie einem schädlichen Domain gehört. Somit kann sich der oben genannte Nachteil in einem Vorteil umwandeln.

Für die Zwecke dieser Arbeit wählen wir die Adresse 99.99.99.99 aus[2].

Bevor wir die Diskussion über die konkrete Implementierung weiterführen, führen wir das Konzept einer Phantom-IP-Adresse ein:

Unter einer Phantom-IP-Adresse werden wir eine IP-Adresse verstehen, die keiner Netzwerkschnittstelle zugewiesen ist, kein Endziel entspricht und nur für das interne Routing in der Box benutzt ist. Solche Adressen sind auf der Abbildung 12 auf Seite 63 mit Fragezeichen dargestellt[3].

Wir stellen zwei neue Phantom-IP-Adressen ein - jeweils eine in jedem Namensraum.

Die hervorragende Möglichkeiten, die uns *nftables* anbietet, um Pakete zu manipulieren haben wir schon im Kapitel „Der Weg eines Netzwerkpakets durch den Linux-Kernel“ auf Seite 22 diskutiert und graphisch auf Abbildung 4 auf Seite 25 dargestellt. Diese Möglichkeiten werden wir jetzt benutzen. Auf dem Weg des Pakets aus dem Heimnetz zu der Box und zurück machen wir folgende Einmischungen:

[1] Z.B. ISP-Geräte, Backbones-Router, etc.

[2] Diese Adresse wurde vollkommen zufällig ausgewählt. Zur Zeit reagiert das Gerät auf dieser Adresse auf keinen der gängigsten Ports (Web, SSH, etc.). Nach Angaben von whois.com gehört die Adresse AT&T in der USA

[3] Die Zieladresse ist tatsächlich eine „echte“ IP-Adresse, wird aber für unsere Ziele als Phantom-IP-Adresse benutzt.

- Das Paket kommt an *eth0* an („0“ auf der Abbildung 12 auf Seite 63). Die Quellenadresse ist 77.77.77.50 und das Ziel - 99.99.99.99;
- Bevor das Paket über veth0 weitergeleitet wird („1“), ändern wir mithilfe von *nft* beide Adressen wie folgt:
  - Quellenadresse: 10.1.0.99
  - Zieladresse: 10.2.0.99
- Das Paket wird über *veth0* geroutet und die Quellenadresse wird nochmals geändert, aufgrund der NAT („2“ => „3“);
- In Namensraum „*default*“ ändern wir nochmals die Zieladresse - diesmal zu 77.77.77.50. Die Änderung passiert bevor die Routing-Entscheidung getroffen wird („3“);
- Das Paket scheint für lokale Bearbeitung zu sein. Das Paket wir lokal bearbeitet und ggf. eine Antwort ausgelöst („4“);
- Die Antwort bearbeiten wir ähnlich, wie auf dem Hinweg:
  - im Namensraum „*default*“ ändern wir die Quellenadresse auf 10.2.0.99 („3“);
  - In Namensraum „*ns0*“ ändern wir die Zieladresse auf 77.77.77.50 („2“). Diese Änderung passiert auch vor der Routingentscheidung;
- Das Paket wird aufgrund der statischen Route über *eth0* zu 77.77.77.50 zum Heimnetz geschickt („0“).

Wie gesehen, werden die Adressen 10.1.0.99, 10.2.0.99 und 99.99.99.99 nie tatsächlich erreicht. Sie dienen nur dafür, dass korrektes Routing ausgeführt wird und dass die NAT auf *veth0* richtig funktioniert.

Die umgekehrte Variante - eine Verbindung aus der Box zum Heimnetz setzt auch offener Port voraus, wie eine Verbindung aus dem externen Netz. Diese Variante ist für unsere Ziele wenig relevant, kann aber ähnlich zu dem schon vorgestellten Szenario ausgeführt werden.

Verbindung zum Heimnetz aus dem Internet (ohne offener Port)

Das Szenario, dass ein internes Gerät im Heimnetz aus dem Internet erreicht werden soll, verdient eine separate Betrachtung. Die Situation ist interessant, weil

sie keine Anforderungen am Heimnetzrouter stellt - es soll nur ein Internetzugang möglich sein, was im Normalfall gegeben ist.

Die Verbindung ist sehr einfach, wenn schon die Möglichkeit vorhanden ist, die Box aus dem Heimnetz zu erreichen. Hilfe dabei bieten die zahlreiche Möglichkeiten von *ssh* an. Der folgende Befehl wird aus dem Heimnetzgerät ausgeführt:

Listing 7: Reverse-SSH

```
1 $ ssh -R 33333:localhost:22 pi@99.99.99.99 [-p ssh-port]
```

Die Bedeutung davon ist wie folgt[1]:

- Die „*-R*“ Option stellt ein Reverse-SSH-Tunnel auf dem Fernserver (in unserem Fall - die IP-lose Box) ein;
- „*33333*“ ist die Nummer eines gewünschten Ports auf dem Fernserver. Wenn der Fernserver auf diesen Port angesprochen wird, wird der dortige SSH-Server die Verbindung zu dem Heimnetz-Gerät weiterleiten. Es muss beachtet werden, dass dieser Port kein privilegierter Port (<1024) sein darf;
- „*localhost:22*“ meint das Heimnetzgerät, der angesprochen wird, bei einem Zugriff auf *Fernserver:33333*[2];
- „*pi*“ ist der Standardbenutzername auf dem Pi, den wir für dieses Projekt für die IP-lose Box benutzen;
- „*99.99.99.99*“ haben wir schon diskutiert - die IP-Adresse, worüber die IP-lose Box aus dem Heimnetz erreicht wird.
- „*-p ssh-port*“ ist eventuell notwendig, wenn wir den SSH-Server auf einen anderen Port als 22 auf der IP-losen Box betreiben.

Die Standardeinstellung vom SSH-Server ist den Reverse-Tunnel nur aus dem *localhost* (IP-lose Box) zu erlauben. Für einen Zugriff von außen muss im */etc/ssh/sshd.config* von der Box „GatewayPorts yes“ eingestellt sein[3].

Solange die Verbindung aus dem Heimnetzgerät aktiv ist, kann es aus dem Internet mittels „*ssh 77.77.77.50:3000*“ erreicht werden.

Diese Überlegungen setzen natürlich voraus, dass die externe IP-Adresse bekannt ist (im oberen Beispiel 77.77.77.50). Wie dieser Zugriff über DNS vereinfacht werden kann, diskutieren wir weiter unten.

[1] https://www.man7.org/linux/man-pages/man1/ssh.1.html, Zugriff am 08.05.2022

[2] Die IP-Adresse vom *Fernserver* ist in unserem Fall gleich die externe IP-Adresse vom Heimnetzrouter.

[3] https://www.man7.org/linux/man-pages/man5/sshd_config.5.html, Zugriff am 08.05.2022

Vollbetrieb: Untere Schichten

ARP

Mit Betracht auf die Kommunikation über ARP sind folgende Varianten denkbar:

1. Der Heimnetzrouter möchte mit einem beliebigen Gerät im Internet kommunizieren. Diese Kommunikation erfolgt über dem Gateway, also in der Sicherungsschicht ist nur die (MAC-)Adresse vom Gateway relevant. Die selbe Situation entsteht, wenn der Router mit dem Gateway kommunizieren möchte;

2. Es wird eine Kommunikation im Rahmen des lokalen Subnetzes vom Router und vom Gateway erfolgen. In diesem Fall ist eine Sicherungsschichtbasierte Kommunikation zwischen den beteiligten Geräte notwendig.

Für unsere Beispielanwendung ist die Variante 2. weniger wahrscheinlich, werden wir aber trotzdem betrachten, um die Flexibilität des Konzepts zu behalten.

Eine Variante dieses Problem zu lösen, liegt in der Benutzung von *nftables*. Eine der unterstützten Optionen ist die Weiterleitung der abgefangenen Pakete zum Userspace für eine weitere Bearbeitung. Wir können also ARP abfangen, die lokale ARP-Tabelle aktualisieren und das Paket im nächsten Namensraum abspielen. Der beschriebene Ansatz ist als „*Queueing to userspace*“ bekannt[1].

Es existiert zusätzlich eine einfachere Variante, die die Kernel-Vorrichtungen benutzt, namens *Proxy-ARP*. Die Grundidee dabei ist, dass das so eingestellte Gerät ARP–Anfragen zwischen den Netzwerkschnittstellen weiterleitet[2]. Die dazu zuständige Option in *sysctl* ist *proxy_arp*[3]. Nach der Aussage in der Debian-Dokumentation werden nur Pakete weitergeleitet, wenn deren Empfänger der Routing-Tabellen bekannt ist[4]. Allerdings werden in unseren Tests mit verschiedenen Rechner und Netzwerkszenarien alle ARP-Anfragen mit der MAC-Adresse vom lokalen Gerät beantwortet. Eventuell liegt das an der default-Route, die alle mögliche Adressen beinhaltet. Weitere Forschung in dieser Richtung ist notwendig.

