ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

STL SYSTEMTECHNIKLABOR

Evaluierung, Design und Implementierung einer Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks

Peter Schmidt

Technical Report – STL-TR-2018-02 – ISSN 2364-7167

stl.htwsaar.de

Technische Berichte des Systemtechniklabors (STL) der htw saar
Technical Reports of the System Technology Lab (STL) at htw saar
ISSN 2364-7167

Peter Schmidt: Evaluierung, Design und Implementierung einer Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks
Technical report id: STL-TR-2018-02

First published: March 2018
Last revision: March 2018
Internal review: Markus Esch

For the most recent version of this report see: https://stl.htwsaar.de/

Title image source: karmarat, http://www.freeimages.com/photo/404031

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. http://creativecommons.org/licenses/by-nc-nd/4.0/

htw saar – Hochschule für Technik und Wirtschaft des Saarlandes (University of Applied Sciences)
Fakultät für Ingenieurwissenschaften (School of Engineering)
STL – Systemtechniklabor (System Technology Lab)
Prof. Dr.-Ing. André Miede (andre.miede@htwsaar.de)
Goebenstraße 40
66117 Saarbrücken, Germany
https://stl.htwsaar.de

ingenieur
wissenschaften
htw saar

Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Informatiker (Dipl.-Inf.)
an der Hochschule für Technik und Wirtschaft des Saarlandes
im Studiengang Praktische Informatik
der Fakultät für Ingenieurwissenschaften

Evaluierung, Design und Implementierung einer Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks

vorgelegt von
Peter Schmidt

betreut und begutachtet von
Prof. Dr. Markus Esch

Saarbrücken, 30. November 2017

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich erkläre hiermit weiterhin, dass die vorgelegte Arbeit zuvor weder von mir noch von einer anderen Person an dieser oder einer anderen Hochschule eingereicht wurde.

Darüber hinaus ist mir bekannt, dass die Unrichtigkeit dieser Erklärung eine Benotung der Arbeit mit der Note „nicht ausreichend“zur Folge hat und einen Ausschluss von der Erbringung weiterer Prüfungsleistungen zur Folge haben kann.

Saarbrücken, 30. November 2017

Peter Schmidt

Zusammenfassung

Bei LAN-LAN-Kopplung mit Hilfe von Virtual Private Networks mit der Zielsetzung der Hochverfügbarkeit genügt es nicht, diese Virtual Private Networks mehrfach redundant auszulegen. Um hochverfügbare LAN-LAN-Kopplung zu erreichen, ist es nötig, die genutzten Virtual Private Networks mehrfach redundant mit unterschiedlichen zu Grunde liegenden Transportpfaden auszulegen, damit wirkliche Redundanz erzielt wird. Overlay-Netzwerke mit niedriger Latenz können als alternative Transportpfade für diese Virtual Private Networks (VPNs) zur redundanten und hochverfügbaren LAN-LAN-Kopplung dienen.
Diese Arbeit beschreibt die Konzeption und die Implementierung einer Routerumgebung zur Kopplung von LANs mit Hilfe von VPNs und mit Overlay-Netzwerken als alternative Transportpfade für diese VPNs in Verbindung mit dynamischem Routing zur Realisierung von Redundanz und damit zur Realisierung von Hochverfügbarkeit. Dabei wird auf die Systemarchitektur und wichtige bei Konzeption und Implementierung getroffene Entscheidungen eingegangen. Bei der Implementierung zu lösende Problemstellungen werden beschrieben und Lösungen für diese Problemstellungen werden dargestellt.
Die Konzeption und Implementierung beinhaltet die Entwicklung einer Software als Laufzeitumgebung in Form eines modularen Frameworks zur Steuerung der benötigten und genutzten Software-Komponenten. Abschliessend wird die Evaluation der Routerumgebung im praktischen Einsatz nebst einer Auswertung von qualitativen und quantitativen diesbezüglichen Indikatoren dargestellt.

Computers can do lots of things. They can add millions of numbers in the twinkling of an eye. They can outwit chess grandmasters. They can guide weapons to their targets. They can book you onto a plane between a guitar-strumming nun and a non-smoking physics professor. Some can even play the bongoes. That's quite a variety! [...] [sic]

— Richard P. Feynman [1]

Danksagung

Ich danke meinem Betreuer Prof. Dr. Markus Esch, der mir diese Arbeit ermöglicht hat und der sich immer Zeit für mich und meine Arbeit genommen hat und mich hervorragend betreut hat, und Prof. Dr.-Ing. André Miede als Leiter des Systemtechniklabors für die Betreuung in organisatorischer Hinsicht.

Ich danke den Ansprechpartnern des Systemtechniklabors, allen voran Wolfgang Pauly, Achim Pick und Frau Margit Meyer, für ihre Unterstützung mit Zeit und Ressourcen.

Ich danke Roman Schließmeyer von der SKYWAY DataCenter GmbH für die Bereitstellung eines VM-Testbeds im Rechenzentrum der SKYWAY DataCenter GmbH in St. Ingbert.

Ich danke meiner Chinesisch-Deutsch-Tandem-Lernpartnerin LIU Huiming für ihre Impulse bezüglich der Hintergründe der Konzeption dieser Arbeit.

Zuletzt möchte ich Prof. Dr. Klaus Huckert danken für seinen Zuspruch und für seine Bestärkung, meinen Weg zu wählen und zu gehen.

Inhaltsverzeichnis

1 Einleitung **1**
1.1 Motivation . . . 1
1.2 Aufgabenstellung und Zielsetzung . . . 2
1.2.1 LAN-LAN-Kopplung . . . 2
1.2.2 Herausforderungen bezüglich der Verfügbarkeit . . . 2
1.3 Aufbau der Arbeit . . . 3
1.4 Projektplanung . . . 4

2 Grundlagen **5**
2.1 Routing im Internet . . . 5
2.1.1 Struktur des Backbones des Internets . . . 6
2.1.2 Verwendete Routing-Protokolle . . . 8
2.1.3 Routing-Software im Internet . . . 15
2.2 Virtual Private Networks . . . 21
2.2.1 Definition von "Virtual Private Network" . . . 22
2.2.2 Framework für IP-basierte VPNs . . . 22
2.2.3 Von ISPs bereitgestellte VPNs . . . 22
2.2.4 Standardisierte VPN-Protokolle . . . 23
2.2.5 Nicht-standardisierte VPN-Protokolle . . . 26
2.2.6 De-facto-Standards bei VPN-Protokollen . . . 33
2.3 Overlay-Netzwerke . . . 34
2.3.1 Verbreitete Overlay-Netzwerke . . . 37

3 Analyse **43**
3.1 Eingesetzte VPN-Protokolle zur LAN-LAN-Kopplung . . . 43
3.2 Geeignete Routing-Protokolle . . . 44
3.3 Overlay-Netzwerke im Internet . . . 45
3.3.1 Veränderung der Netzwerktopologie durch Overlay-Netzwerke . . . 46
3.4 Zusammenfassung . . . 48

4 Verwandte Arbeiten **51**
4.1 HORNET . . . 51
4.2 OnionCat . . . 52

5 Konzeption **55**
5.1 Übersicht . . . 56
5.2 Reihenfolge bei der Auswahl der Komponenten . . . 58
5.3 Auswahl der Routing-Software . . . 58
5.3.1 Kriterien für die Auswahl . . . 58
5.4 Auswahl des Overlay-Netzwerks . . . 59
5.5 Auswahl des VPN-Protokolls . . . 61
5.6 Auswahl des Router-Betriebssystems . . . 63
5.7 Konzeption der Laufzeitumgebung . . . 65
5.7.1 Beschreibung der Systemzustände mit Markov-Ketten . . . 65

5.7.2 Design der Laufzeitumgebung als Zustandsautomat 66
5.8 Zusammenfassung 68

6 Implementierung **69**
6.1 Aufsetzen einer Muster-Appliance als Basissystem 70
6.1.1 Spezifische Anpassungen für Router 71
6.1.2 Testen 73
6.1.3 Klonen der Appliance 73
6.2 Aufsetzen zweier Muster-Appliances mit VPN-Kopplung 74
6.2.1 Public-Key-Infrastruktur für die VPNs 75
6.2.2 Testen 76
6.3 Aufsetzen zweier Muster-Appliances mit Routing 77
6.3.1 Testen 78
6.4 Aufsetzen zweier Muster-Appliances als Overlay-Netzwerkknoten 79
6.4.1 Anpassungen für zwei Overlay-Netzwerkknoten-Paare 81
6.4.2 Testen 81
6.5 Integration in eine Plattform 82
6.5.1 Integration in einen ersten Prototyp 82
6.5.2 Entwicklung der Laufzeitumgebung 84
6.5.3 Vervollständigung der Laufzeitumgebung 87
6.5.4 Testen 88
6.6 Zusammenfassung 90

7 Evaluation **93**
7.1 Auswahl der Indikatoren 93
7.1.1 Verfügbarkeit 93
7.1.2 Rekonvergenz 93
7.1.3 Latenz 94
7.1.4 Datendurchsatz 94
7.2 Auswahl der Erfassungs-Tools 94
7.2.1 iperf3 94
7.2.2 vnStat 96
7.2.3 MRTG 96
7.2.4 Wget, Nginx 98
7.3 Erfassung der Messdaten 100
7.4 Auswertung der Messdaten 101
7.5 Ergebnisse der Auswertung 101

8 Zusammenfassung und Ausblick **107**
8.1 Zusammenfassung 107
8.2 Ausblick 108

Literatur **111**

Abbildungsverzeichnis **127**

Tabellenverzeichnis **128**

Listings **128**

Abkürzungsverzeichnis **131**

A Weiterführende Informationen zur Implementierung **137**
A.1 Weiterführende Informationen zur Muster-Appliance als Basissystem . . . 137
A.1.1 Liste der verfügbaren Pakete des Basissystems 137
A.2 Weiterführende Informationen zu spezifischen Anpassungen für Router . 139
A.2.1 Spezifische Betriebssystemkern-Einstellungen für Router 139
A.3 Weiterführende Informationen zur Archivierung relevanter Dateien der Router-Appliance 140
A.3.1 Datei backup-pull-r1-configs-logs-to-backuphost.sh 140
A.4 Weiterführende Informationen zur Public-Key-Infrastruktur für die VPNs 141
A.4.1 Transkript der Zertifizierungsstelle *Router r2 CA* 141
A.5 Weiterführende Informationen zu den Muster-Appliances mit VPN-Kopplung 143
A.5.1 Setup der VPN-Endpunkte mit Code-Einsprungpunkten (*hooks*) für die Laufzeitumgebung 143
A.6 Weiterführende Informationen zu den Muster-Appliances mit dynamischem Routing 145
A.6.1 Muster-Appliances mit dynamischem Routing mit OSPF 145
A.7 Weiterführende Informationen zu den Muster-Appliances als Overlay-Netzwerkknoten 146
A.7.1 Auszug aus den Implementierungsdetails für die Funktion des Overlay-Netzwerkknotens 146
A.7.2 Auszug aus den Implementierungsdetails für die zweite Instanz des Overlay-Netzwerkknotens 147
A.8 Weiterführende Informationen zur Integration in eine Plattform 148
A.8.1 Auszug aus den Implementierungsdetails der Integration eines Overlay-Netzwerkknotens mit einem darauf aufbauenden VPN-Tunnel . . 148
A.8.2 Integration mit den Komponenten für das dynamische Routing mit OSPF 150
A.8.3 Dynamisches Routing mit OSPF mit mehrfacher Redundanz 151
A.9 Weiterführende Informationen und Code zur Laufzeitumgebung der Router-Appliance 156
A.9.1 Datei /usr/local/sbin/router-framework-runtime.sh 156
A.9.2 Datei /usr/local/lib/router-framework/router-framework-init . . 157
A.9.3 Datei /usr/local/lib/router-framework/router-framework-lib . . 158
A.9.4 Datei /usr/local/lib/router-framework/router-framework-lib-util 158
A.9.5 Datei /usr/local/lib/router-framework/router-framework-lib-connectivity 159
A.9.6 Datei /usr/local/lib/router-framework/router-framework-lib-overlay 159
A.9.7 Datei /usr/local/lib/router-framework/router-framework-lib-vpn 161
A.9.8 Datei /usr/local/lib/router-framework/router-framework-lib-router 164
A.9.9 Datei /usr/local/lib/router-framework/router-framework-lib-qos 165

B Weiterführende Informationen zur Evaluation **167**
B.1 Weiterführende Informationen zur Erfassung der Messdaten 167
B.1.1 Auszug aus dem Setup zur Erfassung der Messdaten für die Indikatoren *Verfügbarkeit* und *Latenz* mit der Software *MRTG* 167
B.1.2 Auszug aus den Datenreihen der Messdaten für die Indikatoren *Verfügbarkeit* und *Latenz* 168
B.1.3 Auszug aus dem Setup zur Erfassung der Messdaten für den Indikator *Datendurchsatz* mit der Software *iperf3* 170
B.1.4 Auszug aus den Datenreihen der Messdaten für den Indikator *Datendurchsatz* bei Verfügbarkeit der nativen VPN-Tunnel 171

B.2 Weiterführende Informationen zur Auswertung der Messdaten 172
B.2.1 Auszug aus dem Setup und dem Transkript zur Auswertung der Messdaten für den Indikator *Datendurchsatz* mit der Software *gnuplot* 172

1 Einleitung

Die Kopplung von räumlich entfernten Local Area Networks (LANs) hat mehrere etablierte Anwendungsbereiche.
Erste Anwendungsbereiche der Kopplung von räumlich entfernten Local Area Networks waren die Vernetzung von Zentralen und Niederlassungen von Unternehmen, die Vernetzung von räumlich entfernten Laboren im wissenschaftlichen Bereich und die Vernetzung von Standorten von Rechenzentren zum Zweck der Spiegelung (*mirroring*) von Daten.
Zur Realisierung der Kopplung von räumlich entfernten Local Area Networks wurden anfänglich bis in die 2000er-Jahre hinein oftmals Standard-Festverbindungen[1] und paketvermittelte Telekommunikationsnetze wie z.B. *Datex-P* verwendet.

Im Zeitverlauf hat das Internet an Bedeutung gewonnen und zur Realisierung der Kopplung von räumlich entfernten Local Area Networks werden häufig Virtual Private Networks (VPNs) mit dem Internet als Transportweg verwendet.

Virtual Private Networks (VPNs) mit dem Internet als Transportweg haben gegenüber Standard-Festverbindungen zur Realisierung von LAN-LAN-Kopplungen den Vorteil niedrigerer Betriebskosten.
Standard-Festverbindungen zur Realisierung von LAN-LAN-Kopplungen - insbesondere redundante Standard-Festverbindungen mit getrennter Leitungsführung - haben jedoch hingegen im Vergleich zu Virtual Private Networks (VPNs) den Vorteil einer höheren Verfügbarkeit.

Es erscheint wünschenswert, LAN-LAN-Kopplung mit hoher Verfügbarkeit *und* mit niedrigen Betriebskosten zu realisieren und zu betreiben.
Eine konventionelle, etablierte Möglichkeit, dies zu tun, besteht in der Realisierung von LAN-LAN-Kopplung mit Hilfe von Virtual Private Networks (VPNs) über das Internet als primärem Transportweg und mit Hilfe von Wählleitungen - z.B. in der Form von Primärmultiplexanschlüssen (*PrimuX/S2M*) - als alternativem Transportweg.
Eine innovative, kostengünstigere Möglichkeit, dies zu tun, besteht in der Realisierung von LAN-LAN-Kopplung mit Hilfe von Virtual Private Networks (VPNs) über das Internet als primärem Transportweg und mit Hilfe von Virtual Private Networks über Overlay-Netzwerke - ebenso über das Internet - als alternativem Transportweg.

1.1 Motivation

Virtual Private Networks mit Overlay-Netzen als Transportweg in Verbindung mit dynamischem Routing zum Zweck der Realisierung von LAN-LAN-Kopplung wurden bislang dem Anschein nach nicht in signifikanter Häufigkeit untersucht.

Bei der Recherche nach vergleichbaren verwandten Arbeiten stellte sich folgende Situation dar:

[1] Synonym zum Begriff *Standard-Festverbindung* wird gemeinhin der Begriff *Standleitung* verwendet.

- Keine der in Kapitel 4 untersuchten verwandten Arbeiten stellte eine befriedigende Lösung für mehr als zwei Teilaspekte der Gesamtanforderung dar.
- Es scheint derzeit praktisch keine Literatur über Virtual Private Networks mit Overlay-Netzen als Transportweg zu existieren, welche diesbezügliche Vorgehensweisen oder *Best Practices* dokumentiert.

Es erschien sinnvoll, eine generische und erweiterbare Lösung zu erarbeiten. Die Konzeption und die Entwicklung einer Laufzeitumgebung als Framework unter Zuhilfenahme von weiteren verbreiteten, zu selektierenden und anzupassenden Software-Komponenten sollte dies bewerkstelligen.

1.2 Aufgabenstellung und Zielsetzung

Aufgabenstellung dieser Arbeit war es, eine Routerumgebung zur hochverfügbaren LAN-LAN-Kopplung mit Hilfe von VPNs und Overlay-Netzwerken zu entwerfen und zu implementieren. Diese Routerumgebung soll Overlay-Netzwerke als alternative Transportpfade für die VPNs - welche die LAN-LAN-Kopplung als Kernbestandteil realisieren - nutzen; Zielsetzung war die Erhöhung der Verfügbarkeit der VPNs in Gesamtheit und resultierend die Erhöhung der Verfügbarkeit der LAN-LAN-Kopplung.

1.2.1 LAN-LAN-Kopplung

Die genauere Betrachtung von LAN-LAN-Kopplung erschliesst Möglichkeiten, die Verfügbarkeit bei der LAN-LAN-Kopplung zu erhöhen.
Bei der LAN-LAN-Kopplung werden zur Wahrung der Privatheit und Integrität der übertragenen Daten Virtual Private Networks (VPNs) eingesetzt; die beteiligten Router realisieren über das Internet mindestens einen gemeinsamen verbindenden VPN-Tunnel. Dieser VPN-Tunnel wird bzw. diese VPN-Tunnel werden zum IP-Pakettransport zwischen den zu koppelnden LANs verwendet. Die beteiligten Router treffen Routing-Entscheidungen bezüglich der zu transportierenden IP-Pakete; als Grundlage für diese Routing-Entscheidungen existieren auf den Routern Routing-Tabellen, durch welche die Wege der zu transportierenden IP-Pakete festgelegt werden. Der eigentliche Transport von IP-Paketen *nach einer getroffenden* Routing-Entscheidung wird in Abgrenzung zum Routing *an sich* mit dem Begriff Paketforwarding (*packet forwarding*) bezeichnet.

Eine Möglichkeit der Erhöhung der Verfügbarkeit der LAN-LAN-Kopplung ist die Realisierung mehrerer VPN-Tunnel als genutzte Transportpfade für die IP-Pakete zwischen den LANs in Verbindung mit dynamischem Routing zur Nutzung dieser Redundanz für die Erhöhung der Verfügbarkeit.

1.2.2 Herausforderungen bezüglich der Verfügbarkeit

Eine Herausforderung bezüglich der Verfügbarkeit bei der LAN-LAN-Kopplung ist das Sicherstellen von *wirklicher* Redundanz bei den primär genutzten und bei den alternativ genutzten Transportpfaden der zu Grunde liegenden Virtual Private Networks (VPNs).
Zwei parallele VPN-Tunnel mit direktem Transportweg über das Internet erfüllen das Kriterium *wirklicher* Redundanz nicht oder in nur rudimentärer Ausprägung - die Möglichkeit der zeitgleichen Ausfalls beider VPN-Tunnel ist inakzeptabel hoch.
Bei Störungen im Internet auf der Backbone-Ebene - z.B. bei Naturkatastrophen oder bei technisch bedingten Ausfällen - passen die Router im Backbone des Internets das Routing

an, um alternative Pfade zu realisieren; üblichenfalls ist die Zeitspanne für diese Anpassungen des Routings im Zeitbereich von 90 Sekunden[2].
Das Vorliegen von geopolitischen Gründen kann spontan zur Nichtverfügbarkeit des Internets als direktem unmittelbarem Transportpfad zwischen zwei Routern für VPNs zur LAN-LAN-Kopplung führen; ein Beispiel für einen solchen Kontext ist die *Great Chinese Firewall* der Volksrepublik China.

1.3 Aufbau der Arbeit

Die Arbeit beschreibt die Hintergründe der Konzeption, die Konzeption und wichtige Aspekte der Implementierung einer Routerumgebung zur redundanten und hochverfügbaren Kopplung von LANs mit Hilfe von Overlay-Netzwerken und Virtual Private Networks (VPNs).

In Kapitel 2 Grundlagen wird das Routing im Internet näher betrachtet, es werden gebräuchliche Routing-Protokolle dargestellt, es werden gebräuchliche Protokolle für Virtual Private Networks dargestellt und es werden Overlay-Netzwerke im Internet dargestellt.
Zu den gebräuchlichen Routing-Protokollen werden diese Routing-Protokolle implementierende, im Internet verwendete *Routing Software Suites* und Routing-Appliances kommerzieller Anbieter vorgestellt. Zu den gebräuchlichen Protokollen für Virtual Private Networks werden diese Protokolle implementierende Softwares vorgestellt. Bezüglich Overlay-Netzwerken werden Arten von Overlay-Netzwerken dargestellt und es werden verbreitete Overlay-Netzwerke und deren Software-Implementierungen vorgestellt.
Vorgreifend auf Kapitel 3 Analyse und Kapitel 5 Konzeption werden in Kapitel 2 bezüglich der Softwares deren wichtigsten relevanten Eigenschaften dargestellt.

Kapitel 3 stellt die Analyse des Themenbereichs dar; es werden Anforderungen an die LAN-LAN-Kopplung mit Hilfe von Virtual Private Networks (VPNs) betrachtet, es werden eingesetzte VPN-Protokolle zur LAN-LAN-Kopplung dargestellt und es werden geeignete dynamische Routing-Protokolle zur LAN-LAN-Kopplung mit Hilfe von VPNs betrachtet. In der Analyse werden untersuchenswerte Arten von Overlay-Netzwerken dargestellt, die zur Erhöhung der Verfügbarkeit bei der LAN-LAN-Kopplung mit Hilfe von VPNs beitragen können - im Kontext des Themenbereichs untersuchenswerte Overlay-Netzwerke sind solche, die alternative Transportpfade für VPNs zur LAN-LAN-Kopplung bereitstellen können.

Eine Auswahl verwandter Arbeiten wird in Kapitel 4 vorgestellt. Diese boten und bieten wertvolle Anregungen und sollen auch als Informationsgrundlage für Weiterentwicklungen dienen; jedoch ist keine dieser Arbeiten direkt mit der hier entwickelten Routerumgebung zur LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks vergleichbar.

Kapitel 5 beschreibt die Konzeption der Routerumgebung, die beim Entwurf gestellten Fragen, Entscheidungen und Entscheidungsketten - mit Schwerpunkt auf der Auswahl der genutzen Software-Komponenten und Protokolle - und mit Schwerpunkt auf dem Entwurf der die Komponenten integrierenden steuernden Laufzeitumgebung; das dem

[2] Prof. Dr. Norbert Pohlmann und Stefan Dierichs: So funktioniert Internet-Routing: Wie Routing dem Netz seine Selbstheilungskräfte verleiht, 2008 [2, S. 4]

Entwurf der Laufzeitumgebung zu Grunde liegende Modell in der Form eines Zustandsautomaten wird dargestellt.

Die Implementierung wird in Kapitel 6 beschrieben; hier wird die Entwicklungsumgebung und die Produktivumgebung beschrieben und die schrittweise Integration der Komponenten und die damit einhergehende Entwicklung der Laufzeitumgebung. Implementierungsdetails, Programmcode und Details zu den Software-Komponenten sind in Anhang A ausgegliedert.

Die Evaluation der implementierten Lösung in Kapitel 7 gliedert sich in die Darstellung geeigneter Indikatoren - qualitativer und quantitiver Eigenschaften - zur Evaluation der Ziele und in die Auswahl geeigneter Messwerkzeuge und deren Anwendung. Weiterführende Informationen zur Evaluation - Details zur Datenerfassung, Auszüge aus den Datenreihen und Details zur Auswertung der Messdaten - befinden sich in Anhang B.

Zuletzt gibt Kapitel 8 eine Zusammenfassung und einen Ausblick auf mögliche Optimierungen und auf erwägenswerte Erweiterungen.

1.4 Projektplanung

Initial wurde eine Mind-Map zum Thema mittels der Software *FreeMind*[3,4] erstellt; aus der Mind-Map heraus wurden die Gliederung dieses Dokuments und Vorgänge der Projektplanung herausgearbeitet[5].

Die Projektplanung vollzog sich mit der Software *GanttProject*[6,7].

Zeitlich einhergehend mit der Bearbeitung des Themas wurden regelmässig in zeitlichen Abständen von je vier Wochen Status-Reports erstellt.

[3] FreeMind Project: FreeMind - free mind-mapping software, URL: `http://freemind.sourceforge.net/` [3]

[4] Die Software *FreeMind* ist eine freie Software zur Erstellung von Mind-Maps; *FreeMind* benötigt die Laufzeitumgebung von Java (*JRE - Java Runtime Environment*).[3]

[5] Der Informationsexport aus der Mind-Map-Software heraus wurde mit Hilfe angepasster *XSLT*-Vorlagen im *XML*-Format bewerkstelligt; die Software *FreeMind* stellt Funktionalität bereit, um die Informationen der Mind-Maps in strukturierter Form in vielen Formaten zu exportieren.[3]

[6] GanttProject Team: GanttProject, URL: `http://www.ganttproject.biz/` [4]

[7] Die Software *GanttProject* ist eine freie Software zur Projektplanung und zum Projektmanagement; *GanttProject* benötigt die Laufzeitumgebung von Java (*JRE - Java Runtime Environment*).[6]

2 Grundlagen

Das Internet besteht aus vielen administrativ voneinander unabhängigen Netzen, die miteinander vernetzt sind.

Die Organisationseinheit für diese administrativ voneinander unabhängigen Netze, die direkt oder indirekt miteinander vernetzt sind und die als Gesamtheit das Internet bilden, ist ein *Autonomes System* (*AS - Autonomous System*).

Ein *Autonomes System* (*AS*) bezeichnet die Gesamtheit aller Router unter einer Administration; *Autonomes System* bezeichnet auch die Entität dieses Routing-Bereichs (*routing domain*).

Die wichtigsten Eigenschaften eines Autonomen Systems sind seine Nummer (*ASN - autonomous system number*) und die Adressbereiche der IP-Netze, aus denen das Autonome System besteht; des Weiteren ist einem Autonomen System ein symbolischer Name (*AS name*) zugeordnet.

Die Koordination der Nummernkreise der Nummern und der Nummern der Autonomen Systeme (*ASN*) ist Aufgabe der *Internet Assigned Numbers Authority* (*IANA*); die *IANA* hat die Funktion der Vergabe der Nummern der Autonomen Systeme an Regionale Internet Registries (*RIR - Regional Internet Registry*) delegiert.[1]

Es existieren fünf Regionale Internet Registries weltweit, die Zuständigkeit der Regionalen Internet Registries ist nach geographischer Region festgelegt, Tabelle 2.1 gibt eine Übersicht der Regionalen Internet Registries.

Tabelle 2.1: Regionale Internet Registries weltweit mit ihren Zuständigkeiten nach Region

RIR	**Geographische Region**
AFRINIC	Region Afrika
APNIC	Region Asien und Pazifik
ARIN	Kanada, USA und einige Karibische Inseln
LACNIC	Lateinamerika und einige Karibische Inseln
RIPE NCC	Europa, Mittlerer Osten und Zentral-Asien

Für die Vergabe eines Autonomen Systems mit einer entsprechenden *AS*-Nummer (*ASN*) in Deutschland ergibt sich die Zuständigkeit des RIPE NCC; das RIPE NCC ist das Network Coordination Centre des Reseaux IP Européens.

2.1 Routing im Internet

Mit Hilfe der Nummern der Autonomen Systeme (*ASN*) wird das Routing des IP-Datenverkehrs unter den Autonomen Systemen untereinander und transitierend[2] von einem Autonomen System über dritte Autonome Systeme hinweg zu einem anderen Autonomen System bewerkstelligt.

[1] IANA: Autonomous System (AS) Numbers, 2016 [5]

[2] transitieren: durchlaufen, passieren

Internet Service Provider sind Betreiber von Autonomen Systemen, ihre Geschäftsgrundlage ist der Transport und die Durchleitung von IP-Datenverkehren; sie treffen diesbezüglich Vereinbarungen und schliessen Verträge. Grosse Unternehmen wie z.B. Google oder Internet-Austauschpunkte wie z.B. der deutsche DE-CIX oder grosse Webhosting-Unternehmen sind ebenfalls Betreiber von Autonomen Systemen.

Es gibt unter Internet Service Providern mehrere Arten von Vereinbarungen bzw. Verträgen mit Bezug auf den Transport oder die Durchleitung von IP-Datenverkehren.

Eine Möglichkeit der Vereinbarung besteht darin, dass ein kleinerer Internet Service Provider (ISP) bei einem grösseren Internet Service Provider für die Durchleitung des eigenen notwendigen IP-Datenverkehrs zahlt; der grössere Internet Service Provider ist in diesem Fall der *Uplink* des kleineren Internet Service Provider, der kleinere Internet Service Provider zahlt für den *Transit*.

Eine andere Möglichkeit der Vereinbarung besteht darin, dass sich zwei Internet Service Provider darauf einigen, ihren *jeweils eigenen* IP-Datenverkehr auf Gegenseitigkeit kostenfrei mit dem anderen Internet Service Provider auszutauschen; in diesem Fall betreiben die beiden Internet Service Provider *privates Peering*.

Eine dritte Möglichkeit der Vereinbarung besteht darin, dass sich zwei grosse Internet Service Provider darauf einigen, *jeglichen* jeweilig anfallenden IP-Datenverkehr auf Gegenseitigkeit auszutauschen und *durchzuleiten*.

Internet Service Provider werden nach der jeweils im Vordergrund stehenden Art der Vertragsbeziehung zu anderen Internet Service Providern in mehrere Klassen unterteilt:

Internet Service Provider, die keinen *Transit* an andere Internet-Service-Provider verkaufen und ausschliesslich IP-Datenverkehr-Durchleitung an Endkunden verkaufen, welche keine Internet Service Provider sind, fallen in die Klasse der *Tier 3*-Internet Service Provider.

Grosse Internet Service Provider, die an keinen anderen Internet Service Provider für die IP-Datenverkehr-Durchleitung zahlen, bilden die Klasse der *Tier 1*-Internet Service Provider; dies sind weltweit insgesamt etwa 12 sehr grosse Internet Service Provider.

Alle Internet Service Provider, die nicht der Klasse der *Tier 1*-Internet Service Provider oder der Klasse der *Tier 3*-Internet Service Provider angehören, bilden die Klasse der *Tier 2*-Internet Service Provider.[3]

2.1.1 Struktur des Backbones des Internets

Die etwa 12 sehr grossen Internet Service Provider aus der Klasse der *Tier 1*-Internet Service Provider bilden mit der Infrastruktur ihrer Autonomen Systeme das *Backbone* des Internets.

Eine Einordnung in die Klasse der *Tier 1*-Internet Service Provider lässt sich nur approximativ treffen, da ein Kriterium für die Einordnung - ob ein Internet Service Provider (*ISP*) an einen anderen Internet Service Provider für die IP-Datenverkehr-Durchleitung zahlt - eine nicht-öffentliche geschäftliche Information der Unternehmen darstellt.

Die Autonomen Systeme (AS) der Internet Service Provider bilden eine Topologie.
Zur Untersuchung dieser Topologie gibt es drei wichtige Kriterien:

- die Anzahl der Autonomen Systeme, die topologisch unterhalb eines Autonomen Systems liegen
- die Anzahl der IP-Adressen, die topologisch unterhalb eines Autonomen Systems liegen

[3] Prof. Dr. Norbert Pohlmann und Stefan Dierichs: So funktioniert Internet-Routing: Wie Routing dem Netz seine Selbstheilungskräfte verleiht, 2008 [2, S. 2]

- den *AS transit degree* - der Wert einer Funktion, die angibt, in welchem Grad ein Autonomes System *Transit* für andere Autonome Systeme ist

Tabelle 2.2 gibt eine Übersicht über 20 wichtige Autonome Systeme im Internet; Grundlage bildet ein Ranking über die Autonomen Systeme vom Center for Applied Internet Data Analysis (CAIDA) mit aggregierten Datensätzen bis ins Jahr 2016.[4]

Tabelle 2.2: 20 wichtige Autonome Systeme mit Topologie-Kriterien

	ASN	Unternehmen	ASe[5]	IPv4-Adressen	AS transit degree
1	3356	Level 3 Communications	29 494	783 401 728	4 138
2	174	Cogent Communications	23 299	616 423 936	4 567
3	1299	Telia Company AB	21 954	667 346 176	1 272
4	2914	NTT America	18 991	642 432 768	1 352
5	3257	GTT (vormals: Tinet)	18 140	565 089 024	1 282
6	6762	Telecom Italia Sparkle	14 394	329 530 624	534
7	6453	Tata Communications	12 300	533 133 824	685
8	6939	Hurricane Electric	8 088	278 942 720	4 809
9	2828	Verizon Enterprise (vormals: XO)	6 251	250 568 448	1 089
10	1273	Vodafone Group	5 878	173 223 936	296
11	3549	Level 3 Communications	4 735	99 505 920	3 409
12	701	Verizon Enterprise (vorm.[6]: MCI)	4 527	414 235 392	1 301
13	6461	Zayo Bandwidth	4 479	120 272 640	1 488
14	4436	GTT (vormals: nLayer)	4 145	135 533 568	838
15	209	CenturyLink (vormals: Qwest)	3 988	278 455 808	1 651
16	9002	RETN Limited	3 820	23 978 496	1 161
17	3491	PCCW Global (vormals: BTN)	3 572	187 765 504	547
18	20485	TransTeleCom	3 447	25 126 400	1 883
19	1239	Sprint	3 439	330 382 848	582
20	3320	Deutsche Telekom AG	3 378	203 363 072	508

Einen Überblick über grössere Autonome Systeme - einen Teil des Internet-Backbones - gibt Abbildung 2.1.

Tier 1-Internet Service Provider - die das Internet-Backbone bilden - und *Tier 2*- und *Tier 3*-Internet Service Provider haben teilweise unterschiedliche Interessen; *Tier 1*-Internet Service Provider können oder wollen mitunter den Anforderungen an den Durchsatz (*throughput*) bei der IP-Datenverkehr-Durchleitung nicht nachkommen.
Eine solche Reduzierung des Durchsatzes bei der IP-Datenverkehr-Durchleitung beim *Tier 1*-Internet Service Provider Cogent Communications, die Kunden der Internet Service Provider Comcast, Time Warner Cable und Verizon in der Region New York City und den Zeitraum April/Juni 2013 bis Februar 2014 betreffend, wurde im Bericht "ISP Interconnection and its Impact on Consumer Internet Performance" des Measurement Lab Consortiums aus dem Jahr 2014 dokumentiert.[7]

[4] CAIDA (Center for Applied Internet Data Analysis): AS Rank: AS Ranking, 2016 [6]
[5] *ASes* ist englischer Sprachgebrauch für den Plural von *Autonomous System*, *ASe* ist deutscher Sprachgebrauch für den Plural von *Autonomes System*.
[6] *vorm.:* vormals
[7] Measurement Lab Consortium: ISP Interconnection and its Impact on Consumer Internet Performance, 2014 [7, S. 9]

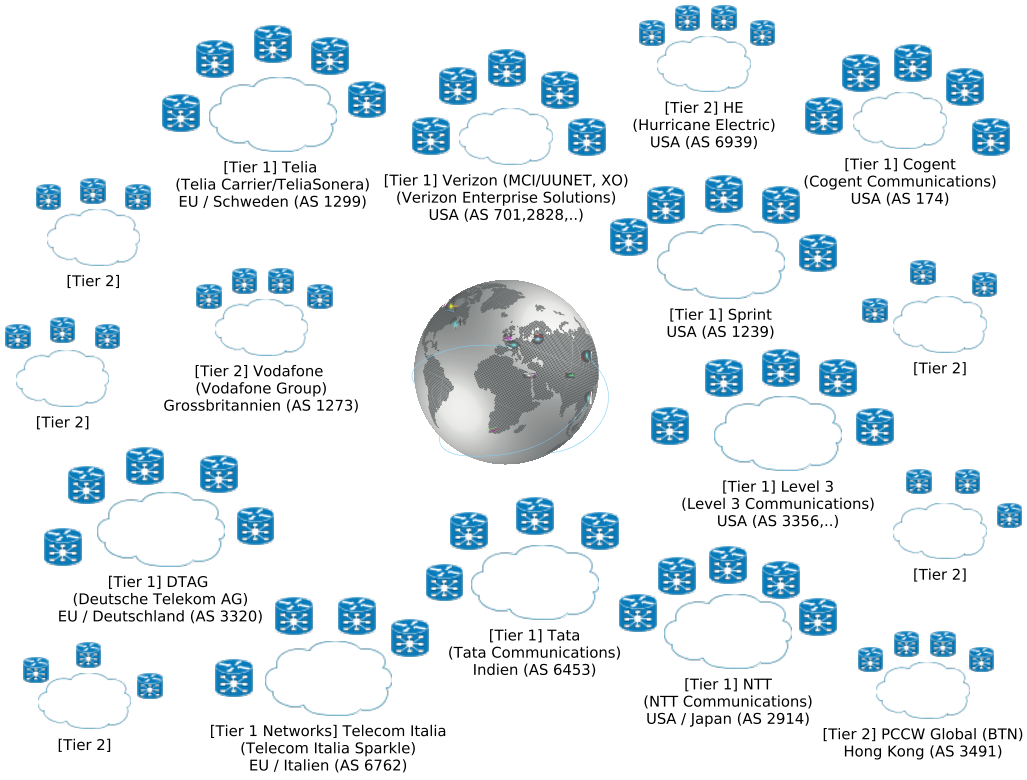

Abbildung 2.1: Grössere Autonome Systeme im Internet - ein Teil des Internet-Backbones

2.1.2 Verwendete Routing-Protokolle

Routing-Protokolle auf der topologischen Ebene von Autonomen Systemen und auf der topologischen Ebene zwischen Autonomen Systemen sind dynamische Routing-Protokolle. Bei dynamischen Routing-Protokollen[8] wird der IP-Datenverkehr je nach äusseren Bedingungen über unterschiedliche Wege zum Ziel transportiert; die Routing-Wege ändern sich also dynamisch.

Je nach verwendeten Algorithmen für das dynamische Routing lassen sich dynamische Routing-Protokolle in verschiedene Klassen einteilen:

- Distanzvektor-Routing-Protokolle[9] (*Distance Vector Routing Protocols*)
- Link-State-Routing-Protokolle[9] (*Link State Routing Protocols*)
- Pfadvektor-Routing-Protokoll (*Path Vector Routing Protocol*)

Distanzvektor-Routing-Protokolle[9] (*Distance Vector Routing Protocols*) nutzen zwei Merkmale, um Routing-Entscheidungen für die zu transportierenden Datenpakete zu treffen: Entfernung des Zielnetzwerks (*Distance*) und Richtung (*Vector*) des Zielnetzwerks. Die Entfernung des Zielnetzwerks entspricht einer Metrik, z.B. die Anzahl der zwischen Quelle und Ziel liegenden Router (*hop count*), die Kosten des Leitungswegs, die Bandbreite des Leitungswegs oder die durch die Eigenschaften des Leitungswegs bedingte Verzögerung (*delay*). Die Richtung ist spezifiziert durch die Wahl des benachbarten Routers oder durch

[8] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Routing Protokoll, dynamisches -, statisches -" [8, S. 715]

[9] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Routing Protokoll, Distanzvektor -, Link State -" [8, S. 715]

die Wahl einer Netzwerkschnittstelle. Bei Distanzvektor-Routing-Protokollen haben die einzelnen Router keine Information über die Struktur des gesamten Netzwerkes.

Link-State-Routing-Protokolle[9] (*Link State Routing Protocols*) nutzen hingegen als Information, um Routing-Entscheidungen für die zu transportierenden Datenpakete zu treffen, die Information über die aktuelle Struktur des gesamten Netzwerks. Zu diesem Zweck tauschen alle Router direkt oder indirekt untereinander Informationen über den Verbindungzustand (*Link State*) zwischen den Routern aus; dies geschieht bei Veränderungen des Verbindungszustands zweier beteiligter Router. Jeder Router ist durch die Informationen über alle Verbindungszustände aller Router in der Lage, die Topologie des gesamten Netzwerks zu berechnen; somit ist die Information über den gesamten Weg von Quelle zum Ziel als Grundlage für die Routing-Entscheidung für den Transport der Datenpakete vorliegend.

Das Pfadvektor-Routing-Protokoll (*Path Vector Routing Protocol*) hat Ähnlichkeiten mit den Distanzvektor-Routing-Protokollen. Das *Border Gateway Protocol* (*BGP*), welches für das Routing zwischen den *Autonomen Systemen* und im Internet-Backbone (siehe Abschnitt 2.1.1 Struktur des Backbones des Internets) eingesetzt wird, ist ein Pfadvektor-Routing-Protokoll. Das Pfadvektor-Routing-Protokoll nutzt zwei Merkmale, um Routing-Entscheidungen für die zu transportierenden Datenpakete zu treffen: eine abstrakte Vorgabe für das Routing (*policy*) und einem Pfad.
Die abstrakte Vorgabe für das Routing (*policy*) kann betriebswirtschaftliche Gründe haben, sie kann die Sicherheit betreffende oder geopolitische Gründe haben. Ein Beispiel für betriebswirtschaftliche Gründe für eine abstrakte Routing-Vorgabe (*routing policy*) ist der Umstand, ob *privates Peering* (siehe Abschnitt 2.1) oder eine Durchleitung von Datenverkehr (siehe Abschnitt 2.1) zwischen den beteiligten Router-Betreibern vereinbart ist; ein Beispiel für geopolitische Gründe für eine abstrakte Routing-Vorgabe ist die Vorgabe, dass Datenverkehr zwischen einem Router in der Europäischen Union und einem Router in Süd-Korea nicht über einen Router in Nord-Korea abgewickelt werden soll.
Das zweite Merkmal des Pfadvektor-Routing-Protokolls, um Routing-Entscheidungen für die zu transportierenden Datenpakete zu treffen - der Pfad - besteht aus einem Router (*next hop router*), einem Pfad von Autonomen Systemen (*AS path*) und einem Präfix (*Prefix*); der besserem Lesbarkeit wegen wird der Pfad in umgekehrter Reihenfolge[10] notiert: *Prefix*, *AS path*, *next hop router*. Der Pfad gibt Informationen über einen Teil des Weges, den Datenverkehr zu einem Ziel nehmen kann. Der mögliche gesamte Pfad von Quelle zum Ziel ergibt sich aus diesen Teil-Pfaden. Die Router geben untereinander Informationen über Routen durch die Übertragung dieser Pfade bekannt (*route advertisements*). Jeder beteiligte Router nutzt die Information über die Topologie des gesamten Netzes, die durch den Austausch der Informationen über die Routen bekannt ist, um zu entscheiden, ob umittelbar für die Routing-Entscheidung zur Auswahl stehende Teil-Pfade für den Transport der Datenpakete nutzbar sind.

Routing-Protokolle im Internet lassen sich unterscheiden bezüglich ihres Einsatzzwecks: Ein Routing-Protokoll, das *zwischen* Autonomen Systemen verwendet wird (*interdomain routing*), heisst Exterior Gateway Protocol (*EGP*); ein Routing-Protokoll, das *innerhalb eines* Autonomen Systems verwendet wird (*intradomain routing*), heisst Interior Gateway Protocol (*IGP*).

Es gibt nur ein Exterior Gateway Protocol (*EGP*): das *Border Gateway Protocol* (*BGP*).

[10] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 481]

Das *Border Gateway Protocol* (*BGP*) wird im Internet eingesetzt zum Routing zwischen Autonomen Systemen; neben *BGP* wird kein anderes Routing-Protokoll zu diesem Zweck verwendet.

Zur Verwendung als Interior Gateway Protocol (*IGP*) zum Internet-Routing innerhalb eines Autonomen Systems existieren folgende Routing-Protokolle:

- *OSPF* (*Open Shortest Path First*)
- *IS-IS* (*Intermediate-System to Intermediate-System*)
- *IGRP* (*Interior Gateway Routing Protocol*) und *EIGRP* (*Enhanced Interior Gateway Routing Protocol*)
- *RIP* (*Routing Information Protocol*)

Als Interior Gateway Protocol (*IGP*) innerhalb eines Autonomen Systems werden gewöhnlich das Routing-Protokoll *OSPF* (*Open Shortest Path First*) sowie das Routing-Protokoll *IS-IS* (*Intermediate-System to Intermediate-System*) eingesetzt.[11]
Das Routing-Protokoll *OSPF* wird als Interior Gateway Protocol bevorzugt in Unternehmens-Netzwerken eingesetzt, das Routing-Protokoll *IS-IS* wird als Interior Gateway Protocol bevorzugt in Netzwerken der Internet Service Provider eingesetzt.[12]
Bei Organisationen mit grossen Netzwerken wird innerhalb eines Autonomen Systems als Interior Gateway Protocol (*IGP*) eine Variante von *BGP* eingesetzt, da *BGP* eine bessere Skalierbarkeit als das Routing-Protokoll *OSPF* in grossen Netzwerken zeigt; diese Variante wird *iBGP* (*internal BGP*) genannt.[13] Das Border Gateway Protocol (*BGP*) als Exterior Gateway Protocol (*EGP*) wird in diesen Einsatzbereichen der besseren Unterscheidbarkeit wegen *eBGP* (*external BGP*) genannt.[13]

2.1.2.1 Exterior Gateway Protocol (EGP): BGP

Das Border Gateway Protocol (BGP) wird als Exterior Gateway Protocol (EGP) - also als äusseres Routing-Protokoll - als einziges Protokoll zum Internet-Routing zwischen den Autonomen Systemen verwendet. Das Border Gateway Protocol (BGP) ist ein Pfadvektor-Routing-Protokoll (*Path Vector Routing Protocol*) (siehe Abschnitt 2.1.2). BGP zeigt gute Skalierbarkeit und schnelles Konvergenz-Verhalten in grossen Netzwerken - z.B. dem Internet -, jedoch ist die Komplexität bei der Implementierung und bei der Wartung beim Border Gateway Protocol (BGP) hoch.
Die aktuelle Version des Border Gateway Protocol ist BGP-4, diese Protokollversion ist in RFC[14,15] 4271 spezifiziert.[16] Das Border Gateway Protocol (BGP) wird eingesetzt, um das Routing des Datenverkehrs von weltweit mehr als 50000 Autonomen Systemen[4] im Internet zu bewerkstelligen. Jeder äussere Router eines Autonomen Systems (*border rou-*

[11] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 479]
[12] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 474]
[13] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 483]
[14] *RFC* ist ein Akronym für "*Request For Comments*" - ein Vorgang der Stellungnahme im Rahmen des Standardisierungs-Prozesses der *IETF* (Internet Engineering Task Force) im Bereich der Internet-Protokolle und Internet-Standards.
[15] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "RFC" [8, S. 705]
[16] RFC 4271: A Border Gateway Protocol 4 (BGP-4), 2006, Draft Standard, URL: `http://www.rfc-editor.org/rfc/rfc4271.txt` [10]

ter[17,18]) nutzt in seinen Routing-Tabellen mehr als 650000 Pfade[19] (siehe Abschnitt 2.1.2) als Grundlage für das Routing.
Die äusseren Router eines Autonomen Systems senden den benachbarten Routern in der Topologie regelmaessig etwa alle 60 Sekunden im Rahmen der Protokollspezifikation Nachrichten;[20] diese Nachrichten dienen als Grundlage für die Entscheidung, ob die benachbarten Router erreichbar sind. Bei Nichterreichbarkeit eines Routers passen dessen Nachbarn ihre Routing-Tabellen an, um alternative Pfade für das Routing wählen zu können. Diese Anpassung an veränderte äussere Bedingungen benötigt etwa 90 Sekunden; nach dieser Zeitspanne sind Router wieder in der Lage, IP-Datenpakete zum Ziel über einen anderen Pfad zu transportieren - dieser Vorgang wird *Rekonvergenz* genannt.
Beim Border Gateway Protocol (BGP) ist in der ursprünglichen Protokollbeschreibung unmittelbar nur das Routing von IPv4-Datenpaketen berücksichtigt; das Border Gateway Protocol wurde in RFC 4760 dahingehend erweitert, dass auch Routing von IPv6-Datenverkehren oder Multicast-Datenverkehren mit Routern, die das Border Gateway Protocol mit dieser Erweiterung nutzen, möglich ist; diese Erweiterung des Routing-Protokolls wurde abwärtskompatibel gestaltet, um Interoperabilität zu gewährleisten.[21]
Das Border Gateway Protocol wurde ursprünglich nicht mit dem Schwerpunkt auf Sicherheit konzipiert, auch die aktuelle Protokollversion BGP-4 hat mehrere Schwachstellen in Bezug auf Sicherheit; RFC 4272 beschreibt Schwachstellen des Protokolls und in Betracht zu ziehende Angriffsvektoren.[22]
Diese Angriffsvektoren werden aktiv genutzt, um Datenverkehre umzuleiten oder um in Regionen den Internetzugang zu verwehren oder zu beschränken; solche Regionen sind z.B. Ägypten, Libyen, Pakistan, Thailand.

Historisch

Das *Border Gateway Protocol* (*BGP*) als Exterior Gateway Protocol (EGP) wurde mit RFC 1105 im Jahr 1989 spezifiziert.[23] Es wurden mit RFC 1163 im Jahr 1990 kleinere Veränderungen am Protokoll spezifiziert[24] und damit einhergehend mit RFC 1164 im Jahr 1990 die Anwendung des Border Gateway Protokolls und die Umsetzung der Protokollspezifikationen beschrieben[25].
Mit RFC 1267 und RFC 1268 im Jahr 1991 wurde das Border Gateway Protocol - mit der Versionsbezeichnung *BGP-3* - als Standard deklariert.[26,27]

[17] Für den Begriff "*border router*" ist der deutsche Sprachgebrauch "*Perimeter-Router*" üblich [8, S. 715].
[18] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Router, Perimeter- oder Border -" [8, S. 715]
[19] BGP Reports: BGP Routing Table Analysis Reports - Growth of the BGP Table - 1994 to Present [11]
[20] Das Zeitintervall ist von der Konfiguration der Router abhängig, spezifisch von den optionalen Session-Attributen (*Optional Session Attributes*) *IdleHoldTime* und *IdleHoldTimer* [10, S. 37].
[21] RFC 4760: Multiprotocol Extensions for BGP-4, 2007, Draft Standard,
URL: `http://www.rfc-editor.org/rfc/rfc4760.txt` [12]
[22] RFC 4272: BGP Security Vulnerabilities Analysis, 2006, Informational,
URL: `http://www.rfc-editor.org/rfc/rfc4272.txt` [13]
[23] RFC 1105: A Border Gateway Protocol (BGP), 1989, Experimental,
URL: `http://www.rfc-editor.org/rfc/rfc1105.txt` [14]
[24] RFC 1163: A Border Gateway Protocol (BGP), 1990, Historic,
URL: `http://www.rfc-editor.org/rfc/rfc1163.txt` [15]
[25] RFC 1164: Application of the Border Gateway Protocol in the Internet, 1990, Historic,
URL: `http://www.rfc-editor.org/rfc/rfc1164.txt` [16]
[26] RFC 1267: A Border Gateway Protocol 3 (BGP-3), 1991, Historic,
URL: `http://www.rfc-editor.org/rfc/rfc1267.txt` [17]
[27] RFC 1268: Application of the Border Gateway Protocol in the Internet, 1991, Historic,
URL: `https://www.rfc-editor.org/rfc/rfc1268.txt` [18]

Mit RFC 1654 und RFC 1655 im Jahr 1994 wurde erstmals das Border Gateway Protokoll mit der Versionsbezeichnung *BGP-4* spezifiziert.[28,29]
Im Jahr 1994 nutzten die Router, die das Border Gateway Protocol einsetzten, weniger als 25000 Pfade in ihren Routing-Tabellen als Grundlage für das Routing.
Um die Auswirkungen eines Teils der bestehenden Schwachstellen beim Border Gateway Protocol zu reduzieren, wurde mit RFC 2385 im Jahr 1998 eine Option, um TCP-Datenpakete kryptographisch gegen unbemerkte Veränderungen zu schützen[30], eingeführt. Diese Massnahme sollte die Gefahr von Angriffen auf die TCP-Verbindungen minimieren, über die mit dem Border Gateway Protocol arbeitende Router ihre Routing-Informationen austauschen. Mit RFC 5925 und RFC 5926 wurde im Jahr 2010 eine verbesserte Option einführt, um TCP-Datenpakete kryptographisch gegen unbemerkte Veränderungen zu schützen,[31,32] da der in RFC 2385 im Jahr 1998 spezifizierte Algorithmus *MD5* (*Message Digest 5*) zum kryptographischen Signieren der TCP-Datenpakete[30] nicht mehr zuverlässig war.[33] RFC 5926 spezifiziert als verwendete Algorithmen zum kryptographischen Signieren in nicht abschliessender Aufzählung die Algorithmen *SHA-1* (*Secure Hash Algorithm*) und *AES-128* (*Advanced Encryption Standard*) als zu Grunde liegende Algorithmen.

2.1.2.2 Interior Gateway Protocol (IGP): OSPF

Als Interior Gateway Protocol (IGP) - also als Routing-Protokoll innerhalb eines Autonomen Systems - ist in Unternehmens-Netzwerken das Routing-Protokoll OSPF (*Open Shortest Path First*) gebräuchlich. Das Routing-Protokoll OSPF ist ein Link-State-Routing-Protokoll (*Link State Routing Protocol*) (siehe Abschnitt 2.1.2).
Das Routing-Protokoll OSPF zeigt gute Skalierbarkeit und schnelles Konvergenz-Verhalten in mittleren und grossen Netzwerken, jedoch ist die Komplexität bei der Implementierung und bei der Wartung bei OSPF hoch.
Die aktuelle Version des Routing-Protokolls OSPF für das Routing von IPv4-Datenverkehren ist OSPFv2, diese Version ist in RFC 2328 spezifiziert;[34] die aktuelle Version von OSPF für das Routing von IPv6-Datenverkehren ist OSPFv3, diese Version ist in RFC 5340 spezifiziert.[35]

Beim Routing-Protokoll OSPF haben alle beteiligten Router die Information über die aktuelle Topologie des gesamten Netzwerks als Grundlage für das Routing. Jeder Router nutzt eine eigene Datenbank mit Informationen über die aktuelle Topologie des gesamten Netzwerkes. Jeder Router berechnet zu jedem anderen Router den kürzesten Pfad (*shortest path)*; dies geschieht durch Berechnung eines Baumes der kürzesten Pfade. Bei jeder Änderung der Topologie des Netzwerkes wird der Baum der kürzesten Pfade als Grundlage

[28] RFC 1654: A Border Gateway Protocol 4 (BGP-4), 1994, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc1654.txt` [19]
[29] RFC 1655: Application of the Border Gateway Protocol in the Internet, 1994, Proposed Standard, URL: `https://www.rfc-editor.org/rfc/rfc1655.txt` [20]
[30] RFC 2385: Protection of BGP Sessions via the TCP MD5 Signature Option, 1998, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc2385.txt` [21]
[31] RFC 5925: The TCP Authentication Option, 2010, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc5925.txt` [22]
[32] RFC 5926: Cryptographic Algorithms for the TCP Authentication Option (TCP-AO), 2010, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc5926.txt` [23]
[33] Niels Ferguson, Bruce Schneier und Tadayoshi Kohno: Cryptography engineering: Design principles and practical applications, 2010 [24, S. 81]
[34] RFC 2328: OSPF Version 2, 1998, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc2328.txt` [25]
[35] RFC 5340: OSPF for IPv6, 2008, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc5340.txt` [26]

für die Routing-Tabelle - also für das Routing - neu berechnet. Die Information über die aktuelle Topologie des gesamte Netzwerkes wird verteilt, indem alle beteiligten Router direkt oder indirekt Informationen über die Verbindungszustände (*link state*s) zwischen den Routern austauschen. Die Verbindungszustände zwischen den Routern werden erkannt, indem sich die Router untereinander im Rahmen der Protokollspezifikation regelmässig ein Hello-Paket (*Hello Packet*) senden. Jeder Router schreibt die Informationen über die Verbindungszustände in einer lokalen Datenbank (*link state database*) fort. Die in dieser Datenbank vorliegenden Informationen verbreitet jeder Router an andere Router in Form von *Link State Advertisements* (LSAs).

Das Routing-Protokoll OSPF hat die hervorzuhebende Eigenschaft, Mehrweg-Routing[36] (*Equal Cost MultiPath - ECMP*)[37] bei kostengleichen Wegen zu ermöglichen; der Transport von IP-Datenpaketen kann also über mehrere Wege verteilt werden (*load balancing*). Eine spezifische Steuerung des Routings ist über zwei Typen (*Type 1*, *Type 2*) von numerischen Metrik-Attributen möglich (*link state metric*).

Für das Routing in grösseren Netzwerken stellt das Routing-Protokoll OSPF das Konzept der Bereiche (*area*s) zur Verfügung; grössere Netzwerke lassen sich auf diese Weise in mehrere Bereiche untergliedern, um das Routing zu vereinfachen und um den nötigen Datenverkehr für das Routing selbst zu reduzieren. Einer dieser Bereiche hat eine besondere Bedeutung, er wird *Area 0* genannt; *Area 0* ist der Backbone-Bereich (*backbone area*) des Autonomen Systems.[37] Teilbereiche des Netzwerkes werden an den Backbone-Bereich angebunden und bilden eigene Areas; ein Bereich, der direkt oder indirekt *über nur einen Router* an den Backbone-Bereich angebunden ist, wird *stub area* genannt.

Beim Routing-Protokoll OSPF - in der Protokollversion OSPFv2 und auch in der Protokollversion OSPFv3 - werden Areas über eine ID mit 32 Bit Länge (*Area ID*) referenziert, Router werden ebenfalls über eine ID mit 32 Bit Länge (*Router ID*) referenziert.

Einen Beispiel für eine Netzwerk-Topologie eines Autonomen Systems, in dem als Routing-Protokoll OSPF eingesetzt wird, stellt Abbildung 2.2 dar.

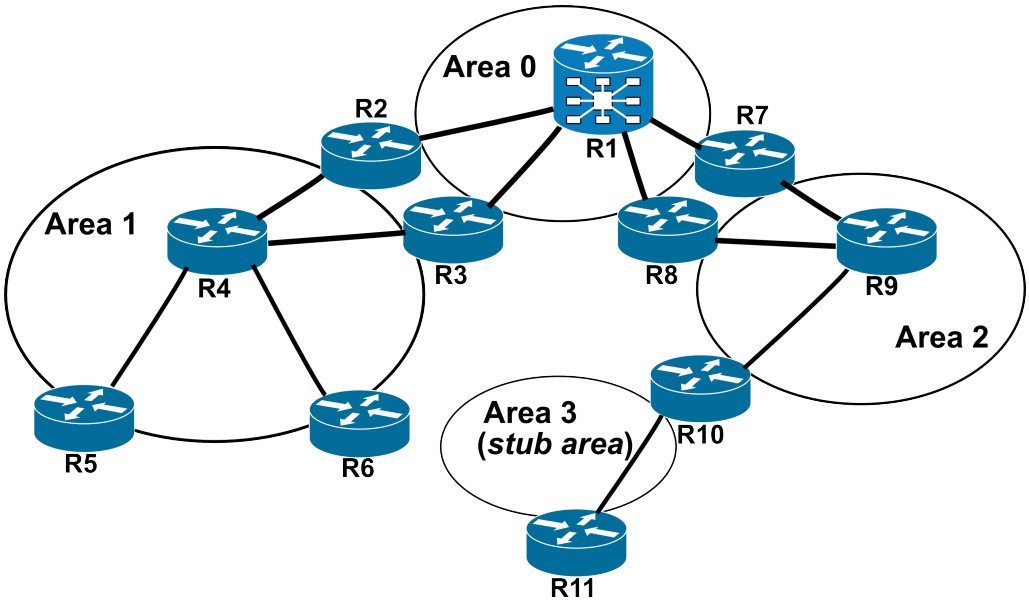

Abbildung 2.2: Beispiel für eine Netzwerk-Topologie mit OSPF als Routing-Protokoll

Die Authentizität und die Integrität der von den beteiligten Routern ausgetauschten

[36] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "OSPF" [8, S. 601]
[37] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 476]

Routing-Informationen kann kryptographisch abgesichert werden; zu diesem Zweck kann als zugrunde liegender Algorithmus der kryptographische Algorithmus MD5 (*Keyed-MD5*) genutzt werden (wie in RFC 2328 spezifiert),[34] durch eine Erweiterung der Spezifikation von OSPF durch RFC 5709 im Jahr 2009 kann zu diesem Zweck als zugrunde liegender Algorithmus auch der kryptographische Algorithmus SHA (*HMAC-SHA-1*, *HMAC-SHA-256*, *HMAC-SHA-384*, *HMAC-SHA-512*) genutzt werden.[38]

Historisch

Die Entwicklung des Routing-Protokolls OSPF als Interior Gateway Protocol (*IGP*) wurde im Jahr 1988 begonnen. Mit RFC 1247 im Jahr 1991 wurde OSPF in der Protokoll-Version OSPFv2 standardisiert.[39] Der Grund für die Entwicklung war der Umstand, das es grosse Netzwerke gab, in denen die bislang als Interior Gateway Protocol (*IGP*) genutzten Routing-Protokolle nur mehr bedingt geeignet waren.[40]

Ein ehemals gebräuchliches Interior Gateway Protocol (*IGP*) ist RIP (Routing Information Protocol); RIP ist ein dynamisches Distanzvektor-Routing-Protokoll. Vom Routing-Protokoll RIP gibt es im Zeitverlauf mehrere Versionen:

RIP / RIPv1: RFC 1058: Routing Information Protocol, 1988, Memo,
URL: `http://www.rfc-editor.org/rfc/rfc1058.txt` [29]

RIP-2 / RIPv2: RFC 2453: RIP Version 2, 1998, Standard,
URL: `http://www.rfc-editor.org/rfc/rfc2453.txt` [30]

RIPng: RFC 2080: RIPng for IPv6, 1997, Standard,
URL: `http://www.rfc-editor.org/rfc/rfc2080.txt` [31]

Ein weiteres älteres Interior Gateway Protocol (*IGP*) ist *IGRP* (Interior Gateway Routing Protocol); *IGRP* ist ein dynamisches Distanzvektor-Routing-Protokoll und wurde vom Routerhersteller Cisco Systems entwickelt. *IGRP* wurde vom Hersteller Cisco Systems als *deprecated* deklariert, der Einsatz von *IGRP* als Interior Gateway Protocol ist also nicht mehr empfehlenswert.

Sowohl das Routing-Protokoll *RIP* als auch das Routing-Protokoll *IGRP* haben die Eigenschaft eines langsamen Konvergenzverhaltens - also eines suboptimalen Laufzeitverhaltens bei Anpassungen der Routing-Tabellen an die äußeren Gegebenheiten, um ein optimales Routing zu erreichen - und die Eigenschaft der Skalierbarkeit in nur kleinen Massstäben bei Netzwerktopologien.

IGRP wurde durch *EIGRP* (Enhanced Interior Gateway Routing Protocol) abgelöst; *EIGRP* ist ein dynamisches Distanzvektor-Routing-Protokoll und wurde wie auch *IGRP* vom Routerhersteller Cisco Systems entwickelt. Das Routing-Protokoll *EIGRP* hat die Eigenschaft eines schnellen Konvergenzverhaltens und hat die Eigenschaft der Skalierbarkeit auch in grossen Massstäben bei Netzwerktopologien; ein Nachteil von *EIGRP* ist jedoch, dass *EIGRP* in der dieses Routing-Protokoll betreffenden *RFC*[41] nicht als Standard deklariert ist.

[38] RFC 5709: OSPFv2 HMAC-SHA Cryptographic Authentication, 2009, Proposed Standard,
URL: `http://www.rfc-editor.org/rfc/rfc5709.txt` [27]
[39] RFC 1247: OSPF Version 2, 1991, Draft Standard,
URL: `http://www.rfc-editor.org/rfc/rfc1247.txt` [28]
[40] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 474]
[41] RFC 7868: Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP), 2016, Informational,

2.1.3 Routing-Software im Internet

Routing-Software im Internet lässt sich in zwei Gruppen einordnen: Routing-Software, die gemeinsam mit speziell für das Routing entworfener Hardware als Appliance eingesetzt wird und Routing-Software, bei der das Routing ohne spezielle Hardware-Unterstützung für das Routing bewerkstelligt wird.

Hersteller von Appliances für das Routing - von Produkten, die eine Kombination von Software und Hardware zum Zweck des Routings darstellen - sind die Unternehmen Cisco Systems und Juniper Networks. Bei deren Appliances werden Routing-Funktionen in Hardware ausgelagert, als Hardware für diese Routing-Funktionen werden spezifisch hergestellte Integrierte Schaltungen (*ASIC*s - *Application Specific Integrated Circuit*s) eingesetzt.
Systemsoftware für diese Appliances für das Routing ist z.B. "IOS" und "IOS XR" vom Router-Hersteller Cisco Systems (siehe Abschnitt 2.1.3.1) und "Junos OS" vom Router-Hersteller Juniper Networks (siehe Abschnitt 2.1.3.2).
Im Report "2011 Report on European IXPs (Euro-IX IXP Report 2011)" der European Internet Exchange Association (*Euro-IX*) - der Vereinigung der Europäischen Internet-Austauschpunkte - aus dem Jahr 2012 werden die Hersteller von genutzten Switches und von genutzten Routern bei den Europäischen Internet-Austauschpunkten nach Marktanteil aufgeschlüsselt,[42] Tabelle 2.3 stellt die Marktanteile dar.

Tabelle 2.3: Hersteller von genutzten Switches und Routern bei den Europäischen Internet-Austauschpunkten nach Marktanteil

	Hersteller	**Marktanteil**
1	Brocade	40%
2	Cisco	31%
3	Extreme	12%
4	Force10	9%
5	Juniper	5%
6	HP	1%
7	Arista	1%
8	Dell	1%

Beim Routing ohne spezielle Hardware-Unterstützung für das Routing hingegen werden Kombinationen von Betriebssystem und Routing-Software eingesetzt; die eingesetzten Betriebssysteme sind überwiegend Betriebssysteme aus der *UNIX*-Betriebssystemfamilie.
Die eingesetzte Software für das Routing ist entweder monolithisch als Daemon implementiert (z.B. *OpenBGPD*[43]) oder sie ist als Paket - bestehend aus mehreren Daemons und

URL: `http://www.rfc-editor.org/rfc/rfc7868.txt` [32]

[42] Bijal Sanghani, European Internet Exchange Association: 2011 Report on European IXPs (Euro-IX IXP Report 2011), URL: `https://www.euro-ix.net/m/cms_page_media/82/Euro-IX%20IXP%20Report%202011.pdf` [33, S. 12]

[43] *OpenBGPD* ist eine freie Implementierung des Exterior Gateway Protokolls BGP-4 als Daemon für das Betriebssystem *OpenBSD*, URL: `http://openbgpd.org/` [34].

Hilfsprogrammen - als *Routing Software Suite* implementiert (z.B. *Quagga*, *Zebra*[44,45]).

Verbreitete Software für das dynamische Routing im Bereich des Routings zwischen Autonomen Systemen und im Bereich des Routings innerhalb Autonomer Systeme ohne spezielle Hardware-Unterstützung ist *Quagga* (siehe Abschnitt 2.1.3.4), *Zebra*, *BIRD* (siehe Abschnitt 2.1.3.5), *XORP*[46] und *OpenBGPD*.

Im Report "2011 Report on European IXPs (Euro-IX IXP Report 2011)" der European Internet Exchange Association (*Euro-IX*) aus dem Jahr 2012 werden die Betriebsysteme und Routing-Softwares der genutzten *Route Server*[47] bei den Europäischen Internet-Austauschpunkten nach Anteil aufgeschlüsselt;[48] Tabelle 2.4 stellt die genutzten Betriebssysteme nach Anteil dar, Tabelle 2.5 stellt die genutzten Routing-Softwares nach Anteil dar.

Tabelle 2.4: Betriebssysteme der genutzten Route Server bei den Europäischen Internet-Austauschpunkten nach Anteil

Betriebssystem	**Anteil**
Cisco IOS	14%
Debian Linux	35%
Fedora Linux	1%
FreeBSD	20%
Gentoo Linux	6%
OpenBSD	23%
Ubuntu Linux	1%

Tabelle 2.5: Routing-Softwares der genutzten Route Server bei den Europäischen Internet-Austauschpunkten nach Anteil

Routing-Software	**Anteil**
BIRD	41%
Cisco	14%
OpenBGPD	22%
Quagga	22%
andere	3%

[44] *Zebra* ist eine freie *Routing Software Suite* für die Betriebssysteme *Linux*, *FreeBSD*, *NetBSD* und *OpenBSD*, URL: `https://www.gnu.org/software/zebra/` [35].

[45] *Zebra* wird nicht aktiv weiterentwickelt, eine Weiterentwicklung eines früheren Entwicklungszweigs von *Zebra* ist *Quagga*.

[46] *XORP* ist eine freie Routingplattform (*routing platform*) für die Betriebssysteme *Linux* und Betriebssysteme der *BSD*-Betriebssystemfamilie; die aktuelle Version ist Version 1.8.5 aus dem Jahr 2012, URL: `http://www.xorp.org/` [36].

[47] *Route Server* haben beim dynamischen Routing die primäre Funktion des Monitorings - also der Überwachung - der Routing-Tabellen.

[48] Bijal Sanghani, European Internet Exchange Association: 2011 Report on European IXPs (Euro-IX IXP Report 2011), URL: `https://www.euro-ix.net/m/cms_page_media/82/Euro-IX%20IXP%20Report%202011.pdf` [33, S. 13-14]

2.1.3.1 Cisco Systems "IOS"

Cisco Systems "IOS", "IOS XR", "IOS XE":

IOS ist die Systemsoftware der *IOS*-Routingplattform vom Hersteller Cisco Systems; sie ist als Produkt ausgerichtet auf den Einsatz in Unternehmens-Netzwerken. Die aktuelle Version der *IOS*-Software ist Cisco IOS Relase 15 (Stand 16.06.2017); die Software bildet abgestimmt auf die Hardware und zusammen mit der Hardware von Cisco Systems (z.B. Cisco 800 Series Industrial Integrated Services Router, Cisco 1000 Series Connected Grid Router, Cisco 3900 Series Integrated Services Router[49]) eine Appliance für das Routing.
IOS unterstützt an dynamischen Routing-Protokollen in der Release 15 die Routing-Protokolle *BGP*, *IS-IS*, *OSPF* (in der Protokollversion OSPFv2 und in der Protokollversion OSPFv3)[50], *RIP*, *EIGRP* sowie *LISP*[51].[52]

IOS XR ist die Systemsoftware der *Cisco Carrier Routing System*-Routingplattform (*Cisco CRS Router*), der *Cisco Aggregation Services Routers*-Routingplattform (*Cisco ASR*) und der *Cisco Network Convergence System Routers*-Routingplattform (*Cisco NCS Router*) vom Hersteller Cisco Systems; sie ist als Produkt ausgerichtet auf den Einsatz in Netzwerken von Internet Service Providern. Die aktuelle Version der *IOS XR*-Software ist Cisco IOS XR 5.1.x (Stand 16.06.2017); die Software bildet abgestimmt auf die Hardware und zusammen mit der Hardware von Cisco Systems (z.B. Cisco CRS Router, Cisco ASR 9000 Series Aggregation Services Router, Cisco Network Convergence System 6000 Series Router) eine Appliance für das Routing.
IOS XR unterstützt an dynamischen Routing-Protokollen in der Release 5.1.x die Routing-Protokolle *BGP*, *IS-IS*, *OSPF* (in der Protokollversion OSPFv2[53] und in der Protokollversion OSPFv3[54]), *RIP*, *EIGRP* sowie *LISP*.[55]

IOS XE ist die Systemsoftware mehrerer Routingplattformen vom Hersteller Cisco Systems; zum Einem ist *IOS XE 3* die Systemsoftware für Appliances in Verbindung mit Hardware von Cisco Systems (z.B. Cisco ASR 1000 Series Aggregation Services Routers, Cisco 4000 Series Integrated Services Routers); zum Anderen ist *IOS XE 16* die Systemsoftware einer virtuellen Routingplattform von Cisco Systems (Virtual Rou-

[49] Cisco Systems: Cross Platform Release Notes for Cisco IOS Release 15.6M&T, URL: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/15-6m-and-t/release/notes/15-6m-and-t-book/156TREQS.html [37] (Stand: 25.04.2017)

[50] Cisco Systems: Cisco IOS IP Routing: OSPF Command Reference, URL: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_ospf/command/iro-cr-book/ospf-a1.html [38] (Stand: 11.07.2017)

[51] RFC 6830: The Locator/ID Separation Protocol (LISP), 2013, Experimental, URL: http://www.rfc-editor.org/rfc/rfc6830.txt [39]

[52] Cisco Systems: Cisco IOS Software Release 15.6(3)M Documentation, URL: http://www.cisco.com/c/en/us/support/ios-nx-os-software/ios-software-release-15-6-3-m/model.html [40]

[53] Cisco Systems: Cisco IOS XR Routing Command Reference for the Cisco CRS Router, Release 5.1.x - OSPF Commands, URL: http://www.cisco.com/c/en/us/td/docs/routers/crs/software/crs_r5-1/routing/command/reference/b_routing_cr51xcrs/b_routing_cr51xcrs_chapter_0100.html [41] (Stand: 02.06.2017)

[54] Cisco Systems: Cisco IOS XR Routing Command Reference for the Cisco CRS Router, Release 5.1.x - OSPFv3 Commands, URL: http://www.cisco.com/c/en/us/td/docs/routers/crs/software/crs_r5-1/routing/command/reference/b_routing_cr51xcrs/b_routing_cr51xcrs_chapter_0101.html [42] (Stand: 02.06.2017)

[55] Cisco Systems: Cisco IOS XR Routing Command Reference for the Cisco CRS Router, URL: http://www.cisco.com/c/en/us/td/docs/routers/crs/software/crs_r5-1/routing/command/reference/b_routing_cr51xcrs.html [43] (Stand: 02.06.2017)

ting ISRv, Virtual Routing CSR1000v).[56] Die Routingplattform CSR1000v kann mittels Betriebssystem-Virtualisierung als Gast-Betriebssystem (Virtuelle Maschine - *VM*) unter den Betriebssystem-Virtualisierungs-Hypervisoren VMware *vSphere*, Microsoft *Hyper-V*, Citrix *Xen* bzw. Linux *Xen* und Linux *KVM* eingesetzt werden.[57]
Es existieren mehrere Produktlinien von *IOS XE*: *IOS XE 16*, *IOS XE 3S*, *IOS XE 3E* sowie *IOS XE 3SG*, *IOS XE 3CE*, *IOS XE 3SQ* und *IOS XE 3SE* (Stand 16.06.2017).
IOS XE 16 für die virtuelle Routingplattform Virtual Routing CSR1000v unterstützt an dynamischen Routing-Protokollen in der aktuellen Release IOS XE Everest 16.5 die Routing-Protokolle *BGP*, *IS-IS*, *OSPF* (in der Protokollversion OSPFv2 und in der Protokollversion OSPFv3)[58] sowie *LISP*.[59]

Auf der Webpräsenz von Cisco Systems ist auf der Seite "Netzwerksoftware (IOS und NX-OS) - Netzwerksoftware - Cisco" eine Übersicht der Netzwerksoftware, die von Cisco Systems vertrieben wird, dargestellt; die Seite ist unter der URL `http://www.cisco.com/c/de_de/products/ios-nx-os-software/index.html` [48] abrufbar.