Die einfachste Lösung in diesem Fall wäre das Subnetz Router-Gateway zu ignorieren und die beiden Adressen vom externen Gateway - MAC, sowie auch IP

[1] https://wiki.nftables.org/wiki-nftables/index.php/Queueing_to_userspace, (Zugriff am 08.05.2022)

[2] https://wiki.debian.org/BridgeNetworkConnectionsProxyArp, (Zugriff am 08.05.2022)

[3] https://www.kernel.org/doc/html/latest/networking/ip-sysctl.html, (Zugriff am 08.05.2022)

[4] https://wiki.debian.org/BridgeNetworkConnectionsProxyArp, (Zugriff am 08.05.2022)

- auf der internen Netzwerkschnittstelle abzubilden. ARP-Anfragen würden dabei ganz normal von der internen Netzwerkschnittstelle befriedigt und *nftables* kann sich danach um das Routing zwischen den Namensräume kümmern. Das würde aber alle Kommunikationen im besagten Subnetz ausschließen. Der beigefügte Skript benutzt die Variante mit *proxy_arp*, weil sie zu keine Probleme bei den Tests geführt hat, wobei auch eine Kommunikation im Subnetz möglich war.

An dieser Stelle ist es interessant zu vermerken, dass der DHCP-Standard folgendes vorschreibt:

„Wenn der Client feststellt, dass die [vergebene] Adresse von einem anderen Gerät in Benutzung ist (z.B. durch die Benutzung von ARP), dann MUSS der Client eine DHCPDECLINE Nachricht am Server senden und den Konfigurationsprozess neu starten.[1]“ (Übersetzung durch den Verfasser)

Dieses Verhalten wurde bei keinem der getesteten DHCP-Clients beobachtet und bleibt auch als ein Thema für die weitere Forschung.

DHCP: Allgemeine Überlegungen

Wie schon kommentiert wollen wir das Konzept flexibel halten - sie soll mit statischen, sowie auch mit dynamischen Netzwerkeinstellungen arbeiten können. Bei DHCP muss besondere Aufmerksamkeit geschenkt werden. Der Grund dafür ist, dass die ersten Phasen der Kommunikation in der Sicherungsschicht verlaufen - es kann auch nicht anders sein, weil der Client noch keine Netzwerkdaten hat und gerade nach diesen fragt[2]. Die Folge davon ist, dass diese Meldungen auch in die andere Netzsegmente weitergegeben werden müssen. Die zwei Standardlösungen wären:

1. Ein DHCP-Relay zu installieren. Diese Software hört am Subnetz zu, wo der Client ist und beim Empfang einer Anfrage leitet dieser über der Vermittlungsschicht an einem vorgegebenen DHCP-Server weiter. Ein Teil der Kommunikation ist die Angabe von Information über dem Client-Subnetz, damit der Server eine korrekte IP-Adresse anbieten kann[3]. Zusätzlich gibt das Relay auch die Adresse der Netzwerkschnittstelle an, wo es läuft;

2. Einen vollständiger DHCP-Server im Namensraum zum internen Netz zu installieren.

Bei beiden Varianten haben sich in den Tests Probleme ergeben, wobei die Software durch die ungewöhnliche Netzkonfiguration verwirrt wird. Die Situation lässt

[1] Network Working Group (1997b), S.16
[2] Network Working Group (1997b), S.13-15
[3] https://kb.isc.org/docs/isc-dhcp-44-manual-pages-dhcrelay, (Zugriff am 08.05.2022)

sich aber stark vereinfachen, wenn wir betrachten, dass wir keine vollständige und umfangreiche Software brauchen - in dem DHCP-Szenario läuft ein vollständiger DHCP-Server auf dem ISP-Gateway.

Durch ein Standardclient, installiert im extern-gerichteten Namensraum fragen wir den DHCP-Server ab, dann speichern wir seine Antworten und spielen diese an der internen Seite ab. Für das Abspielen hat sich die Python-Bibliothek *Scapy* als besonders geeignet gezeigt. *Scapy* bietet uns die Möglichkeit an, beliebige Netzwerkpakete zu bilden und diese zu senden, sowie auch das Netz abzuhören und die Pakete nach den eigenen Wünsche zu ändern[1]. Der Nachteil der hohen Abstraktion von Python soll sich in unserem Projekt nicht zeigen, weil wir *Scapy* nur für die vergleichsweise selten auftretende DHCP-Pakete benutzen. Bei den Tests wurde keine messbare Differenz zwischen direkte DHCP Client-Server Kommunikation und eine solche, die über *Scapy* abläuft.

Ein einfacher DHCP-Server, der auf *Scapy* basiert, hat der Benutzer „*ptef*“ in GitHub Quell-offen zur Verfügung gestellt[2]. Die Software hört auf Anfragen zu und beantwortet diese mit vorgegebenen Werte. Den Skript erweitern wir mit einer Möglichkeit diese Werte immer neu zu laden, sowie auch mit einem Verfahren den Hauptskript über den Anfragen zu informieren. Der modifizierte *scapy* DHCP-Skript ist im Dateienanhang dieser Arbeit zu finden.

DHCP: Erfassung von DHCP-Daten mit tcpdump

Der DHCP-Standard erwähnt mehrere Typen von Nachrichten[3]. Für unsere Ziele ist die DHCPACK Nachricht von besonderen Bedeutung, aus folgenden Gründen:

- Die Nachricht kommt am Ende der Kommunikation und beinhaltet somit endgültige und bestätigte Daten;
- Die Nachricht beinhaltet genau die Daten, wovon wir interessiert sind. Darunter:
  - IP-Adresse für den Client;
  - Netzmaske;
  - DNS-Server;
  - Hostname vom Client;

[1] https://scapy.readthedocs.io/en/latest/introduction.html, (Zugriff am 08.05.2022)
[2] ptef (2018), Internetquelle
[3] Network Working Group (1997b), S.13

Wir wollen genau diese Nachrichten mit *tcpdump* abfangen, damit wir auch sofort darauf reagieren können. Allerdings bietet *tcpdump* keinen Filter für DHCPACK an. Die Lösung des Problem finden wir, wenn wir den Standard beachten:

- DHCP benutzt UDP als Transportprotokoll[1];
- Der UDP-Header ist genau 8 Oktetten lang[2];
- Die Optionen in der DHCP-Nachricht fangen ab dem Oktett 236 an (berechnet durch Summierung aller unteren Felder)[3];
- Die Optionen starten mit einem 4 Oktetten langen „Magic-Cookie“[4]. Der Wert des Cookies ist genau 63.82.53.63 in Hex[5];
- Der Wert der DHCPACK-Option ist genau 35.01.05 in Hex (Code + Len + Type)[6].

Also wir wollen die Pakete filtern, die ab dem Oktett 248 (UDP Header + Optionenanfang + Magic Cookie) den Wert 0x350105 haben. *tcpdump* kann aber nur mit Filter arbeiten, die 1,2 oder 4 Oktetten lang sind. Aus diesem Grund müssen wir entweder die unteren Bits des Wertes benutzen, oder einige Bits aus dem Magic-Cookie mitbenutzen und etwas früher den Vergleich anfangen. Wir entscheiden uns für die zweite Option, weil sie präziser ist. Der endgültige Filter ist somit:

Listing 8: tcpdump: Filter für ARP-Reply

```
1 'udp[247:4] = 0x63350105'
```

Die so abgefangene DHCPACK-Nachrichten filtern wir mit den gewöhnlichen Linux-Werkzeuge und erfassen die benötigten Daten. Diese speichert der Skript in einer gesonderten Datei (mehr dazu im nächsten Paragraph).

Der Hostname vom Client interessiert uns insoweit, dass unser DHCP-Client im Namensraum „*default*“ nicht mit unserem Hostname den DHCP-Server abfragt (z.B. „*pi3*“). Nach der Erfassung des Hostname vom Client kann der DHCP-Client mit der Option „*-H*“ den korrekten Name beim Server angeben[7].

[1] Network Working Group (1997b), S.22
[2] Tanenbaum; Wetherall (2014), S.542
[3] Network Working Group (1997b), S.8
[4] Network Working Group (1997b), S.22
[5] Network Working Group (1997a), S.3
[6] Network Working Group (1997a), S.26
[7] https://manpages.debian.org/bullseye/isc-dhcp-client/dhclient.8.en.html, (Zugriff am 08.05.2022)

DHCP: Synchronisierung von Daten

Für die Synchronisierung der Information, die *tcpdump* in Namensraum „*default*“ sammelt und den kleinen Python-basierten DHCP-Server, der im Namensraum „*ns0*“ läuft, übernehmen wir folgende Ansätze:

1. Jedes Mal, wenn *tcpdump* DHCP-Daten im Namensraum „*default*“ sammelt, werden die benötigte Daten ausgewertet und in einer gesonderten Datei gespeichert;

2. Jedes Mal, wenn eine DHCP-Anfrage im Namensraum „*ns0*“ ankommt, wird die Datei mit DHCP-Daten von Punkt 1. aus dem Python-Skript gelesen. Die Daten werden benutzt, um die Anfrage mit der aktuellsten Information zu beantworten. Der Skript speichert einen Timestamp in einer extra Datei;

3. In jedem „Tick“ vom Hauptskript wird die Datei mit dem Timestamp aus Punkt 2. ausgelesen und darauf basierend wird die Aktualität der DHCP-Variablen geprüft. Wenn aktuellere Information vorliegt, wird die DHCP-Datei aus Punkt 1. gelesen und die Variablen werden dadurch aktualisiert.