2.1.3.2 Juniper Networks "Junos OS"

Juniper Networks "Junos OS":

Junos OS ist die Systemsoftware mehrerer Produktreihen von Routingplattformen (z.B. Core Router T Series, Universal Access Router ACX Series, PTX Series)[60] vom Hersteller Juniper Networks; sie ist als Produkt ausgerichtet auf den Einsatz in Netzwerken von Internet Service Providern. Die aktuelle Version der *Junos OS*-Software ist Junos OS Release 17.1R2 (Stand 20.07.2017)[61]; die Software bildet abgestimmt auf die Hardware und zusammen mit der Hardware von Juniper Networks eine Appliance für das Routing.
Junos OS unterstützt an dynamischen Routing-Protokollen die Routing-Protokolle *BGP*, *IS-IS*, *OSPF* (in der Protokollversion OSPFv2 und in der Protokollversion OSPFv3)[62] sowie *RIP*.[63]

Junos OS ist auch die Systemsoftware einer virtuellen Routingplattform von Juniper

[56] Cisco Systems: Cisco IOS XE, URL: `http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-xe/index.html` [44]
[57] Cisco Systems: Cisco Integrated Services Virtual Router Data Sheet, URL: `http://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/enterprise-network-functions-virtualization-nfv/datasheet-c78-736768.html` [45], (Stand: 26.04.2017)
[58] Cisco Systems: IP Routing: OSPF Configuration Guide, URL: `http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_ospf/configuration/xe-16/iro-xe-16-book.html` [46] (Stand: 04.08.2016)
[59] Cisco Systems: Cisco Cloud Services Router 1000V Series - Configuration Guides, URL: `http://www.cisco.com/c/en/us/support/routers/cloud-services-router-1000v-series/products-installation-and-configuration-guides-list.html` [47]
[60] Juniper Networks: Routing-Produkte, URL: `http://www.juniper.net/de/de/products-services/routing/` [49]
[61] Juniper Networks: Junos OS - Technical Documentation - Support, URL: `https://www.juniper.net/documentation/en_US/release-independent/junos/information-products/pathway-pages/junos/product/index.html` [50]
[62] Juniper Networks: OSPF Feature Guide - Technical Documentation - Support, URL: `https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/config-guide-routing/config-guide-ospf.html` [51] (Stand: 14.03.2017)
[63] Juniper Networks: Junos OS Routing Protocols Library - Technical Documentation - Support, URL: `https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/config-guide-routing/index.html` [52]

Networks (Virtual Router Software vMX Series);[64] diese virtuelle Routingplattform wird in einer virtualisierten Umgebung unter dem Betriebssystem *Linux* ausgeführt, es können neben der Software auch Container-Anwendungen und andere Gast-Betriebssysteme ausgeführt werden.[65]

Auf der Webpräsenz von Juniper Networks sind auf der Seite "Junos - Netzwerkbetriebssystem - Juniper Networks" weitere Informationen über die Software *Junos OS* dargestellt; die Seite ist unter der URL `http://www.juniper.net/de/de/products-services/nos/junos/` [54] abrufbar.

2.1.3.3 Brocade "vRouter"

Brocade vRouter (*Brocade 5600 vRouter*) ist eine virtuelle Routingplattform des Herstellers Brocade Communications Systems;[66] sie ist als Produkt ausgerichtet auf den Einsatz in Netzwerken von Internet Service Providern und auf den Einsatz in grösseren Unternehmens-Netzwerken. *Brocade vRouter* baut auf dem Betriebssystemkern (*kernel*) des Betriebssystems *Linux* auf.[67] Die Routingplattform *Brocade vRouter* kann mittels Betriebssystem-Virtualisierung als Gast-Betriebssystem (Virtuelle Maschine - *VM*) unter den Betriebssystem-Virtualisierungs-Hypervisoren VMware *ESXi*, Microsoft *Hyper-V*, Citrix *Xen* (*XenServer*) und Linux *KVM* eingesetzt werden.[68]

Brocade vRouter unterstützt an dynamischen Routing-Protokollen die Routing-Protokolle *BGP*, *OSPF* (in der Protokollversion OSPFv2 und in der Protokollversion OSPFv3) sowie *RIP*.[69]

Hervorhebenswert ist, dass *Brocade vRouter* mehrere VPN-Technologien und -Protokolle, z.B. *OpenVPN*, *IPsec* und *L2TP* (*L2TPv3*) in Verbindung mit *IPsec* unterstützt.[70]

2.1.3.4 Quagga

Quagga ist eine frei zugängliche und frei verfügbare *Routing Software Suite* (siehe Abschnitt 2.1.3), die zusammen mit einem geeigneten Betriebssystem zum Routing eingesetzt werden kann; *Quagga* ist in Kombination mit den Betriebssystemen *Linux*, *FreeBSD*, *NetBSD* und *Solaris* als Routing-Software einsetzbar.[71]
Quagga wird aktiv gepflegt, es existiert eine Mailingliste (*mailing list*) für Anwender und

[64] Juniper Networks: Virtual Router Software, URL: `http://www.juniper.net/us/en/products-services/routing/mx-series/vmx/` [53]
[65] Juniper Networks: Junos – Netzwerkbetriebssystem, URL: `http://www.juniper.net/de/de/products-services/nos/junos/` [54]
[66] Brocade Communications Systems: Brocade vRouter (formerly Vyatta vRouter), URL: `http://www.brocade.com/en/products-services/software-networking/network-functions-virtualization/vrouter.html` [55]
[67] Brocade Communications Systems: Brocade vRouter Architecture, URL: `http://www.brocade.com/content/html/en/technical-documentation-library/5600-vrouter-41R1-library/GUID-B07BD5C1-5E8A-497C-9FE6-DA9B31D27369.html` [56]
[68] Brocade Communications Systems: Brocade vRouter Data Sheet, URL: `http://www.brocade.com/content/dam/common/documents/content-types/datasheet/brocade-vrouter-ds.pdf` [57, S. 7]
[69] Brocade Communications Systems: Brocade vRouter Data Sheet, URL: `http://www.brocade.com/content/dam/common/documents/content-types/datasheet/brocade-vrouter-ds.pdf` [57, S. 5-6]
[70] Brocade Communications Systems: Brocade vRouter Data Sheet, URL: `http://www.brocade.com/content/dam/common/documents/content-types/datasheet/brocade-vrouter-ds.pdf` [57, S. 3]
[71] Quagga Software Routing Suite, URL: `http://www.nongnu.org/quagga/` [58]

eine Mailingliste für Entwickler,[72] über freigegebene neue Versionen von *Quagga* wird in einem Neuigkeiten-Bereich informiert.[73]
Quagga besteht als *Routing Software Suite* aus einem zentralen *Daemon* (*zebra*) und mehreren auf die unterschiedlichen Routing-Protokolle spezialisierten *Daemon*s.[71]

Quagga unterstützt an dynamischen Routing-Protokollen die Routing-Protokolle *BGP*, *RIP* und *OSPF* in der Protokollversion OSPFv2 sowie in nicht vollständiger Funktionalität und in nicht ausreichender Stabilität die Routing-Protokolle *IS-IS* und *OSPF* in der Protokollversion OSPFv3.[71]

Die aktuelle Version der Software ist Version 1.2.1 vom 10.03.2017 (Stand 20.07.2017).[73] Unter der URL `http://download.savannah.gnu.org/releases/quagga/` [61] ist der Quellcode der Software abrufbar; des Weiteren ist der Quellcode des aktuellen Entwicklungszweiges der Software auf der Internetseite "quagga.git - quagga" unter der URL `http://git.savannah.gnu.org/cgit/quagga.git` zugreifbar.[74]

Hervorhebenswert ist, dass *Quagga* über einen Kommandozeilen-Interpreter (*Command Line Interface - CLI*) konfigurierbar und steuerbar ist[71,75] und dass dieser Kommandozeilen-Interpreter eine ähnliche Semantik, Grammatik und Syntax besitzt wie die Systemsoftware *IOS* für die Router des Herstellers Cisco Systems (siehe Abschnitt 2.1.3.1 Cisco Systems "IOS").

2.1.3.5 BIRD

BIRD - The BIRD Internet Routing Daemon - ist eine frei zugängliche und frei verfügbare Routing-Software, die als *Daemon* implementiert ist[76] und die zusammen mit einem geeigneten Betriebssystem zum Routing eingesetzt werden kann (siehe Abschnitt 2.1.3); *BIRD* ist in Kombination mit den Betriebssystemen *Linux*, *FreeBSD*, *NetBSD* und *OpenBSD* als Routing-Software einsetzbar.[77]
BIRD wird aktiv gepflegt und weiterentwickelt, es existiert eine Mailingliste für Anwender,[78] über freigegebene neue Versionen von *BIRD* wird auf der Webpräsenz des Projekts informiert;[77] detaillierte Informationen über neue Funktionalität, über Änderungen und über Fehlerbeseitigungen von neuen Versionen werden fortgeschrieben und sind abrufbar.[79]

BIRD unterstützt an dynamischen Routing-Protokollen die Routing-Protokolle *BGP*, *RIP* und *OSPF* in der Protokollversion OSPFv2 und in der Protokollversion OSPFv3[77,79]

[72] Quagga Software Routing Suite: Mailing Lists, URL: `http://www.nongnu.org/quagga/lists.html` [59]
[73] Quagga Software Routing Suite: News, URL: `http://savannah.nongnu.org/news/?group=quagga` [60] (Stand: 10.03.2017)
[74] Quagga Software Routing Suite: Development, URL: `http://www.nongnu.org/quagga/devel.html` [62]
[75] Quagga Software Routing Suite: Documentation - Quagga Info pages - Command Index, URL: `http://www.nongnu.org/quagga/docs/docs-info.html#Command-Index` [63]
[76] The BIRD Internet Routing Daemon Project: BIRD Programmer's Documentation - BIRD Design, URL: `http://bird.network.cz/?get_doc&f=prog-1.html` [64]
[77] The BIRD Internet Routing Daemon Project, URL: `http://bird.network.cz/` [65]
[78] The BIRD Internet Routing Daemon Project: Mailing List, URL: `http://bird.network.cz/?m_list` [66]
[79] The BIRD Internet Routing Daemon Project: News Archive, URL: `http://bird.network.cz/?o_news` [67]

sowie das dynamische Distanzvektor-Routing-Protokoll *Babel*[80,81,82].[77,83]

Die aktuelle Version der Software ist Version 1.6.3 vom 22.12.2016 (Stand 21.07.2017).[77,79]
Der Quellcode der Software ist auf der Internetseite "The BIRD Internet Routing Daemon Project - Download Current releases" unter der URL `http://bird.network.cz/?download` [72] abrufbar; des Weiteren ist der Quellcode des aktuellen Entwicklungszweiges der Software unter der URL `https://gitlab.labs.nic.cz/labs/bird/tree/master` zugreifbar.[77]
Es existiert ein Entwicklungszweig der Software, aus dem heraus experimentelle - nicht für Produktivumgebungen freigegebene - Versionen zur Verfügung gestellt werden; die aktuelle experimentelle Version ist Version 2.0.0-pre1 vom 30.04.2017 (Stand 21.07.2017).[77,79]

Hervorhebenswert ist, dass beim Routing mit *BIRD* die Routing-Tabellen und deren Verwendung mit einer Filter-Programmiersprache - diese Filter-Programmiersprache ist von der Semantik, Grammatik und Syntax an die Semantik, Grammatik und Syntax der Programmiersprache C angelehnt - gesteuert werden können.[84]

2.2 Virtual Private Networks

Mit Hilfe von Virtual Private Networks werden räumlich entfernte Netzwerke miteinander verbunden und erscheinen für die Benutzer und für die Anwendungen als *ein* logisches Netzwerk; ein Vorteil ist hierbei, dass Benutzer auf Ressourcen - Datenbanken, Dokumente, Drucker oder Dienste - *transparent*[85,86] zugreifen können.
Ein Anwendungsfall von Virtual Private Networks ist die Verbindung von Firmenniederlassungen und Firmenzentralen, ein anderer Anwendungsfall ist die Telearbeit[87].

Ein *Virtual Private Network* (*VPN*) hat diese Eigenschaften:

- Es dient dem Zweck, Netzwerke miteinander zu einem logischen Netzwerk zu verbinden zur *transparenten*[85,86] *Nutzung* von Ressourcen.
- Der zu Grunde liegende Übertragungsweg ist öffentlich, die übertragenen Nutzdaten sind *nicht-öffentlich* und *kryptographisch gesichert*.
- Es werden *simulierte* - virtuelle - *Netzwerkschnittstellen* verwendet.

[80] RFC 6126: The Babel Routing Protocol, 2011, Experimental, URL: `http://www.rfc-editor.org/rfc/rfc6126.txt` [68]
[81] RFC 7298: Babel Hashed Message Authentication Code (HMAC) Cryptographic Authentication, 2014, Experimental, URL: `http://www.rfc-editor.org/rfc/rfc7298.txt` [69]
[82] RFC 7557: Extension Mechanism for the Babel Routing Protocol, 2015, Experimental, URL: `http://www.rfc-editor.org/rfc/rfc7557.txt` [70]
[83] The BIRD Internet Routing Daemon Project: BIRD User's Guide - Protocols, URL: `http://bird.network.cz/?get_doc&f=bird-6.html` [71]
[84] The BIRD Internet Routing Daemon Project: BIRD User's Guide - Filters, URL: `http://bird.network.cz/?get_doc&f=bird-5.html` [73]
[85] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "transparent. Transparenz" [8, S. 856]
[86] Andrew S. Tanenbaum und Maarten Van Steen: Distributed systems: Principles and Paradigms: Second Edition, 2006 [74, S. 4]
[87] Telearbeit ist eine Form der Arbeit, bei der der Arbeitnehmer über Datenleitungen mit der räumlich entfernten IT-Infrastruktur des Arbeitgebers verbunden ist.

2.2.1 Definition von "Virtual Private Network"

Definition von *Virtual Private Network*: "*VPN*; durch strenge Authentisierung, Autorisierung und Verschlüsselung gesicherte und deshalb vertrauliche Koppelung zweier geschlossener Netzwerke über eine öffentliche und unsichere Netzwerkinfrastruktur; [...]" [Hervorhebung nicht im Original][88].

Begriffsverwendung

Der Begriff *VPN* (*Virtual Private Network*) wird in mehreren Formen verwendet: Einerseits bezeichnet der Begriff *VPN* die Struktur, mit der Netzwerke miteinander zu einem Virtual Private Network verbunden sind; andererseits bezeichnet der Begriff auch die Software, welche für den Aufbau von Virtual Private Networks eingesetzt wird.

2.2.2 Framework für IP-basierte VPNs

Verschiedene Typen von Virtual Private Networks werden von Internet Service Providern über Internet-Backbones betrieben, um für Kunden Netzwerke miteinander zu verbinden. In RFC 2764[89] wurden im Jahr 2000 die im Internet-Backbone-Bereich üblichen und technisch möglichen VPN-Varianten dokumentiert, um als Framework für die Weiterentwicklung der technischen Spezifikationen für Virtual Private Networks im Internet-Backbone-Bereich zu dienen und um als Grundlage für Interoperabilität[90] zu dienen.

2.2.3 Von ISPs bereitgestellte VPNs

Von Internet Service Providern bereitgestellte Virtual Private Networks lassen sich gliedern in Virtual Private Networks, die Netzwerke von Kunden auf OSI Layer 2 - auf Ethernet-Ebene (*Data Link Layer*[91]) - miteinander verbinden und in Virtual Private Networks, die Netzwerke von Kunden auf OSI Layer 3 - auf IP-Ebene (*Network Layer*[91]) - miteinander verbinden.[92]

Die VPN-Kopplung von Netzwerken von Kunden auf OSI Layer 2 - auf Ethernet-Ebene - hat den Vorteil, dass für nutzende Anwendungen auf den darüberliegenden, darauf aufbauenden Schichten hohe Kompatibilität - vergleichbar mit dem direkten Zugriff im LAN - gegeben ist; diese Art der Kopplung hat jedoch den Nachteil, dass sie in grösseren zu koppelnden Netzwerken weniger gut anwendbar ist. Der Grund dafür ist, dass bei VPN-Kopplung von Netzwerken auf OSI Layer 2 - auf Ethernet-Ebene - die gekoppelten Netzwerke eine Broadcastdomäne[93] bilden.
Im Internet-Draft "Framework for Layer 2 Virtual Private Networks (L2VPNs)"[94] der Net-

[88] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Virtual Private Network" [8, S. 905]
[89] RFC 2764: A Framework for IP Based Virtual Private Networks, 2000, Informational, URL: `http://www.rfc-editor.org/rfc/rfc2764.txt` [75]
[90] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Interoperabilität" [8, S. 416]
[91] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "OSI-Schichten, Namen und Nummern der -" [8, S. 601]
[92] RFC 4026: Provider Provisioned Virtual Private Network (VPN) Terminology, 2005, Informational, URL: `http://www.rfc-editor.org/rfc/rfc4026.txt` [76, S. 7]
[93] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Broadcastdomäne" [8, S. 120]
[94] Loa Andersson und Eric C. Rosen: Internet-Draft: Framework for Layer 2 Virtual Private Networks, 2004, URL: `http://www.ietf.org/internet-drafts/draft-ietf-l2vpn-l2-framework-05.txt` [77]

work Working Group der IETF (Internet Engineering Task Force) wurden im Jahr 2004 die üblichen und technisch möglichen Layer 2-VPN-Varianten dokumentiert, um als Framework für die Weiterentwicklung der technischen Spezifikationen für Layer 2-Virtual Private Networks zu dienen und um als Grundlage für Interoperabilität[90] zu dienen.

Die VPN-Kopplung von Netzwerken von Kunden auf OSI Layer 3 - auf IP-Ebene - hat den Vorteil, dass die Kopplung von grösseren Netzwerken möglich ist; diese Art der Kopplung hat jedoch den Nachteil der nicht vollständigen Kompatibilität für Anwendungen im Vergleich mit dem direkten Zugriff im LAN.
Die VPN-Kopplung von Netzwerken von Kunden auf OSI Layer 3 - auf IP-Ebene - kann durch den Internet Service Provider *für den Kunden* oder *durch den Kunden selbst* bewerkstelligt werden.[92]

Bei der durch den Internet Service Provider bewerkstelligten VPN-Kopplung von Kunden-Netzwerken auf IP-Ebene ist ein mögliches dynamisches Routing-Protokoll das *Interior Gateway Protocol* (*IGP*) *OSPF* (siehe Abschnitt 2.1.2) in Verbindung mit dem *Border Gateway Protocol* (*BGP*) (siehe Abschnitt 2.1.2) oder in Verbindung mit *Multiprotocol Label Switching* (*MPLS*)[95].[92,96,97,98]
Kryptografische Absicherung der Vertraulichkeit der zwischen den gekoppelten Netzwerken transportierten IP-Pakete ist bei durch Internet Service Provider für Kunden bereitgestellte BGP/MPLS IP Virtual Private Networks nicht gegeben.[99]

2.2.4 Standardisierte VPN-Protokolle

Die folgenden Abschnitte stellen standardisierte VPN-Protokolle, die im Internet[100] und in Intranets[101] bzw. nicht-öffentlichen Netzwerken Verwendung finden, als Informationsgrundlage vor; vorgreifend auf das Kapitel 3 Analyse und auf das Kapitel 5 Konzeption werden relevante Eigenschaften der VPN-Protokolle dargestellt.

[95] *Multiprotocol Label Switching* (*MPLS*) ist eine Datagramm-Transporttechnik, bei der die Datagramme eingekapselt transportiert werden; hierbei wird jedem eingekapselten Datagramm ein Header vorangestellt. Ein Bestandteil des vorangestellten Headers ist ein Kennzeichner der Länge 20 Bit, der *Label* genannt wird [78] [9, S. 360,470,471] [79, S. 34].
[96] RFC 4577: OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs), 2006, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4577.txt` [80]
[97] RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs), 2006, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4364.txt` [81]
[98] RFC 6178: Label Edge Router Forwarding of IPv4 Option Packets, 2011, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc6178.txt` [82]
[99] RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs), 2006, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4364.txt` [81, S. 39]
[100] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Internet" [8, S. 414,415]
[101] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Intranet" [8, S. 417]

2.2.4.1 IPsec

IPsec ist eine Sicherheitsarchitektur für das *IP*-Protokoll, die 1998 standardisiert wurde;[102,103,104,105] der Zweck von *IPsec* ist es, die Vertraulichkeit und die Integrität von Kommunikationsverbindungen, die auf dem *IP*-Protokoll basieren, zu schützen.
IPsec sieht mehrere Möglichkeiten vor, Kommunikationsverbindungen zu schützen: Möglich ist der Schutz einzelner *TCP*-Verbindungen, der Schutz aller *IP*-Verbindungen zweier miteinander kommunizierender Rechner oder der Schutz jeglicher *IP*-Datenverkehre, die über zwei *IPsec* einsetzende Router übertragen werden.
Die Architektur von *IPsec* ist untergliedert in den Bereich der Verbindung zweier Endpunkte (*Security Association - SA*) und in den Bereich der kryptographischen Schlüsselverwaltung (*Internet Security Association and Key Management Protocol - ISAKMP* und *Internet Key Exchange - IKE*).
Der Pakettransport für die kryptografische Schlüsselverwaltung (*IKEv2*) basiert auf dem UDP-Protokoll (UDP-Port 500 oder alternativ UDP-Port 4500).[106]

Die Verschlüsselung des Datenverkehrs erfolgt mit den Verschlüsselungsverfahren *AES*, *TripleDES* oder *DES*.[107]

Die Authentifizierung kann durch Zertifikate oder durch Passwörter erfolgen, bei der Authentifizierung durch Passwörter wird aus dem Passwort mit Hilfe einer Einwegfunktion ein *Pre-Shared Key* (*PSK*) erzeugt; es kommt *RSA*-Verschlüsselung, *DSA*-Verschlüsselung oder auf elliptischen Kurven basierende *ECDSA*-Verschlüsselung in Verbindung mit dem Algorithmus *SHA* und dem *Diffie-Hellman*-Verfahren (*DH*) zum Einsatz.[113,114]

Eine zusätzliche optionale Funktionalität von *IPsec* ist die Kompression des Datenverkehrs, die zugrunde liegenden Kompressionsalgorithmen sind *DEFLATE*, *LZS* und *LZJH*.[108]

IPsec hat zwei Modi für die Verbindung zweier Endpunkte: den Transportmodus (*transport mode*) und den Tunnelmodus (*tunnel mode*). Der Tunnelmodus von *IPsec* eignet sich zum Aufbau von Virtual Private Networks, da in diesem Modus ein zu übertragendes IP-Datagramm vollständig in ein neues IP-Datagramm mit einem neuen IP-Header gekapselt wird.[109]

Der aktuelle Standard der Protokollspezifikation von *IPsec* ist aus dem Jahr 2005[110,111,112]

---

[102] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 814]
[103] RFC 2401: Security Architecture for the Internet Protocol, 1998, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc2401.txt` [83]
[104] RFC 2402: IP Authentication Header, 1998, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc2402.txt` [84]
[105] RFC 2406: IP Encapsulating Security Payload (ESP), 1998, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc2406.txt` [85]
[106] RFC 7296: Internet Key Exchange Protocol Version 2 (IKEv2), 2014, Standard, URL: `http://www.rfc-editor.org/rfc/rfc7296.txt` [86, S. 63]
[107] RFC 7321: Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH), 2014, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc7321.txt` [87, S. 4]
[108] RFC 7296: Internet Key Exchange Protocol Version 2 (IKEv2), 2014, Standard, URL: `http://www.rfc-editor.org/rfc/rfc7296.txt` [86, S. 62]
[109] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 815]
[110] RFC 4301: Security Architecture for the Internet Protocol, 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4301.txt` [88]
[111] RFC 4303: IP Encapsulating Security Payload (ESP), 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4303.txt` [89]
[112] RFC 4305: Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH), 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4305.txt` [90]

in Verbindung mit Ergänzungen und Erweiterungen bis ins Jahr 2016[113,114,115,116].

IPsec ist für viele Betriebssysteme verfügbar, für Betriebssysteme der *Unix*-Betriebssystemfamilie wie *Linux*, *FreeBSD*, *NetBSD*, *OpenBSD*, für Microsoft *Windows*, für Apple *MacOS/X* und für Betriebssyteme für mobile Geräte wie Google *Android* und Apple *iOS*.

2.2.4.2 L2TP (Layer Two Tunneling Protocol)

Mit Hilfe des Protokolls *L2TP* (*Layer Two Tunneling Protocol*) können zwischen zwei Computern über IP-basierende Transit-Netzwerke hinweg OSI Layer 2-Verbindungen[117] aufgebaut werden; das Protokoll *L2TP* ermöglicht die Tunnelung[118] von Ethernet-Frames[119] in *IP*-Datenpaketen.[120]

Der Pakettransport auf dem benutzten Transit-Netz basiert auf dem UDP-Protokoll (UDP-Port 1701) oder auf dem IP-Protokoll (IP Protocol ID 115).[122]

Das Protokoll *L2TP* enthält keine kryptographische Komponente; die Sicherung der IP-basierenden Kommunikation über das benutzte Transit-Netz mit Hilfe von *IPsec* wird in der Protokollspezifikation gefordert.[121]

L2TP in Verbindung mit *IPsec* ist clientseitig für viele Betriebssysteme verfügbar: für *Linux*, Microsoft *Windows*, Apple *MacOS/X* und für Betriebssysteme für Mobile Geräte wie Google *Android*, Apple *iOS* und Windows Phone; *L2TP* in Verbindung mit *IPsec* ist serverseitig für kommerzielle Routingplattformen und für Microsoft Windows Server und für Betriebssysteme der Unix-Betriebssystemfamilie wie *Linux*, *FreeBSD*, *NetBSD* und *OpenBSD* verfügbar.

L2TP findet Verwendung bei Internet Service Providern, um breitbandige Internet-Anschlüsse von Endkunden an bündelnde Netzelemente (Konzentratoren) anzubinden.

Die aktuelle Version der Protokollspezifikation von *L2TP* ist *Version 3* (*L2TPv3*).[122]

L2TPv3 eignet sich in Verbindung mit *IPsec* im Transportmodus (*transport mode*) und in Verbindung mit *IPsec* im Tunnelmodus (*tunnel mode)* zum Aufbau von Virtual Private Networks.[120,122,123]

---

[113] RFC 7296: Internet Key Exchange Protocol Version 2 (IKEv2), 2014, Standard, URL: `http://www.rfc-editor.org/rfc/rfc7296.txt` [86]

[114] RFC 7321: Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH), 2014, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc7321.txt` [87]

[115] RFC 7619: The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2), 2015, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc7619.txt` [91]

[116] RFC 7670: Generic Raw Public-Key Support for IKEv2, 2016, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc7670.txt` [92]

[117] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "OSI-Schichten, Namen und Nummern der -" [8, S. 601]

[118] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Tunneling, Tunnelung" [8, S. 863]

[119] Ethernet-Pakete auf OSI Layer 2 (*Data Link Layer*) im Datenformat nach dem Standard IEEE 802.3

[120] RFC 4719: Transport of Ethernet Frames over Layer 2 Tunneling Protocol Version 3 (L2TPv3), 2006, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4719.txt` [93]

[121] RFC 3931: Layer Two Tunneling Protocol - Version 3 (L2TPv3), 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3931.txt` [94, S. 20,21]

[122] RFC 3931: Layer Two Tunneling Protocol - Version 3 (L2TPv3), 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3931.txt` [94]

[123] RFC 3193: Securing L2TP using IPsec, 2001, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3193.txt` [95]

2.2.4.3 PPTP (Point-to-Point Tunneling Protocol)

Mit Hilfe des Protokolls *PPTP* (*Point-to-Point Tunneling Protocol*)[124] können zwischen zwei Computern über IP-basierende Transit-Netzwerke hinweg IP-basierende Datagramme transportiert werden; das Protokoll *PPTP* ist ein Tunnel-Protokoll, um das Protokoll IP eingekapselt über das Protokoll IP zu transportieren.

Der Pakettransport auf dem benutzten Transit-Netzwerk basiert auf dem TCP-Protokoll (TCP-Port 1723[125]) für die Kontrollverbindung und für die zu transportierenden Daten auf der *Generic Routing Encapsulation* (*GRE*)[126] des IP-Protokolls. *PPTP* nutzt zum Kapseln des zu transportierenden Datenverkehrs das Protokoll *PPP*[127,128,129,130].
Die Sicherheit bei der Authentifizierung basiert auf den Mechanismen, die vom Protokoll *PPP* vorgegeben sind,[127,128,129] eine ausreichende kryptographische Sicherung der zu transportierenden Daten ist nicht gegeben.[131]

Das *Point-to-Point Tunneling Protocol* wurde von einem Hersteller-Konsortium - darunter die Unternehmen Microsoft, Ascend, 3Com und ECI - entwickelt und ist in RFC 2637 aus dem Jahr 1999 spezifiziert.[132]

Der Einsatz von *PPTP* ist aufgrund sicherheitsspezifischer Schwächen nicht mehr sinnvoll.

2.2.5 Nicht-standardisierte VPN-Protokolle

Die folgenden Abschnitte stellen nicht-standardisierte VPN-Protokolle und deren Software-Implementierungen als Informationsgrundlage vor; vorgreifend auf das Kapitel 3 Analyse und auf das Kapitel 5 Konzeption werden relevante Eigenschaften der VPN-Protokolle und Software-Implementierungen dargestellt.

2.2.5.1 PPP over SSH

PPP over SSH ist eine einfache Möglichkeit zum Realisieren eines Virtual Private Networks. Mit *PPP over SSH* lässt sich zwischen Computern ein virtuelles, privates Netzwerk mit einer Punkt-zu-Punkt-Topologie aufbauen; über Punkt-zu-Punkt-Topologien hinausgehende Topologien wie z.B. Stern-Topologien sind zwar grundsätzlich möglich, müssen jedoch händisch eingerichtet werden.
Die kryptographischen Funktionen und die Tunnelungsfunktionen sind mit der Software *SSH* (*Secure Shell*)[133,134] realisiert, die Kapselung der Nutzdaten erfolgt mit dem Protokoll

[124] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "PPTP" [8, S. 646]
[125] RFC 2637: Point-to-Point Tunneling Protocol (PPTP), 1999, Informational, URL: `http://www.rfc-editor.org/rfc/rfc2637.txt` [96, S. 8]
[126] RFC 2637: Point-to-Point Tunneling Protocol (PPTP), 1999, Informational, URL: `http://www.rfc-editor.org/rfc/rfc2637.txt` [96, S. 9]
[127] RFC 1661: The Point-to-Point Protocol (PPP), 1994, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc1661.txt` [97]
[128] RFC 1663: PPP Reliable Transmission, 1994, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc1663.txt` [98]
[129] RFC 2153: PPP Vendor Extensions, 1997, Informational, URL: `http://www.rfc-editor.org/rfc/rfc2153.txt` [99]
[130] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "PPP" [8, S. 646]
[131] RFC 2637: Point-to-Point Tunneling Protocol (PPTP), 1999, Informational, URL: `http://www.rfc-editor.org/rfc/rfc2637.txt` [96, S. 54]
[132] RFC 2637: Point-to-Point Tunneling Protocol (PPTP), 1999, Informational, URL: `http://www.rfc-editor.org/rfc/rfc2637.txt` [96]
[133] Daniel J. Barrett und Richard E. Silverman: SSH, The Secure Shell: The Definitive Guide, 2001 [100]
[134] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "SSH" [8, S. 799]

PPP[127,128]; für die Verschlüsselung des Datenverkehrs wird als zugrunde liegender Algorithmus *AES*, *CAST*[135], *Arcfour* (*RC4*), *Blowfish*[136] oder *3DES* genutzt, die Authentifizierung nutzt *RSA*- oder *DSA*-Verschlüsselung. Eine zusätzliche, optionale Funktionalität ist die Kompression des Datenverkehrs, der zugrunde liegende Kompressionsalgorithmus ist *DEFLATE*,[137] welcher auch beim Unix-Kommandozeilenprogramm *gzip* verwendet wird.
SSH ist frei verfügbar; z.B. ist *OpenSSH*[138] eine frei zugängliche und frei verwendbare Implementierung von *SSH*. Der primäre Zweck von *SSH* ist es, für Clients eine gesicherte interaktive Terminalsitzung zu einem Server zu ermöglichen.

Der Pakettransport auf dem benutzten Transit-Netzwerk - üblicherweise dem Internet - basiert auf dem TCP-Protokoll (der Standard-Port von SSH ist TCP-Port 22, vom Standard-Port abweichende TCP-Ports sind wählbar).

Zum Realisieren der VPN-Funktionalität kommuniziert eine Client-Instanz von SSH auf einem Computer mit einer Server-Instanz von SSH auf einem anderen Computer; der Client nutzt die Funktionalität des SSH-Protokolls zum Erzeugen einer Terminalsitzung auf dem Server, in dieser Terminalsitzung erzeugen Client und Server eine PPP-Sitzung zur Kapselung und Übertragung der IP-Datagramme.

PPP over SSH lässt sich unter dem Betriebssystem *Linux* zum Realisieren von Virtual Private Networks nutzen, es existiert eine diesbezügliche Dokumentation.[139]

2.2.5.2 VTun

VTun ist eine frei zugängliche und frei verwendbare VPN-Software, die es ermöglicht, zwischen Computern ein virtuelles, privates Netzwerk mit einer Punkt-zu-Punkt-Topologie aufzubauen; über Punkt-zu-Punkt-Topologien hinausgehende Topologien wie z.B. Stern-Topologien sind zwar grundsätzlich möglich, müssen jedoch händisch eingerichtet werden.
VTun baut für die kryptographischen Funktionen auf der Softwarebibliothek *OpenSSL*[140] auf. Um die zusätzliche optionale Funktionalität der Kompression des Datenverkehrs zu realisieren, baut VTun auf den Softwarebibliotheken *zlib* und *LZO* auf.
Zum Realisieren der VPN-Funktionalität kommuniziert eine Client-Instanz von *VTun* auf einem Computer mit einer Server-Instanz von *VTun* auf einem anderem Computer.
Der Pakettransport zwischen einer Client-Instanz von *VTun* und einer Server-Instanz von *VTun* auf dem benutzten Transit-Netz - üblicherweise dem Internet - basiert wahlweise auf dem TCP-Protokoll oder auf dem UDP-Protokoll.
Die Verschlüsselung des Datenverkehrs wird mit dem AES-Verschlüsselungsverfahren oder dem Blowfish-Verschlüsselungsverfahren bewerkstelligt, die Funktionalität der Softwarebibliothek *OpenSSL*[140] wird hierfür genutzt; Client-Instanz und Server-Instanz von *VTun* nutzen zu diesem Zweck einen gemeinsamen 128 Bit oder 256 Bit langen symmetrischen Schlüssel. Dieser genutzte symmetrische Schlüssel wird aus einem in einer Konfigurationsdatei hinterlegten gemeinsamen Passwort hergeleitet.

---

[135] Henk C.A. van Tilborg, Hrsg.: Encyclopedia of Cryptography and Security, Eintrag "CAST" [101, S. 62-63]
[136] Henk C.A. van Tilborg, Hrsg.: Encyclopedia of Cryptography and Security, Eintrag "Blowfish" [101, S. 48]
[137] Daniel J. Barrett und Richard E. Silverman: SSH, The Secure Shell: The Definitive Guide, 2001 [100, S. 56]
[138] *OpenSSH* ist eine freie - client- wie serverseitige - Implementierung des Protokolls *SSH* und wurde primär für das Betriebssystem OpenBSD entwickelt; *OpenSSH* ist unter vielen Betriebssystemen der Unix-Betriebssystemfamilie wie *OpenBSD*, *NetBSD*, *FreeBSD*, *Linux* oder Apple *MacOS/X* lauffähig, URL: `https://www.openssh.com/` [102].
[139] Scott Bronson: VPN PPP-SSH Mini-HOWTO, 2002, URL: `http://www.tldp.org/HOWTO/pdf/ppp-ssh.pdf` [103]
[140] OpenSSL Software Foundation: OpenSSL Cryptography and SSL/TLS Toolkit, URL: `https://www.openssl.org/` [104]

VTun ist verfügbar für eine Vielzahl von Betriebssystemen; die Betriebssysteme *Linux*, *Solaris*, *FreeBSD*, *OpenBSD*, *NetBSD* sowie andere Abkömmlinge der *BSD*-Betriebssystemfamilie werden von *VTun* direkt unterstützt, d.h. der Quellcode ist - ohne dass weitere Anpassungen notwendig sind - auf diesen Betriebssystemen übersetzbar und funktional lauffähig.
Um die VPN-Funktionalität mittels *VTun* zu realisieren, wird auf der Ebene des jeweils eingesetzten Betriebssystems Software für die Bereitstellung einer generischen, virtuellen Netzwerkschnittstelle - Treibersoftware für generische *tap*-Devices bzw. generische *tun*-Devices - benötigt; diese ist verfügbar auf der Internetseite "Universal TUN/TAP driver" unter der URL `http://vtun.sourceforge.net/tun/index.html`.[141,142]
Die aktuelle Version der Software ist Version 3.0.4 vom 18.09.2016 (Stand 29.06.2017).[142]
Informationen zur Funktionalität von *VTun* und Dokumentation zu *VTun* sind auf der Internetseite "VTun - Virtual Tunnels over TCP/IP networks" unter der URL `http://vtun.sourceforge.net/` abrufbar.[142]
Der Quellcode der Software ist auf der Internetseite "Download VTun" unter der URL `http://vtun.sourceforge.net/download.html` abrufbar;[142,143] des Weiteren ist der Quellcode des aktuellen Entwicklungszweiges der Software in der Version 3.0.4-1 vom 19.03.2017 (Stand 29.06.2017) auf der Internetseite "VTun - Virtual Tunnels download | SourceForge.net" unter der URL `https://sourceforge.net/projects/vtun/` abrufbar.[142]

2.2.5.3 vpnd

vpnd ist eine frei zugängliche und frei verwendbare VPN-Software, die es ermöglicht, zwischen Computern ein virtuelles, privates Netzwerk mit einer Punkt-zu-Punkt-Topologie aufzubauen; über Punkt-zu-Punkt-Topologien hinausgehende Topologien wie z.B. Stern-Topologien sind zwar grundsätzlich möglich, müssen jedoch händisch eingerichtet werden.
Zum Realisieren der VPN-Funktionalität kommuniziert eine Client-Instanz von *vpnd* auf einem Computer mit einer Server-Instanz von *vpnd* auf einem anderem Computer.
Der Pakettransport zwischen den teilnehmenden Computern auf dem benutzten Transit-Netz - üblicherweise dem Internet - basiert auf dem TCP-Protokoll.
Zur Verschlüsselung des Datenverkehrs wird ein Schlüssel genutzt, den Client-Instanz und Server-Instanz teilen; als Verschlüsselungsalgorithmus wird *Blowfish* eingesetzt.
vpnd wurde primär entwickelt für die Betriebssysteme *Linux* und *FreeBSD*. Auf den Betriebssystemen *Linux*, *FreeBSD*, *NetBSD*, *OpenBSD* und *Solaris* ist die Software ohne Modifikationen des Betriebssystemkerns lauffähig und funktional nutzbar. Zur Bereitstellung der virtuellen Netzwerkschnittstelle nutzt *vpnd* ein *PPP*-Interface oder ein *SLIP*-Interface des Betriebssystems, unter dem Betriebssytem *Solaris* wird für die Bereitstellung der virtuellen Netzwerkschnittstelle ein *PPP*-Interface[144] genutzt.
Die aktuelle Version der Software ist Version 1.1.4 vom 30.12.2007 (Stand 05.08.2017).
Der Quellcode der Software ist unter der URL `http://www.mirrorservice.org/sites/ftp.wiretapped.net/pub/security/network-security/vpnd/vpnd-1.1.4.tar.gz` abruf-

[141] Maxim Krasnyansky: Universal TUN/TAP driver - Virtual Point-to-Point(TUN) and Ethernet(TAP) devices, URL: `http://vtun.sourceforge.net/tun/index.html` [105]
[142] Maxim Krasnyansky: VTun - Virtual Tunnels over TCP/IP networks, URL: `http://vtun.sourceforge.net/` [106]
[143] Maxim Krasnyansky: VTun - Virtual Tunnels over TCP/IP networks - Download VTun, URL: `http://vtun.sourceforge.net/download.html` [107]
[144] RFC 1661: The Point-to-Point Protocol (PPP), 1994, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc1661.txt` [97]

bar.[145]
Die ursprüngliche Projektseite zu *vpnd* auf der Internetseite "Virtual Private Network Daemon" unter der URL `http://sunsite.dk/vpnd/` ist nicht mehr verfügbar, eine archivierte Version der Seite mit dem Stand vom 30.08.2005 ist jedoch unter der URL `http://web.archive.org/web/20050830024341/http://sunsite.dk/vpnd/` abrufbar.[146]
Hervorhebenswert ist, dass eine Client-Instanz von *vpnd* eine indirekte Verbindung über einen *SOCKS*-Proxyserver[147,148] zu einer Server-Instanz von *vpnd* herstellen kann; diese Funktionalität ist ab Version 1.1.1 der Software gegeben.