Graphisch ist dieses Konzept auf der Abbildung 13 auf der nächsten Seite vorgestellt.

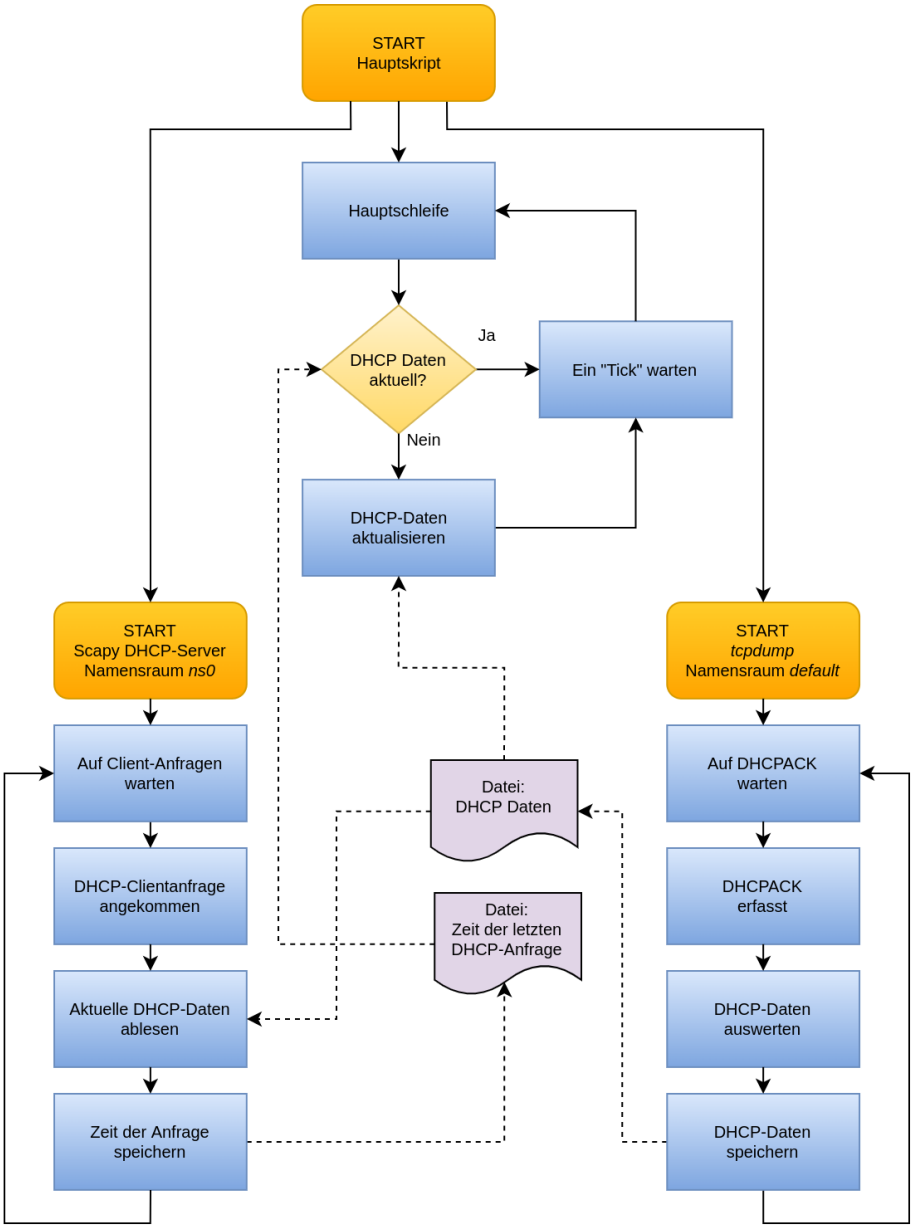

Abbildung 13: IP-lose Box: Synchronisierung

Vollbetrieb: Dynamic-DNS und DNS-Filter

Die Hauptfunktion der IP-losen Box ist schon vorhanden. Wir wollen aber, wie schon bei den Anforderungen diskutiert, einige Software darauf installieren, die als Beispiel für Einsätze in realen Benutzerumgebungen dienen kann. Aus diesem Grund stellen wir einen Dynamic-DNS-Client und einen DNS-Filter vor.

Dynamic-DNS

Bei der Diskussion über Verbindungen zum Heimnetz aus dem Internet mittels Reverse-SSH-Tunnel haben wir schon festgestellt, dass für die Verbindung die

externe IP-Adresse des Heimnetzes benötigt ist. Wenn wir überlegen, dass sie eventuell dynamisch vom Netzbetreiber zugewiesen ist, wird es schnell unpraktisch mit der konkreten IP-Adresse zu arbeiten. Nach dem selben Prinzip, wie die Webseiten im Internet erreicht werden, kann auf unsere Box auch über einen Domain-Name zugegriffen werden. Die Option einen „echten“ Domain-Name zu kaufen und einzustellen widerspricht der Anforderungen an Benutzerfreundlichkeit die das Projekt hat. Zusätzlich wäre das für unsere Ziele ein Overkill, wenn wir die zahlreiche kostenlose Dienste betrachten, die sich genau um dieses Problem kümmern.

Eine der möglichen Optionen ist „Duck DNS“ - ein Anbieter von Dynamic-DNS[1]. Nach Registrierung erhält der Benutzer seine zugewiesene Domain (z.B. in der Form von mycooldomain.duckdns.org) und einen Token, der die Domain eindeutig identifiziert. Eine einfache URL-Anfrage zu dem DuckDNS- Aktualisierungsserver, die auch den Token enthält, führt zu einer Aktualisierung der IP-Adresse vom Gerät, aus dem die Anfrage gemacht wurde[2]. Empfohlen wird die Einstellung über *cron* für die automatische Pflege der aktuellen Adresse.

Nach dieser Einstellung erfolgt der Zugriff auf dem gewünschten Gerät über seinen Domainname.

Der pi-hole DNS-Filter

Das *pi-hole* Projekt ist eine Sammlung aus Software, die als DNS-Filter für das gesamte Heimnetz dienen kann[3]. Die DNS-Anfragen werden von den Clients angenommen, unerwünschte oder schädliche Domäne werden anhand Sperrlisten gefiltert und erlaubte Anfrage werden über einem oder mehrere DNS-Server des Vertrauens beantwortet. Anfragen über Domäne in den Sperrlisten geben „0.0.0.0“ zurück, somit kann der unerwünschte Inhalt nicht geladen werden.

Manuelle Änderungen der DNS-Einstellungen von allen Heimnetzgeräte wären mühsam und oftmals nicht trivial (z.B. Smartphones) oder gar unmöglich (Smart-Geräte wie Fernseher, etc.). Der Ansatz von pi-hole ist die Einstellung im Router von *pi-hole* als lokaler DNS-Server und die Übernahme dadurch aller DNS-Anfragen aus dem Heimnetz[4].

Die Installation erfolgt über dem angebotenen Skript, der auf der IP-losen Box ohne Probleme ausgeführt wird. Bei der Ausführung werden Fragen über einer statischen IP-Adresse der Box gestellt, sowie auch über der Netzwerkschnittstelle,

[1] https://www.duckdns.org/about.jsp, (Zugriff am 08.05.2022)
[2] https://www.duckdns.org/install.jsp, (Zugriff am 08.05.2022)
[3] https://docs.pi-hole.net/, (Zugriff am 08.05.2022)
[4] https://docs.pi-hole.net/main/post-install/, (Zugriff am 08.05.2022)

wo pi-hole zuhören soll. Diese Einstellungen können nicht-konfiguriert/beliebig ausgewählt werden, weil sie in unserer Umgebung keine Rolle spielen und nur für DHCP-Dienste über *pi-hole* gedacht sind, die wir nicht benutzen.

Ein Aufruf aus dem Heimnetz über einem Browser von

http://99.99.99.99/admin[1] zeigt die Web-Oberfläche von *pi-hole* an, wo wir uns einloggen und verschiedene Statistiken sehen oder manche Einstellungen übernehmen können. Das Passwort für die Admin-Seite wird bei der Skript-Installation erstellt und im Terminal angezeigt, kann aber auch bei Bedarf mit „*pihole -a -p secretpassword*“ geändert werden[2].

Für die tatsächliche DNS-Funktion werden schon bei der Installation die Upstream-DNS-Server ausgewählt. Allerdings sind alle angebotene Adressen von „normalen“ DNS-Server und wir möchten unsere DNS Anfragen über HTTPS ausführen. Das *pi-hole* Projekt bietet in der Dokumentation auch eine detaillierte Information über der Benutzung von Cloudflare als DNS-Resolver an, über der von Cloudflare angebotenen Software namens „*cloudflared*“[3]. Nach dieser Anleitung erstellen wir den Installationsskript *cloudflared_install.sh*, der auch im Dateienanhang dieser Arbeit zu finden ist.

Als letzter Schritt stellen wir sicher, dass alle DNS-Anfragen über *pi-hole* verlaufen werden. Dieser Schritt ist aus zwei Gründen bedingt:

1. Unsere Anforderungen an Benutzerfreundlichkeit und einen *Plug-n-Play* Dienst widersprechen Einmischungen im Router, um DNS-Einstellungen zu ändern;

2. Selbst wenn die Einstellungen im Router geändert sind, gibt das keine Garantie, dass alle Geräte im Heimnetz diese Einstellungen beachten.

DNS, haben wir in den Grundlagen gesehen, läuft über UDP und den Port 53. Mit dieser Information fangen wir mittels *nftables* alle solche Pakete, die aus dem Heimnetz stammen und über UDP einen Server am Port 53 erreichen wollen. Die Einmischung erfolgt im Namensraum *default* vor jeder Routingentscheidung gleich an *veth1*. Die abgefangene Pakete leiten wir an *localhost* im Namensraum *default* weiter, wo *pi-hole* sie eventuell filtert und über DNS-over-HTTPS ausführt. Die Antworten versehen wir mit der ursprünglichen IP-Adresse des gewünschten DNS-Servers als Quelle und schicken sie zurück. Die Antwort sieht so aus, als würde sie vom gefragten Server kommen. Das ist auch eine Form von

[1] Vorausgesetzt wir benutzen die IP-Adresse aus den oberen Beispiele 99.99.99.99 für den Zugang aus dem Heimnetz.

[2] https://docs.pi-hole.net/core/pihole-command/#admin=, (Zugriff am 08.05.2022)

[3] https://docs.pi-hole.net/guides/dns/cloudflared/, (Zugriff am 08.05.2022)

NAT und zwar DNAT (Destination-NAT)[1]. Ausgedrückt in nft-Regeln sieht das wie folgt aus:

Listing 9: nftables: Regel zum Abfangen von DNS

```
1     chain prerouting {
2        type nat hook prerouting priority dstnat; policy accept;
3        iifname "veth1" meta l4proto {tcp, udp} th dport 53 counter dnat to \
4     77.77.77.50
5     }
```

Die korrekte Funktion des Filters können wir über einer Abfrage eines bekannten unerwünschten Domains testen, z.B. „*doubleclick.net*“. Zuerst führen wir die Abfrage über einem beliebigen DNS-Server aus der Box aus, z.B. 8.8.8.8 (Google DNS):

Listing 10: DNS: Standardanfrage ohne Filter

```
1   pi@pi3:~ $ dig @8.8.8.8 doubleclick.net
2
3   ; <<>> DiG 9.16.27-Raspbian <<>> @8.8.8.8 doubleclick.net
4   ; (1 server found)
5   ;; global options: +cmd
6   ;; Got answer:
7   ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15537
8   ;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
9
10  ;; OPT PSEUDOSECTION:
11  ; EDNS: version: 0, flags:; udp: 512
12  ;; QUESTION SECTION:
13  ;doubleclick.net. IN A
14
15  ;; ANSWER SECTION:
16  doubleclick.net. 300 IN A 172.217.19.78
17
18  ;; Query time: 19 msec
19  ;; SERVER: 8.8.8.8#53(8.8.8.8)
20  ;; WHEN: Thu Apr 21 13:14:50 CEST 2022
21  ;; MSG SIZE rcvd: 60
```

Die selbe Anfrage gemacht aus einem beliebigen Rechner hinter dem DNS-Filter ergibt das gewollte Ergebnis. Die Antwort sieht so aus, als würde sie von 8.8.8.8 kommen:

[1] https://wiki.nftables.org/wiki-nftables/index.php/Performing_Network_Address_ Translation_(NAT), (Zugriff am 08.05.2022)

Listing 11: DNS: Standardanfrage mit Filter

```
1  home:~$ dig @8.8.8.8 doubleclick.net
2
3  ; <<>> DiG 9.16.27-Debian <<>> @8.8.8.8 doubleclick.net
4  ; (1 server found)
5  ;; global options: +cmd
6  ;; Got answer:
7  ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35535
8  ;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
9
10 ;; OPT PSEUDOSECTION:
11 ; EDNS: version: 0, flags:; udp: 4096
12 ;; QUESTION SECTION:
13 ;doubleclick.net.  IN A
14
15 ;; ANSWER SECTION:
16 doubleclick.net. 2 IN A 0.0.0.0
17
18 ;; Query time: 4 msec
19 ;; SERVER: 8.8.8.8#53(8.8.8.8)
20 ;; WHEN: Thu Apr 21 13:16:16 CEST 2022
21 ;; MSG SIZE rcvd: 60
```

Im Log von *pi-hole* tauchen die Anfragen aus dem Heimnetz mit der Quelle 10.0.0.1 auf - die Adresse unserer (virtuellen) *veth0* Netzwerkkarte, die auch NAT durchführt.

6.4 Skriptausgaben

Nach den Anforderungen muss die Box zwei Arten von Ausgaben leisten: einfache und schnell lesbare Information für den unerfahrenen Benutzer und erweiterte Informationen für den Entwickler oder für den fortgeschrittenen Benutzer.

Einfache LED-Anzeige

Die schnell lesbare, einfache Ausgabe implementieren wir durch vier LED-Lampen. Dank dem GPIO-Header auf dem Pi, können die LEDs unkompliziert angeschlossen werden. Es entsteht eine Standardschaltung mit Strombegrenzungswiderstände, die nebeneinander stehende Header-Pins benutzt. Zusätzlich fügen wir einen Schalter zu, auch über einem Schutzwiderstand, der für die Betriebsmodusauswahl zuständig ist. Die komplette Schaltung ist auf Abbildung 14 auf der nächsten Seite dargestellt.

Als LED-Farben wählen wir wie folgt:

- Eine grüne LED wird dauerhaft leuchten, wenn der Skript läuft, um die Hauptfunktion anzuzeigen;

- Eine gelbe LED wird dauerhaft leuchten, solange die Box im Brückenmodus ist;
- Eine weiße LED wird dauerhaft leuchten, wenn die Box im Vollbetrieb ist;
- Eine rote LED wird dauerhaft leuchten, wenn ein Fehler aufgetreten ist.

Zusätzlich wird die weiße LED blinken, um den Ablauf der mehreren Phasen der Netzwerkanalyse und die Umstellung in den Vollbetrieb zu zeigen:

- *Lang AN, lang AUS*: Erfassung vom ankommenden Verkehr;
- *Lang AN, lang AUS, kurz AN, kurz AUS*: Erfassung vom ausgehenden Verkehr;
- *Lang AN, lang AUS, kurz AN, kurz AUS, kurz AN, kurz AUS*: Warten auf ARP-Reply;
- *Kurz AN, kurz AUS*: Firewall-Umstellung, Netzwerkschnittstellen- und Routen-Einstellung.

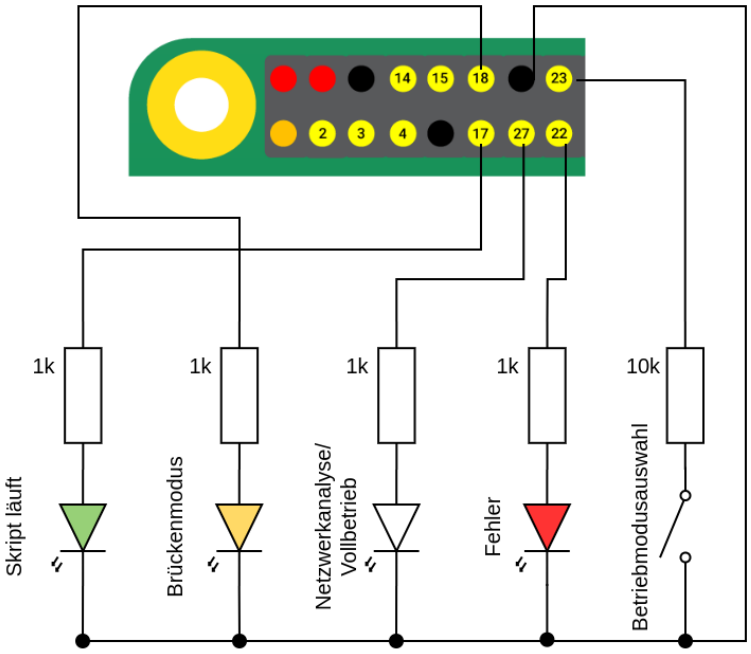

Abbildung 14: GPIO-Schaltung

Für die Software-Seite der GPIO-Steuerung hat sich die Software p*igpio*[1] als sehr geeignet gezeigt, weil sie einen direkten Zugriff über *bash* erlaubt, was die

[1]http://abyz.me.uk/rpi/pigpio/, Zugriff am 08.05.2022

Sprache unseres Skriptes ist. Die Installation auf dem Pi und die Einschaltung der Software als Service erfolgt mit:

Listing 12: Installation von pigpio

```
1  sudo apt install pigpio
2  sudo systemctl enable --now pigpiod.service
```

Die Kommunikation mit dem pigpio-Daemon erfolgt über das Programm *pigs*, wofür eine ausführliche Dokumentation angeboten ist[1].