2.2.5.4 n2n

n2n ist eine frei zugängliche und frei verwendbare VPN-Software, die es ermöglicht, zwischen Computern ein virtuelles, privates Netzwerk aufzubauen.
Die Erzeugung und Aufrechterhaltung der virtuellen Netzwerktopologie wird von zwei redundanten, zentralen Netzwerk-Knoten koordiniert, die Supernodes genannt werden.[149]
n2n baut für die kryptographischen Funktionen auf der Softwarebibliothek *OpenSSL*[150] auf. Der Pakettransport zwischen den teilnehmenden Computern auf dem benutzten Transit-Netz - üblicherweise dem Internet - basiert auf dem UDP-Protokoll, für den Pakettransport der beteiligten Computer mit den Supernodes kommt als Protokoll das UDP-Protokoll oder auch alternativ das TCP-Protokoll zum Einsatz. Zur Verschlüsselung des Datenverkehrs wird ein Schlüssel genutzt, den alle beteiligten Computer teilen; als Verschlüsselungsalgorithmus wird *Twofish* eingesetzt.[151]
n2n ist primär verfügbar für die Betriebssysteme *Linux*, Apple *MacOS/X* und Microsoft *Windows*; des Weiteren werden die Betriebssysteme *FreeBSD*, *NetBSD* und das auf dem Betriebssystem *Linux* basierende Router-Betriebssystem *OpenWrt* unterstützt.
Auf den Betriebssystemen *Linux*, *FreeBSD* und *NetBSD* ist die Software ohne Modifikationen des Betriebssystemkerns lauffähig und funktional nutzbar; das fußt auf dem Umstand, dass für die Bereitstellung der Funktionalität der benötigten virtuellen Ethernet-Schnittstelle generische *tap*-Devices verwendet werden,[152] die bereits in diesen Betriebssystemen enthalten sind.
Für die Betriebssysteme Microsoft *Windows* und Apple *MacOS/X* ist *n2n* verfügbar, bei diesen Betriebssystemen wird jedoch zusätzlich Software für die Bereitstellung der benötigten generischen, virtuellen Netzwerkschnittstelle - Treibersoftware für generische *tap*-Devices - benötigt; diese ist verfügbar.
Die aktuelle Version der Software ist Version 2.1.0 (Stand 19.06.2017); *n2n* wird derzeit nicht weiterentwickelt.
Der Quellcode der Software ist auf der Internetseite "GitHub - ntop/n2n: Peer-to-peer VPN" unter der URL `https://github.com/ntop/n2n` abrufbar; des Weiteren ist der Quellcode der Software auf der Internetseite "ntop - Revision 9349: /trunk/n2n/n2n_v2" unter

[145] University of Kent, School of Computing: UK Mirror Service, URL: `http://www.mirrorservice.org/` [108]
[146] Internet Archive: Wayback Machine, URL: `https://web.archive.org/` [109]
[147] *SOCKS* ist ein Protokoll für *Circuit Level Proxies*.[148]
[148] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "SOCKS" [8, S. 779]
[149] Luca Deri und Richard Andrews: N2N: A Layer Two Peer-to-Peer VPN, 2008, URL: `http://luca.ntop.org/n2n.pdf` [110, S. 5]
[150] OpenSSL Software Foundation: OpenSSL Cryptography and SSL/TLS Toolkit, URL: `https://www.openssl.org/` [104]
[151] Luca Deri und Richard Andrews: N2N: A Layer Two Peer-to-Peer VPN, 2008, URL: `http://luca.ntop.org/n2n.pdf` [110, S. 7]
[152] Luca Deri und Richard Andrews: N2N: A Layer Two Peer-to-Peer VPN, 2008, URL: `http://luca.ntop.org/n2n.pdf` [110, S. 6]

der URL `https://svn.ntop.org/svn/ntop/trunk/n2n/n2n_v2/` abrufbar.
Es existiert ein Entwicklungszweig der Software, in welchem anstatt der Softwarebibliothek *OpenSSL*[150] die Softwarebibliothek *PolarSSL*[153] für die Bereitstellung der kryptographischen Funktionen genutzt wird; der Quellcode dieses Entwicklungszweigs der Software ist auf der Internetseite "GitHub - lukablurr/n2n_v2_fork" unter der URL `https://github.com/lukablurr/n2n_v2_fork/` abrufbar.
Die ursprüngliche Projektseite zu *n2n* auf der Internetseite "n2n - a Layer Two Peer-to-Peer VPN" unter der URL `http://www.ntop.org/products/n2n/` ist nicht mehr verfügbar, eine archivierte Version der Seite mit dem Stand vom 08.04.2015 ist jedoch unter der URL `http://web.archive.org/web/20150408190359/http://www.ntop.org/products/n2n/` abrufbar.[154]
Informationen zur Funktionalität von *n2n* und Dokumentation zu *n2n* sind im Dokument "N2N: A Layer Two Peer-to-Peer VPN" von der Autoren der Software - Luca Deri und Richard Andrews - unter der URL `http://luca.ntop.org/n2n.pdf` abrufbar.[155]

2.2.5.5 OpenVPN

OpenVPN[156] ist eine frei zugängliche und frei verwendbare VPN-Software, die es ermöglicht, zwischen Computern ein virtuelles, privates Netzwerk aufzubauen.
Mit *OpenVPN* ist die Realisierung eines virtuellen, privaten Netzwerks mit einer Punkt-zu-Punkt-Topologie möglich;[157] auch eine Stern-Topologie mit zentralem VPN-Server ist möglich.[158]
Bei der VPN-Funktionalität mit einer Punkt-zu-Punkt-Topologie kommuniziert eine Client-Instanz von *OpenVPN* auf einem Computer mit einer Server-Instanz von *OpenVPN* auf einem anderem Computer;[157] bei der VPN-Funktionalität mit einer Stern-Topologie kommunizieren Client-Instanzen von *OpenVPN* auf mehreren Computern mit *einer* Server-Instanz von *OpenVPN* auf einem zentralen VPN-Server.[158]
Mit *OpenVPN* sind Virtual Private Networks möglich, die Netzwerke auf OSI Layer 3 - auf IP-Ebene (*Network Layer*[159]) - miteinander verbinden (*routed VPN*) und es sind alternativ dazu Virtual Private Networks möglich, die Netzwerke auf OSI Layer 2 - auf Ethernet-Ebene (*Data Link Layer*[159]) - miteinander verbinden (*bridged VPN*).[160]
Als Transportprotokoll kommt bei *OpenVPN* das *SSL*-Protokoll bzw. *TLS*-Protokoll[161] zum Einsatz.

OpenVPN baut auf der Softwarebibliothek *OpenSSL*[162] für die kryptographischen Funktionen auf, alternativ kann zu diesem Zweck die Softwarebibliothek *PolarSSL*[163] genutzt werden.[164] Um die zusätzliche optionale Funktionalität der Kompression des Datenver-

[153] ARM Limited: SSL Library mbed TLS / PolarSSL, URL: `http://polarssl.org/` [111]
[154] Internet Archive: Wayback Machine, URL: `https://web.archive.org/` [109]
[155] Luca Deri und Richard Andrews: N2N: A Layer Two Peer-to-Peer VPN, 2008, URL: `http://luca.ntop.org/n2n.pdf` [110]
[156] OpenVPN Inc.: OpenVPN - Open Source VPN, URL: `https://openvpn.net/` [112]
[157] Jan Just Keijser: OpenVPN 2 Cookbook, 2011 [113, S. 22,23]
[158] Jan Just Keijser: OpenVPN 2 Cookbook, 2011 [113, S. 38,39]
[159] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "OSI-Schichten, Namen und Nummern der -" [8, S. 601]
[160] OpenVPN Inc.: HOWTO - Determining whether to use a routed or bridged VPN, URL: `https://openvpn.net/index.php/open-source/documentation/howto.html#vpntype` [114]
[161] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2, 2008, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc5246.txt` [115]
[162] OpenSSL Software Foundation: OpenSSL Cryptography and SSL/TLS Toolkit, URL: `https://www.openssl.org/` [104]
[163] ARM Limited: SSL Library mbed TLS / PolarSSL, URL: `http://polarssl.org/` [111]
[164] OpenVPN Inc.: OpenVPN - Installation Notes,

kehrs zu realisieren, baut *OpenVPN* auf der Softwarebibliothek *LZO*[165] auf.[164]
Zur Verschlüsselung des Datenverkehrs existiert sowohl die Möglichkeit feste - statische - Schlüssel einzusetzen als auch die Möglichkeit Zertifikat-basierte Public-Key-Verschlüsselung einzusetzen;[166] als Verschlüsselungsalgorithmen stehen die Verschlüsselungsalgorithmen der Softwarebibliothek *OpenSSL* bzw. *PolarSSL* zur Auswahl.[166] Der Verschlüsselungsalgorithmus *Blowfish* ist der verwendete Standardverschlüsselungsalgorithmus, wenn keine spezifische Auswahl aus den zur Verfügung stehenden Algorithmen getroffen wird.[167]
Der Pakettransport zwischen den teilnehmenden Computern auf dem benutzten Transit-Netz - üblicherweise dem Internet - basiert auf dem TCP-Protokoll oder auf dem UDP-Protokoll.[168] Der Standard-Port von *OpenVPN* ist TCP-Port 1194 bzw. UDP-Port 1194, ein abweichender TCP-Port bzw. UDP-Port ist wählbar.[169]
OpenVPN ist verfügbar für eine Vielzahl von Betriebssystemen; die Betriebssysteme *Linux*, *OpenBSD*, *FreeBSD*, *NetBSD*, Apple *MacOS/X* sowie das Betriebsystem *Solaris* werden von *OpenVPN* direkt unterstützt,[170] d.h. der Quellcode ist - ohne dass weitere Anpassungen notwendig sind - auf diesen Betriebssystemen übersetzbar und funktional lauffähig. Für das Betriebssystem Microsoft *Windows* ist *OpenVPN* verfügbar,[170] bei diesem Betriebssystem wird jedoch zusätzlich Software für die Bereitstellung einer generischen, virtuellen Netzwerkschnittstelle - Treibersoftware für generische *tap*-Devices - benötigt; diese ist verfügbar.[164,171,172]
Die aktuelle Version der Software ist Version 2.4.3 vom 21.06.2017 (Stand 06.08.2017);[172] es existiert ein Entwicklungszweig der Software mit der Versionsnummer 2.3.17 vom 21.06.2017 (Stand 06.08.2017), die mit dem Label *old stable* gekennzeichnet ist.[172] Es sind für mehrere Betriebssystemplattformen vorbereitete Pakete[173] von *OpenVPN* verfügbar, z.B. für die Open-Source-Betriebssysteme *Debian Linux*, *Ubuntu Linux*, *Fedora Linux*, *FreeBSD* und *Maemo*.[172]

Hervorhebenswert ist, dass eine Client-Instanz von *OpenVPN* eine indirekte Verbindung über einen *SOCKS*-Proxyserver[174,175] oder eine indirekte Verbindung über einen *HTTP*-Proxyserver zu einer Server-Instanz von *OpenVPN* herstellen kann.[176]

URL: `https://openvpn.net/index.php/open-source/documentation/install.html` [116]

[165] Markus F.X.J. Oberhumer, oberhumer.com GmbH: LZO real-time data compression library, URL: `http://www.oberhumer.com/opensource/lzo/` [117]

[166] OpenVPN Inc.: OpenVPN - What is OpenVPN?, URL: `https://openvpn.net/index.php/open-source/333-what-is-openvpn.html` [118]

[167] OpenVPN Inc.: HOWTO - Hardening OpenVPN Security, URL: `https://openvpn.net/index.php/open-source/documentation/howto.html#security` [119]

[168] OpenVPN Inc.: OpenVPN - Release Notes, URL: `https://openvpn.net/index.php/open-source/documentation/release-notes.html` [120]

[169] OpenVPN Inc.: OpenVPN - HOWTO - Creating configuration files for server and clients, URL: `https://openvpn.net/index.php/open-source/documentation/howto.html#config` [121]

[170] OpenVPN Inc.: OpenVPN - Community Project Overview, URL: `https://openvpn.net/index.php/open-source/245-community-open-source-software-overview.html` [122]

[171] GitHub Inc.: GitHub - OpenVPN/tap-windows: Windows TAP driver, URL: `https://github.com/OpenVPN/tap-windows` [123]

[172] OpenVPN Inc.: OpenVPN - Downloads, URL: `https://openvpn.net/index.php/open-source/downloads.html` [124]

[173] Diese vorbereiteten Pakete werden *packages* bzw. beim Betriebssystem FreeBSD *ports* genannt.

[174] *SOCKS* ist ein Protokoll für Circuit Level Proxies.[175]

[175] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "SOCKS" [8, S. 779]

[176] Jan Just Keijser: OpenVPN 2 Cookbook, 2011 [113, S. 294,297,300]

2.2.5.6 Tinc VPN

Tinc VPN ist eine frei zugängliche und frei verwendbare VPN-Software, die es ermöglicht, zwischen Computern ein virtuelles, privates Netzwerk mit einer Full-Mesh-Topologie aufzubauen.[177]
Mit *Tinc VPN* sind Virtual Private Networks möglich, die Netzwerke auf OSI Layer 3 - auf IP-Ebene (*Network Layer*[178]) - miteinander verbinden und es sind alternativ dazu Virtual Private Networks möglich, die Netzwerke auf OSI Layer 2 - auf Ethernet-Ebene (*Data Link Layer*[178]) - miteinander verbinden (*bridging Ethernet segments*).[179]
Tinc VPN baut für die kryptographischen Funktionen auf der Softwarebibliothek *LibreSSL*[180] oder alternativ auf der Softwarebibliothek *OpenSSL*[181] auf. Um die zusätzliche optionale Funktionalität der Kompression des Datenverkehrs zu realisieren, baut *Tinc VPN* auf den Softwarebibliotheken *zlib* und *LZO* auf.[177]
Der Pakettransport auf dem benutzten Transit-Netz - üblicherweise dem Internet - basiert für die Kontrollverbindungen zwischen den einzelnen verteilten Instanzen von *Tinc VPN* auf dem TCP-Protokoll[182] und für den Nutzdatentransport zwischen den einzelnen verteilten Instanzen von *Tinc VPN* auf dem UDP-Protokoll.[183] Der Standard-Port von *Tinc VPN* ist TCP-Port 655 bzw. UDP-Port 655.[184]
Die Verschlüsselung des Datenverkehrs wird mit dem Verschlüsselungsalgorithmus *Blowfish* bewerkstelligt, die Funktionalität der Softwarebibliothek *LibreSSL*[180] oder alternativ der Softwarebibliothek *OpenSSL*[181] wird hierfür genutzt; sich miteinander verbindende Instanzen der Software nutzen zur Authentifizierung jeweils ein Public-Key-/Private-Key-Schlüsselpaar nach dem *RSA*-Verfahren.[185]
Tinc VPN ist verfügbar für eine Vielzahl von Betriebssystemen; die Betriebssysteme *Linux*, *FreeBSD*, *OpenBSD*, *NetBSD*, *DragonFlyBSD* sowie die *sparc32*-Architektur des Betriebsystems *Solaris* werden von *Tinc VPN* direkt unterstützt, d.h. der Quellcode ist - ohne dass weitere Anpassungen notwendig sind - auf diesen Betriebssystemen übersetzbar und funktional lauffähig.[186] Für die Betriebssysteme Microsoft *Windows* und Apple *MacOS/X* ist *Tinc VPN* verfügbar, bei diesen Betriebssystemen wird jedoch zusätzlich Software für die Bereitstellung einer generischen, virtuellen Netzwerkschnittstelle - Treibersoftware für generische *tap*-Devices bzw. generische *tun*-Devices - benötigt; diese ist verfügbar.
Die aktuelle Version der Software ist Version 1.0.31 (Stand 08.08.2017).[177,187]
Informationen zur Funktionalität von *Tinc VPN* und Dokumentation zu *Tinc VPN* sind auf der Internetseite "Tinc VPN" unter der URL `http://www.tinc-vpn.org/` abrufbar.[177]
Auch für das Betriebssystem Google *Android* sowie für das Betriebssystem Apple *iOS* ist

[177] Tinc VPN - tinc, URL: `http://www.tinc-vpn.org/` [125]
[178] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "OSI-Schichten, Namen und Nummern der -" [8, S. 601]
[179] Tinc VPN - Example: bridging Ethernet segments using tinc under Linux, URL: `http://www.tinc-vpn.org/examples/bridging/` [126]
[180] LibreSSL, URL: `https://www.libressl.org/` [127]
[181] OpenSSL Software Foundation: OpenSSL Cryptography and SSL/TLS Toolkit, URL: `https://www.openssl.org/` [104]
[182] Ivo Timmermans und Guus Sliepen: tinc Manual, 2016, URL: `http://www.tinc-vpn.org/documentation/tinc.pdf` [128, S. 32]
[183] Ivo Timmermans und Guus Sliepen: tinc Manual, 2016, URL: `http://www.tinc-vpn.org/documentation/tinc.pdf` [128, S. 31]
[184] Ivo Timmermans und Guus Sliepen: tinc Manual, 2016, URL: `http://www.tinc-vpn.org/documentation/tinc.pdf` [128, S. 8]
[185] Ivo Timmermans und Guus Sliepen: tinc Manual, 2016, URL: `http://www.tinc-vpn.org/documentation/tinc.pdf` [128, S. 34,35]
[186] Tinc VPN - Supported platforms, URL: `http://www.tinc-vpn.org/platforms/` [129]
[187] Tinc VPN - Download, URL: `http://www.tinc-vpn.org/download/` [130]

Tinc VPN verfügbar.[186]

2.2.5.7 PeerVPN

PeerVPN ist eine frei zugängliche und frei verwendbare VPN-Software, die es ermöglicht, zwischen Computern ein virtuelles, privates Netzwerk mit einer Full-Mesh-Topologie aufzubauen.[188]
Mit *PeerVPN* sind Virtual Private Networks möglich, die Netzwerke auf OSI Layer 3 - auf IP-Ebene (*Network Layer*[189]) - miteinander verbinden und es sind alternativ dazu Virtual Private Networks möglich, die Netzwerke auf OSI Layer 2 - auf Ethernet-Ebene (*Data Link Layer*[178]) - miteinander verbinden (*Ethernet tunneling support*).[188]
PeerVPN baut für die kryptographischen Funktionen auf der Softwarebibliothek *LibreSSL*[190] oder alternativ auf der Softwarebibliothek *OpenSSL*[191] auf. Bei der Verwendung von OpenSSL als unterstützende Softwarebibliothek ist aus Kompatibilitätsgründen zwingend eine Version von *OpenSSL vor* Version 1.1.x erforderlich.[188]
Der Pakettransport auf dem benutzten Transit-Netz - üblicherweise dem Internet - basiert auf dem UDP-Protokoll.[192] Die Verschlüsselung des Datenverkehrs nutzt einen Schlüssel, den alle beteiligten Computer teilen (PSK).[188,192]
PeerVPN ist verfügbar für die Betriebssysteme *Linux* und *FreeBSD*, d.h. der Quellcode ist - ohne dass weitere Anpassungen notwendig sind - auf diesen beiden Betriebssystemen übersetzbar.[188]
Die Software ist auf den beiden Betriebssystemen *Linux* und *FreeBSD* ohne Modifikationen des Betriebssystemkerns lauffähig und funktional; das fußt auf dem Umstand, dass für die Bereitstellung der Funktionalität der benötigten virtuellen Ethernet-Schnittstelle generische *tap*-Devices verwendet werden.[188]
Der Quellcode der Software - im Detail: die Quellcode-Datei `platform/ifconfig.c` - legt den Schluss nahe, dass an einer Portierung auf das Betriebssystem Microsoft *Windows* gearbeitet wird.[193]
Die aktuelle Version der Software ist Version 0.044 vom 21.03.2016 (Stand 08.08.2017);[188] die Versionsnummerierung legt den Schluss nahe, dass die Software in einer frühen Phase des Systemlebenszyklus ist. Es ist nicht eindeutig, ob die Software in der Entwicklungsphase oder in der Nutzungsphase des Systemlebenszyklus ist.
Informationen zur Funktionalität von *PeerVPN* und Dokumentation zu *PeerVPN* sind auf der Internetseite "PeerVPN - the open source peer-to-peer VPN" unter der URL `https://peervpn.net/` abrufbar.[188]

2.2.6 De-facto-Standards bei VPN-Protokollen

VPN-Protokolle lassen sich einteilen in durch Normierungs- und Standardisierungsgremien standardisierte VPN-Protokolle (*de jure*)[194] (siehe Abschnitt 2.2.4 Standardisierte VPN-Protokolle) und in VPN-Protokolle, die durch ihren weitverbreiteten praktischen Einsatz faktisch zum Standard geworden sind (*de facto*)[194].

[188] PeerVPN - the open source peer-to-peer VPN, URL: `https://peervpn.net/` [131]
[189] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "OSI-Schichten, Namen und Nummern der -" [8, S. 601]
[190] LibreSSL, URL: `https://www.libressl.org/` [127]
[191] OpenSSL Software Foundation: OpenSSL Cryptography and SSL/TLS Toolkit, URL: `https://www.openssl.org/` [104]
[192] PeerVPN - Tutorial, URL: `https://peervpn.net/tutorial/` [132]
[193] PeerVPN - Download source code, URL: `https://peervpn.net/files/peervpn-0-044.tar.gz` [133]
[194] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 76]

2.2.6.1 IPsec

IPsec ist ein Standard bei VPN-Protokollen durch die Standardisierung und Spezifikation von *IPsec* durch die *IETF* (*Internet Engineering Task Force*)[195];[196,197,198,199,200,201] die *IETF* hat die Autorität zur Normierung und ist Standardisierungsgremium. Ein *RFC* (*Request for Comment*)[202] ist eine Form der Veröffentlichung durch die *IETF* mit dem Zweck, diese Funktion zu erfüllen.

Diese De-jure-Standardisierung von *IPsec* hat durch die Definition der Spezifikationen die Grundlage für Interoperabilität unterschiedlicher Implementierungen von *IPsec* geschaffen und damit *IPsec* als VPN-Protokoll verbreitet. *IPsec* ist auf vielen Betriebssystemen verfügbar und hat breite Unterstützung von Herstellern von Routingplattformen (siehe Abschnitt 2.1.3 Routing-Software im Internet). *IPsec* ist ein De-jure-Standard bei VPN-Protokollen und zugleich auch implizit ein De-facto-Standard bei VPN-Protokollen.

2.2.6.2 OpenVPN

OpenVPN ist durch weite Verbreitung ein De-facto-Standard bei VPN-Protokollen; sowohl in Unternehmen als auch bei Einzelnen ist *OpenVPN* eines der weltweit verbreitetsten Pakete (*packages*) zur Realisierung von Virtual Private Networks.[203] Die Hauptgründe für die Verbreitung von *OpenVPN* sind dessen Einfachheit in der Anwendung, dessen geringere Komplexität im Vergleich mit *IPsec* sowie die Verwendung des SSL-Protokolls als zugrunde liegendes Transportprotokoll.[204]

2.3 Overlay-Netzwerke

Ein *Overlay-Netzwerk* ist ein Netzwerk, das auf einem anderen Netzwerk - welches die darunterliegende Transportinfrastruktur bildet - aufbaut. Das darunterliegende Netzwerk, auf dem aufgebaut wird, wird *Underlay* genannt; das darüberliegende logische Netzwerk selbst wird *Overlay* genannt.[205]

[195] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Internet Engineering Task Force" [8, S. 415]
[196] RFC 4301: Security Architecture for the Internet Protocol, 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4301.txt` [88]
[197] RFC 4303: IP Encapsulating Security Payload (ESP), 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4303.txt` [89]
[198] RFC 4305: Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH), 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4305.txt` [90]
[199] RFC 3193: Securing L2TP using IPsec, 2001, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3193.txt` [95]
[200] RFC 3931: Layer Two Tunneling Protocol - Version 3 (L2TPv3), 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3931.txt` [94]
[201] RFC 4719: Transport of Ethernet Frames over Layer 2 Tunneling Protocol Version 3 (L2TPv3), 2006, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4719.txt` [93]
[202] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "RFC" [8, S. 705]
[203] Jan Just Keijser: OpenVPN 2 Cookbook, 2011 [113, S. 1]
[204] Charlie Hosner: OpenVPN and the SSL VPN Revolution, SANS Institute, 2004, URL: `https://www.sans.org/reading-room/whitepapers/vpns/openvpn-ssl-vpn-revolution-1459` [134, S. 5]
[205] Thomas D. Nadeau und Ken Gray: SDN: Software Defined Networks, 2013 [79, S. 11,34]

MPLS (*Multiprotocol Label Switching*) in der Telekommunikation und im Bereich der Internet Service Provider erzeugt Overlay-Netzwerke; in *MPLS*-Netzwerken[206] wird aufbauend auf IP-Netzwerken als Underlay ein Netzwerk zum generischen Pakettransport als Overlay erzeugt.[207]

Multicast[208] in IP-Netzwerken - z.B. dem Internet - erzeugt Overlay-Netzwerke; das bei *Multicast* erzeugte Overlay dient dem Versenden von IP-Datagrammen von einem Sender an mehrere Empfänger (Ein-zu-Mehrpunkt-Verbindung) oder von mehreren Sendern an mehrere Empfänger (Mehrpunkt-zu-Mehrpunkt-Verbindung); Gruppen von Sendern und Empfängern sind definierbar, das Underlay ist ein IP-Netzwerk. *Internet Multicast* ist *Multicast* über das Internet als Underlay;[209] eine Möglichkeit zum Bewerkstelligen von *Internet Multicast* stellt das Protokoll *IGMP* (*Internet Group Management Protocol*)[210] in Verbindung mit den Multiprotokoll-Erweiterungen des BGP-Protokolls[211] (siehe Abschnitt 2.1.2.1) zur Verfügung.

Peer-to-Peer-File-Sharing-Systeme nutzen ein Peer-to-Peer-Overlay - bestehend aus Computern mit Speicherplatz als Ressource -, aufbauend auf einem IP-Netzwerk - dem Internet - als Underlay.[212]

Structured Search Overlays sind Overlay-Netzwerke, die aus Computern mit Speicherplatz und mit Routing-Funktionalität bestehen und die genutzt werden, um im Overlay gespeicherte Datenobjekte zu suchen und aufzufinden; ein *Structured Search Overlay* hat eine vordefinierte Topologie für die teilnehmenden Computer mit einer Routing-Tabelle (*routing table*) als Datenstruktur.[213] Eine Suche mit und in einem *Structured Search Overlay* wird bewerkstelligt, indem die Suchanfrage mit Hilfe einer *Hash Key*-Funktion - z.B. der kryptographischen Einwegfunktion[214] (*one-way function*) *MD5* oder der kryptographischen Einwegfunktion[214] *SHA-1* - in einen *Hash Key* transformiert wird. Dieser *Hash Key* wird mit einer Routing-Funktion über das *Structured Search Overlay* versandt; das passende, zu suchende Datenobjekt besitzt als Eigenschaft den entsprechenden *Hash Key* und wird über diesen *Hash key* gefunden.[213] Aufgrund dieser Funktionsweise wird eine solche Overlay-Struktur *Distributed Hash Table* (*DHT*) genannt.[215]

Chord/DHash[216], *Kademlia*[217] und *Pastry*[218] sind Implementierungen von *Structured*

[206] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 470,471]
[207] Thomas D. Nadeau und Ken Gray: SDN: Software Defined Networks, 2013 [79, S. 34]
[208] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Multicast" [8, S. 553]
[209] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 485]
[210] RFC 3376: Internet Group Management Protocol, Version 3, 2002, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3376.txt` [135]
[211] RFC 4760: Multiprotocol Extensions for BGP-4, 2007, Draft Standard, URL: `http://www.rfc-editor.org/rfc/rfc4760.txt` [12]
[212] Kamill Panitzek und Thorsten Strufe: A Formal Model for Peer-to-Peer Systems, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [136][137, S. 15-18]
[213] Christian Groß, Björn Richerzhagen und Max Lehn: Structured Search Overlays, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [138, S. 50][137, S. 49-67]
[214] Wolfgang Ertel: Angewandte Kryptographie, 2007 [139, S. 101]
[215] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 753]
[216] Chord Project: Overview, URL: `https://pdos.csail.mit.edu/chord/` [140]
[217] Petar Maymounkov und David Mazières: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, In: LNCS (Lecture Notes in Computer Science): Peer-to-Peer Systems: IPTPS 2002 (International Workshop on Peer-to-Peer Systems), 2002 [141][142, 53–65]
[218] Pastry Project: A scalable, decentralized, self-organizing and fault-tolerant substrate for peer-to-peer applications, URL: `http://www.freepastry.org/` [143]

Search Overlays[219] als *Distributed Hash Table* (*DHT*). *CAN*[220], *Freenet*[221,222] und Oceanstore/*Tapestry*[223] sind ebenso Overlay-Netzwerke, die als *Distributed Hash Table*-System implementiert sind.[218]
Peer-to-Peer-Overlay-Netzwerke, die auf der Struktur eines *Distributed Hash Table* (*DHT*) aufbauen, werden in die Gruppe der strukturierten Peer-to-Peer-Netzwerke (*structured P2P networks*) eingeordnet.[224]

Service Overlays sind Overlay-Netzwerke, die aus Computern mit Speicherplatz, mit Rechenleistung und mit Routing-Funktionalität bestehen und die sowohl genutzt werden, um im Overlay gespeicherte Datenobjekte zu suchen und aufzufinden und die auch genutzt werden, um im Overlay befindliche ausführbare Objekte (*service objects*) entfernt aufzurufen und auszuführen.[225] *Tapestry* ist eine Overlay-Netzwerk-Implementierung, welche als *Service Overlay* nutzbar ist.[223]

Content Delivery Overlays (CDOs) sind Overlay-Netzwerke, die aus Computern mit Speicherplatz als Ressource bestehen und die genutzt werden, um den Transport auszuliefernder Inhalte (*content*) an Nutzer zu organisieren;[226] bei Content Delivery Overlays (CDOs) kann das Internet als Underlay dienen, es kann auch ein nicht-öffentliches IP-Netzwerk als Underlay dienen. Ein typisches Beispiel für die Implementierung eines *Content Delivery Overlays* (CDOs) mit offenen Protokollen und offenen Technologien ist *BitTorrent*[227];[226] ein Beispiel für ein *Content Delivery Overlay* im kommerziellen Bereich zum Zweck der Auslieferung von Video-Inhalten ist Octoshape.[226] Content Delivery Overlays mit dem Zweck der Auslieferung von Video-Inhalten werden *Video-Streaming Overlays* genannt.[228]

Onion-Routing-Systeme - wie z.B. *Tor*[229,230] - erzeugen ein Overlay-Netzwerk aus Computern, die als Relays für Datenverkehre dienen[231] - aufbauend auf einem IP-Netzwerk - dem Internet - als Underlay. Die Funktionsweise von Onion-Routing-Systemen besteht darin, zu transportierende Datagramme mehrfach geschachtelt einzupacken und diese Datagramme über mehrere Relays des Overlay-Netzwerkes hinweg zum Ziel zu transportieren; beim Transport der Datagramme zum Ziel hin werden die mehrfach geschachtelt eingepackten Datagramme von Relay zu Relay schrittweise ausgepackt.[232] Primäres

[219] Christian Groß, Björn Richerzhagen und Max Lehn: Structured Search Overlays, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [138, S. 64][137, S. 49-67]
[220] Sylvia Ratnasamy: A Scalable Content-Addressable Network, Ph.D. Thesis, University of California at Berkeley, 2002, URL: `http://www.icir.org/sylvia/thesis.ps` [144]
[221] Freenet Project Inc.: Freenet, URL: `https://freenetproject.org/` [145]
[222] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 863]
[223] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph und John D. Kubiatowicz: Tapestry: A Resilient Global-Scale Overlay for Service Deployment, In: IEEE Journal on Selected Areas in Communications 22.1, 2004 [146, S. 41-53]
[224] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 754]
[225] Kamill Panitzek, Muhammad Ikram und Michael Stein: Service Overlays, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [147, S. 113][137, S. 113-140]
[226] Elias Weingärtner, René Glebke und Alexander Hocks: Content Delivery Overlays, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [148, S. 69][137, S. 69-79]
[227] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 750]
[228] Osama Abboud und Julius Rückert: Video-Streaming Overlays, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [149, S. 169][137, 169–193]
[229] The Tor Project Inc.: Tor Project: Anonymity Online URL: `https://www.torproject.org/` [150]
[230] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151][152, 303–320]
[231] George Danezis, Claudia Diaz und Paul Syverson: Systems for Anonymous Communication, 2009 [153, S. 35]
[232] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 863]

Design-Ziel von Onion-Routing-Systemen ist die Anonymisierung von bidirektionaler Kommunikation mit niedriger Latenz.[233] Ein weiteres Onion-Routing-System mit dem Zweck der Anonymisierung von Datenverkehren neben *Tor*[229,230,237] ist *Tarzan*[234,235].

Garlic-Routing-Systeme - wie z.B. *I2P*[236] - erzeugen ebenso ein Overlay-Netzwerk aus Computern, die als Relays für Datenverkehre dienen, aufbauend auf dem Internet als Underlay. *Garlic Routing* ist eine Variante von *Onion Routing*;[237,238] bei *Garlic Routing* werden die zu transportierenden Datagramme ebenfalls mehrfach geschachtelt eingepackt und diese Datagramme über mehrere Relays des Overlay-Netzwerkes hinweg zum Ziel transportiert;[237] als Erweiterung werden bei *Garlic Routing* mehrere gekapselte Pakete unterschiedlicher Sender und Empfänger bei jedem Relay zu Bündeln (*bulbs* bzw. *cloves*[238]) umgepackt.[237] Dies dient dem primären Ziel, den Grad der Anonymität bei der Kommunikation unter Verwendung von *Garlic Routing* zu erhöhen.[237] Ein Vorteil von *Garlic Routing* gegenüber *Onion Routing* ist eine höhere Zuverlässigkeit und Robustheit durch höhere Pfad-Redundanz (*path redundancy*).[237,238]

2.3.1 Verbreitete Overlay-Netzwerke

Es scheint keine naheliegende oder praktikable verhältnismässige Möglichkeit zu geben, die Verbreitung existierender Overlay-Netzwerke *direkt* zu messen; eine *indirekte* Möglichkeit, die Verbreitung existierender Overlay-Netzwerke zu messen, besteht darin, den Grad des öffentlichen Interesses an Ressourcen zu ihrem praktischen Einsatz zu ermitteln.

Ein Indikator für den Grad des öffentlichen Interesses an Ressourcen zum praktischen Einsatz eines bestimmten Overlay-Netzwerks ist die Anzahl der Ergebnisse einer Internet-Suchmaschine zu einer Suchanfrage bezüglich diesem Overlay-Netzwerk. Die jeweilige Anzahl der Ergebnisse der Internet-Suchmaschine *Google Search*[239] zur Suchanfrage[240]

```
Suchbegriff AND ("overlay network" OR "overlay networks" OR
  "distributed system" OR "distributed systems") AND
(howto OR "how to" OR faq OR Tutorial) AND
(Installation OR install) AND Software
```

zeigt Tabelle 2.6.

[233] George Danezis, Claudia Diaz und Paul Syverson: Systems for Anonymous Communication, 2009 [153, S. 32]

[234] George Danezis, Claudia Diaz und Paul Syverson: Systems for Anonymous Communication, 2009 [153, S. 37]

[235] Michael J. Freedman: A Peer-to-Peer Anonymizing Network Layer, Master Thesis, Massachusetts Institute of Technology, 2002, URL: https://www.cs.princeton.edu/~mfreed/docs/tarzan-thesis.pdf [154, S. 16]

[236] I2P Project: The Invisible Internet Project: I2P Anonymous Network, URL: https://geti2p.net/ [155]

[237] Roger R. Dingledine: The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven, Master Thesis, Massachusetts Institute of Technology, 2000, URL: https://dspace.mit.edu/bitstream/handle/1721.1/86475/46861930-MIT.pdf?sequence=2 [156, S. 104]

[238] I2P Project: The Invisible Internet Project: I2P Anonymous Network - Documentation - Garlic Routing, URL: https://geti2p.net/en/docs/how/garlic-routing [157]

[239] Google Inc.: Google Search, URL: https://www.google.com/ [158]

[240] Für den Suchbegriff *Tapestry* - das Overlay-Netzwerk *Tapestry* betreffend - ergibt sich die aus der Suchanfrage resultierende und an die Internet-Suchmaschine *Google Search* übertragene URL https://www.google.com/search?q=Tapestry+("overlay+network"+|+"overlay+networks"+|+"distributed+system"+|+"distributed+systems")+(howto+|+"how+to"+|+faq+|+Tutorial)+(Installation+|+install)+Software.

Tabelle 2.6: Anzahl der Ergebnisse der Suchmaschine *Google Search* bezüglich verschiedener Overlay-Netzwerke

Suchbegriff / Overlay-Netzwerk	**Ergebnisse**
MPLS	7 960
Multicast	15 400
Internet Multicast[241]	83
Chord	7 240
Kademlia	1 760
Pastry	1 440
Content-addressable network	176
Freenet	1 320
Tapestry	4 000
BitTorrent	9 870
Octoshape	17
Tor	13 200
Tarzan	195
I2P	521

Die folgenden Abschnitte stellen zwei Overlay-Netzwerke - *Tor*[242,243] und *I2P*[244] - vor, deren Zweck darin besteht, Overlays mit Knoten (*nodes*) zu erzeugen, die als Relays für IP-basierte Datenkommunikation mit niedriger Latenz dienen; das Netzwerk, auf dem aufgebaut wird - das Underlay - ist jeweils das Internet. Vorgreifend auf das Kapitel 3 Analyse und auf das Kapitel 5 Konzeption werden relevante Eigenschaften der Overlay-Netzwerke dargestellt.

2.3.1.1 Tor

Tor[242,243,245] ist ein Overlay-Netzwerk aus Computern, die als Relays für Datenverkehre dienen, aufbauend auf dem Internet als Underlay. Die Relay-Funktionalität der einzelnen Knoten des Overlay-Netzwerks umfasst den Pakettransport von Datagrammen nach dem TCP-Protokoll. Als Schnittstelle für darauf aufbauende Anwendungen wird das *SOCKS*-

[241] Für den Suchbegriff *Internet Multicast* ergibt sich die aus der Suchanfrage resultierende und an die Internet-Suchmaschine *Google Search* übertragene URL `https://www.google.com/search?q="Internet+Multicast"+("overlay+network"+|+"overlay+networks"+|+"distributed+system"+|+"distributed+systems")+(howto+|+"how+to"+|+faq+|+Tutorial)+(Installation+|+install)+Software`.

[242] The Tor Project Inc.: Tor Project: Anonymity Online URL: `https://www.torproject.org/` [150]

[243] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151][152, 303–320]

[244] I2P Project: The Invisible Internet Project: I2P Anonymous Network, URL: `https://geti2p.net/` [155]

[245] Roger R. Dingledine: The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven, Master Thesis, Massachusetts Institute of Technology, 2000, URL: `https://dspace.mit.edu/bitstream/handle/1721.1/86475/46861930-MIT.pdf?sequence=2` [156, S. 104]

Protokoll[246,247] verwendet.[248] Der Erzeugung und Aufrechterhaltung der Struktur des Tor-Netzwerks dienen mehrere redundante zentrale Knoten, die *Directory Server* genannt werden.[249,250]

Der Pakettransport auf dem Underlay basiert sowohl für die Verbindungen der Knoten zu dem Directory Servern als auf für die Verbindungen der Knoten untereinander auf dem TCP-Protokoll.[251,252,253]

Tor setzt für einen Teil der kryptographischen Funktionalität das *SSL/TLS*-Protokoll[254] ein; das sichernde Transportprotokoll auf dem Underlay - dem Internet - ist das *TLS*-Protokoll.[255]

Zur Sicherung der Authentizität der Knoten wird Zertifikat-basierte Public-Key-Verschlüsselung mit dem *RSA*-Verfahren[256] und zur Sicherung der Integrität der Kommunikation wird der kryptographische Algorithmus *SHA-1*[257] eingesetzt, für den Schlüsselaustausch wird das Diffie-Hellman-Verfahren (*DH*)[258] eingesetzt; exklusiv alternativ zu *RSA*-Schlüsseln können auf elliptischen Kurven basierende *ECDSA*-Schlüssel genutzt werden.[259] Für die Verschlüsselung der Nutzdaten wird das *AES*-Verfahren mit 128 Bit Länge eingesetzt.[256]

Die Software zum Betreiben eines Tor-Knotens - die Software wird ebenso wie das Overlay-Netzwerk selbst *Tor* genannt[255] - ist frei zugängliche und frei verfügbare Software, die im Quellcode vorliegt.[260]

Die Tor-Software sollte einfach auf verbreiteten Betriebssystem-Plattformen implementiert werden können[255] und erfordert keine Modifikationen des jeweiligen Betriebssystems oder am Betriebssystemkern (*kernel*).

Die aktuelle Version der Software ist Version 0.3.0.10 vom 02.08.2017 (Stand 18.08.2017);[260] es existiert ein Entwicklungszweig der Software mit der Versionsnummer 0.3.1.5-alpha vom 01.08.2017 (Stand 18.08.2017), die mit dem Label *unstable/alpha* gekennzeichnet ist.[260]

---

[246] *SOCKS* ist ein Protokoll für *Circuit Level Proxies*; eine Anwendung nutzt einen *SOCKS*-Server als Stellvertreter für den Pakettransport, die nutzende Anwendung wird *SOCKS*-Client genannt.[247]

[247] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "SOCKS" [8, S. 779]

[248] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151, S. 308][152, 303–320]

[249] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151, S. 313][152, 303–320]

[250] George Danezis, Claudia Diaz und Paul Syverson: Systems for Anonymous Communication, 2009 [153, S. 35]

[251] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151, S. 310][152, 303–320]

[252] The Tor Project Inc.: Tor Project: Documentation - FAQ - Running a Tor relay, URL: `https://www.torproject.org/docs/faq.html.en#RunningATorRelay` [159]

[253] The Tor Project Inc.: Tor Project: Documentation - FAQ - Running a Tor relay: I'm behind a NAT/Firewall, URL: `https://www.torproject.org/docs/faq.html.en#BehindANAT` [160]

[254] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2, 2008, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc5246.txt` [115]

[255] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151, S. 306][152, 303–320]

[256] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151, S. 307][152, 303–320]

[257] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151, S. 309][152, 303–320]

[258] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151, S. 311][152, 303–320]

[259] The Tor Project Inc.: Tor Project: Documentation - FAQ - Running a Tor relay: How do offline ed25519 identity keys work? What do I need to know?, URL: `https://www.torproject.org/docs/faq.html.en#OfflineED25519` [161]

[260] The Tor Project Inc.: Tor Project: Download Tor - Source Code, URL: `https://www.torproject.org/download/download.html.en#source` [162]

Der Quellcode der Software ist unter der URL `https://www.torproject.org/dist/tor-0.3.0.10.tar.gz` abrufbar;[260] des Weiteren ist der Quellcode des aktuellen Entwicklungszweiges der Software auf der Internetseite "tor - Tor's source code" unter der URL `https://gitweb.torproject.org/tor.git/` zugreifbar.
Die Quellcode der Software Tor ist in der Programmiersprache C geschrieben.

Tor baut für einen Teil der kryptographischen Funktionen auf der Softwarebibliothek *OpenSSL*[261] auf, für die Funktionalität der Kompression des Datenverkehrs nutzt Tor die Softwarebibliothek *zlib*. Zur effizienteren Netzwerk-Kommunikation baut Tor auf der Softwarebibliothek *Libevent*[262] auf; diese Bibliothek stellt ein Framework für *TCP*-Sockets zur Verfügung, um die Systemaufrufe *select*[263], *poll*, *kqueue* und *epoll* für diese *TCP*-Sockets - diesen sind *TCP*-Verbindungen zugeordnet - auf effiziente Weise zu handhaben.[264]

Die Tor-Software ist primär verfügbar für die Betriebssysteme Microsoft *Windows*, *Linux*, *Solaris*, *Unix*-artige Betriebssysteme der *BSD*-Betriebssystemfamilie wie *FreeBSD*, *OpenBSD* und *NetBSD* und für das Betriebssystem Apple *MacOS/X*.[255,265]

Auch für das Betriebssystem Google *Android* ist *Tor* verfügbar.[266]

Es sind für viele Betriebssystemplattformen vorbereitete Pakete[267] von *Tor* verfügbar, z.B. für die Betriebssysteme *Debian Linux*, *Ubuntu Linux*, *Knoppix Linux*, *CentOS Linux*, *Fedora Linux*, *Gentoo Linux*, *FreeBSD*, *OpenBSD* und *NetBSD*.[265]

2.3.1.2 I2P

I2P[268] ist ein Overlay-Netzwerk aus Computern, die als Relays und als Router für Datenverkehre dienen, aufbauend auf dem Internet als Underlay.
Die Relay-Funktionalität der einzelnen Knoten des Overlay-Netzwerks umfasst den Pakettransport von Datagrammen nach dem *TCP*-Protokoll und den Pakettransport von Datagrammen nach dem *UDP*-Protokoll. Es existieren mehrere Schnittstellen für darauf aufbauende Anwendungen: zum Einen sind Protokoll-Proxys für verschiedene Protokolle (*HTTP*, *HTTPS*, *IRC*)[271] implementiert, zum Anderen wird die Realisierung von Portweiterleitungen sowohl nach dem *TCP*-Protokoll[269,270,271] als auch nach dem *UDP*-Protokoll[271] ermöglicht. Daneben ist als Schnittstelle für darauf aufbauende Anwendun-

[261] OpenSSL Software Foundation: OpenSSL Cryptography and SSL/TLS Toolkit, URL: `https://www.openssl.org/` [104]
[262] Libevent: libevent - an event notification library, URL: `http://libevent.org/` [163]
[263] W. Richard Stevens und Gary R. Wright: TCP/IP Illustrated, Volume 2: The Implementation, 1995 [164, S. 536]
[264] The Tor Project Inc.: Tor Project: Documentation - FAQ - Development: What is Libevent?, URL: `https://www.torproject.org/docs/faq.html.en#WhatIsLibevent` [165]
[265] The Tor Project Inc.: Tor Project: Download for Linux/Unix, URL: `https://www.torproject.org/download/download-unix.html.en` [166]
[266] The Tor Project Inc.: Tor Project: Download Tor - Tor for Smartphones, URL: `https://www.torproject.org/download/download.html.en#smartphone` [167]
[267] Diese vorbereiteten Pakete werden *packages* bzw. beim Betriebssystem FreeBSD *ports* genannt.
[268] I2P Project: The Invisible Internet Project: I2P Anonymous Network, URL: `https://geti2p.net/` [155]
[269] I2P Project: The Invisible Internet Project: Introduction - I2P: A scalable framework for anonymous communication, URL: `https://geti2p.net/en/docs/how/tech-intro` [168]
[270] I2P Project: The Invisible Internet Project: I2P Anonymous Network - Documentation - API - I2PTunnel, URL: `https://geti2p.net/en/docs/api/i2ptunnel` [169]
[271] I2P Project: The Invisible Internet Project: I2P Anonymous Network - Index to Technical Documentation, URL: `https://geti2p.net/en/docs` [170]

gen auch das *SOCKS*-Protokoll[272,273] implementiert.[269,274,271]

Der Pakettransport auf dem Underlay basiert auf dem *TCP*-Protokoll[275] und auf dem *UDP*-Protokoll[276].[269,277] Der initialen Erzeugung der Struktur des I2P-Netzwerks dienen WWW-Server im Internet, die Listen bereits bestehener Instanzen von Knoten des I2P-Netzwerks vorhalten (*Bootstrapping*);[269] das *TCP*-Protokoll wird hierfür verwendet.

Zur Sicherung der Authentizität der Knoten wird Zertifikat-basierte Public-Key-Verschlüsselung mit dem DSA-Verfahren und das auf elliptischen Kurven basierende ECDSA-Verfahren eingesetzt;[278] für den Schlüsselaustausch wird das Diffie-Hellman-Verfahren (DH) eingesetzt.[278] Zur Sicherung der Authentizität von Softwareupdates wird RSA-Verschlüsselung benutzt.[278] Zur Sicherung der Integrität der Kommunikation wird der kryptographische Algorithmus SHA-256 eingesetzt;[278] für die Verschlüsselung der Nutzdaten wird das AES-Verfahren mit 128 Bit Länge in Verbindung mit dem ElGamal-Verfahren[279,280,281,282] genutzt.[278]

Die I2P-Software nutzt *nicht* das SSL/TLS-Protokoll[283] als sicherndes Transportprotokoll auf dem Underlay.[278]

Die Software zum Betreiben eines I2P-Knotens ist frei zugängliche und frei verfügbare Software, die im Quellcode vorliegt.[284]

Die aktuelle Version der Software ist Version 0.9.3.1 vom 07.08.2017 (Stand 20.08.2017).[284]

Der Quellcode der Software ist unter der URL `https://download.i2p2.de/releases/0.9.31/i2psource_0.9.31.tar.bz2` abrufbar;[284] des Weiteren ist der Quellcode des aktuellen Entwicklungszweiges der Software auf der Internetseite "GitHub - i2p/i2p.i2p" unter der URL `https://github.com/i2p/i2p.i2p` zugreifbar.[285]

Der Quellcode der Software für I2P ist in der Programmiersprache Java geschrieben.[285]

Die Software für I2P benötigt die Laufzeitumgebung von Java (*JRE - Java Runtime Environment*) in Version 7 oder eine neuere Version der Laufzeitumgebung.[284] Es werden keine weiteren externen Softwarebibliotheken zur Ausführung der Software für I2P benötigt, die benutzten Softwarebibliotheken sind im Quellcodebaum der Software integriert.

Die Software für I2P ist grundsätzlich auf jeder Betriebssystemplattform lauffähig, für

[272] *SOCKS* ist ein Protokoll für *Circuit Level Proxies*; eine Anwendung nutzt einen *SOCKS*-Server als Stellvertreter für den Pakettransport, die nutzende Anwendung wird *SOCKS*-Client genannt.[273]

[273] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "SOCKS" [8, S. 779]

[274] I2P Project: The Invisible Internet Project: I2P Anonymous Network - Documentation - API - SOCKS, URL: `https://geti2p.net/en/docs/api/socks` [171]

[275] Die Pakettransport-Implementierung über das *TCP*-Protokoll wird bei der I2P-Software *NTCP* genannt.

[276] Die Pakettransport-Implementierung über das *UDP*-Protokoll wird bei der I2P-Software *SSU* genannt.

[277] Khalid Shahbar und A. Nur Zincir-Heywood: Effects of Shared Bandwidth on Anonymity of the I2P Network Users, In: 38th IEEE Symposium on Security and Privacy Workshops, 2nd International Workshop on Traffic Measurements for Cybersecurity (WTMC 2017), 2017 [172, S. 5]

[278] I2P Project: The Invisible Internet Project: I2P Anonymous Network - Specifications - Low-level Cryptography Specification, URL: `https://geti2p.net/spec/cryptography` [173]

[279] Das ElGamal-Verfahren steht bezüglich seiner mathematischen Grundlagen aus der Gruppentheorie in Verbindung mit dem Diffie-Hellman-Verfahren.[280]

[280] Henk C.A. van Tilborg, Hrsg.: Encyclopedia of Cryptography and Security, Eintrag "ElGamal public key encryption scheme" [101, S. 183]

[281] Die verwendete Primzahl für das bei der I2P-Software eingesetzte ElGamal-Verfahren ist den Spezifikationen des IPsec-Standards entnommen (RFC 3526)[282]

[282] RFC 3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE), 2003, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3526.txt` [174]

[283] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2, 2008, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc5246.txt` [115]

[284] I2P Project: The Invisible Internet Project: I2P Anonymous Network - Download, URL: `https://geti2p.net/en/download` [175]

[285] I2P Project: The Invisible Internet Project: I2P Anonymous Network - New Developer's Guide, URL: `https://geti2p.net/en/get-involved/guides/new-developers` [176]

die eine Laufzeitumgebung der Programmiersprache Java in Version 7 oder in einer neueren Version verfügbar ist; zur Nutzung der Software für I2P sind keine Modifikationen des jeweiligen Betriebssystems oder am Betriebssystemkern (*kernel*) notwendig.

Es sind für einige Betriebssystemplattformen vorbereitete Pakete der Software für I2P verfügbar, z.B. für die Betriebssysteme Microsoft *Windows*, Apple *MacOS/X*, *Debian Linux* und *Ubuntu Linux*.[284] Die Software für I2P ist auch unter den *Unix*-artigen Betriebssystemen der *BSD*-Betriebssystemfamilie und unter dem Betriebssystem *Solaris* lauffähig;[284] für das Betriebssystem Google *Android* ist I2P ebenso verfügbar.[284]

3 Analyse

Um die Anforderung zu erfüllen, räumlich weit voneinander entfernte *Local Area Networks* (LANs) miteinander zuverlässig zu koppeln, erscheint es naheliegend, durch redundante Auslegung der Leitungswege die Ausfallsicherheit zu erhöhen.

Die redundante Auslegung der Leitungswege durch *mehrere feste Verbindungen* - z.B. Standard-Festverbindungen[1,2] - zwischen den Standorten der zu koppelnden *Local Area Networks* ist betriebswirtschaftlich in finanzieller Hinsicht oftmals *nicht* darstellbar. Eine digitale Standard-Festverbindung des Unternehmens Telekom Deutschland GmbH hat eine *vereinbarte* mittlere Verfügbarkeit von 98,5 % im Jahresdurchschnitt.[1]

Um räumlich weit voneinander entfernte *Local Area Networks* unter Berücksichtigung der betriebswirtschaftlichen Bedingungen (*constraints*) miteinander zu koppeln, werden *Virtual Private Networks* (VPNs) genutzt; *Virtual Private Networks* nutzen das Internet als Übertragungsweg.

3.1 Eingesetzte VPN-Protokolle zur LAN-LAN-Kopplung

Zur Realisierung der Kopplung von räumlich weit entfernten *Local Area Networks* mit Hilfe von *Virtual Private Networks* werden an den Standorten der zu koppelnden *Local Area Networks* breitbandige Zugänge ins Internet betrieben und mit Hilfe geeigneter VPN-Software wird darauf aufbauend ein virtueller Leitungsweg als Übertragungsweg erzeugt.

Es lassen sich zwei Arten der Realisierung von *Virtual Private Networks* zur Kopplung von *Local Area Networks* (LANs) unterscheiden: zum Einen die Realisierung des *Virtual Private Networks* durch die Betreiber der *Local Area Networks* und zum Anderen die Realisierung des *Virtual Private Networks* durch einen Internet Service Provider (siehe Abschnitt 2.2.3)[3]. Voraussetzung für die Realisierung des *Virtual Private Networks* zur Kopplung von LANs durch einen Internet Service Provider ist die Abdeckung der Standorte durch diesen Internet Service Provider (ISP).

Bei Realisierung des *Virtual Private Networks* durch einen Internet Service Provider können als VPN-Protokolle *IPsec* (siehe Abschnitt 2.2.4.1)[4], *IPsec* mit *L2TP* (*Layer Two Tunneling*

[1] Telekom Deutschland GmbH: Leistungsbeschreibung Standard-Festverbindungen Digital (dSFV), URL: `http://www.telekom.de/dlp/agb/pdf/43602.pdf` [177] (Stand: 01.03.2016)

[2] Synonym zum Begriff *Standard-Festverbindung* wird gemeinhin der Begriff *Standleitung* verwendet.

[3] RFC 4026: Provider Provisioned Virtual Private Network (VPN) Terminology, 2005, Informational, URL: `http://www.rfc-editor.org/rfc/rfc4026.txt` [76]

[4] RFC 2764: A Framework for IP Based Virtual Private Networks, 2000, Informational, URL: `http://www.rfc-editor.org/rfc/rfc2764.txt` [75]

Protocol) (siehe Abschnitt 2.2.4.2)[5,6,7] und *MPLS* (*Multiprotocol Label Switching*[8]) mit *BGP* (siehe Abschnitt 2.2.3)[9,10] verwendet werden.

Bei Realisierung des *Virtual Private Networks* durch die Betreiber der *Local Area Networks* wählen die Betreiber das verwendete *VPN*-Protokoll; verbreitete VPN-Protokolle sind diesbezüglich *IPsec* (siehe Abschnitt 2.2.6.1) und *OpenVPN*[11] (siehe Abschnitt 2.2.6.2 sowie Abschnitt 2.2.6 De-facto-Standards bei VPN-Protokollen).

3.2 Geeignete Routing-Protokolle

Es gibt mehrere Möglichkeiten zum Koppeln von *Local Area Networks* (LANs) mit Hilfe von *Virtual Private Networks*: zum Einen lassen sich LANs mit Hilfe von hierfür *geeigneten Virtual Private Networks* auf OSI Layer 2 - auf Ethernet-Ebene (*Data Link Layer*[12]) - miteinander verbinden, zum Anderen lassen sich LANs mit Hilfe von *Virtual Private Networks* auf OSI Layer 3 - auf IP-Ebene (*Network Layer*[12]) - miteinander verbinden.

Bei der Kopplung von LANs mit Hilfe von *Virtual Private Networks* auf Ethernet-Ebene (*Data Link Layer*) ist *keine zwingende* Notwendigkeit gegeben, Routing einzusetzen; die gekoppelten LANs bilden auf Ethernet-Ebene eine Broadcastdomäne[13] (*broadcast domain*). Die VPN-Kopplung von LANs auf OSI Layer 2 - auf Ethernet-Ebene - hat den Vorteil, dass für nutzende Anwendungen auf den darüberliegenden, darauf aufbauenden Schichten hohe Kompatibilität - vergleichbar mit dem direkten Zugriff - gegeben ist; diese Art der Kopplung hat jedoch den Nachteil, dass sie in grösseren zu koppelnden Netzwerken weniger gut anwendbar ist.

Bei der Kopplung von LANs mit Hilfe von *Virtual Private Networks* auf IP-Ebene (*Network Layer*) ist Routing *notwendig*.
In kleineren mit Hilfe von *Virtual Private Networks* zu koppelnden LANs kann das Routing durch statisches Routing geschehen; auf jedem Router sind die Routen zu jedem zu erreichenden IP-Netz der LANs fest - statisch - hinterlegt.

[5] RFC 3931: Layer Two Tunneling Protocol - Version 3 (L2TPv3), 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3931.txt` [94]
[6] RFC 4719: Transport of Ethernet Frames over Layer 2 Tunneling Protocol Version 3 (L2TPv3), 2006, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4719.txt` [93]
[7] RFC 3193: Securing L2TP using IPsec, 2001, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3193.txt` [95]
[8] *Multiprotocol Label Switching* (*MPLS*) ist eine Datagramm-Transporttechnik, bei der die Datagramme eingekapselt transportiert werden; hierbei wird jedem eingekapselten Datagramm ein Header vorangestellt. Ein Bestandteil des vorangestellten Headers ist ein Kennzeichner der Länge 20 Bit, der *Label* genannt wird [78] [9, S. 360,470,471] [79, S. 34].
[9] RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs), 2006, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4364.txt` [81]
[10] RFC 6178: Label Edge Router Forwarding of IPv4 Option Packets, 2011, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc6178.txt` [82]
[11] Charlie Hosner: OpenVPN and the SSL VPN Revolution, SANS Institute, 2004, URL: `https://www.sans.org/reading-room/whitepapers/vpns/openvpn-ssl-vpn-revolution-1459` [134, S. 5]
[12] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "OSI-Schichten, Namen und Nummern der -" [8, S. 601]
[13] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Broadcastdomäne" [8, S. 120]

In grösseren mit Hilfe von *Virtual Private Networks* gekoppelten LANs geschieht das Routing mit dynamischen Routing-Protokollen[14]; dies fusst zum Einen darauf, dass der administrative Aufwand wie z.B. für die Pflege der Routing-Tabellen auf jedem einzelnen Router reduziert wird und zum Anderen darauf, dass beim Einsatz von dynamischen Routing-Protokollen oftmals zusätzliche Funktionalität wie z.B. Mehrweg-Routing (*Equal Cost MultiPath - ECMP*)[15] gegeben ist. Bei Mehrweg-Routing (*Equal Cost MultiPath - ECMP*) kann der Transport von IP-Datenpaketen über mehrere Wege verteilt werden (*load balancing*).

Geeignete Routing-Protokolle für das dynamische Routing bei der LAN-Kopplung mit Hilfe von *Virtual Private Networks* sind solche Routing-Protokolle, die auch als Interior Gateway Protocol (IGP) zum Internet-Routing innerhalb eines Autonomen Systems (AS) genutzt werden (siehe Abschnitt 2.1.2); dies fusst auf der Eigenschaft der guten Skalierbarkeit und auf der Eigenschaft des schnellen Konvergenzverhaltens bei gebräuchlichen Routing-Protokollen zum Internet-Routing innerhalb Autonomer Systeme.

Als Interior Gateway Protocol (IGP) zum Internet-Routing innerhalb eines Autonomen Systems (*intradomain routing*) werden gewöhnlich das dynamische Routing-Protokoll *OSPF* (*Open Shortest Path First*) sowie das dynamische Routing-Protokoll *IS-IS* (*Intermediate-System to Intermediate-System*) eingesetzt;[16] das dynamische Routing-Protokoll *OSPF* wird als Interior Gateway Protocol bevorzugt in Unternehmens-Netzwerken eingesetzt, das dynamische Routing-Protokoll *IS-IS* wird als Interior Gateway Protocol bevorzugt in Netzwerken der Internet Service Provider eingesetzt.[17]

Als grundsätzlich geeignet für das dynamische Routing bei der LAN-Kopplung mit Hilfe von *Virtual Private Networks* erscheint sowohl das dynamische Routing-Protokoll *OSPF* als auch das dynamische Routing-Protokoll *IS-IS*, wobei die bevorzugte Verwendung des Routing-Protokolls *OSPF* in Unternehmens-Netzwerken und die häufigere Implementierung des Routing-Protokolls *OSPF* in verfügbarer Routing-Software im Internet (siehe Abschnitt 2.1.3) die Wahl zur Verwendung des dynamischen Routing-Protokolls *OSPF* forciert.

3.3 Overlay-Netzwerke im Internet

Bestehende Overlay-Netzwerke im Internet dienen einem oder mehreren Zwecken; sie lassen sich dem primären Zweck und Nutzen nach unterteilen:

Es existieren Overlay-Netzwerke, die primär dem Speichern und dem Abruf von Datenobjekten dienen - Peer-to-Peer-File-Sharing-Systeme als *Peer-to-Peer-Overlays*, *Structured-Search-Overlays* und *Content Delivery Overlays* (*CDO*s) -; es existieren Overlay-Netzwerke, die primär dem entfernten Ausführen von Objekten dienen - *Service Overlays*[18] -; und es existieren Overlay-Netzwerke, die primär dem Transport von Datenströmen mit niedriger Latenz dienen, z.B. TCP-Overlay-Netzwerke (*TCP overlay networks*).

[14] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Routing Protokoll, dynamisches -, statisches -" [8, S. 715]

[15] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 476]

[16] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 479]

[17] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 474]

[18] Kamill Panitzek, Muhammad Ikram und Michael Stein: Service Overlays, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [147, S. 113][137, S. 113-140]

Overlay-Netzwerke im Internet, die dem primären Zweck des Speicherns und des Abrufs von Datenobjekten dienen, sind *BitTorrent*[19,20] und *Freenet*[21] - *Peer-to-Peer-Overlays*[22] mit der Struktur eines *Distributed Hash Table* (*DHT*)[23] -, Implementierungen von *Kademlia*[24] - ein *Structured Search Overlay*[25] mit der Struktur eines *Distributed Hash Table* (*DHT*)[23] -, und *Video-Streaming Overlays*[26] als spezialisierte *Content Delivery Overlays* (*CDO*s)[20] wie z.B. das *Video-Streaming Overlay*[26] *Octoshape*[20].

Eine Overlay-Netzwerk-Implementierung im Internet, welche zum Zweck des entfernten Ausführens von Objekten (*service objects*) genutzt werden kann - ein solches Overlay-Netzwerk wird *Service Overlay*[18] genannt -, ist *Tapestry*[27], auch *Tapestry* nutzt die Struktur eines *Distributed Hash Table* (*DHT*).

Overlay-Netzwerke im Internet, die dem primären Zweck des Transports von Datenströmen mit niedriger Latenz dienen, sind *Tor*[28,29] - ein *Onion-Routing-System*[30,31] -, *I2P*[32] - ein *Garlic-Routing-System*[33] -, und IP-*Multicast*[34] mit dem Protokoll *IGMP* (*Internet Group Management Protocol*)[35] in Verbindung mit den Multiprotokoll-Erweiterungen des BGP-Protokolls[36] (siehe Abschnitt 2.1.2.1).

3.3.1 Veränderung der Netzwerktopologie durch Overlay-Netzwerke

Overlay-Netzwerke ermöglichen es den Kunden von Internet Service Providern, Dienste zu betreiben, die unabhängig von der Netzwerkinfrastruktur der jeweils genutzten Inter-

[19] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 750]
[20] Elias Weingärtner, René Glebke und Alexander Hocks: Content Delivery Overlays, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [148, S. 69][137, S. 69-79]
[21] Freenet Project Inc.: Freenet, URL: `https://freenetproject.org/` [145]
[22] Kamill Panitzek und Thorsten Strufe: A Formal Model for Peer-to-Peer Systems, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [136][137, S. 15-18]
[23] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 753]
[24] Petar Maymounkov und David Mazières: Kademlia: A Peer-to-Peer Information System Based on the XOR Metric, In: LNCS (Lecture Notes in Computer Science): Peer-to-Peer Systems: IPTPS 2002 (International Workshop on Peer-to-Peer Systems), 2002 [141][142, 53–65]
[25] Christian Groß, Björn Richerzhagen und Max Lehn: Structured Search Overlays, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [138, S. 50][137, S. 49-67]
[26] Osama Abboud und Julius Rückert: Video-Streaming Overlays, In: LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, 2013 [149, S. 169][137, 169–193]
[27] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph und John D. Kubiatowicz: Tapestry: A Resilient Global-Scale Overlay for Service Deployment, In: IEEE Journal on Selected Areas in Communications 22.1, 2004 [146, S. 41-53]
[28] The Tor Project Inc.: Tor Project: Anonymity Online URL: `https://www.torproject.org/` [150]
[29] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151][152, 303–320]
[30] George Danezis, Claudia Diaz und Paul Syverson: Systems for Anonymous Communication, 2009 [153, S. 32]
[31] Roger R. Dingledine: The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven, Master Thesis, Massachusetts Institute of Technology, 2000, URL: `https://dspace.mit.edu/bitstream/handle/1721.1/86475/46861930-MIT.pdf?sequence=2` [156, S. 104]
[32] I2P Project: The Invisible Internet Project: I2P Anonymous Network, URL: `https://geti2p.net/` [155]
[33] I2P Project: The Invisible Internet Project: I2P Anonymous Network - Documentation - Garlic Routing, URL: `https://geti2p.net/en/docs/how/garlic-routing` [157]
[34] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Multicast" [8, S. 553]
[35] RFC 3376: Internet Group Management Protocol, Version 3, 2002, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3376.txt` [135]
[36] RFC 4760: Multiprotocol Extensions for BGP-4, 2007, Draft Standard, URL: `http://www.rfc-editor.org/rfc/rfc4760.txt` [12]

net Service Provider sind;[37] die zu Grunde liegende Netzwerktopologie des Transportnetzes für die darauf aufbauenden Protokolle und für die darauf aufbauenden Dienste ist verändert.

Beispielsweise ist beim Zugriff mit Hilfe des Protokolls *HTTP*[38,39,40] auf die Ressourcen eines Webservers das zu Grunde liegende Transportnetz das Internet; die Netzwerktopologie des Internets ist bestimmt durch die Verbindungen der Autonomen Systeme (ASe[41]) der Internet Service Provider untereinander (siehe Abschnitt 2.1.1 Struktur des Backbones des Internets).

Abbildung 3.1 zeigt die zu Grunde liegende Connectivity der Autonomen Systeme des Internets am 02.01.2013.

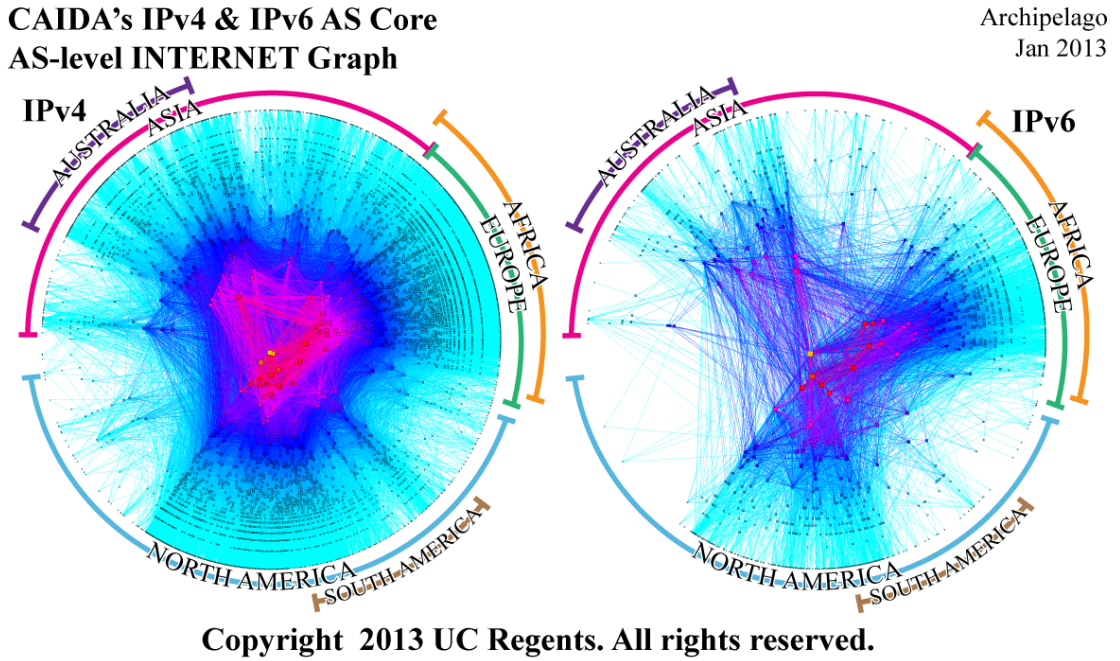

Abbildung 3.1: Kern Autonome Systeme (IPv4 und IPv6), Januar 2013[42]

Beim Zugriff mit Hilfe des Protokolls *HTTP*[38,39,40] auf die Ressourcen eines Webservers *über ein TCP-Overlay-Netzwerk* wie z.B. über das Tor-Netzwerk hingegen ist das *unmittelbar* zu Grunde liegende Transportnetz für das Protokoll *HTTP* das TCP-Overlay-Netzwerk; die Netzwerktopologie des TCP-Overlay-Netzwerks ist bestimmt durch die Router des

[37] Matthias Wählisch und Thomas C. Schmidt: Peer the Peers: An Overlay ID Assignment Service at Internet Exchange Points, In: Proceedings of the 5th International Student Workshop on Emerging Networking Experiments and Technologies, Co-Next Student Workshop '09, Rome, Italy, 2009 [178, 45–46]

[38] RFC 1945: Hypertext Transfer Protocol – HTTP/1.0, 1996, Informational, URL: `http://www.rfc-editor.org/rfc/rfc1945.txt` [179]

[39] RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, 2014, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc7230.txt` [180]

[40] RFC 7540: Hypertext Transfer Protocol Version 2 (HTTP/2), 2015, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc7540.txt` [181]

[41] *ASes* ist englischer Sprachgebrauch für den Plural von *Autonomous System*, *ASe* ist deutscher Sprachgebrauch für den Plural von *Autonomes System*.

[42] Die Grafik zeigt "Figure 1. IPv4 and IPv6 AS Core Jan 2013." aus: CAIDA (Center for Applied Internet Data Analysis): IPv4 and IPv6 AS Core: Visualizing IPv4 and IPv6 Internet Topology at a Macroscopic Scale in 2013, 2013 [182]

TCP-Overlay-Netzwerks - beim Tor-Netzwerk heissen diese Router *Tor-Relays*.

Abbildung 3.2 zeigt die Connectivity der 105 schnellsten *Tor-Relays* am 14.03.2012.