Die für unsere Ziele bedeutendste Befehle sind:

- *pigs m PIN w* : stellt den Pin PIN im Ausgabemodus ein;
- *pigs m PIN r* : stellt den Pin PIN im Eingabemodus ein;
- *pigs pud PIN u* : schaltet einen internen Pull-Up Widerstand auf Pin PIN ein;
- *pigs proc PROGRAMM* : speichert das Programm PROGRAMM und gibt die ID-Nummer zurück, die für den Zugriff darauf benutzt wird;
- *pigs procr PROGID* : startet das gespeicherte Programm mit ID PROGID;
- *pigs procs PROGID* : stoppt die Ausführung des Programms mit ID PROGID;
- *pigs procd PROGID* : löscht das mit der ID PROGID gespeicherte Programm.

Die Programme, die *pigpio* unterstützt, bestehen aus einfachen Anweisungen mit eventuellen Sprünge und Parameter. Als Beispiel folgt unser Programm für die „Erfassung von ankommenden Daten“ (Zeile 77 und 158 im Hauptskript):

Listing 13: pigpio: Blink-Skript

```
1  SCRIPTBLINK1ID=$(pigs proc "tag 123 w 27 1 mils 500 w 27 0 mils 500 jmp 123")
```

Die ID des Programms wird in SCRIPTBLINK1IP gespeichert. Das tatsächliche Programm ist zwischen den Ausrufezeichen. Ein Tag mit beliebigem Namen wird erzeugt um später als Ziel für Sprünge zu dienen. Danach folgen die folgende Anweisungen:

[1] http://abyz.me.uk/rpi/pigpio/pigs.html, Zugriff am 08.05.2022

- Schalte Pin 27 ein; Warte 500 Millisekunden; Schalte Pin 27 aus; Warte 500 Millisekunden; Springe zu dem Tag „123“.

Die Schleife wird solange laufen, bis das Programm mit „*pigs procs ID*“ unterbrochen wird.

Protokollierung in Datei

Die Statusanzeige durch den LEDs ist praktisch und liefert schnelle und einfache Informationen. Für erweiterte und mehr detaillierte Informationen speichert der Skript regelmäßig Daten in externer Datei. Jede Zeile wird mit einem Timestamp versehen für eine schnelle Orientierung. Um weiter die Lesbarkeit zu verbessern, benutzt auch der Skript farbige Darstellungen zu einer Abgrenzung bestimmter Meldungen, wie z.B. rot für Fehler. Bei einer Ausgabe in *bash* werden diese vom Shell interpretiert und sorgen für eine leicht ablesbare Anzeige.

6.5 Aufbau vom Hauptskript; Aufbau vom Projekt

Der Hauptskript „*BOXrun.sh*“ ist in folgenden Teile aufgebaut:

- Startteil mit Definitionen von Variablen, Konstanten u.a.
- Eine Sammlung von Funktionen die für kleinere Aufgaben zuständig sind;
- Die Funktion *set_bridge* für die Einstellung von Brückenmodus;
- Die Funktion *parse_network_data* für die Erfassung und Auswertung von Netzwerkdaten;
- Die Funktion *set_full_mode*, die die Box im Vollbetrieb umstellt;
- Wrapper-Funktionen, die für den Aufruf von anderen Funktionen notwendig sind;
- Hauptteil des Skriptes, wo die Aufrufparameter analysiert werden und dementsprechend den Ablauf gesteuert wird.

Die Standardausgabe vom Skript, wenn keine oder falsche Parameter eingegeben werden ist eine kleine formatierte Anleitung auszugeben. Durch die Parameter können alle Hauptfunktionen einzeln aufgerufen werden, oder es kann auch die vollautomatische Funktion gestartet werden. Diese basiert auf einer endlosen Schleife, die auf Benutzereingaben reagiert und regelmäßig die DHCP-Daten aktualisiert. Die Schleife arbeitet mit „Ticks“ und in der aktuellen Version des

Skriptes ein „Tick“ ist 5 Sekunden lang. Die vollautomatische Funktion ist über dem Parameter „*onboot*“ eingestellt und eignet sich gut aus Cron gestartet zu werden, wie z.B. mittels:

Listing 14: cron: Eintrag für die IP-lose Box

```
1 @reboot /home/pi/BOX/BOXrun.sh onboot 2>/home/pi/BOX/logs/errors
```

Der aktuelle Status der Box speichert der Skript in einer Datei. Die Information aus dieser Datei, kombiniert mit der Angabe vom Benutzerschalter entscheiden den potenziellen Wechsel zwischen den Modi.

Eine Darstellung des allgemeinen Ablaufs vom Skript[1] ist auf Abbildung 15 auf der nächsten Seite zu sehen.

[1] ohne Synchronisierung und ohne den Scapy-DHCP-Server - diese sind auf Abbildung 13 auf Seite 74 zu finden

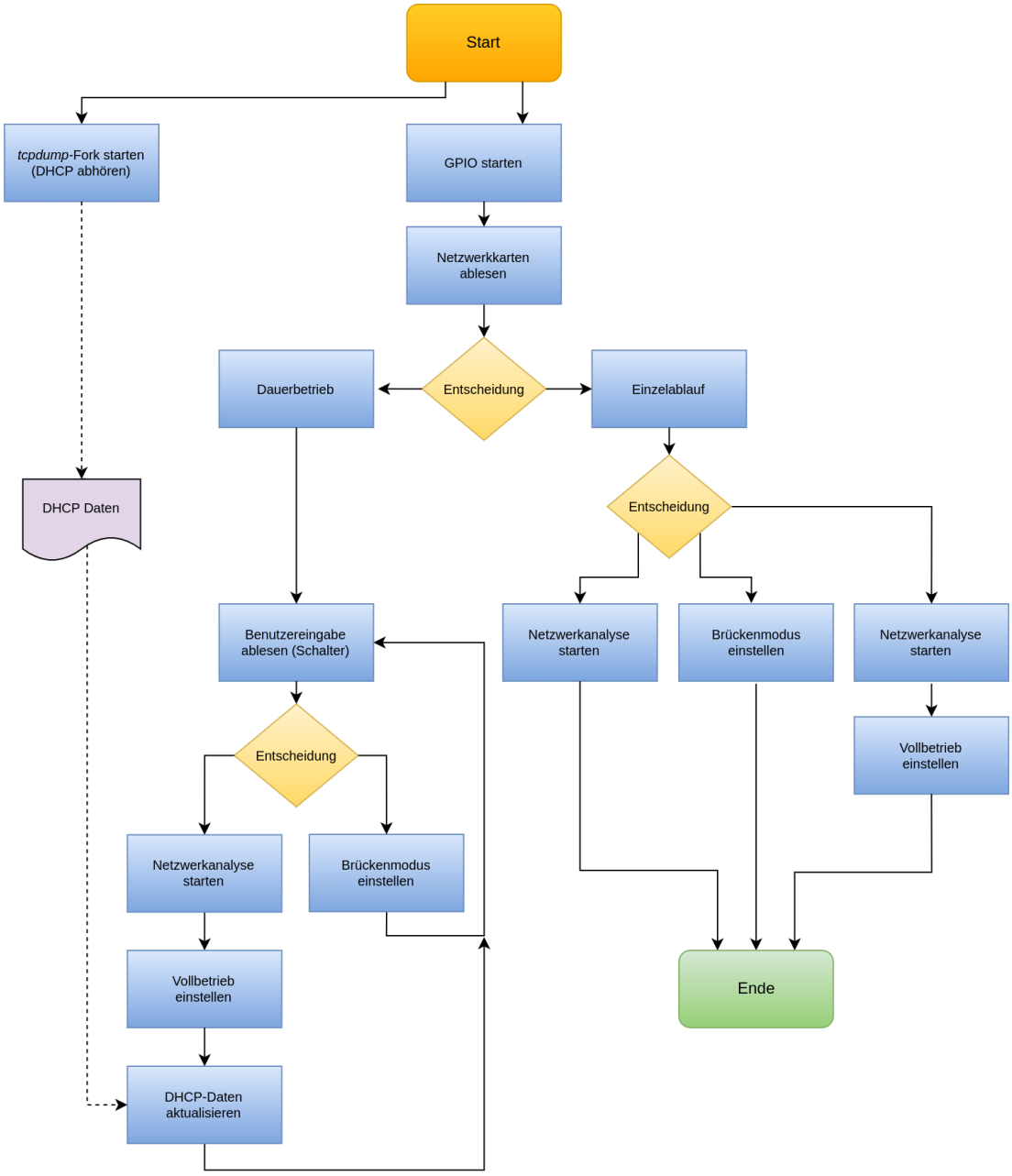

Abbildung 15: IP-lose Box: Ablaufdiagramm

Mit Bezug auf die Dateiorganisation besteht das Projekt aus dem Hauptskript, dem Python Skript, Konfigurationsdateien, und Hilfsskripten. Zusätzlich sind die Installationsskripten und die zugehörigen Dateien von „cloudflared“ und „duckdns“ in separate Ordner gesammelt. Die gesamte Dateiorganisation ist auf Abbildung 16 auf der nächsten Seite dargestellt.