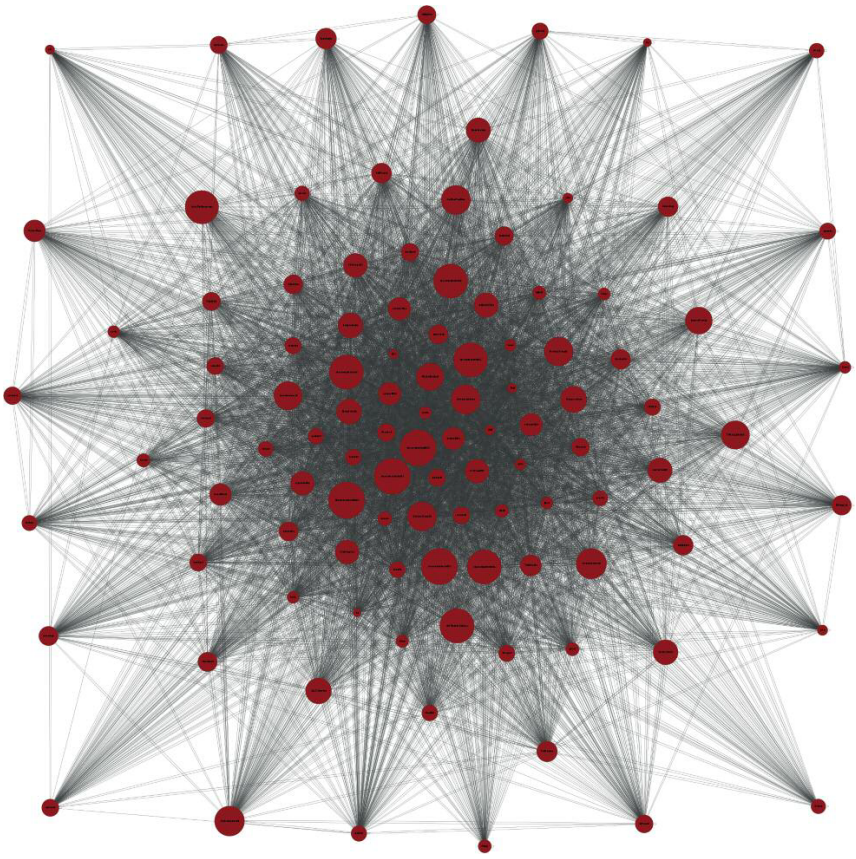

Abbildung 3.2: Connectivity der 105 schnellsten Tor-Relays am 14.03.2012, 13:59 GMT[43]

3.4 Zusammenfassung

Die hauptsächlichen Anforderungen an die Kopplung von LANs (LAN-LAN-Kopplung) sind hohe Kompatibilität für darauf aufbauende Protokolle und Dienste, hohe Verfügbarkeit und Erweiterbarkeit bzw. Anpassbarkeit an den zukünftigen Bedarf. Eine weitere Anforderung ist Interoperabilität, z.B. bezüglich verschiedener Router-Betriebssystemplattformen.

Die Anforderung nach hoher Kompatibilität bei der Kopplung von LANs lässt sich erfüllen durch die Verwendung von (gebräuchlichen) dynamischen Routing-Protokollen, wie sie auch bei Internet Service Providern genutzt werden (siehe Abschnitt 2.1.2).

Eine implizite Anforderung bei der Kopplung von LANs ist die Wahrung der Vertraulichkeit und der Integrität der über die LANs und zwischen den LANs transportierten

[43]Die Grafik zeigt "Figure 2: Connectivity of the 105 fastest Tor relays on March 14th, 2012, 13:59 GMT." aus: Alex Biryukov, Ivan Pustogarov und Ralf-Philipp Weinmann: TorScan: Deanonymizing Connections Using Topology Leaks, In: ERCIM News 90: Special theme: Cybercrime and Privacy Issues, Hrsg. von ERCIM (European Research Consortium for Informatics and Mathematics) EEIG, 2012 [183][184, 29–30]

Daten; diese Anforderung erzwingt die Nutzung von VPN-Software zur Kopplung der LANs.

Die Anforderung nach hoher Verfügbarkeit (*availability*) stösst die Anforderung an, auf mehrere gleichzeitig verfügbare Übertragungswege für die LAN-LAN-Kopplung zurückgreifen zu können, um durch Redundanz die Verfügbarkeit gegenüber der Existenz nur eines Übertragungswegs zu erhöhen.
Die Anforderung nach mehreren gleichzeitig verfügbaren Übertragungswegen zwischen Standorten für die LAN-LAN-Kopplung lässt sich erfüllen durch die Implementierung mehrerer VPN-Verbindungen, welche möglichst auf unterschiedlichen darunterliegenden Transportwegen aufbauen sollten.

Die Implementierung mehrerer *direkter* (paralleler) VPN-Verbindungen über das Internet erfüllt die Forderung nach möglichst unterschiedlichen von den VPN-Verbindungen genutzten Transportwegen *in nicht ausreichendem Umfang*; bei mehreren *direkten* parallelen VPN-Verbindungen zwischen Standorten über das Internet ist die Möglichkeit gegeben, dass z.B. bei Störungen des Routings im Backbone des Internets alle VPN-Verbindungen bezüglich ihres Transportweges durch diese Störungen beeinträchtigt sind.

Bei der LAN-LAN-Kopplung von Standorten mit Hilfe mehrerer paralleler VPN-Verbindungen *über TCP-Overlay-Netzwerke hinweg* ist die Möglichkeit hingegen verringert, dass z.B. bei Störungen des globalen Routings im Internet alle VPN-Verbindungen zwischen den Standorten durch diese Störungen beeinträchtigt sind; dies fusst darauf, dass bei der Verwendung von TCP-Overlay-Netzwerken eine (transitive) Vermaschung von TCP-Overlay-Netzwerkknoten über viele Regionen des Internets erzeugt wird; die Auswirkungen von z.B. Störungen des Routings im Backbone des Internets auf die auf das TCP-Overlay-Netzwerk aufbauenden VPN-Verbindungen werden verringert.

Wünschenswert ist also die Möglichkeit, zusätzliche VPN-Verbindungen zwischen Standorten mit TCP-Overlay-Netzwerken als Transportweg zu realisieren, um damit im Fall von Störungen oder Ausfällen im Internet zusätzliche potentiell verfügbare Transportpfade zwischen den zu koppelnden LANs nutzen zu können.

4 Verwandte Arbeiten

Die folgenden Abschnitte stellen verwandte Arbeiten als Vergleichsgrundlage bezüglich der ermittelten Anforderungen aus Kapitel 3 Analyse vor; sie sollen zugleich auch als Informationsgrundlage für Weiterentwicklungen dienen.

4.1 HORNET

HORNET[1,2] ist ein System, um anonyme Kommunikationskanäle mit hoher Geschwindigkeit zwischen vielen Endpunkten zur Verfügung zu stellen. *HORNET* transportiert Daten in Form von Paketen mit Nutzlast (*payload*) über anonyme Kommunikationskanäle, die es bereitstellt; das System ist ein Onion-Routing-System mit niedriger Latenz.

Die Art der Nutzlast der Pakete sind IP-Pakete; die Art der Nutzlast der Pakete ist jedoch durch das *HORNET*-Protokolldesign nicht explizit auf IP-Pakete beschränkt. Da die Art der Daten der Nutzlast-Pakete generisch, also nicht vom Format her festgelegt ist, ist in *HORNET* keine Funktionalität zum Routing der transportierten Pakete auf der Ebene des IP-Protokolls enthalten.
Das Routing der Pakete wird von den sendenden Endpunkten der Datenübertragungen gesteuert, die den Datenpaketen einen Teil des Pfades über die Knoten, den die Pakete beim Transport nehmen, über einen zusätzlichen Header mitgeben; dieser Header wird *AHDR - anonymous header* - genannt.[3]

Als darunterliegende Transportschicht nutzt *HORNET* Netzwerke, die auf dem Protokoll TCP/IP aufbauen.

HORNET nutzt zur Verschlüsselung des Datenverkehrs sowohl Public-/Private-Key-basierende Verfahren als auch symmetrische Kryptographieverfahren. Zum Etablieren der Sessions für Datenübertragungen zwischen den Endpunkten kommt Public-/Private-Key-basierende Kryptographie in Verbindung mit der Methode des Diffie-Hellman-Schlüsselaustauschs (*DH*) zur Anwendung;[3] für die Sicherung der Datenübertragung der Paket-Nutzlast kommt ein symmetrisches Kryptographieverfahren (AES) zur Anwendung.[3] *HORNET* baut für die kryptographischen Funktionen auf den Softwarebibliotheken *PolarSSL*[4]

[1] Chen Chen, Daniele Enrico Asoni, David Barrera, George Danezis und Adrian Perrig: HORNET: High-speed Onion Routing at the Network Layer, 2016, URL: `https://arxiv.org/pdf/1507.05724v3` [185]
[2] Die Entwicklung von *HORNET* wurde im Rahmen des *7th Framework Programme* des Europäischen Forschungsrats (ERC) der Europäischen Union gefördert.[1]
[3] Chen Chen, Daniele Enrico Asoni, David Barrera, George Danezis und Adrian Perrig: HORNET: High-speed Onion Routing at the Network Layer, 2016, URL: `https://arxiv.org/pdf/1507.05724v3` [185, S. 3]
[4] ARM Limited: SSL Library mbed TLS / PolarSSL, URL: `http://polarssl.org/` [111]

und *curve25519-donna*[5,6] auf; darüber hinaus wird die Softwarebibliothek *AESNI*[7,8] der Firma Intel genutzt.[9]

HORNET ist aufbauend auf dem *Data Plane Development Kit* (*DPDK*)[10];[9] das *Data Plane Development Kit* ist eine frei zugängliche und frei verwendbare Software, die aus Bibliotheken und Treibern für schnelle (generische) Paketverarbeitung besteht.[10]
Das *Data Plane Development Kit* ist primär für das Betriebssystem *Linux* verfügbar, in eingeschränkter Funktionalität ist es auch für das Betriebssystem *FreeBSD* verfügbar; weitere Informationen zur Funktionalität von *DPDK* sind unter der URL `http://dpdk.org/` abrufbar.[10]

Weitere Informationen zur Funktionalität von *HORNET* und Dokumentation zu *HORNET* sind im Dokument "HORNET: High-speed Onion Routing at the Network Layer" unter der URL `https://arxiv.org/pdf/1507.05724v3` abrufbar.[1]

Die vorgestellte verwandte Arbeit deckt bezüglich der Anforderungen an die LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks (VPNs) einen grossen Teil der Overlay-Netzwerk-Funktionalität und einen grossen Teil der VPN-Funktionalität ab; bei der Implementierung der Overlay-Netzwerk-Funktionalität wird hier das Designziel verfolgt, kurze Pfade (*short paths*) des zu Grunde liegenden Transitnetzes zu nutzen;[1] dies geschieht hier, um das übergeordnete Ziel des hohen Datendurchsatzes zu erfüllen. Durch die Nutzung von kurzen Pfaden des zu Grunde liegenden Transitnetzes wird im Gegensatz zur Nutzung von langen Pfaden (*long paths*) die Ausfallsicherheit der hier zur Verfügung gestellten Kommunikationskanäle bei Störungen z.B. des Routings im Backbone des Internets nicht als zu erfüllendes Ziel in den Vordergrund gestellt.
Bei der vorgestellten verwandten Arbeit ist die Funktionalität des Routings auf der Protokoll-Ebene des IP-Pakettransports bzw. des dynamischen Routings auf der Protokoll-Ebene des IP-Pakettransports nicht gegeben.

4.2 OnionCat

OnionCat[11] ist eine frei zugängliche und frei verwendbare Software, die es Computern, die am Overlay-Netzwerk *Tor* teilnehmen, ermöglicht, ein transparentes und anonymes TCP/IP-basierendes Overlay-Netzwerk aufzubauen. Alle Computer, die *OnionCat* nut-

[5] Google Inc.: Google Code Archive: curve25519-donna, URL: `https://code.google.com/archive/p/curve25519-donna/` [186]

[6] Die Softwarebibliothek *curve25519-donna* implementiert die kryptographischen Funktionen des Diffie-Hellman-Schlüsselaustauschs (DH) mit Hilfe von elliptischen Kurven, das Verfahren wird *Elliptic Curve Diffie-Hellman* (*ECDH*) genannt.[5]

[7] Jeffrey Rott, Intel Corporation: Intel AESNI Sample Library, 2011, URL: `https://software.intel.com/en-us/articles/download-the-intel-aesni-sample-library` [187]

[8] Die Softwarebibliothek *AESNI* implementiert die kryptographischen Funktionen des *Advanced Encryption Standard* (*AES*) mit Hilfe von CPU-Instruktionen, die auf einigen Prozessoren des Herstellers Intel verfügbar sind.[7]

[9] Chen Chen, Daniele Enrico Asoni, David Barrera, George Danezis und Adrian Perrig: HORNET: High-speed Onion Routing at the Network Layer, 2016, URL: `https://arxiv.org/pdf/1507.05724v3` [185, S. 12]

[10] DPDK Project: DPDK - Data Plane Development Kit, URL: `http://dpdk.org/` [188]

[11] Bernhard R. Fischer: The OnionCat Project, URL: `http://www.cypherpunk.at/onioncat/` [189]

zen, bilden *ein* grosses gemeinsames VPN.

Der Pakettransport an der Schnittstelle zum benutzten Transitnetz - hier: dem Overlay-Netzwerk *Tor* - basiert auf dem TCP-Protokoll. *OnionCat* nutzt hierzu die vom Overlay-Netzwerk *Tor* bereitgestellte Funktionalität, Kommunikation von einem *Tor*-Netzwerkknoten zu einem anderen *Tor*-Netzwerkknoten mit dem TCP-Protokoll zu bewerkstelligen; *Tor*-Netzwerkknoten können zu diesem Zweck symbolisch adressierbare Endpunkte definieren, die *Hidden Services* genannt werden.[12,13]

OnionCat nutzt zur Verschlüsselung des Datenverkehrs die Funktionalität des zu Grunde liegenden *Tor*-Netzwerks; die Kommunikation der symbolisch adressierbaren TCP-Protokoll-basierenden Endpunkte wird mit jeweils einem *RSA*-Public-Key-/Private-Key-Schlüsselpaar für jeden Endpunkt kryptographisch abgesichert.

OnionCat ist verfügbar für die Betriebssysteme *Debian Linux*, *FreeBSD*, *OpenBSD* und Apple *MacOS/X*, d.h. der Quellcode ist - ohne dass weitere Anpassungen notwendig sind - auf diesen Betriebssystemen übersetzbar. Die Software ist auf den Betriebssystemen *Debian Linux*, *FreeBSD* und *OpenBSD* ohne Modifikationen des jeweiligen Betriebssystemkerns lauffähig und funktional; das fußt auf dem Umstand, dass für die Bereitstellung der Funktionalität der benötigten virtuellen Ethernet-Schnittstelle generische *tap*-Devices bzw. generische *tun*-Devices verwendet werden.
Für die Betriebssysteme Microsoft *Windows* und Apple *MacOS/X* ist *OnionCat* verfügbar, bei diesen Betriebssystemen wird jedoch zusätzlich Software für die Bereitsstellung einer generischen, virtuellen Netzwerkschnittstelle - Treibersoftware für generische *tap*-Devices bzw. generische *tun*-Devices - benötigt; diese ist verfügbar.
Beim Betriebssystem Microsoft *Windows* wird zudem die frei verfügbare Umgebung *Cygwin*[14] benötigt, um ein API (*Application Programming Interface*) unter Windows zur Verfügung zu stellen, welches mit dem API von POSIX[15,16] vergleichbar ist.

Die aktuelle Version der Software ist Version 0.2.2.r578 vom 02.05.2017 (Stand 07.07.2017); die Versionsnummerierung legt den Schluss nahe, dass die Software in einer frühen Phase des Systemlebenszyklus ist. Es ist nicht eindeutig, ob die Software in der Entwicklungsphase oder in der Nutzungsphase des Systemlebenszyklus ist. Der Quellcode der Software ist unter der URL `https://www.cypherpunk.at/ocat/download/Source/0.2.2/` abrufbar;[17] des Weiteren ist das Entwicklungs-Repository des Quellcodes von *OnionCat* unter der URL `https://www.cypherpunk.at/onioncat_trac/browser` zugreifbar. Der Quellcode der Software - im Detail: das Build-Script `trunk/android/android_configure` im aktuellen Entwicklungszweig der Software im Entwicklungs-Repository - lässt den Schluss zu, dass an einer zumindest experimentellen Portierung der Software für das Betriebssystem Google *Android* entwickelt wird.

[12] The Tor Project Inc.: Tor Project: Documentation - Hidden Service Protocol, URL: `https://www.torproject.org/docs/hidden-services.html.en` [190]
[13] The Tor Project Inc.: Tor Project: Documentation - Hidden Service Configuration Instructions, URL: `https://www.torproject.org/docs/tor-hidden-service.html.en` [191]
[14] Cygwin Project: The Cygwin Project Page, URL: `http://www.cygwin.com/` [192]
[15] The Open Group: ISO/IEC 9945 - Portable Operating System Interface (POSIX), URL: `http://www.unix.org/version3/iso_std.html` [193]
[16] The Open Group: IEEE Portable Applications Standards Committee (PASC Committee), URL: `https://collaboration.opengroup.org/external/pasc.org/plato/#POSIX` [194]
[17] Bernhard R. Fischer und Ferdinand Haselbacher: The OnionCat Download Page, URL: `http://www.cypherpunk.at/ocat/download` [195]

Informationen zur Funktionalität von *OnionCat* und Dokumentation zu *OnionCat* sind unter der URL `http://www.cypherpunk.at/onioncat/` abrufbar; weiterführende Dokumentation zum Design von *OnionCat* ist unter der URL `https://www.cypherpunk.at/ocat/download/Docs/` abrufbar.

Die vorgestellte verwandte Arbeit deckt bezüglich der Anforderungen an die LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks (VPNs) einen sehr grossen Teil der Overlay-Netzwerk-Funktionalität und einen grossen Teil der VPN-Funktionalität ab; der Overlay-Netzwerk-Funktionalität liegt hier die Overlay-Netzwerk-Funktionalität des Tor-Netzwerks als Transitnetz zu Grunde; die Overlay-Netzwerk-Funktionalität ist in weniger grossem Mass bei Störungen z.B. des Routings im Backbone des Internets beeinträchtigt.
Die VPN-Funktionalität erfüllt hier das Designziel *eines* grossen gemeinsamen Virtual Private Networks (VPNs); die Anforderung an die LAN-LAN-Kopplung mit Virtual Private Networks ist hingegen die Nutzung *eines exklusiven* Virtual Private Networks für die zu koppelnden LANs.

Bei der vorgestellten verwandten Arbeit ist die Funktionalität des Routings auf der Protokoll-Ebene des IP-Pakettransports bzw. des dynamischen Routings auf der Protokoll-Ebene des IP-Pakettransports nicht gegeben.

5 Konzeption

Die folgenden Abschnitte legen die Konzeption für eine Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks (VPNs) dar.

Bei der Kopplung von LANs mit Hilfe von Virtual Private Networks ergibt es Sinn, durch mehrfache Auslegung der Virtual Private Networks - durch Redundanz -, die Verfügbarkeit der LAN-LAN-Kopplung zu erhöhen.

Diese Form der Erhöhung der Verfügbarkeit bei der LAN-LAN-Kopplung durch mehrfache Auslegung der Virtual Private Networks (VPNs) veranschaulicht Abbildung 5.1.

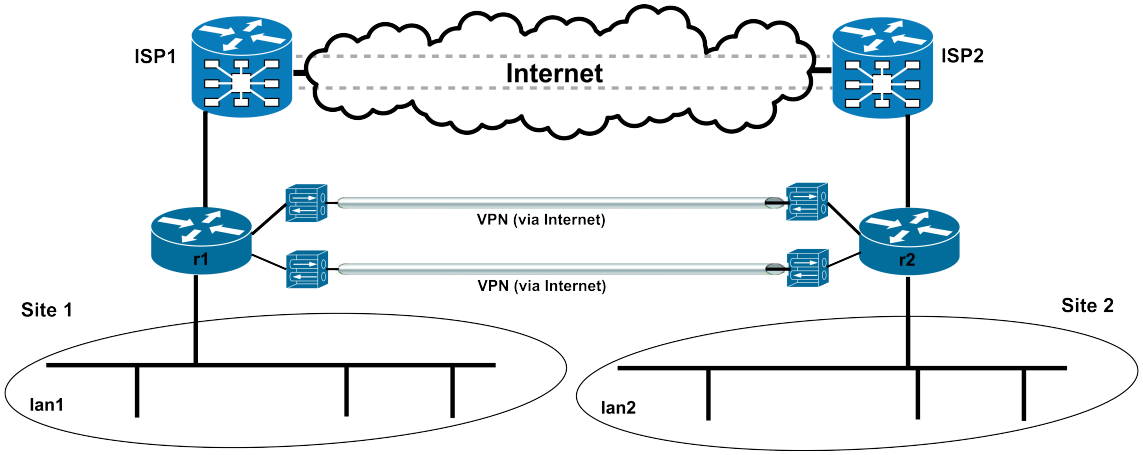

Abbildung 5.1: LAN-LAN-Kopplung mit mehrfacher Auslegung der VPNs

Bei der einfachsten Form der LAN-LAN-Kopplung durch mehrfache Auslegung der Virtual Private Networks wird die Verfügbarkeit dadurch erhöht, dass eines der beiden VPNs die LAN-LAN-Kopplung realisiert und beim Ausfall dieses VPNs das zweite VPN diese Funktion zeitweilig für die Dauer des Ausfalls des ersten VPNs übernimmt.[1] Das zeitweilige Übernehmen der Funktion des ersten VPNs durch das zweite VPN bei Ausfall des ersten ist bedingt durch den Einsatz von statischem Routing mit unterschiedlichen Metriken erst dann möglich, wenn der Ausfall des ersten VPNs feststeht; dieser Vorgang kann unter der Massgabe der möglichst ununterbrochenen Verfügbarkeit eine inakzeptable Zeitspanne in Anspruch nehmen.

Um die durch die mehrfache Auslegung der Virtual Private Networks zur Verfügung stehenden redundanten Transportpfade zum Zweck einer möglichst ununterbrochenen Verfügbarkeit der LAN-LAN-Kopplung nutzen zu können, wird ein dynamisches Routing-Protokoll eingesetzt.

[1] Diese Form der LAN-LAN-Kopplung mit Hilfe von VPNs nutzt statisches Routing zur Abbildung der beschriebenen Funktionalität; es wird auf den Routern jeweils statisches Routing mit unterschiedlichen Metriken für die VPNs genutzt. Die Funktion des statischen Routings mit unterschiedlichen Metriken für die VPNs erfüllt das auf den Routern eingesetzte Betriebssystem.

Als dynamisches Routing-Protokoll zu diesem Zweck wird das dynamische Routing-Protokoll *OSPF* (*Open Shortest Path First*) gewählt (siehe Abschnitt 3.2 Geeignete Routing-Protokolle).

Die Wahl des dynamischen Routing-Protokolls *OSPF* fusst zum Einen auf der Eigenschaft dieses dynamischen Routing-Protokolls, gute Rekonvergenz-Eigenschaften zu besitzen - d.h. dieses Protokoll passt rasch die Routing-Tabellen an, um alternative Pfade für das Routing zu realisieren; zum Anderen wird das dynamische Routing-Protokoll *OSPF* bevorzugt in Unternehmens-Netzwerken eingesetzt (siehe Abschnitt 2.1.2.2). Des Weiteren ist das dynamische Routing-Protokoll *OSPF* häufig in verfügbarer Routing-Software im Internet implementiert (siehe Abschnitt 2.1.3 Routing-Software im Internet).
Eine zusätzliche erwünschte Eigenschaft des dynamischen Routing-Protokolls *OSPF* ist die unter bestimmten Voraussetzungen gegebene Funktionalität, Mehrweg-Routing (*Equal Cost MultiPath - ECMP*) zu ermöglichen[2]; bei Mehrweg-Routing (*Equal Cost MultiPath - ECMP*) kann der Transport von IP-Datenpaketen über mehrere Wege verteilt werden (*load balancing*).

Kapitel 3 Analyse legt dar, dass es wünschenswert ist, durch weitere Erhöhung der Redundanz die Verfügbarkeit der LAN-LAN-Kopplung nochmals zu verbessern, indem weitere *unabhängige* Transportpfade für die VPN-Verbindungen hinzugefügt werden; weitere unabhängige Transportpfade für die VPN-Verbindungen werden mit Hilfe eines TCP-Overlay-Netzwerks realisiert.

5.1 Übersicht

Die Übersicht in Abbildung 5.2 auf Seite 57 veranschaulicht die Konzeption der redundanten und hochverfügbaren LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks (VPNs).

Die Realisierung der redundanten und hochverfügbaren LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks (VPNs) bedingt Hardware und Software.

Als Hardware werden für die Laborumgebung zur Software-Entwicklung Standard-PCs ausgewählt; als Hardware für die Produktivumgebung werden aus Standard-Komponenten bestehende Rack-Serversysteme mit 1 HE (1 Höhen-Einheit) in 19”-Einbaurahmen in den Rechenzentren genutzt.

Die benötigten aus Software bestehenden Komponenten sind:

- die *Betriebssysteme der Router*,
- die *Software für das dynamische Routing*,
- die *Software zur Realisierung der VPN-Funktionalität*,
- *Software zur Realisierung von Overlay-Netzwerkknoten* für das gewählte TCP-Overlay-Netzwerk,
- eine zu entwickelnde *Laufzeitumgebung zum Steuern, Überwachen und Protokollieren*.

[2] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 476]

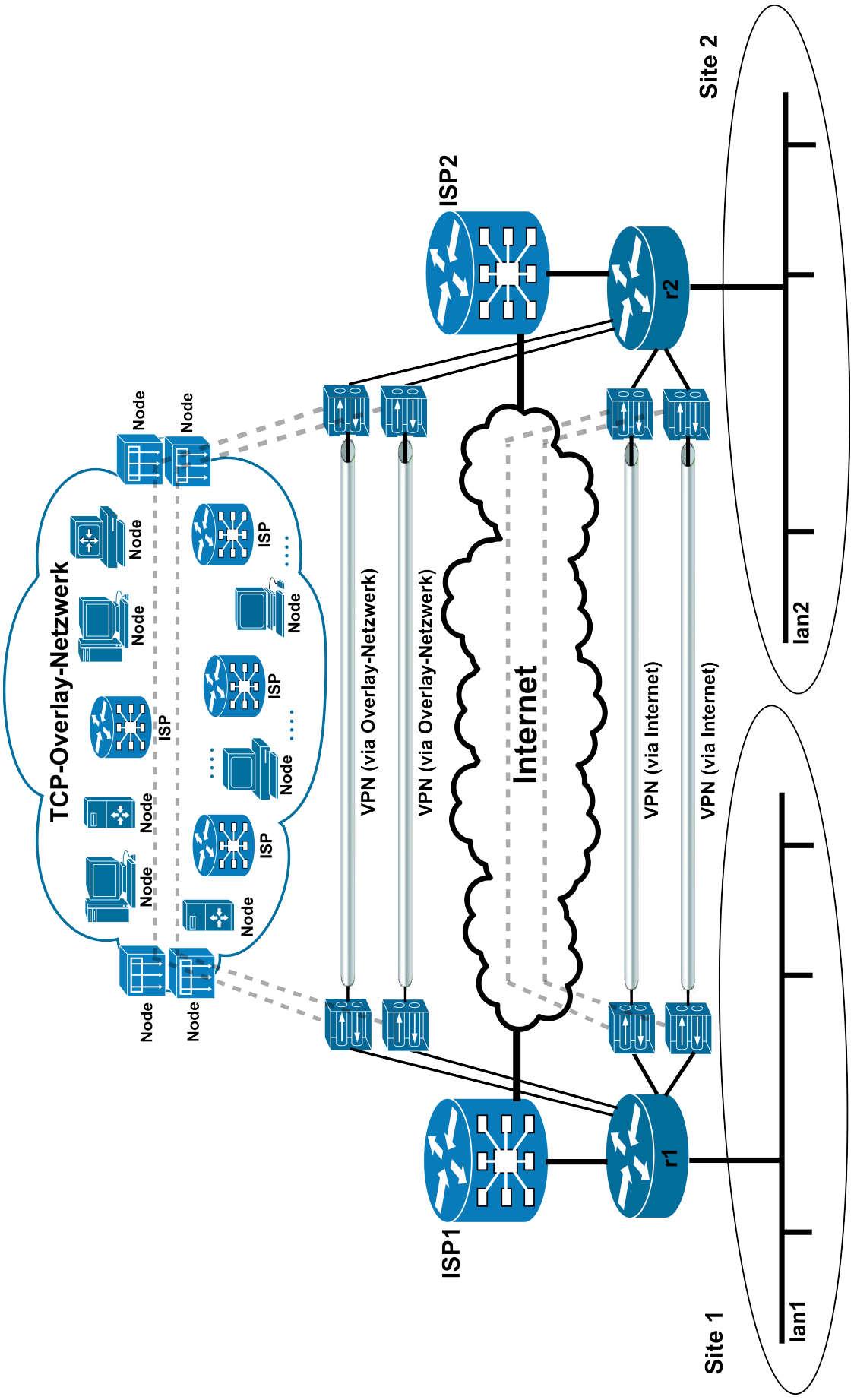

Abbildung 5.2: Übersicht der Konzeption der LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und VPNs

5.2 Reihenfolge bei der Auswahl der Komponenten

Die benötigten Software-Komponenten für die LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks (VPNs) bauen aufeinander auf und hängen voneinander ab.

Bei der Auswahl der Software-Komponenten im Vordergrund stehend sind gemeinsame Schnittstellen der Software-Komponenten.

Die Reihenfolge der Auswahl der Software-Komponenten orientiert sich an den Freiheitsgraden bei die Auswahl der jeweiligen Software-Komponente; es existieren Abhängigkeiten bei der Auswahl, ein Beispiel für eine Abhängigkeit bei der Auswahl der Software-Komponenten ist die Verfügbarkeit einer geeigneten VPN-Software für ein geeignetes Router-Betriebssystem.

Die Auswahl des dynamischen Routing-Protokolls ist bereits auf das dynamische Routing-Protokoll *OSPF* festgelegt (siehe Abschnitt 3.2 Geeignete Routing-Protokolle); aus der Auswahl des Routing-Protokolls resultiert die Auswahl der Routing-Software.

5.3 Auswahl der Routing-Software

Als zu verwendende Routing-Software steht solche Routing-Software zur Auswahl, welche das dynamische Routing-Protokoll *OSPF*[3,4] implementiert (siehe Abschnitt 2.1.2.2).

5.3.1 Kriterien für die Auswahl

Die wichtigsten Kriterien für die Auswahl der Routing-Software sind:

- die *stabile* und *standard-konforme* Implementierung des Routing-Protokolls *OSPF*,
- die Verfügbarkeit der Routing-Software im Quellcode (*Open-Source-Software*),
- die Lauffähigkeit der Routing-Software auf möglichst vielen Betriebssystem-Plattformen.

Das Kriterium der stabilen Implementierung des Routing-Protokolls *OSPF* leitet sich vom Ziel der hohen Verfügbarkeit der darauf aufbauenden LAN-LAN-Kopplung ab.

Das Kriterium der standard-konformen Implementierung des Routing-Protokolls *OSPF* leitet sich vom abstrakten Ziel der Kompatibilität mit anderen (zukünftigen) Implementierungen der Routerumgebung mit Hilfe von anderer Routing-Software ab, die das Routing-Protokoll *OSPF* implementiert.

Das Kriterium der Verfügbarkeit der auszuwählenden Routing-Software im Quellcode begründet sich mit der Forderung nach freier Zugänglichkeit und nach freier Verwendbarkeit der Routing-Software und mit der Forderung nach Überprüfbarkeit der Standard-Konformität und der Sicherheit der Routing-Software.

[3] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "OSPF" [8, S. 601]
[4] RFC 2328: OSPF Version 2, 1998, Internet Standard,
URL: `http://www.rfc-editor.org/rfc/rfc2328.txt` [25]

Das Kriterium der Lauffähigkeit der auszuwählenden Routing-Software auf möglichst vielen Betriebssystem-Plattformen leitet sich vom abstrakten Ziel der möglichst einfachen Portierbarkeit der Routerumgebung auf andere Betriebssysteme ab.

Optional und wünschenswert wäre die Existenz eines API (*Application Programming Interface*) der auszuwählenden Routing-Software, um feingranuliert das Routing zu steuern.

Das Auswahlkriterum der stabilen und standard-konformen Implementierung des Routing-Protokolls OSPF durch die Routing-Software *und* das Auswahlkriterum der Verfügbarkeit der Routing-Software in Form ihres Quellcodes *und* das Auswahlkriterum der Lauffähigkeit der Routing-Software auf möglichst vielen Betriebssystemplattformen wird durch zwei Routing-Softwares erfüllt: zum Einen durch die Routing-Software *Quagga* (siehe Abschnitt 2.1.3.4 Quagga), zum Anderen durch die Routing-Software *BIRD* (siehe Abschnitt 2.1.3.5 BIRD).

Die Routing-Software *Quagga* - eine *Routing Software Suite* - ist in Verbindung mit den Betriebssystemen *Linux*, *FreeBSD*, *NetBSD* und *Solaris* als Routing-Software einsetzbar;[5] die Routing-Software *BIRD* - ein *Routing Daemon* - ist in Verbindung mit den Betriebssystemen *Linux*, *FreeBSD*, *NetBSD* und *OpenBSD* als Routing-Software einsetzbar.[6]

Die Routing-Software *Quagga* hat gegenüber der Routing-Software *BIRD* die hervorzuhebende Eigenschaft eines Kommandozeileninterpreters, welcher eine ähnliche Semantik, Grammatik und Syntax aufweist wie die Systemsoftware *IOS* für die Router des Herstellers Cisco Systems (siehe Abschnitt 2.1.3.1 Cisco Systems "IOS"); dies gibt den Ausschlag für die Wahl der Routing-Software *Quagga* zur Verwendung als Routing-Software für das dynamische Routing mit dem dynamischen Routing-Protokoll OSPF.

5.4 Auswahl des Overlay-Netzwerks

Als zu verwendendes Overlay-Netzwerk (siehe Abschnitt 2.3 Overlay-Netzwerke) steht ein TCP-Overlay-Netzwerk zur Auswahl - ein Overlay-Netzwerk, dass primär dem Transport von Datenströmen mit niedriger Latenz dient.

Die wichtigsten Kriterien für das auszuwählende TCP-Overlay-Netzwerk und für die auszuwählende Software für Overlay-Netzwerkknoten dieses TCP-Overlay-Netzwerks sind:

- das TCP-Overlay-Netzwerk sollte bestehen - d.h. es sollten in ausreichender Anzahl Instanzen von Overlay-Netzwerkknoten dieses Overlay-Netzwerks existieren -,
- es sollte eine übliche gemeinsame Schnittstelle zur auf der TCP-Overlay-Netzwerk-Software aufbauenden VPN-Software existieren,
- die Verfügbarkeit der Software für einen Overlay-Netzwerkknoten im Quellcode (*Open-Source-Software*) sollte gegeben sein,
- die Lauffähigkeit der Software für einen Overlay-Netzwerkknoten auf möglichst vielen Betriebssystem-Plattformen sollte gegeben sein,

[5] Quagga Software Routing Suite, URL: `http://www.nongnu.org/quagga/` [58]
[6] The BIRD Internet Routing Daemon Project, URL: `http://bird.network.cz/` [65]

- die Software für einen Overlay-Netzwerkknoten des TCP-Overlay-Netzwerks sollte leichtgewichtig (*lightweight*) sein, d.h. die Software sollte die Eigenschaften einer geringen Prozessorauslastung und eines geringen Hauptspeicherbedarfs (*memory footprint*) besitzen.

Das Kriterium, dass in ausreichender Anzahl Instanzen von Overlay-Netzwerkknoten des TCP-Overlay-Netzwerks existieren sollten, leitet sich vom Ziel der hohen Verfügbarkeit der darauf aufbauenden Virtual Private Networks (VPNs) zur Realisierung der LAN-LAN-Kopplung ab; durch eine ausreichende Anzahl Instanzen von Overlay-Netzwerkknoten des TCP-Overlay-Netzwerks wird eine (transitive) Vermaschung der Knoten des Overlay-Netzwerks über viele Regionen des Internets erzeugt (siehe Abschnitt 3.4).

Das Kriterium, dass eine übliche gemeinsame Schnittstelle zur auf der TCP-Overlay-Netzwerkknoten-Software aufbauenden VPN-Software existieren sollte, ist darin begründet, dass die Notwendigkeit von Anpassungen an der TCP-Overlay-Netzwerkknoten-Software und die Notwendigkeit von Anpassungen an der VPN-Software nicht erwünscht ist; dies wiederum fusst auf dem Motiv, nach Möglichkeit vorbereitete Pakete der TCP-Overlay-Netzwerkknoten-Software und der VPN-Software nutzen zu können.

Das Kriterium der Verfügbarkeit der Overlay-Netzwerkknoten-Software des auszuwählenden TCP-Overlay-Netzwerks in Form ihres Quellcodes begründet sich mit der Forderung nach freier Zugänglichkeit und nach freier Verwendbarkeit der TCP-Overlay-Netzwerkknoten-Software und mit der Forderung nach Überprüfbarkeit der Sicherheit der TCP-Overlay-Netzwerkknoten-Software.

Das Kriterium der Lauffähigkeit der Overlay-Netzwerkknoten-Software des auszuwählenden TCP-Overlay-Netzwerks auf möglichst vielen Betriebssystem-Plattformen leitet sich vom abstrakten Ziel der möglichst einfachen Portierbarkeit der Routerumgebung auf andere Betriebssysteme ab.

Das Kriterium der Forderung an die Overlay-Netzwerkknoten-Software des auszuwählenden TCP-Overlay-Netzwerks, leichtgewichtig (*lightweight*) zu sein, liegt im Motiv der Skalierbarkeit durch Bereitstellung *mehrerer* TCP-Overlay-Netzwerkknoten begründet; dies ist bei einer geringen Prozessorauslastung und bei einem geringen Hauptspeicherbedarf eines TCP-Overlay-Netzwerkknotens einfacher zu bewerkstelligen.

Das Auswahlkriterium der ausreichenden Anzahl von Instanzen von Overlay-Netzwerkknoten des Overlay-Netzwerks wird vom Overlay-Netzwerk *I2P*[7] in hohem Mass und vom Overlay-Netzwerk *Tor*[8,9] in sehr hohem Mass erfüllt (siehe Abschnitt 3.4).

Das Auswahlkriterium, dass eine übliche gemeinsame Schnittstelle zur auf der TCP-Overlay-Netzwerkknoten-Software aufbauenden VPN-Software existieren sollte, wird sowohl von der Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *I2P* (siehe Abschnitt 2.3.1.2 I2P) als auch von der Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *Tor* (siehe Abschnitt 2.3.1.1 Tor) erfüllt; beide Softwares stellen für darauf aufbauende Anwendungen die TCP-Relay-Funktionalität durch Implementierung des

[7] I2P Project: The Invisible Internet Project: I2P Anonymous Network, URL: `https://geti2p.net/` [155]
[8] The Tor Project Inc.: Tor Project: Anonymity Online URL: `https://www.torproject.org/` [150]
[9] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151][152, 303–320]

SOCKS-Protokolls[10,11] zur Verfügung.

Das Auswahlkriterium der Lauffähigkeit der Overlay-Netzwerkknoten-Software des auszuwählenden TCP-Overlay-Netzwerks auf möglichst vielen Betriebssystemen wird sowohl von der Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *I2P* als auch von der Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *Tor* in hohem Mass erfüllt; die Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *I2P* ist auf Betriebssystemen lauffähig, auf denen eine Laufzeitumgebung der Programmiersprache Java (*JRE - Java Runtime Environment*) in Version 7 oder in neuerer Version verfügbar ist;[12] die Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *Tor* ist auf den Betriebssystemen Microsoft *Windows*, *Linux*, *Solaris*, auf *Unix*-artigen Betriebssystemen der *BSD*-Betriebssystemfamilie wie *FreeBSD*, *OpenBSD* und *NetBSD* und auf dem Betriebssystem Apple *MacOS/X* lauffähig[13].

Das Auswahlkriterium der Forderung an die Overlay-Netzwerkknoten-Software des auszuwählenden TCP-Overlay-Netzwerks, leichtgewichtig (*lightweight*) zu sein, wird von der Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *I2P nicht* erfüllt; von der Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *Tor* wird die Forderung, leichtgewichtig (*lightweight*) zu sein, in hohem Mass erfüllt; die Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *I2P* wird durch die Laufzeitumgebung der Programmiersprache Java (*JRE - Java Runtime Environment*) ausgeführt,[12,14] die Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *Tor* hingegegen ist in der Programmiersprache C geschrieben und es wird nativer Maschinencode ausgeführt.[13,15]

Die Eigenschaft der Overlay-Netzwerkknoten-Software des Overlay-Netzwerks *Tor*, leichtgewichtig (*lightweight*) zu sein, forciert die Wahl des Overlay-Netzwerks *Tor*; dies liegt im Motiv der Skalierbarkeit durch die Möglichkeit der Bereitstellung mehrerer TCP-Overlay-Netzwerkknoten begründet.

5.5 Auswahl des VPN-Protokolls

Als zu verwendendes VPN-Protokoll (siehe Abschnitt 2.2 Virtual Private Networks) steht ein VPN-Protokoll und daraus resultierend eine Implementierung des VPN-Protokolls in Form von VPN-Software zur Auswahl, welche es in Verbindung mit dem dynamischen Routing-Protokoll OSPF ermöglicht, LANs redundant und mit hoher Verfügbarkeit miteinander zu koppeln.

Die wichtigsten Kriterien für die Auswahl des VPN-Protokolls und für die Auswahl einer Implementierung des VPN-Protokolls in Form von VPN-Software sind:

[10] *SOCKS* ist ein Protokoll für *Circuit Level Proxies*; eine Anwendung nutzt einen *SOCKS*-Server als Stellvertreter für den Pakettransport, die nutzende Anwendung wird *SOCKS*-Client genannt.[11]
[11] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "SOCKS" [8, S. 779]
[12] I2P Project: The Invisible Internet Project: I2P Anonymous Network - Download, URL: `https://geti2p.net/en/download` [175]
[13] The Tor Project Inc.: Tor Project : Download for Linux/Unix, URL: `https://www.torproject.org/download/download-unix.html.en` [166]
[14] I2P Project: The Invisible Internet Project: I2P Anonymous Network - New Developer's Guide, URL: `https://geti2p.net/en/get-involved/guides/new-developers` [176]
[15] The Tor Project Inc.: Tor Project: Download Tor - Source Code, URL: `https://www.torproject.org/download/download.html.en#source` [162]

- das VPN-Protokoll und die auszuwählende Implementierung des VPN-Protokolls in Form von VPN-Software sollten verbreitet sein,
- die auszuwählende Implementierung des VPN-Protokolls in Form von VPN-Software sollte eine übliche gemeinsame Schnittstelle mit der TCP-Overlay-Netzwerk-Software aufweisen (siehe Abschnitt 5.4), auf die sie zur Realisierung zusätzlicher Transportpfade aufbaut,
- die auszuwählende Implementierung des VPN-Protokolls in Form von VPN-Software sollte im Quellcode vorliegen (*Open-Source-Software*),
- die auszuwählende Implementierung des VPN-Protokolls in Form von VPN-Software sollte auf möglichst vielen Betriebssystem-Plattformen verfügbar sein.

Das Kriterium der Verbreitung des VPN-Protokolls und der Verbreitung von Implementierungen des VPN-Protokolls in Form von VPN-Software leitet sich von einer hohen zu erwartenden Stabilität und einer hohen zu erwartenden Sicherheit des VPN-Protokolls und der VPN-Software im Fall der weiten Verbreitung ab.

Das Kriterium, dass eine übliche gemeinsame Schnittstelle der VPN-Software zur TCP-Overlay-Netzwerkknoten-Software existieren sollte, auf deren bereitgestellten Transportpfaden die VPN-Software aufbaut, ist darin begründet, dass die Notwendigkeit von Anpassungen an der VPN-Software und die Notwendigkeit von Anpassungen an der TCP-Overlay-Netzwerkknoten-Software nicht erwünscht ist (siehe Abschnitt 5.4); dies wiederum fusst auf dem Motiv, möglichst vorbereitete Pakete der VPN-Software und der TCP-Overlay-Netzwerkknoten-Software nutzen zu können.

Das Kriterium der Verfügbarkeit der VPN-Software in Form ihres Quellcodes begründet sich mit der Forderung nach freier Zugänglichkeit und nach freier Verwendbarkeit der VPN-Software und mit der Forderung nach Überprüfbarkeit der Sicherheit der VPN-Software und des mit ihrer Hilfe implementierten VPN-Protokolls.

Das Kriterium der Lauffähigkeit der VPN-Software auf möglichst vielen Betriebssystem-Plattformen leitet sich vom abstrakten Ziel der möglichst einfachen Portierbarkeit der Routerumgebung auf andere Betriebssysteme ab.

Das Auswahlkriterium der Verbreitung des VPN-Protokolls und der Verbreitung von Implementierungen des VPN-Protokolls in Form von VPN-Software ist in hohem Mass sowohl vom VPN-Protokoll *IPsec*[16,17,18] als auch in hohem Mass vom VPN-Protokoll *OpenVPN*[19,20] erfüllt (siehe Abschnitt 2.2.6 De-facto-Standards bei VPN-Protokollen).

Das Auswahlkriterium, dass eine übliche gemeinsame Schnittstelle der VPN-Software zur TCP-Overlay-Netzwerkknoten-Software existieren sollte, auf deren bereitgestellten

[16] RFC 4301: Security Architecture for the Internet Protocol, 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4301.txt` [88]
[17] RFC 3193: Securing L2TP using IPsec, 2001, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3193.txt` [95]
[18] RFC 4719: Transport of Ethernet Frames over Layer 2 Tunneling Protocol Version 3 (L2TPv3), 2006, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4719.txt` [93]
[19] OpenVPN Inc.: OpenVPN - Open Source VPN, URL: `https://openvpn.net/` [112]
[20] Charlie Hosner: OpenVPN and the SSL VPN Revolution, SANS Institute, 2004, URL: `https://www.sans.org/reading-room/whitepapers/vpns/openvpn-ssl-vpn-revolution-1459` [134, S. 5]

Transportpfaden die VPN-Software aufbaut, wird von Implementierungen des VPN-Protokolls *IPsec nicht* erfüllt, hingegen von der Standard-Implementierung des VPN-Protokolls *OpenVPN* in hohem Mass erfüllt; Implementierungen des VPN-Protokolls *IPsec* besitzen keine unmittelbare gemeinsame Schnittstelle mit der TCP-Overlay-Netzwerkknoten-Software, da das VPN-Protokoll *IPsec* für den Pakettransport auf dem UDP-Protokoll basiert (siehe Abschnitt 2.2.4.1 IPsec), die TCP-Overlay-Netzwerkknoten-Software stellt jedoch Transportpfade für das TCP-Protokoll bereit (siehe Abschnitt 5.4). Die VPN-Software *OpenVPN* besitzt hingegen eine gemeinsame Schnittstelle mit der TCP-Overlay-Netzwerkknoten-Software, die VPN-Software *OpenVPN* kann die bereitgestellten Transportpfade der TCP-Overlay-Netzwerkknoten-Software mit Hilfe des *SOCKS*-Protokolls[21,22] nutzen (siehe Abschnitt 2.2.5.5 OpenVPN).

Die bislang angewandten Auswahlkriterien zur Auswahl des VPN-Protokolls und zur Auswahl der Implementierung des VPN-Protokolls in Form von VPN-Software grenzen die Auswahl auf das VPN-Protokoll *OpenVPN* und dessen Standard-Implementierung in Form der VPN-Software *OpenVPN*[23] ein.

Das weiter anzuwendende Auswahlkriterium, dass die auszuwählende Implementierung des VPN-Protokolls in Form von VPN-Software im Quellcode vorliegen sollte (*Open-Source-Software*), ist bei der VPN-Software *OpenVPN*[23] gegeben.

Das weiter anzuwendende Auswahlkriterium, dass die auszuwählende Implementierung des VPN-Protokolls in Form von VPN-Software auf möglichst vielen Betriebssystemplattformen verfügbar sein sollte, ist bei der VPN-Software *OpenVPN* gegeben; die VPN-Software *OpenVPN* ist verfügbar für die Betriebssysteme *Linux*, *OpenBSD*, *FreeBSD*, *NetBSD*, Apple *MacOS/X*, *Solaris* sowie Microsoft *Windows*.[23,24]

Das durch die VPN-Software *OpenVPN* erfüllte Auswahlkriterium, dass eine gemeinsame Schnittstelle der VPN-Software zur TCP-Overlay-Netzwerkknoten-Software existieren sollte, forciert die Wahl der VPN-Software *OpenVPN* als einzusetzende VPN-Software; dies liegt im Motiv begründet, dass die Notwendigkeit von Anpassungen an der VPN-Software nicht erwünscht ist; dies fusst wiederum auf dem Motiv, möglichst vorbereitete Pakete der VPN-Software nutzen zu können.

5.6 Auswahl des Router-Betriebssystems

Als zu verwendendes Router-Betriebssystem steht ein solches Betriebssystem zur Auswahl, welches in Verbindung mit einer geeigneten Routing-Software (siehe Abschnitt 5.3 Auswahl der Routing-Software) mit dem dynamischen Routing-Protokoll OSPF[25,26] für das IP-Routing geeignet ist.

[21] *SOCKS* ist ein Protokoll für *Circuit Level Proxies*; eine Anwendung nutzt einen *SOCKS*-Server als Stellvertreter für den Pakettransport, die nutzende Anwendung wird *SOCKS*-Client genannt.[22]
[22] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "SOCKS" [8, S. 779]
[23] OpenVPN Inc.: OpenVPN - Open Source VPN, URL: `https://openvpn.net/` [112]
[24] OpenVPN Inc.: OpenVPN - Community Project Overview, URL: `https://openvpn.net/index.php/open-source/245-community-open-source-software-overview.html` [122]
[25] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "OSPF" [8, S. 601]
[26] RFC 2328: OSPF Version 2, 1998, Internet Standard,
URL: `http://www.rfc-editor.org/rfc/rfc2328.txt` [25]

Die wichtigsten Kriterien für die Auswahl des Router-Betriebssystems sind:

- das Router-Betriebssystems sollte verbreitet sein,
- das Router-Betriebssystem sollte im Quellcode vorliegen (*Open-Source-Software*),
- das Router-Betriebssystem sollte für möglichst viele Hardware-Plattformen und Maschinen-Architekturen verfügbar sein.

Das Kriterium der Verbreitung des Router-Betriebssystems leitet sich von einer hohen zu erwartenden Stabilität und einer hohen zu erwartenden Sicherheit des Router-Betriebssystems im Fall der weiten Verbreitung ab.

Das Kriterium der Verfügbarkeit des Router-Betriebssystems im Quellcode (*Open-Source-Software*) begründet sich mit der Forderung nach freier Zugänglichkeit und nach freier Verwendbarkeit des Router-Betriebssystems und mit der Forderung nach Überprüfbarkeit der Sicherheit des Router-Betriebssystems.

Das Kriterium der Verfügbarkeit des Router-Betriebssystems für möglichst viele Hardware-Plattformen bzw. Maschinen-Architekturen leitet sich vom abstrakten Ziel ab, andere (zukünftige) Implementierungen der Routerumgebung mit Hilfe von auf das Routing spezialisierten Hardware-Plattformen zu realisieren.

Das Auswahlkriterium der Verbreitung des Betriebssystems als Router-Betriebssystem ist in hohem Mass sowohl beim Betriebssystem *Linux* als auch beim Betriebssystem *FreeBSD* als auch beim Betriebssystem *OpenBSD* erfüllt; im Report "2011 Report on European IXPs (Euro-IX IXP Report 2011)" der European Internet Exchange Association (*Euro-IX*) aus dem Jahr 2012 werden die Betriebsysteme der genutzten *Route Server* bei den Europäischen Internet-Austauschpunkten nach Anteil aufgeschlüsselt[27] (siehe Tabelle 2.4 , Abschnitt 2.1.3 Routing-Software im Internet), der Anteil des Betriebssystems *Linux* ist dort 43%, der Anteil des Betriebssystems *FreeBSD* ist dort 20% und der Anteil des Betriebssystems *OpenBSD* ist dort 23%; der Anteil der Verwendung des Betriebssystems *Linux* schlüsselt sich weiter auf in *Debian Linux* mit 35%, in *Fedora Linux* mit 1%, in *Gentoo Linux* mit 6% und in *Ubuntu Linux* mit 1% Anteil.

Das Auswahlkriterium der Verbreitung des Betriebssystems *Linux* als Router-Betriebssystem in Verbindung mit der Verbreitung des Betriebssystems *Linux* an sich legt die Wahl des Betriebssystems *Linux* nah.

Die Voraussetzung für die Wahl des Betriebssystems *Linux* als Router-Betriebssystem, dass alle vorgenannten Komponenten unter dem Betriebssystem *Linux* lauffähig sind, ist gegeben (siehe Abschnitt 5.3 Auswahl der Routing-Software, siehe Abschnitt 5.4 Auswahl des Overlay-Netzwerks, siehe Abschnitt 5.5 Auswahl des VPN-Protokolls).

Das Auswahlkriterium, dass das Router-Betriebssystem im Quellcode vorliegen sollte (*Open-Source-Software*), wird von den Betriebssystemen *Debian Linux*, *Fedora Linux*, *Gentoo Linux*, *Ubuntu Linux*, *FreeBSD* und *OpenBSD* erfüllt.

[27] Bijal Sanghani, European Internet Exchange Association: 2011 Report on European IXPs (Euro-IX IXP Report 2011), URL: `https://www.euro-ix.net/m/cms_page_media/82/Euro-IX%20IXP%20Report%202011.pdf` [33, S. 13-14]

Die Erfüllung der vorgenannten Auswahlkriterien durch das Betriebssystem *Debian Linux* und der höchste Anteil des Betriebssystems *Debian Linux* bei den Betriebssystemen der genutzten Route-Server bei den Europäischen Internet-Austauschpunkten[27] forciert die Wahl des Betriebssystems *Debian Linux* als einzusetzendes Router-Betriebssystem; das Auswahlkriterium der Verfügbarkeit für möglichst viele Hardware-Plattformen und Maschinen-Architekturen wird vom Betriebssystem *Debian Linux* erfüllt - das Betriebssystem *Debian Linux* ist für zehn Maschinen-Architekturen verfügbar[28,29].

5.7 Konzeption der Laufzeitumgebung

Die Konzeption der Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks (VPNs) bedingt eine zu entwickelnde Laufzeitumgebung, deren primärer Zweck das Steuern, Überwachen und Protokollieren ist.

Die zu steuernden und zu überwachenden Komponenten sind die Software für das dynamische Routing, die VPN-Software und die Overlay-Netzwerkknoten-Software sowie das Router-Betriebssystem.

5.7.1 Beschreibung der Systemzustände mit Markov-Ketten

Die Routerumgebung bildet ein System; dieses System ist in ein äusseres, umgebendes System eingebettet - das Internet. Der Transport von IP-Datenpaketen über mehrere Pfade des Internets ist ein (stochastischer) Prozess; es existieren äussere - extrinsische - Ereignisse wie z.B. Ausfälle von physischen Leitungen in der Folge von Naturkatastrophen, das umgebende System - das Internet - ist in seiner Gesamtheit nicht deterministisch und nicht berechenbar.

Als Kalkül zur Beschreibung der Systemzustände geeignet erscheint die Beschreibung des Systems als Markov-Prozess (*Markov process*[30]) und die Beschreibung der Systemzustände und Zustandsübergänge (Transitionen - *transitions*) durch Markov-Ketten (*Markov chains*[31,32]).

Zum Design und zur Modellierung der Laufzeitumgebung werden die für die Zwecke der Laufzeitumgebung wesentlichen Systemzustände und Zustandsübergänge definiert.

[28] Software in the Public Interest, Inc. (SPI): Debian Project - About Debian, URL: `https://www.debian.org/intro/about` [196]

[29] Software in the Public Interest, Inc. (SPI): Debian Project - Debian Installation Guide, URL: `https://www.debian.org/releases/stable/installmanual` [197]

[30] B. S. Everitt und A. Skrondal: The Cambridge Dictionary of Statistics, Eintrag "Markov process" [198, S. 270]

[31] B. S. Everitt und A. Skrondal: The Cambridge Dictionary of Statistics, Eintrag "Markov chain" [198, S. 268-269]

[32] Christopher Clapham und James Nicholson: The Concise Oxford Dictionary of Mathematics, Eintrag "Markov chain" [199, S. 501-502]

Tabelle 5.1 gibt eine Übersicht über die definierten und erwünschten Systemzustände der Routerumgebung.

Tabelle 5.1: Übersicht über die definierten und erwünschten Systemzustände

Zustand	Beschreibung des Zustands
S0	*Start* der Router-Appliance
S1	*Offline* - fehlende Internet-Connectivity
S2	*Online* - Internet-Connectivity ist gegeben
S3	*Keine VPN-Verbindung* besteht
S4	*Native VPN-Verbindung* besteht
S5	*Native und Overlay-VPN-Verbindung* besteht
S6	*Overlay-VPN-Verbindung* besteht

Abbildung 5.3 veranschaulicht damit korrespondierend diese Systemzustände.

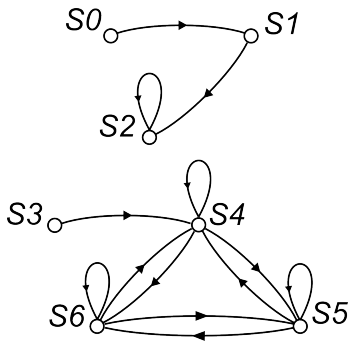

Abbildung 5.3: Übersicht über die definierten und erwünschten Systemzustände

5.7.2 Design der Laufzeitumgebung als Zustandsautomat

Zum Überführen der definierten Systemzustände und Zustandsübergänge (siehe Abschnitt 5.7.1) in eine Grundlage für die praktische Umsetzung erscheint als Modell ein Zustandsautomat (*Finiter Automat*[33], *Endlicher Automat*, *Finite State Machine*) geeignet.

Ein Zustandsautomat (*Endlicher Automat - EA*, *Finite State Machine - FSM*) als Modell erscheint naheliegend, da beispielsweise beim Modell zur Implementierung des Routings im Backbone des Internets mit Hilfe des Protokolls BGP die Verbindungszustände als Zustandsautomat beschrieben sind[34].

Als Grundlage für die praktische Umsetzung der Laufzeitumgebung werden die für die Zwecke der Laufzeitumgebung wesentlichen Zustände, Bedingungen und Aktionen des Zustandsautomaten definiert[35].

[33] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Automat, finiter -" [8, S. 76]
[34] RFC 1163: A Border Gateway Protocol (BGP), 1990, Historic,
URL: `http://www.rfc-editor.org/rfc/rfc1163.txt` [15, S. 18,25]
[35] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Automat; Moore-, Mealy-" [8, S. 76-77]

Tabelle 5.2 gibt eine Übersicht über die wichtigsten Zustandsübergänge der Routerumgebung als Zustandsautomat.

Tabelle 5.2: Übersicht über die wichtigsten Zustandsübergänge der Routerumgebung

Momentaner Zustand	**Übergang (*Transition*)**	**Nächster Zustand**
S0: *Start* der Router-Appliance	Bedingung: -	*S1*: *Offline* - Connectivity fehlend
S1: *Offline* - Connectivity fehlend	Bedingung: Connectivity prüfen; Aktion: Connectivity herstellen	*S2*: *Online* - Connectivity gegeben
S2: *Online* - Connectivity gegeben	Bedingung: Connectivity prüfen	*S2*: *Online* - Connectivity gegeben
S3: *Keine VPN-Verbindung*	Bedingung: Native VPNs prüfen; Aktion: Native VPNs neustarten	*S4*: *Native VPN-Verbindung*
S4: *Native VPN-Verbindung*	Bedingung: Native VPNs prüfen	*S4*: *Native VPN-Verbindung*
S4: *Native VPN-Verbindung*	Bedingung: Overlay-VPNs prüfen; Aktion: Overlay-VPNs neustarten	*S5*: *Native und Overlay-VPN-Verbindung*
S5: *Native und Overlay-VPN-Verbindung*	Bedingung: Native VPNs prüfen, Bedingung: Overlay-VPNs prüfen	*S5*: *Native und Overlay-VPN-Verbindung*
S6: *Overlay-VPN-Verbindung*	Bedingung: Overlay-VPNs prüfen	*S6*: *Overlay-VPN-Verbindung*

Zur Auswahl als Programmiersprache zur Implementierung des Zustandsautomaten für die Laufzeitumgebung der Routerumgebung steht eine Programmiersprache, welche die Eigenschaft der weiten Verbreitung und die Eigenschaft der Robustheit besitzen sollte.

Als Programmiersprachen, welche die Eigenschaft der weiten Verbreitung und die Eigenschaft der Robustheit besitzen, erscheinen die Programmiersprache C und die Skriptsprache der klassischen *Bourne shell* (*AT&T Shell*[36], *sh*) grundsätzlich geeignet zum Zweck der Implementierung der Laufzeitumgebung.

Als Programmiersprache zum Zweck der Implementierung der Laufzeitumgebung wird die Skriptsprache der klassischen *Bourne shell* (*AT&T Shell*, *sh*) gewählt; dies fusst auf dem Grund, dass das Unix-Laufzeitsystem (*Unix Runlevel*[37]) in der Skriptsprache der klassischen *Bourne shell* implementiert ist - es scheint folgerecht, dass Code in der Skriptsprache der klassischen *Bourne shell* in ausreichendem Mass robust und zur Implementierung der Laufzeitumgebung geeignet ist.

Um die Laufzeitumgebung der Routerumgebung zur spezifischeren Nutzung anpassen zu können (*customizing*) oder um Erweiterungen z.B. in Form von zusätzlicher Funktionalität implementieren zu können, erscheint es sinnvoll, bei der Implementierung die Struktur eines Frameworks zu wählen, bei dem Kernfunktionen und genutzte Funktionen separiert sind - ein Framework, welches in einen Hauptbestandteil und einen Bibliotheksbestandteil aufgegliedert ist.

[36] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Shell" [8, S. 757]
[37] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Runlevel" [8, S. 720]

5.8 Zusammenfassung

Abbildung 5.4 gibt einen Überblick über die Konzeption der Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung in Hinblick auf das Zusammenwirken der Software-Komponenten.

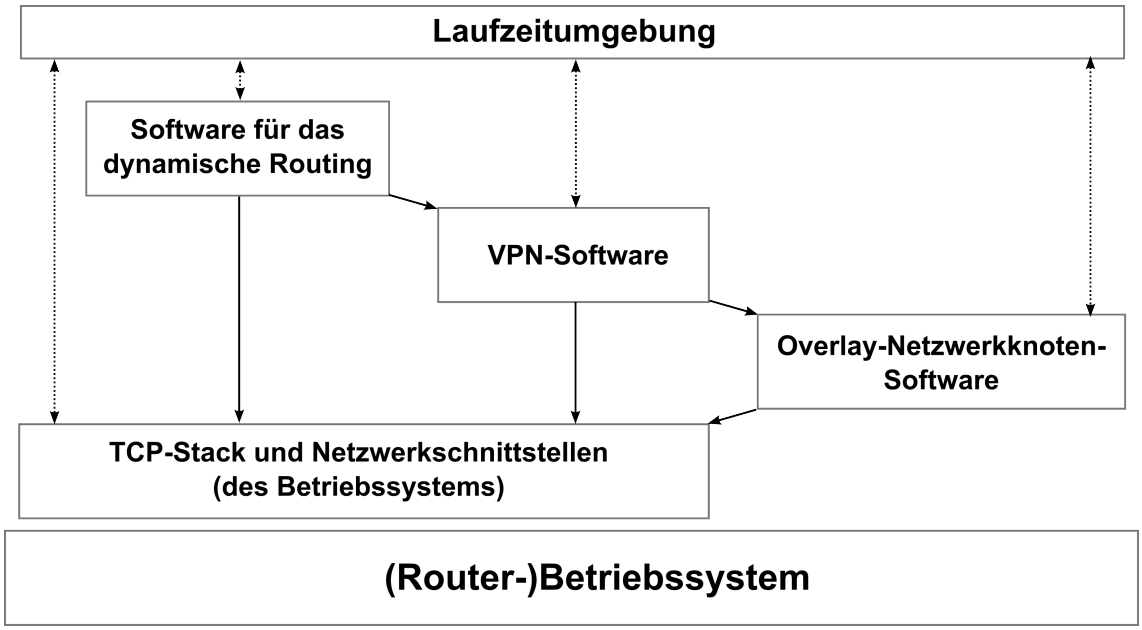

Abbildung 5.4: Übersicht der Konzeption der Laufzeitumgebung mit Software-Komponenten

Die Komponenten stellen sich wie folgt dar:

- Als Router-Betriebssystem wird das Betriebssystem *Debian Linux* gewählt (siehe Abschnitt 5.6),
- als Programmiersprache zum Zweck der Implementierung der Laufzeitumgebung wird die Skriptsprache der klassischen *Bourne shell* (*AT&T Shell*) gewählt (siehe Abschnitt 5.7),
- als Routing-Software - welche das dynamische Routing-Protokoll *OSPF* implementiert - wird die Routing-Software *Quagga* gewählt (siehe Abschnitt 5.3),
- als Overlay-Netzwerk wird das TCP-Overlay-Netzwerk *Tor* und daraus resultierend die Software *Tor* zur Implementierung der TCP-Overlay-Netzwerk-Knoten gewählt (siehe Abschnitt 5.4) und
- als VPN-Protokoll wird *OpenVPN* und daraus resultierend als Implementierung des VPN-Protokolls in Form von VPN-Software wird die VPN-Software *OpenVPN* gewählt (siehe Abschnitt 5.5).

6 Implementierung

In der ersten Stufe erfolgt die Implementierung und Entwicklung in einer Laborumgebung.

Die Laborumgebung für die Implementierung in der ersten Stufe besteht aus zwei Standard-PCs an einem gemeinsamen *LAN* mit Internet-Connectivity. Als Betriebssystem wird auf diesen zwei Standard-PCs *Ubuntu Linux* eingesetzt. Als Besonderheit ist auf diesen beiden Computern mit *Ubuntu Linux* als Betriebssystem der Festplattenplatz über eine Abstraktions-Zwischenschicht - über den Logical Volume Manager (*LVM*)[1] - zugeordnet; dieser ermöglicht das Erstellen von Abbild-Kopien (*snapshots*) des jeweiligen Entwicklungszustands.

Darauf aufbauend kommt in der Laborumgebung auf den beiden Computern mit *Ubuntu Linux* als Betriebssystem die Virtualisierungs-Software *VirtualBox*[2] des Herstellers Oracle zum Einsatz. *VirtualBox* ist eine Virtualisierungssoftware, die für die Betriebssysteme Microsoft *Windows*, *Linux*, Apple *MacOS/X* und *Solaris* als Wirtsbetriebssystem verfügbar ist und die dem Zweck dient, virtuelle Maschinen (VMs) mit unterschiedlichsten Betriebssystemen als Gast zu betreiben. Als Gast können Maschinen mit den Betriebssystemen Microsoft *Windows*, *DOS*, *Linux*, *Solaris* und *OpenBSD* betrieben werden.[2]
Mit Hilfe der beiden Computer mit *Ubuntu Linux* als Betriebssystem und jeweils darauf aufbauend mit der Virtualisierungssoftware *VirtualBox* stehen als Umgebung für die Implementierung zwei mal zwei virtuelle Maschinen (VMs) zur Verfügung.

Auf Computer `A` mit *Ubuntu Linux* als Betriebssystem und darauf aufbauend mit der Virtualisierungssoftware *VirtualBox* werden zwei Betriebssystemgäste betrieben; auf Computer `B` mit *Ubuntu Linux* als Betriebssystem und darauf aufbauend mit der Virtualisierungssoftware *VirtualBox* werden ebenso zwei Betriebssystemgäste betrieben.

Auf Computer `A` werden als virtualisierte Betriebssystemgäste der Router `r1-test` und die LAN-Arbeitsstation `lan1-11-10` betrieben; auf Computer `B` werden als virtualisierte Betriebssystemgäste der Router `r2-test` und die LAN-Arbeitsstation `lan2-12-10` betrieben. Als Gastbetriebssystem der virtuellen Gäste wird *Debian Linux* in der Betriebssystem-Release 7 mit der Release-Bezeichnung *Wheezy* einsetzt.

Die beiden Router `r1-test` und `r2-test` verfügen über Internet-Connectivity; die LAN-Arbeitsstation `lan1-11-10` ist an einem separaten LAN-Segment `lan1` über eine Netzwerkschnittstelle des Routers `r1-test` angebunden; die LAN-Arbeitsstation `lan2-12-10` ist an einem separaten LAN-Segment `lan2` über eine Netzwerkschnittstelle des Routers `r2-test` angebunden.
Abbildung 6.1 stellt die Struktur der Laborumgebung dar.

In der zweiten Stufe erfolgt zeitlich versetzt das nochmalige Testen und die Integration in einer Produktivumgebung; die sich anschliessende Evaluation findet ebenso in dieser Produktivumgebung statt.

Die Produktivumgebung besteht aus zwei Maschinen an unterschiedlichen Standorten.

Am ersten Standort - an der HTW des Saarlandes, Saarbrücken - wird Router `r1` betrieben; als Betriebssystem wird bei Router `r1` *Ubuntu Linux* in der Betriebssystem-Release 15

[1] LVM2 Resource Page - Logical Volume Manager (Linux), URL: `https://www.sourceware.org/lvm2/` [200]
[2] Oracle Corporation: Oracle VM VirtualBox, URL: `https://www.virtualbox.org/` [201]

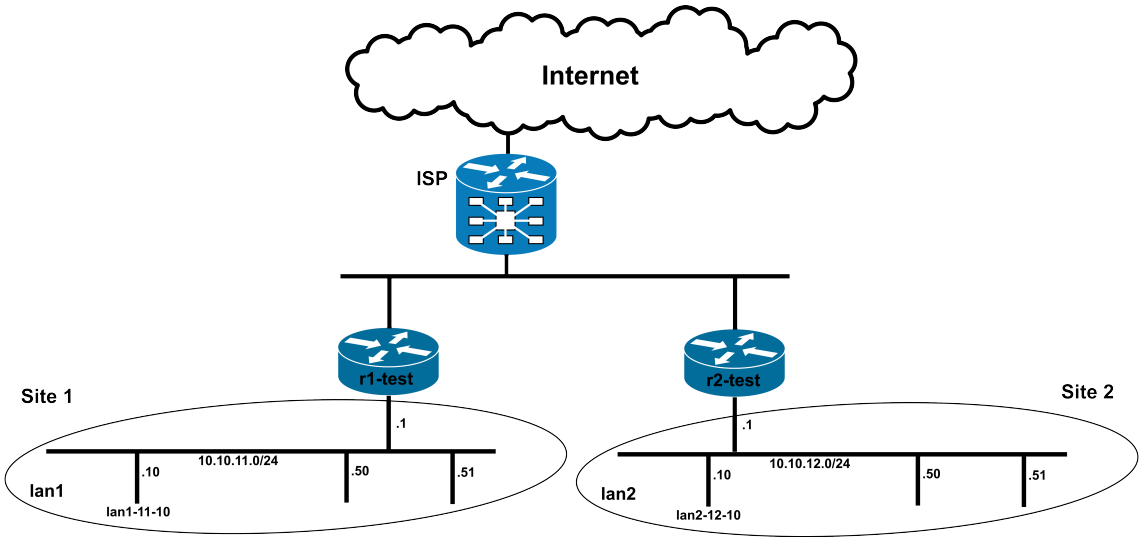

Abbildung 6.1: Struktur der Laborumgebung mit zwei Routern und zwei Arbeitsstationen an LAN-Segmenten

mit der Release-Bezeichnung *Wily Werewolf* einsetzt. Router `r1` steht ein Prozessorkern, 1 GB Hauptspeicher und 20 GB Festplattenplatz zur Verfügung. Die Internetanbindung von Router `r1` erfolgt über das DFN[3] als Internet Service Provider mit einer Bandbreite von ca. 300 MBit/s. An Router `r1` ist die LAN-Arbeitsstation `lan1-11-10` an einem separaten LAN-Segment `lan1` über eine Netzwerkschnittstelle des Routers angebunden.

Am zweiten Standort - im Rechenzentrum der intersaar GmbH, Saarbrücken - wird Router `r2` betrieben; als Betriebssystem wird bei Router `r2` *Debian Linux* in der Betriebssystem-Release 7 mit der Release-Bezeichnung *Wheezy* einsetzt. Router `r2` steht ein Prozessorkern, 2 GB Hauptspeicher und 20 GB Festplattenplatz zur Verfügung. Die Internetanbindung von Router `r2` erfolgt über die intersaar GmbH als Internet Service Provider mit einer Bandbreite von ca. 100 MBit/s. An Router `r2` ist die LAN-Arbeitsstation `lan2-12-10` an einem separaten LAN-Segment `lan2` über eine Netzwerkschnittstelle des Routers angebunden.

Abbildung 6.2 stellt die Struktur der Produktivumgebung dar.

6.1 Aufsetzen einer Muster-Appliance als Basissystem

Als Betriebssystem für das Aufsetzen der Muster-Appliance als Basissystem kommt *Debian Linux*[4] in der Betriebssystem-Release 7 mit der Release-Bezeichnung *Wheezy* zum Einsatz.

Initial werden der Muster-Appliance als Basissystem ein Prozessorkern, 512 MB Hauptspeicher, eine Netzwerkschnittstelle und 8 GB Festplattenplatz zugeordnet; der Festplattenplatz ist in Form einer virtuellen Festplatte zugeordnet, diese ist auf der Festplatte des Wirtssystems (*hosts*) als Datei alloziert.

Das Aufsetzen der Muster-Appliance als Basissystem wird, um das initial aufgesetzte Betriebssystem revisionssicher zu halten, von optischem Datenträger durchgeführt.

Die Bezeichnung für die Muster-Appliance als Basissystem wird auf die Bezeichnung

[3] DFN - Deutsches Forschungsnetz - ist das Kommunikationsnetz für Wissenschaft und Forschung in Deutschland, URL: `https://www.dfn.de/` [202]

[4] Software in the Public Interest, Inc. (SPI): Debian Project - About Debian, URL: `https://www.debian.org/intro/about` [196]

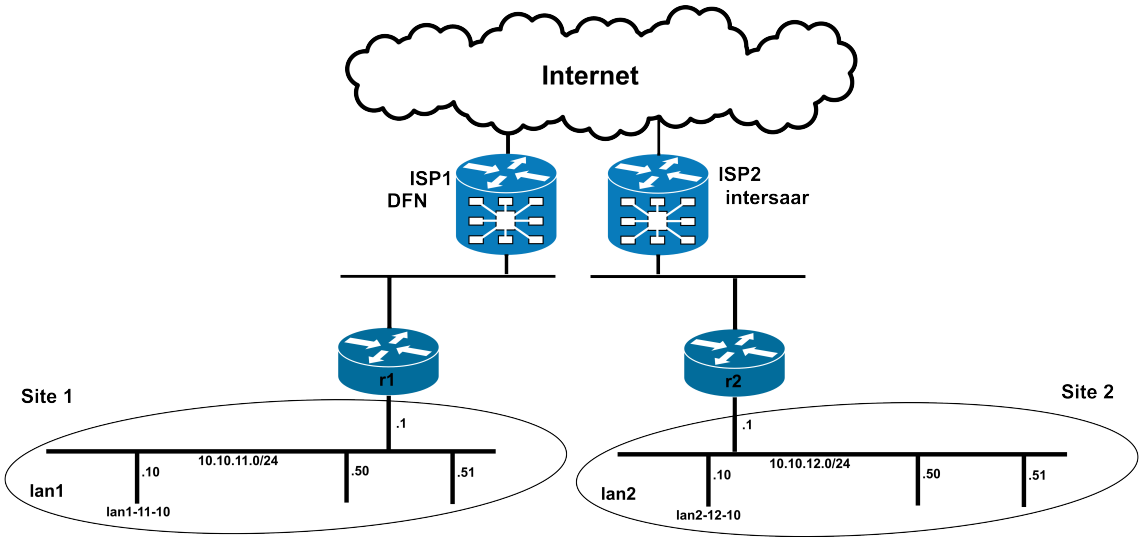

Abbildung 6.2: Struktur der Produktivumgebung mit zwei Routern und zwei Arbeitsstationen an LAN-Segmenten

`base` festgelegt, dies stellt zugleich den Maschinennamen und Hostnamen des Basissystems dar.