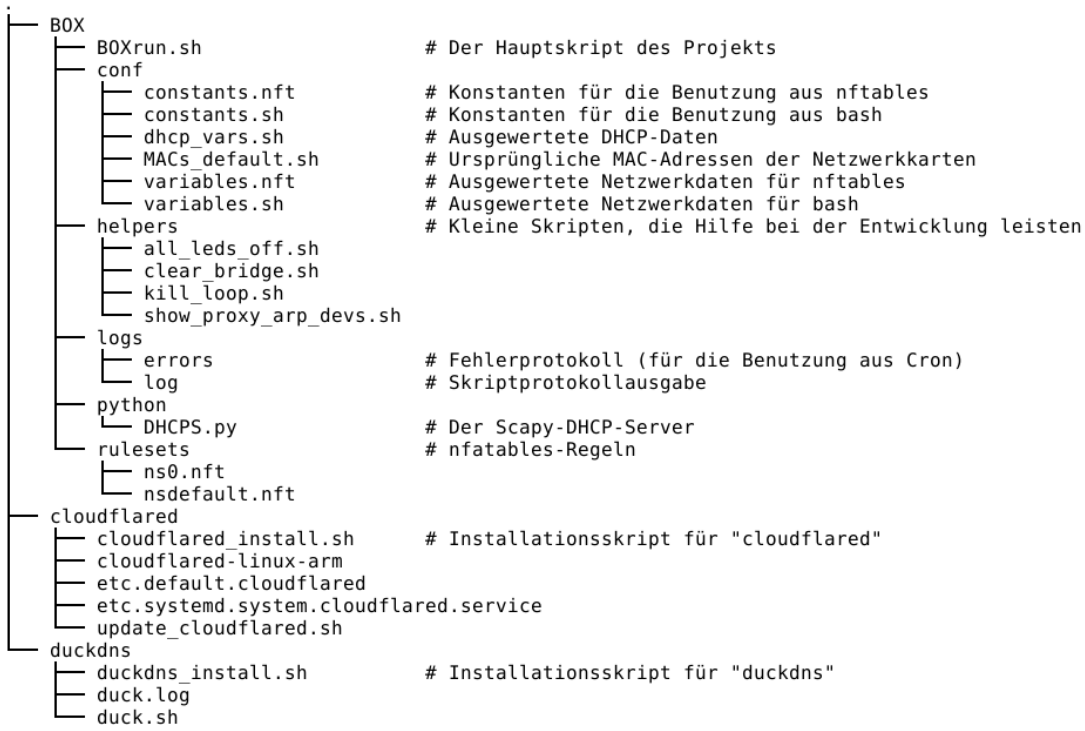

Abbildung 16: IP-lose Box: Projektstruktur

7 Schluss

In dieser Arbeit wurde das Konzept für ein Netzwerkgerät vorgestellt, das aktiv am Netz teil nimmt, ohne den Bedarf jeglicher Ein- oder Umstellungen bei den anderen vorhandene Geräte. Für das neue Gerät wurde die Bezeichnung „*IP-lose Box*“ gebraucht. Durch die Benutzung einer schon im Netz gültigen IP-Adresse, und mithilfe von Virtualisierung, NAT und andere Technologien, kann die Voraussetzung für eine eigene IP-Adresse für die Netzwerkteilnahme erfüllt werden, auch wenn keine neue Adresse im Subnetz vergeben wird. Als Motivation für das Konzept sind die Gelegenheitsbenutzer, die durch den komplexen Anforderungen einer Netzwerkkonfiguration überfordert werden. Deren Bedürfnisse von Netzwerkdienste werden aufgrund der Komplexität einer eigenen Installation entweder nicht bedeckt, oder zu Anbieter mit einer eventuell fragbaren Datenschutzpolitik weitergeleitet. Die implementierte Lösung hat das Ziel als ein „*Plug-and-Play*“ Gerät zu arbeiten - ohne weiterer Einmischung vom Benutzer nach dem Anschließen. Durch Analyse des Netzverkehrs berechnet das neue Gerät selber die für seine weitere Funktion benötigte Daten, sowie auch die Richtung des Anschlusses. Die automatische Aktualisierungen wurden auch als Teil der selbstständigen Funktion konzipiert. Die Kommunikation mit dem Benutzer erfolgt über einfache LED-Anzeigen mit mehrere Modi, um die verschiedene Phasen des Betriebs darzustellen. Für die fortgeschrittene Benutzer ist erweiterte Information in Dateien vorgesehen. Ein DNS-Filter wurde in der Implementierung integriert um eines der Anwendungsszenarien zu demonstrieren. Als Anschlussposition wurde den Engpass an der Grenze des Heimnetzwerkes ausgewählt, damit das ganze Heimnetz davon profitieren kann.

Nach dem Anschluss des neuen Geräts werden automatisch alle Geräte im Heimnetz durch den DNS-Filter geschützt. Eine Kommunikation mit dem Gerät ist aus dem Heimnetz möglich, sowie auch aus dem Internet, durch die Benutzung von Standardsoftware - SSH. Ein Zugriff aus dem Internet auf einen Rechner im Heimnetz ist über Reverse-SSH möglich, auch wenn keine Ports am Heimnetzrouter geöffnet sind. Für die Erreichbarkeit aus dem Internet kümmert sich Dynamic-DNS.

Für die Funktion des DNS-Filters muss kein Gerät neu konfiguriert werden, weil alle DNS-Anfragen automatisch abgefangen und vom neu-angeschlossenen Gerät beantwortet werden. Als extra Schutz wird der Zugriff auf dem Heimnetz aus dem Internet gesperrt und nur für ausgewählte Anwendungen, oder gar nicht, zugelassen. Alle andere Kommunikationen funktionieren ungestört vom neuen Anschluss weiter.

Als Hardware für die Implementierung wurde ein Raspberry-Pi 3 ausgewählt. Das Gerät hat einige Hardware-bedingte Einschränkungen, wie etwa die Geschwindigkeit des Ethernet-Ports. Zusätzlich ist die Situation durch die Benutzung eines USB-zu-Ethernet Adapter als zweite Netzwerkschnittstelle weiter verschlechtert. Für die Ziele der Demonstration zeigte sich der Pi3 als gut genug geeignet, weitere Tests mit anderer Hardware sind notwendig. Es muss ein Kompromiss zwischen den wirtschaftlichen Aspekten der Implementierung und der Leistung der Hardware gefunden werden.

Das vorgestellte Konzept setzt einige Annahmen voraus, die die Umgebung vereinfachen. IP Version 6 wurde überhaupt nicht betrachtet. Von den Netzwerk-Protokolle haben wir nur die nach unserer Meinung bedeutendste kommentiert. Weitere Tests und Forschung für die Funktion in möglichst viele Netzwerkumgebungen sind notwendig.

Die vorgestellte Beispielumgebung eines Heimnetzes mit Kabelanschluss hat Einschränkungen bei der Anschlussmöglichkeiten gezeigt, aufgrund des schwierigen Zugangs von der externen Seite des DOCSIS-Modems. Wenn der Router und das Modem zwei separate Geräte sind, ist das kein Problem. Leider kann das nicht immer vorausgesetzt sein. Weitere Forschung in dieser Richtung soll mögliche Lösungen erkennen oder Umwege anbieten, die die gezielte „*Plug-and-Play*“ Funktionalität behalten.

Die Arbeit mit ARP und DHCP ist in der aktuellen Implementierung nicht vollständig und unterstützt nur die Grundfunktionen. Weitere Recherche in diesem Bereich soll die Situation verbessern. Die Reaktion auf ARP-Anfragen auf dem internen Netz benötigt weitere Aufmerksamkeit, sowie auch die begrenzte Fähigkeiten des gezeigten DHCP-Konzepts. In der vorgestellten Implementierung werden z.B. nicht alle DHCP-Nachrichten bearbeitet und solche wie etwa DHCP-NAK werden einfach ignoriert. Bei der Erfassung von DHCP-Daten wird auch die Annahme gemacht, dass die DHCP-Optionen mit „*Message Type*“ anfangen. Das ist vom Standard nicht vorgeschrieben und sich darauf zu verlassen kann eventuell zu Probleme führen.

Bei dem Zugang aus dem Internet bekommt zuerst die IP-lose Box alle ankommenden Anfragen. Es gibt keinen direkten Weg zu wissen, ob und welche Ports am Router aus dem Internet erreichbar sein sollen. Es kann dadurch vorkommen, dass Ports am Router und am neu-angeschlossenen Gerät sich überlappen. Eventuelle Lösungen wie etwa Portscanning würden die Netzwerkanalyse komplizierter machen und wurden hier nicht kommentiert.

Die aktuell angebotene Skripten dienen als Demonstration und nicht als eine

komplette und robuste Implementierung. Kleine Fehlkonditionen und unerwartete Ereignisse führen schnell zu einem Fehlermodus im Betrieb. Die Fehlererkennung und die Fehlerbehandlung sind in den Skripten kaum vorhanden. Die Rückmeldungen der meisten Befehle müssen über bash-Umleitungen abgefangen werden, was sicherlich keine optimale Lösung ist.

Erweiterte Tests mit verschiedener Hardware und Netzumgebungen sind notwendig um das Konzept und die Implementierung zu verbessern. In ihrer vorgestellten Form sollen sie hier als reine Demonstration dienen und machen keinen Anspruch auf Vollständigkeit.

Anhang

Anhang 1: DNS und TLS bei der www.akad.de Webseite

DNS-Anfrage

```
1  $ dig +trace www.akad.de @83.169.185.161
2
3  ; <<>> DiG 9.16.22-Debian <<>> +trace www.akad.de @83.169.185.161
4  ;; global options: +cmd
5  .   27320 IN NS f.root-servers.net.
6  .   27320 IN NS g.root-servers.net.
7  .   27320 IN NS h.root-servers.net.
8  .   27320 IN NS a.root-servers.net.
9  .   27320 IN NS i.root-servers.net.
10 .   27320 IN NS j.root-servers.net.
11 .   27320 IN NS k.root-servers.net.
12 .   27320 IN NS l.root-servers.net.
13 .   27320 IN NS m.root-servers.net.
14 .   27320 IN NS b.root-servers.net.
15 .   27320 IN NS c.root-servers.net.
16 .   27320 IN NS d.root-servers.net.
17 .   27320 IN NS e.root-servers.net.
18 .   27320 IN RRSIG NS 8 0 518400 20220323220000 20220310210000 9799 .
       f63CqF1Eq4B/mLXFnOM8ccQFZm40OVfY2mpDiAgOu+MPiALcpsvEqU7g
       vbUGP3A5JySfg7IlLjso9tYoJ2hyLXtGfkmhTjALbJnAB9U3E0S/bKdO
       ZHf9vs1yiXwgS4kf88bijjwm47REbWUGx3NklcnSCISHAyh2eEPFHgga VERVgr/
       qlSxxZuJF7x3xYJLtBngI37lG6mGkxv476MmmxsbWwzSgfdqN
       g7mdZR7LWc9EiMnfGwWSOVZzyE+psTfQboIUfxwbR+khTB39m494BgYF 9
       raZWYI0Z24ebhJWfEpHHGV5u2a5v1pavX+8WNnEECQ2PjpSc98jVyZd 3m5Brw==
19 ;; Received 1097 bytes from 83.169.185.161#53(83.169.185.161) in 28 ms
20
21 de.   172800 IN NS a.nic.de.
22 de.   172800 IN NS f.nic.de.
23 de.   172800 IN NS l.de.net.
24 de.   172800 IN NS n.de.net.
25 de.   172800 IN NS s.de.net.
26 de.   172800 IN NS z.nic.de.
27 de.   86400 IN DS 26755 8 2
       F341357809A5954311CCB82ADE114C6C1D724A75C0395137AA397803 5425E78D
28 de.   86400 IN RRSIG DS 8 1 86400 20220324050000 20220311040000 9799 .
       FfObdfGyLwbDKs7vdJBWpwumFIz/75uMIAOiJI64npeRBQeASf12lVcM 15
       meYeDTekALepu9f8SNjglg62REF362W26Hn4NYAQhkH4h+QzxSHVfq
       LpSntHh7eSA73qU7GqRHDjoailSCC1mBnygOqT3WIxDCKJMb4TYlDVNc dBdfEBUByeDPmj/
       PqgwSgoEGH9Qmue+G4EpaBdKdVyLr5lk/ny1QW5Le
       uPKHUasEnbEeMATz6f61OEfOqQIDyBMlBX1JtFotSkZ6rCHJRHCiZkfq
       cpfBTi0lJgWY9iE3FYRI4jFIoP8TLtZcyzNe1KTpv+BvYYVr7b5Nv4Mi fEDnjw==
29 ;; Received 745 bytes from 192.203.230.10#53(e.root-servers.net) in 28 ms
30
31 akad.de.  86400 IN NS ns01.agenturserver.co.
32 akad.de.  86400 IN NS ns01.agenturserver.de.
33 akad.de.  86400 IN NS ns01.agenturserver.it.
34 tjlb7qbojvmlf1s6gdriru7vsms1lg16.de. 7200 IN NSEC3 1 1 15 CA12B74ADB90591A
       TJLFJPRVCMSTH243PMI98VQTJ9PGI7US NS SOA RRSIG DNSKEY NSEC3PARAM
```

```
35  q2c62r68boc158rg1flpc1g2cv8l5cd7.de. 7200 IN NSEC3 1 1 15 CA12B74ADB90591A
        Q2CCL4JH29QMFL9AKGP0IMRH8A530TCN
36  tjlb7qbojvmlf1s6gdriru7vsms1lg16.de. 7200 IN RRSIG NSEC3 8 2 7200
        20220323101708 20220309084708 58473 de. yzCAC2UBBJY/EJSpu62L9BeaN+7P/
        BmrrmaDRvn3djl8Tt0z2fiXdqHy L/Rb77SIPJRGmujDY8eJgv+
        kfWE3SSKkbIjmy5PNulXjKQdHoTC9TYr5 nhQbP9h42x9yTBQshXm093XO3WVmopcgPiBsg5J/
        CjCx+09nMvdm9h9y qAM=
37  q2c62r68boc158rg1flpc1g2cv8l5cd7.de. 7200 IN RRSIG NSEC3 8 2 7200
        20220323112131 20220309095131 58473 de.
        vABjQ8KYgVBld5LnlgIEW8V17ehZfHYZ43KbvGJAq+dKLAEk9WtpfJaG 39
        wEpzlHfud6Ctk23eNkaXhDsP4qc8YMGekZsFMD6UyFJ8DEwmcD/2tW +0UX+YL/9+
        UPIPfi3U3PI2jkFvnI3xUzwwcKD4A0os+VWCVtYqnPKSyW zqU=
38  ;; Received 652 bytes from 195.243.137.26#53(s.de.net) in 20 ms
39
40  www.akad.de.  10800 IN A 188.94.253.154
41  ;; Received 56 bytes from 185.247.150.113#53(ns01.agenturserver.co) in 28 ms
```

Authentifizierung

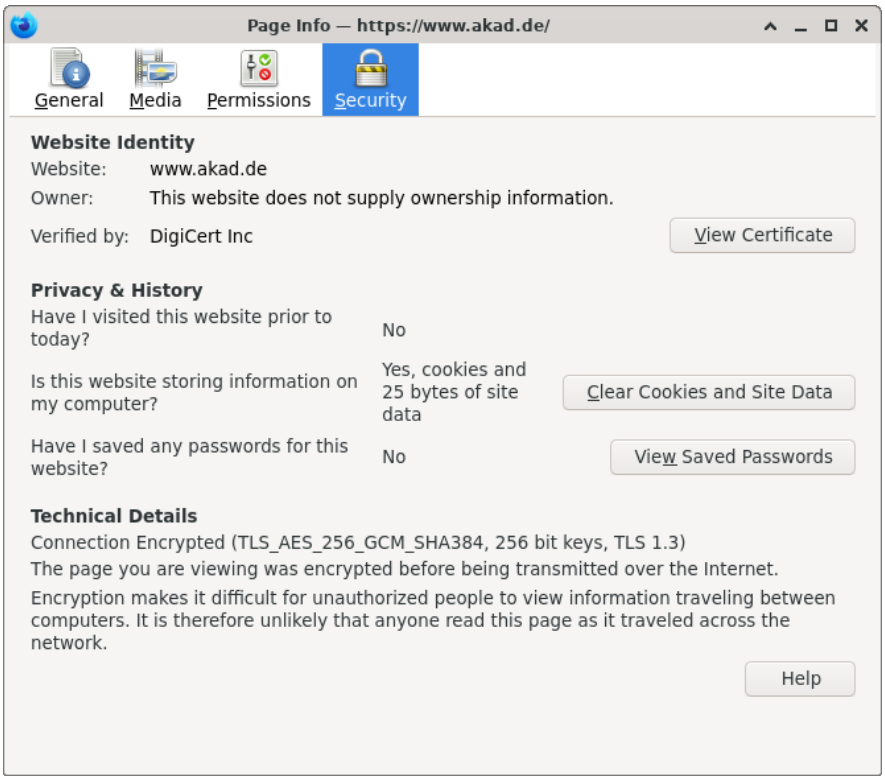

Abbildung 17: Authentifizierung der Akad-Webseite

Anhang 2: GPIO-Pins von Raspberry-Pi

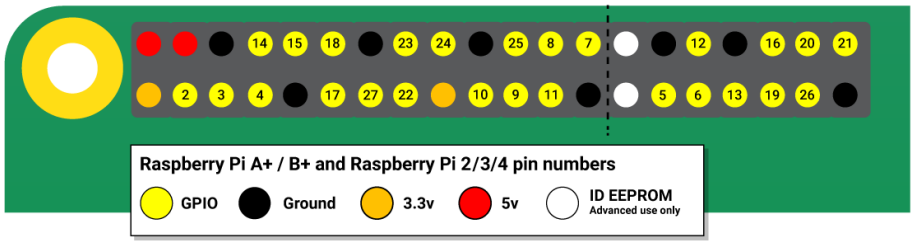

Abbildung 18: GPIO-Pins von Raspberry-Pi

(Quelle: https://www.raspberrypi.com/documentation/computers/os.html#gpio-and-the-40-pin-header, Zugriff am 08.05.2022)

Literaturverzeichnis

Anderson, D. (2020)
How NAT traversal works, https://tailscale.com/blog/how-nat-traversal-works/ (Zugriff am 08.05.2022)

Baun, C. (2020)
Computernetze kompakt: Eine an der Praxis orientierte Einführung für Studium und Berufspraxis, 5. Auflage, Berlin

Beaupré, A. et al. (2022)
Debian Manpages, https://manpages.debian.org/ (Zugriff am 08.05.2022)

Benvenuti, C. (2006)
Understanding Linux Network Internals, Beijing et al.

Biondi, P.; the Scapy community (2022)
Scapy's documentation, https://scapy.readthedocs.io/en/latest/ (Zugriff am 08.05.2022)

Bovet, D. P.; Cesati, M. (2005)
Understanding the Linux Kernel, Third Edition, Beijing et al.

Brown, B. (2019)
Facebook's Catastrophic Blackout Could Cost USD 90 Million in Lost Revenue, CCN - Capital & Celeb News, https://www.ccn.com/facebooks-blackout-90-million-lost-revenue/ (Zugriff am 08.05.2022)

Bundesministerium für Verkehr und digitale Infrastruktur (2021)
Aktuelle Breitbandverfügbarkeit in Deutschland (Stand Mitte 2021), o.O.

Bundesnetzagentur (2021)
Jahresbericht 2020: Märkte im digitalen Wandel, o.O.

Cable Television Laboratories, Inc. (2017)
Data-Over-Cable Service Interface Specifications: Cable Modem to Customer Premise Equipment Interface Specification, o.O.

Cable Television Laboratories, Inc. (2015)
Data-Over-Cable Service Interface Specifications DOCSIS® 3.1: MAC and Upper Layer Protocols Interface Specification, o.O.

Cable Television Laboratories, Inc. (2016)
Data-Over-Cable Service Interface Specifications DOCSIS® 3.1: Security Specification, o.O.

Chimata, A. K. (2005)
Path of a packet in the Linux kernel stack, Kansas

Cimpanu, C. (2019)
DNS-over-HTTPS causes more problems than it solves, experts say, https://www.zdnet.com/article/dns-over-https-causes-more-problems-than-it-solves-experts-say/ (Zugriff am 08.05.2022)

Cloudflare, Inc. (2022)
Learning Center, https://www.cloudflare.com/learning/ (Zugriff am 08.05.2022)

Debian Wiki team (2022)
Debian Wiki, https://wiki.debian.org/ (Zugriff am 08.05.2022)

Deutscher Bundestag (2015)
Deutscher Bundestag, Stenografischer Bericht, 133. Sitzung, Berlin

Deutscher Bundestag (2021)
Gesetz zur Umsetzung der Richtlinie (EU) 2018/1972 des Europäischen Parlaments und des Rates vom 11. Dezember 2018 über den europäischen Kodex für die elektronische Kommunikation (Neufassung) und zur Modernisierung des Telekommunikationsrechts (Telekommunikationsmodernisierungsgesetz), in: Bundesgesetzblatt Jahrgang 2021 Teil I Nr. 35, Bonn

Duckbill Holdings (Canada) Ltd (2022)
Duck DNS, https://www.duckdns.org/ (Zugriff am 08.05.2022)

Excentis (o.J.)
XRA-31 DOCSIS Protocol Analyzer, https://www.excentis.com/products/xra-31-docsis-protocol-analyzer (Zugriff am 08.05.2022)

Group, N. W. (2006)
The Secure Shell (SSH) Protocol Architecture, o.O.

Internet Engineering Task Force (2018)
The Transport Layer Security (TLS) Protocol Version 1.3, o.O.

Internet Security Research Group (2022)
Let's Encrypt Stats, https://letsencrypt.org/stats/ (Zugriff am 08.05.2022)

Internet Systems Consortium, Inc. (2022)
ISC's Knowledgebase, https://kb.isc.org/v1/docs (Zugriff am 08.05.2022)

Joan und Mitwirkende (2021)
The pigpio library, http://abyz.me.uk/rpi/pigpio/ (Zugriff am 08.05.2022)

Kerrisk, M.; Mitwirkende (2022)
The Linux man-pages project, https://www.kernel.org/doc/man-pages/ (Zugriff am 08.05.2022)

Liu, H. (2018)
Introduction to Linux interfaces for virtual networking, https://developers.redhat.com/blog/2018/10/22/introduction-to-linux-interfaces-for-virtual-networking (Zugriff am 08.05.2022)

Luntovskyy, A.; Gütter, D. (2020)
Moderne Rechnernetze: Protokolle, Standards und Apps in kombinierten drahtgebundenen, mobilen und drahtlosen Netzwerken, Wiesbaden

Mandl, P. (2019)
Internet Internals: Vermittlungsschicht, Aufbau und Protokolle, Wiesbaden

McHardy, P.; Ayuso, P. N. (2020)
Man page of NFT, https://www.netfilter.org/projects/nftables/manpage.html (Zugriff am 08.05.2022)

Microsoft Inc. (2021)
Windows Network Architecture and the OSI Model, https://docs.microsoft.com/en-us/windows-hardware/drivers/network/windows-network-architecture-and-the-osi-model (Zugriff am 08.05.2022)

Network Working Group (1982)
An Ethernet Address Resolution Protocol, o.O.

Network Working Group (1987a)
Domain Names - Concepts and Facilities, o.O.

Network Working Group (1987b)
Domain Names - Implementation and Specification, o.O.

Network Working Group (1997a)
DHCP Options and BOOTP Vendor Extensions, o.O.

Network Working Group (1997b)
Dynamic Host Configuration Protocol, o.O.

Network Working Group (2000)
Architectural Implications of NAT, o.O.

Network Working Group (2001)
Traditional IP Network Address Translator (Traditional NAT), o.O.

Parziale, L. et al. (2006)
TCP/IP tutorial and technical overview, o.O.

Pi-hole LLC (2022)
Pi-hole documentation, https://docs.pi-hole.net/ (Zugriff am 08.05.2022)

Prince, M. (2020)
Introducing 1.1.1.1 for Families, https://blog.cloudflare.com/introducing-1-1-1-1-for-families/ (Zugriff am 08.05.2022)

Prytuluk, M. (2022)
Preventing Circumvention of Cisco Umbrella with Firewall Rules, https://support.umbrella.com/hc/en-us/articles/230904088-Preventing-Circumvention-of-Cisco-Umbrella-with-Firewall-Rules (Zugriff am 08.05.2022)

ptef (2018)
dhcp-scapy-server, https://github.com/ptef/dhcp-scapy-server (Zugriff am 08.05.2022)

Ramey, C.; Fox, B. (2020)
Bash Reference Manual, o.O.

Raspberry Pi Foundation (2022)
Raspberry Pi Documentation, https://www.raspberrypi.org/documentation/ (Zugriff am 08.05.2022)

Sandvine Corporation (2022)
The Global Internet Phenomena Report, o.O.

Sharpe, R.; Warnick, E.; Lamping, U. (o.J.)
Wireshark User's Guide, https://www.wireshark.org/docs/wsug_html/ (Zugriff am 08.05.2022)

Statista, Inc. (2021)
Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030, https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (Zugriff am 08.05.2022)

Sullivan, N. (2014)
DNSSEC: An Introduction, https://blog.cloudflare.com/dnssec-an-introduction/ (Zugriff am 08.05.2022)

Tanenbaum, A. S.; Wetherall, D. J. (2014)
Computer Networks, Fifth Edition, o.O.

The kernel development community (2022)
The Linux Kernel documentation, https://www.kernel.org/doc/html/latest/ (Zugriff am 08.05.2022)

The Linux Foundation (2022)
Linux Foundation DokuWiki, https://wiki.linuxfoundation.org/start (Zugriff am 08.05.2022)

The netfilter.org project (2021)
nftables wiki, https://wiki.nftables.org/wiki-nftables/index.php (Zugriff am 08.05.2022)

The netfilter.org project (2022)
The netfilter.org project, https://www.netfilter.org/ (Zugriff am 08.05.2022)

The Tcpdump Group (2022)
TCPDUMP & LIBPCAP, https://www.tcpdump.org/ (Zugriff am 08.05.2022)

Verma, T.; Singanamalla, S. (2020)
Improving DNS Privacy with Oblivious DoH in 1.1.1.1, https://blog.cloudflare.com/oblivious-dns/ (Zugriff am 08.05.2022)

Vodafone Deutschland GmbH (2022)
Kabel Hardware & Optionen, https://zuhauseplus.vodafone.de/internet-telefon/kabel/router-optionen/ (Zugriff am 08.05.2022)

Whittaker, Z. (2019)
Internet group brands Mozilla 'internet villain' for supporting DNS privacy feature, in: TechCrunch, https://techcrunch.com/2019/07/05/isp-group-mozilla-internet-villain-dns-privacy/ (Zugriff am 08.05.2022)

Ich versichere, dass ich die beiliegende Abschlussarbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe.

....., 08.05.2022

_____

(Vladimir Zhelezarov)