Die Einrichtung der Muster-Appliance erfolgt weitgehend den Vorschlägen der Installationsroutinen von *Debian Linux* folgend. Um die Kompatibilität zu gewährleisten und die Portabilität zu erhöhen - z.B. bezüglich der Ausgaben von Unix-Systemkommandos - wird als Standard für die Spracheinstellung des Systems die Sprache *Englisch* gewählt; entsprechend wird als Standard für die Locale-Einstellungen (*locale settings*) die Einstellung `en_US.UTF-8` gewählt.

Die initiale Softwareauswahl - die Auswahl der zur Nutzung verfügbaren Softwarepakete des Systems - ist auf die notwendigen Pakete zum Betreiben eines *Linux*-Betriebssystems ohne graphische Benutzeroberfläche reduziert; diese Wahl fusst auf der Erwägung, dass Fehler in nicht benötigter installierter Software mögliche Sicherheitsschwachstellen darstellen können.

Zum Zweck des Aufbringens von Aktualisierungen von Software, zum Zweck der Wartung und zum Zweck des Transfers von Log-Dateien (*log files*) ist in die initiale Softwareauswahl des Systems das Paket *SSH server* einbezogen, die Paketbezeichnung ist `openssh-server`.

Anhang A.1.1 listet die aus der initialen Softwareauswahl resultierende Liste der verfügbaren Pakete des Basissystems auf.

6.1.1 Spezifische Anpassungen für Router

Router unterscheiden sich von Arbeitsstationen bezüglich der Verarbeitung von IP-Datenpaketen; Router leiten über Netzwerkschnittstellen eingehende Pakete basierend auf Routing-Entscheidungen über gewählte Netzwerkschnittstellen weiter.

Die Netzwerkschnittstelle mit der Bezeichnung `eth0` wird als Standard-Netzwerkschnittstelle des Systems und zur Bereitstellung der Internet-Connectivity verwandt, eine zusätzliche Netzwerkschnittstelle wird für das LAN, für das Routing-Funktionalität bereitgestellt wird, über eine Netzwerkbrücke (*network bridge*) mit der Bezeichnung `br1` zugeordnet; ein Vorteil der Zuordnung mittels einer Netzwerkbrücke liegt darin, dass zur Laufzeit dynamisch ohne Verlust der Verfügbarkeit oder der Routing-Funktionalität für das LAN weitere LAN-Segmente integriert werden können.

Die Spezifikationen für Router, welche IPv4-Pakete weiterleiten, sind hauptsächlich in RFC 1812 definiert.[5]

Die spezifischen Anpassungen für Router lassen sich in drei Bereiche unterteilen: zum Einen das Festlegen der Routing-Funktionalität an sich, zum Anderen das Festlegen entsprechend grösser dimensionierter Speicherbereiche für die Netzwerk-Puffer und Routingtabellen im Betriebssystemkern und des Weiteren sicherheitsspezifische Einstellungen, die beispielsweise die Angriffsfläche für DoS-Attacken[6,7] reduzieren.[8]

Eine weitere Massnahme, die Angriffsfläche für DoS-Attacken zu reduzieren, ist die Reduktion der auf dem Router laufenden Dienste (*services*) auf solche Dienste, welche zwingend für das Routing benötigt werden.

6.1.1.1 Notwendige Dienste für Router

Für die Funktionalität des Protokollierens von Ereignissen wird eine ausreichend genaue Systemzeit auf der Router-Appliance benötigt. Das *Network Time Protocol* (NTP) stellt im Internet die Funktionalität zur Synchronisation von Echtzeituhren von Computern zur Verfügung.[9] Eine ausreichend genaue Systemzeit auf der Router-Appliance ist gegeben durch die Nutzung eines Clients für die *Network Time Protocol*-Funktionalität; ein *Network Time Protocol*-Server als laufender Dienst auf der Router-Appliance ist nicht zwingend notwendig und in der Abwägung gegenüber Sicherheitsrisiken verzichtbar.

Unter anderem für die Funktionalität des Protokollierens von Verbindungsversuchen zum Router ist es eine Notwendigkeit, Namensauflösung durch DNS-Server zu nutzen[10,11,12,13]. Hierfür ist ausschliesslich die Funktionalität eines DNS-Resolvers - eines DNS-Namen und -Adressen auflösenden Clients - notwendig, ein DNS-Server als Dienst auf der Router-Appliance ist nicht zwingend notwendig und in der Abwägung gegenüber Sicherheitsrisiken verzichtbar; der DNS-Resolver des Routers nutzt die DNS-Server des jeweils genutzten Internet Service Providers.

Ein SSH-Server[14,15] als Dienst auf der Router-Appliance ist zwingend notwendig, um Software-Aktualisierungen vorzunehmen, Wartung zu betreiben und Log-Dateien zu transferieren.

6.1.1.2 Spezifische Betriebssystemkern-Einstellungen

Der beim verwendeten Betriebssystem *Debian Linux* als Paket bereitgestellte Betriebssystemkern (*kernel*) ist mit der Funktionalität übersetzt, Routing für IP-basierte Datenver-

[5] RFC 1812: Requirements for IP Version 4 Routers, 1995, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc1812.txt` [203]

[6] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "DOS, DoS" [8, S. 242-243]

[7] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Denial of Service" [8, S. 217]

[8] *Denial of Service*-Attacken haben zum Ziel, die Verfügbarkeit von Systemen oder die Verfügbarkeit von Diensten zu stören.[7]

[9] RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification, 2010, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc5905.txt` [204]

[10] RFC 1034: Domain names - concepts and facilities, 1987, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc1034.txt` [205]

[11] RFC 1035: Domain names - implementation and specification, 1987, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc1035.txt` [206]

[12] RFC 4033: DNS Security Introduction and Requirements, 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4033.txt` [207]

[13] RFC 4035: Protocol Modifications for the DNS Security Extensions, 2005, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4035.txt` [208]

[14] Daniel J. Barrett und Richard E. Silverman: SSH, The Secure Shell: The Definitive Guide, 2001 [100]

[15] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "SSH" [8, S. 799]

kehre bewerkstelligen zu können. Diese Funktionalität wird gesteuert über Systemvariablen (*system variables*). Die Werte für Systemvariablen des Betriebssystems *Debian Linux* werden in einer frühen Phase des Systemstarts aus der Datei `/etc/sysctl.conf` sowie aus Dateien aus der Dateisystemhierarchie `/etc/sysctl.d` ausgelesen; die Systemvariable `net.ipv4.ip_forward` steuert die Routing-Funktionalität des Betriebssystemkerns bezüglich IPv4-basiertem Routing.

Anhang A.2.1 listet Werte für Systemvariablen des Betriebssystemkerns auf, die spezifische Werte für Router darstellen.

6.1.2 Testen

Eine Möglichkeit, die Funktionalität des Routings bei Router `r1` zu testen, besteht darin, mit dem Kommandozeilen-Netzwerkhilfsprogramm `ping` von einer am LAN über den Router `r1` angebundenen Arbeitsstation IP-Datagramme vom Typ *ICMP Echo Request*[16,17,18] über den Router hinweg zu versenden:

```
root@lan1-11-10:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         r1-br1          0.0.0.0         UG    0      0        0 eth0
lan1            *               255.255.255.0   U     0      0        0 eth0
root@lan1-11-10:~#

root@lan1-11-10:~# ping -c 1 r1-ext | head -2
PING r1-ext (134.96.216.137) 56(84) bytes of data.
64 bytes from r1-ext (134.96.216.137): icmp_req=1 ttl=64 time=0.160 ms
root@lan1-11-10:~#
```

6.1.3 Klonen der Appliance

Es existieren zwei naheliegende Möglichkeiten, die Router-Appliance zu klonen: zum Einen durch Erzeugung einer Abbild-Kopie (*image*) des Datenträgers und zum Anderen durch Archivierung der relevanten Dateien und Dateisystemteilbäume des Dateisystems der Router-Appliance.

Die Erzeugung einer Abbild-Kopie des Datenträgers der Router-Appliance erscheint für Anwendungsfälle wie das vollständige Wiederherstellen des Systems nach einem Datenträgerausfall sinnvoll.

Um die Konsistenz und Integrität der Abbild-Kopie des Systems zu gewährleisten, erfolgt das Erstellen der Abbild-Kopie des Datenträgers von einer Betriebssystem-Instanz, die von optischem Datenträger gestartet wird.

Die Vorgehensweise für die Erzeugung einer Abbild-Kopie des Datenträgers der Router-Appliance stellt sich wie folgt dar:

```
root@backup-host:~# file /dev/disk/by-uuid/ade7c0d4-867b-4e5b-b997-8a1be98138b0 | \
  tr ':' '\n'
/dev/disk/by-uuid/ade7c0d4-867b-4e5b-b997-8a1be98138b0
 symbolic link to `../../sda1'
```

[16] Bruce Hartpence: Packet Guide to Core Network Protocols, 2011 [209, S. 97]
[17] RFC 792: Internet Control Message Protocol, 1981, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc792.txt` [210]
[18] RFC 4884: Extended ICMP to Support Multi-Part Messages, 2007, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4884.txt` [211]

```
root@backup-host:~#
root@backup-host:~# fdisk -l /dev/sda | grep -e "Device Boot" -e "^/dev/sda1"
   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048    15988735     7993344   83  Linux
root@backup-host:~#
root@backup-host:~# dd if=/dev/sda1 bs=16M | \
  xz -9 --block-size=16MiB - > /backup-ext/routerumgebung/r1/sda1-rootfs.img.xz
487+1 records in
487+1 records out
8185184256 bytes (8,2 GB) copied, 3853,77 s, 2,1 MB/s
root@backup-host:~#
root@backup-host:~# xzcat \
  /backup-ext/routerumgebung/r1/sda1-rootfs.img.xz | file -b - | \
  tr ',' '\n'
Linux rev 1.0 ext4 filesystem data
 UUID=ade7c0d4-867b-4e5b-b997-8a1be98138b0 (extents) (large files) (huge files)
root@backup-host:~#
```

Die zweitgenannte Möglichkeit, die Router-Appliance zu klonen - die Archivierung der relevanten Dateien und Dateisystemteilbäume des Dateisystems des Systems - erscheint geeignet, um revisionssicher auf eine neuere oder andere Betriebssystemversion zu migrieren oder um mit Hilfe von Automatisierungswerkzeugen und -frameworks Installationen der Router-Appliance durchzuführen. Die Vorgehensweise, die relevanten Dateien und Dateisystemteilbäume des Dateisystems der Router-Appliance zu archivieren, stellt sich wie folgt dar:

```
root@backup-host:~# ${HOME}/bin/backup-pull-r1-configs-logs-to-backuphost.sh
remote tar via ssh to local destination...
```

Die Datei `backup-pull-r1-configs-logs-to-backuphost.sh` ist in Anhang A.3.1 aufgelistet.

6.2 Aufsetzen zweier Muster-Appliances mit VPN-Kopplung

Zur Implementierung der VPN-Tunnel wird die VPN-Software OpenVPN[19] (siehe Abschnitt 2.2.5.5 OpenVPN) verwendet.

Als Pakettransport zwischen den teilnehmenden Routern wird das TCP-Protokoll gewählt; dies fusst auf der Erwägung, dass das TCP-Protokoll gegenüber dem alternativ möglichen UDP-Protokoll unter widrigen Bedingungen eine höhere Wahrscheinlichkeit auf einen gelungenen Verbindungsaufbau zwischen den VPN-Endpunkten erwarten lässt.

Als Transportprotokoll wird das *SSL*-Protokoll bzw. *TLS*-Protokoll[20] eingesetzt. Um Angriffsvektoren auf die VPN-Endpunkte zu reduzieren, ist dem eingehenden Verbindungsaufbau mit dem *SSL*-Protokoll eine *TLS*-Authentifizierung (`tls-auth`) mit statischem Schlüssel von 2048 Bit Länge vorgeschaltet.

Es wird Zertifikat-basierte Public-Key-Verschlüsselung eingesetzt; den beiden Appliances für die VPN-Kopplung wird jeweils eine eigene Zertifizierungsstelle (*Certificate Authority*[21], *CA*[22]) für die digitalen Zertifikate zugeordnet; dies fusst auf der Erwägung,

[19] OpenVPN Inc.: OpenVPN - Open Source VPN, URL: `https://openvpn.net/` [112]
[20] RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2, 2008, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc5246.txt` [115]
[21] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Certificate Authority" [8, S. 150]
[22] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "CA" [8, S. 132]

dass für den Fall der Kompromittierung einer Appliance durch z.B. Diebstahl die negativen Auswirkungen einer solchen Kompromittierung hierdurch begrenzt werden.

6.2.1 Public-Key-Infrastruktur für die VPNs

Die Bereitstellung einer Public-Key-Infrastruktur[23] (PKI[24]) für die VPNs ist motiviert durch die Anforderung, dass der VPN-Client-Endpunkt die Authentizität des VPN-Server-Endpunkts zu prüfen hat, sowie durch die Anforderung, dass der VPN-Server die Authentizität und die Zugriffsberechtigung des VPN-Client-Endpunkts zu prüfen hat[25].

Die Public-Key-Infrastruktur[23] (PKI[24]) für die beiden Router-Appliances besteht aus zwei Zertifizierungsstellen; jeder einzelnen Router-Appliance ist eine Zertifizierungsstelle zugeordnet. Nachfolgend wird eine der beiden Zertifizierungsstellen - die Zertifizierungsstelle *Router r2 CA* - beschrieben, die Struktur der beiden Zertifizierungsstellen ist bis auf die Bezeichnung der Zertifizierungsstelle und die Bezeichnungen der ausgestellten Zertifikate für die VPN-Endpunkte identisch.

Die Zertifizierungsstelle *Router r2 CA* besteht aus einem *nicht* mit anderen Systemen verbundenen Computer; als Betriebssystem wird *Debian Linux* in der Betriebssystem-Release 7 mit der Release-Bezeichnung *Wheezy* eingesetzt. Die Zertifizierungsstelle baut auf den kryptographischen Funktionen der Softwarebibliothek *OpenSSL*[26] auf. Die Softwarebibliothek *OpenSSL* wird beim Betriebssystem *Debian Linux* durch das Paket `openssl` bereitgestellt. Zur Vereinfachung der Handhabung und zur Vermeidung von Fehlern werden die kryptographischen Funktionen der Softwarebibliothek *OpenSSL* zur Nutzung durch die Zertifizierungsstelle mit Hilfe eines Management-Pakets (*management package*) genutzt[27].

Die Serverschlüssel, digitalen Zertifizierungsanforderungen (*certificate requests*) sowie die digitalen Zertifikate werden mit Hilfe von Wechseldatenträgern mit der Zertifizierungsstelle ausgetauscht.

Die Zertifizierungsstelle *Router r2 CA* entsteht durch Erzeugung eines privaten RSA-Schlüssels (`ca.key`) und eines damit verbundenen öffentlichen selbst-signierten Zertifikats (`ca.crt`); dieses Zertifikat wird den VPN-Client- und Server-Endpunkten zur Verfügung gestellt. Für den VPN-Server-Endpunkt wird ein privater RSA-Schlüssel erzeugt (`server.key`); für diesen privaten RSA-Schlüssel des VPN-Server-Endpunkts wird durch die Zertifizierungsstelle mittels einer Zertifizierungsanforderung (`server.csr`) ein Zertifikat (`server.crt`) erzeugt. Der private RSA-Schlüssel des VPN-Server-Endpunkts sowie das damit verbundene Zertifikat werden beim VPN-Server-Endpunkt hinterlegt. Für den VPN-Client-Endpunkt wird ein privater RSA-Schlüssel erzeugt (`r1.key`); für diesen privaten RSA-Schlüssel des VPN-Client-Endpunkts wird durch die Zertifizierungsstelle mittels einer Zertifizierungsanforderung (`r1.csr`) ein Zertifikat (`r1.crt`) erzeugt. Der privaten RSA-Schlüssel des VPN-Client-Endpunkts sowie das damit verbundene Zertifikat werden beim VPN-Client-Endpunkt hinterlegt.

Das Transkript der Zertifizierungsstelle *Router r2 CA* ist in Anhang A.4.1 dargestellt.

Mit Hilfe des Zertifikats der Zertifizierungsstelle (`ca.crt`) ist der VPN-Client-Endpunkt befähigt, die Authentizität des VPN-Server-Endpunkts zu prüfen und der VPN-Server-

[23] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Public-Key-Infrastruktur bzw. Verfahren" [8, S. 664]

[24] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "PKI" [8, S. 632]

[25] Niels Ferguson, Bruce Schneier und Tadayoshi Kohno: Cryptography engineering: Design principles and practical applications, 2010 [24, S. 276]

[26] OpenSSL Software Foundation: OpenSSL Cryptography and SSL/TLS Toolkit, URL: `https://www.openssl.org/` [104]

[27] OpenVPN Inc.: OpenVPN - Documentation - RSA Key Management, URL: `https://openvpn.net/index.php/open-source/documentation/miscellaneous/77-rsa-key-management.html` [212]

Endpunkt ist hierdurch befähigt, die Authentiziät und Zugriffsberechtigung des VPN-Client-Endpunkts zu prüfen, da diese Zertifikate die Zugehörigkeit des VPN-Server-Schlüssels zum VPN-Server-Endpunkt und die Zugehörigkeit des VPN-Client-Schlüssels zum VPN-Client-Endpunkt bestätigen.

Für die VPN-Kopplung werden zwei VPN-Tunnel parallel bereitgestellt, um durch Redundanz eine höhere Verfügbarkeit erreichen zu können. Der erste der beiden VPN-Tunnel nutzt den TCP-Port 1194 serverseitig, der zweite der beiden VPN-Tunnel nutzt den TCP-Port 1195 serverseitig.

Die VPN-Endpunkte erzeugen paarweise Pfade für den Nutzdatentransport auf Ethernet-Ebene (*Data Link Layer*[28]) - OSI Layer 2 -, um grösstmögliche Kompatibilität mit zukünftigen Protokollen sicherzustellen.

Die VPN-Endpunkte bei Router `r1` und bei Router `r2` werden bei der VPN-Software OpenVPN bei Nutzdatentransport auf Ethernet-Ebene in Form von generischen, virtuellen Netzwerkschnittstellen - *tap*-Devices - bereitgestellt; diese *tap*-Devices (`tap2`, `tap10`) werden jeweils einer Netzwerkbrücke (*network bridge*) (`br2`, `br10`) zugeordnet. Dies hat den Vorteil, dass zur Laufzeit dynamisch ohne Verlust der Verfügbarkeit Änderungen an den Netzwerkschnittstellen vorgenommen werden können, beispielsweise zum Zweck der Wartung.

Abbildung 6.3 stellt die Muster-Appliances mit VPN-Kopplung in der Produktivumgebung dar.

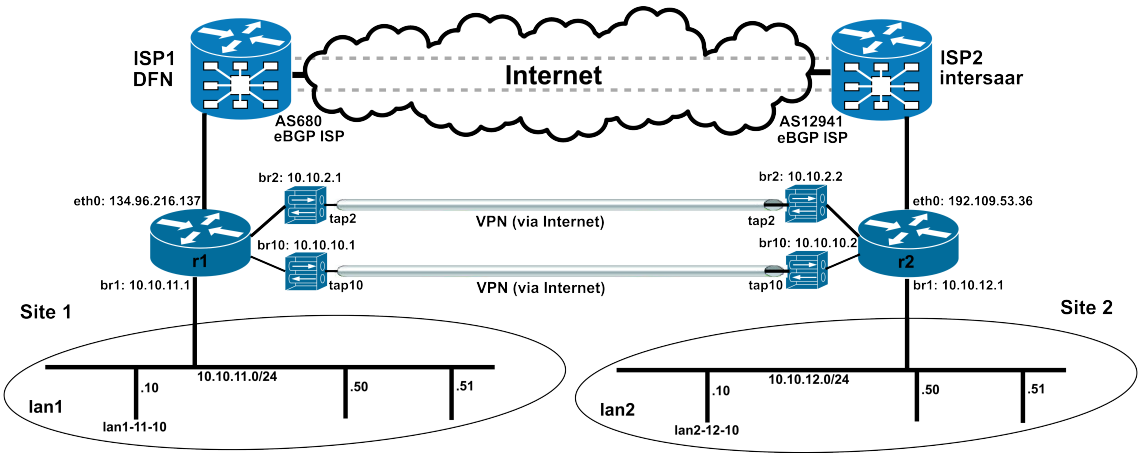

Abbildung 6.3: Muster-Appliances mit VPN-Kopplung

Ein Auszug aus dem Setup der VPN-Endpunkte mit Code-Einsprungpunkten (*hooks*) für die Laufzeitumgebung auf Router `r1` und auf Router `r2` ist in Anhang A.5.1 aufgelistet.

6.2.2 Testen

Eine Möglichkeit, die Funktionalität der VPN-Tunnel und der VPN-Kopplung zu testen, besteht darin, mit dem Kommandozeilen-Netzwerkhilfsprogramm `ping` von Router `r1`

[28] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "OSI-Schichten, Namen und Nummern der -" [8, S. 601]

IP-Datagramme vom Typ *ICMP Echo Request*[29,30,31] an den Tunnel-Endpunkt bei Router r2 zu versenden:

```
root@r1:~# brctl show | grep -e "^bridge name" -e "^br2"
bridge name     bridge id               STP enabled     interfaces
br2             8000.e259c8908639       no              tap2
root@r1:~#

root@r1:~# route -n | grep -e "^Destination" -e "br2$"
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.10.2.0       0.0.0.0         255.255.255.0   U     0      0        0 br2
10.10.12.0      10.10.2.2       255.255.255.0   UG    20     0        0 br2
root@r1:~#

root@r1:~# ping -c 1 r2-br2 | head -2
PING r2-br2 (10.10.2.2) 56(84) bytes of data.
64 bytes from r2-br2 (10.10.2.2): icmp_seq=1 ttl=64 time=53.0 ms
root@r1:~#
```

6.3 Aufsetzen zweier Muster-Appliances mit Routing

Zur Implementierung des Routings mit dem dynamischen Routing-Protokoll OSPF wird die *Routing Software Suite* Quagga[32] (siehe Abschnitt 2.1.3.4 Quagga) verwendet.

Die *Routing Software Suite* Quagga wird beim Betriebssystem *Debian Linux* durch das Paket quagga bereitgestellt. Die beim Betriebssystem *Debian Linux* in der Betriebssystem-Release 7 bereitgestellte Version der Software ist Version 0.99.x,[33] die aktuell veröffentlichte Version der Software ist Version 1.2.1 vom 10.03.2017 (Stand 20.07.2017);[34] die bereitgestellten Versionen der Software durch das Betriebssystem sind produktiv verwendbar und enthalten Rückportierungen (*backports*) von sicherheitsspezifischen Änderungen aus der jeweiligen aktuell veröffentlichten Version.

Bei dynamischen Routing mit der *Routing Software Suite* Quagga erfolgt der Pakettransport durch den *TCP-Stack*[35] des verwendeten Betriebssystems; auch wird für die Routing-Entscheidung die Routing-Tabelle des Betriebssystemkerns des verwendeten Betriebssystems - hier: des Betriebssystems *Debian Linux* - verwendet. Die Routing-Tabelle des Betriebssystemkerns des verwendeten Betriebssystems wird durch den Routing-Tabellen-Manager (*kernel routing table manager*) der *Routing Software Suite* Quagga überwacht und verwaltet; der Routing-Tabellen-Manager wird mit *zebra* bezeichnet und ist als *Daemon* implementiert.[36] Das Routing mit dem dynamischen Routing-Protokoll OSPF ist ebenso als *Daemon* implementiert, der *Daemon* hierfür wird mit *ospfd* bezeichnet.

Für das Routing mit dem dynamischen Routing-Protokoll OSPF zwischen den beiden Routern r1 und r2 werden periodisch Informationen über die Verbindungszustände

[29] Bruce Hartpence: Packet Guide to Core Network Protocols, 2011 [209, S. 97]
[30] RFC 792: Internet Control Message Protocol, 1981, Internet Standard, URL: http://www.rfc-editor.org/rfc/rfc792.txt [210]
[31] RFC 4884: Extended ICMP to Support Multi-Part Messages, 2007, Proposed Standard, URL: http://www.rfc-editor.org/rfc/rfc4884.txt [211]
[32] Quagga Software Routing Suite, URL: http://www.nongnu.org/quagga/ [58]
[33] Die bereitgestellte Version der *Routing Software Suite* Quagga ist beim Betriebssystem *Debian Linux* durch das Kommando `apt-cache showpkg quagga | head -3` abrufbar.
[34] Quagga Software Routing Suite: News, URL: http://savannah.nongnu.org/news/?group=quagga [60]
[35] Für den Begriff *TCP-Stack* wird synonym auch der Begriff *TCP-Protokollstapel* genutzt.
[36] Kunihiro Ishiguro et al.: Quagga - A routing software package for TCP/IP networks - Quagga 1.2.0, 2017, URL: http://www.nongnu.org/quagga/docs/quagga.pdf [213, S. 2]

ermittelt; dies geschieht durch das Versenden von IP-Datagrammen als *Multicast*[37] zwischen den beteiligten Routern. Ein solches IP-Datagramm wird beim Routing-Protokoll OSPF als Hello-Paket (*Hello Packet*) bezeichnet (siehe Abschnitt 2.1.2.2). Die Periode für das Versenden dieser Hello-Pakete zur Ermittlung von Informationen über die Verbindungszustände ist einstellbar, der übliche Wert für das zeitliche Intervall zwischen solchen Hello-Paketen beträgt 10 Sekunden.[38] Der Wert für das zeitliche Intervall zwischen Hello-Paketen wird hier auf 2 Sekunden festgelegt; dies fusst auf der Motivation, die Verfügbarkeit durch Verkleinerung der maximalen Rekonvergenzzeit zu erhöhen.

Ein Auszug aus dem Setup von Router `r1` für das dynamische Routing mit OSPF ist in Anhang A.6.1 dargestellt.

Abbildung 6.4 stellt die Muster-Appliances mit dynamischem Routing dar.

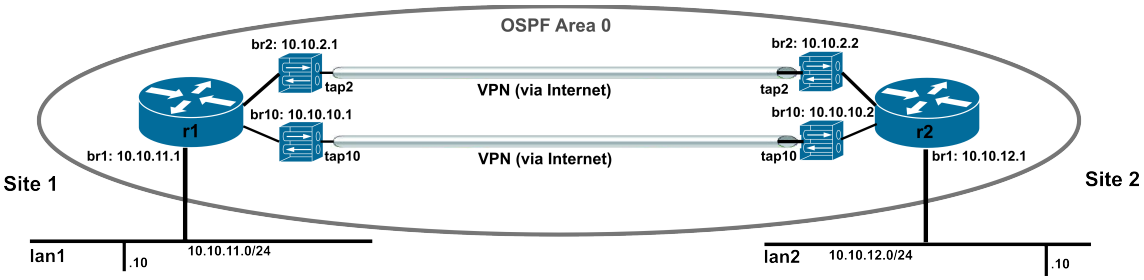

Abbildung 6.4: Muster-Appliances mit dynamischem Routing

6.3.1 Testen

Eine Möglichkeit, die Funktionalität des dynamischen Routings mit OSPF zu *prüfen*, besteht darin, mit Hilfe der von der *Routing Software Suite* Quagga bereitgestellten virtuellen Konsole (*Virtual Terminal Interface*, *VTY*)[39] den Zustand der Datenbank über die OSPF-Verbindungszustände (*link state database*) (siehe Abschnitt 2.1.2.2) abzufragen[40]:

```
r1> show ip ospf database

       OSPF Router with ID (10.10.11.1)

                Router Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum  Link count
10.10.11.1      10.10.11.1       107 0x800013e7 0x3bfa 2
10.10.12.1      10.10.12.1       107 0x80000862 0xd40c 8

                Net Link States (Area 0.0.0.0)

Link ID         ADV Router      Age  Seq#       CkSum
10.10.2.2       10.10.12.1       109 0x8000014a 0x573e
10.10.10.2      10.10.12.1       108 0x8000014a 0xfe8e
```

[37] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Multicast" [8, S. 553]

[38] Kunihiro Ishiguro et al.: Quagga - A routing software package for TCP/IP networks - Quagga 1.2.0, 2017, URL: `http://www.nongnu.org/quagga/docs/quagga.pdf` [213, S. 46]

[39] Kunihiro Ishiguro et al.: Quagga - A routing software package for TCP/IP networks - Quagga 1.2.0, 2017, URL: `http://www.nongnu.org/quagga/docs/quagga.pdf` [213, S. 13]

[40] Der Kommandozeilen-Interpreter der virtuellen Konsole der *Routing Software Suite* Quagga besitzt eine ähnliche Sematik, Grammatik und Syntax wie die Systemsoftware IOS für die Router des Herstellers Cisco Systems.

```
                AS External Link States

Link ID         ADV Router      Age  Seq#       CkSum  Route
10.10.11.0      10.10.11.1       107 0x8000046b 0x58b8 E2 10.10.11.0/24 [0x0]
10.10.12.0      10.10.12.1       108 0x80000469 0x4ac6 E2 10.10.12.0/24 [0x0]

r1>
```

Eine Möglichkeit, die Funktionalität des dynamischen Routings mit OSPF zu *testen*, besteht darin, mit dem Kommandozeilen-Netzwerkhilfsprogramm `ping` von einer am LAN `lan1` über den Router `r1` angebundenen Arbeitsstation IP-Datagramme vom Typ *ICMP Echo Request*[41,42,43] an eine über den Router `r2` angebundenen Arbeitsstation am LAN `lan2` zu versenden.

Das nachfolgende Transkript zeigt die Ausgabe mit *zwei verfügbaren* VPN-Links mit dynamischem Routing:

```
root@r1:~# ip route list 10.10.12.0/24
10.10.12.0/24  proto zebra  metric 20
        nexthop via 10.10.2.2  dev br2 weight 1
        nexthop via 10.10.10.2  dev br10 weight 1
root@r1:~#

root@lan1-11-10:~# ping -c 1 lan2-12-10 | head -2
PING lan2-12-10 (10.10.12.10) 56(84) bytes of data.
64 bytes from lan2-12-10 (10.10.12.10): icmp_req=1 ttl=62 time=43.9 ms
root@lan1-11-10:~#
```

Das nachfolgende Transkript hingegen zeigt die Ausgabe mit einem *nicht verfügbaren VPN-Link und einem verfügbaren* VPN-Link mit dynamischem Routing:

```
root@r1:~# ip route list 10.10.12.0/24
10.10.12.0/24 via 10.10.10.2 dev br10  proto zebra  metric 20
root@r1:~#

root@lan1-11-10:~# ping -c 1 lan2-12-10 | head -2
PING lan2-12-10 (10.10.12.10) 56(84) bytes of data.
64 bytes from lan2-12-10 (10.10.12.10): icmp_req=1 ttl=62 time=26.8 ms
root@lan1-11-10:~#
```

6.4 Aufsetzen zweier Muster-Appliances als Overlay-Netzwerkknoten

Zur Implementierung der Funktionalität der Overlay-Netzwerkknoten (*Nodes*) wird die Software Tor und das Overlay-Netzwerk Tor[44] verwendet.

Die Software Tor wird beim Betriebssystem *Debian Linux* durch die Kombination der Pakete `tor`, `tor-geoipdb` und `torsocks` bereitgestellt. Die beim Betriebssystem *Debian Linux*

[41] Bruce Hartpence: Packet Guide to Core Network Protocols, 2011 [209, S. 97]
[42] RFC 792: Internet Control Message Protocol, 1981, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc792.txt` [210]
[43] RFC 4884: Extended ICMP to Support Multi-Part Messages, 2007, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4884.txt` [211]
[44] The Tor Project Inc.: Tor Project: Anonymity Online URL: `https://www.torproject.org/` [150]

in der Betriebssystem-Release 7 bereitgestellte Version der Software ist Version 0.2.4.x,[45] die aktuell veröffentlichte Version der Software ist Version 0.3.0.10 vom 02.08.2017 (Stand 18.08.2017);[46] die bereitgestellten Versionen der Software durch das Betriebssystem sind produktiv verwendbar.

Als Pakettransport auf dem Underlay wird von am Overlay-Netzwerk Tor teilnehmenden Overlay-Netzwerkknoten das TCP-Protokoll genutzt.[47,48,49] Die Wahl von Tor als verwendetes Overlay-Netzwerk fusst unter anderem auf der Erwägung, dass das TCP-Protokoll unter widrigen Bedingungen eine hohe Wahrscheinlichkeit auf einen gelungenen Verbindungsaufbau zwischen Overlay-Netzwerkknoten auf dem Underlay erwarten lässt.

Das Overlay-Netzwerk Tor ermöglicht den Pakettransport von Datagrammen nach dem TCP-Protokoll auf dem Overlay; eine TCP-Socket-basierte Verbindung von einer TCP-Client-Anwendung an einem Tor-Knoten *A* zu einer TCP-Server-Anwendung an einem anderen Tor-Knoten *B* wird ermöglicht.

Für die TCP-Server-Anwendung an Tor-Knoten *B* (*Node B*) wird ein Overlay-Netzwerk-interner Endpunkt erzeugt, der *Hidden Service* genannt wird;[50] dieser Overlay-Netzwerk-interne Endpunkt wird in Form eines Hostnamen-Port-Tupels adressiert - z.B. mit dem Bezeichner `ryqoqnzfmomxora2.onion:22` - und ist durch andere teilnehmende Knoten des Overlay-Netzwerks ansprechbar.

Eine eingehende TCP-Verbindung einer TCP-Client-Anwendung über das Tor-Netzwerk zu diesem Overlay-Netzwerk-internen Endpunkt wird über eine TCP-Portweiterleitung auf Tor-Knoten *B* (*Node B*) an die TCP-Server-Anwendung weitergeleitet.[51]

Der Aufbau einer TCP-Verbindung durch eine TCP-Client-Anwendung über den Tor-Knoten *A* (*Node A*) zu einem Overlay-Netzwerk-internen Endpunkt (*Hidden Service*) wird durch eine SOCKS-Client-Verbindung[52,53] der TCP-Client-Anwendung zum im Tor-Knoten *A* (*Node A*) integrierten SOCKS-Proxy-Server[54] realisiert.

Abbildung 6.5 veranschaulicht die Weiterleitung einer TCP-Verbindung über das Overlay-Netzwerk Tor.

Die integrierte Funktion des SOCKS-Proxy-Servers wird von der Software Tor für SOCKS-Clients über den lokalen TCP-Port 9050 zur Verfügung gestellt. Es erscheint angemessen und geboten, aus Sicherheitserwägungen heraus den Zugriff auf die Funktionalität des SOCKS-Proxy-Servers auf lokale SOCKS-Clients zu beschränken; die Software Tor bietet hierfür die Möglichkeit von Zugriffskontrolllisten (*Access Control Lists*) für das Ge-

[45] Die bereitgestellte Version der Software Tor ist beim Betriebssystem *Debian Linux* durch das Kommando `apt-cache showpkg tor | head -3` abrufbar.

[46] The Tor Project Inc.: Tor Project: Download Tor - Source Code, URL: `https://www.torproject.org/download/download.html.en#source` [162]

[47] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151, S. 310][152, 303–320]

[48] The Tor Project Inc.: Tor Project: Documentation - FAQ - Running a Tor relay, URL: `https://www.torproject.org/docs/faq.html.en#RunningATorRelay` [159]

[49] The Tor Project Inc.: Tor Project: Documentation - FAQ - Running a Tor relay: I'm behind a NAT/Firewall, URL: `https://www.torproject.org/docs/faq.html.en#BehindANAT` [160]

[50] The Tor Project Inc.: Tor Project: Documentation - Hidden Service Protocol, URL: `https://www.torproject.org/docs/hidden-services.html.en` [190]

[51] The Tor Project Inc.: Tor Project: Documentation - Hidden Service Configuration Instructions, URL: `https://www.torproject.org/docs/tor-hidden-service.html.en` [191]

[52] *SOCKS* ist ein Protokoll für *Circuit Level Proxies*; eine Anwendung nutzt einen *SOCKS*-Server als Stellvertreter für den Pakettransport, die nutzende Anwendung wird *SOCKS*-Client genannt.[53]

[53] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "SOCKS" [8, S. 779]

[54] Roger Dingledine, Nick Mathewson und Paul F. Syverson: Tor: The Second-Generation Onion Router, In: Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004 [151, S. 308][152, 303–320]

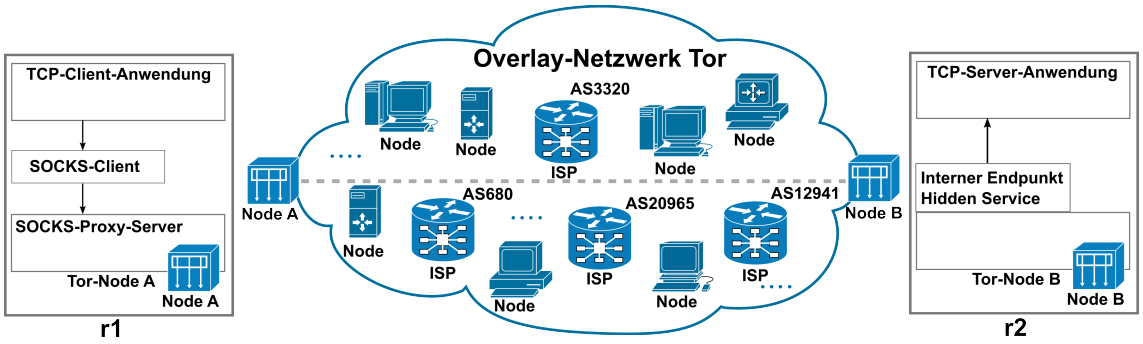

Abbildung 6.5: Weitergeleitete TCP-Verbindungen über das Overlay-Netzwerk Tor

währen oder das Verweigern des Zugriffs auf die integrierte Funktion des SOCKS-Proxy-Servers an. Für diese Zugriffskontrolllisten wird bei der Software Tor das Schlüsselwort `SocksPolicy` verwendet.

Die Implementierung der Funktion *eines* Overlay-Netzwerkknotens für Router `r1` mit der Weiterleitung von TCP-Verbindungen ist auszugsweise in Anhang A.7.1 dargestellt.

6.4.1 Anpassungen für zwei Overlay-Netzwerkknoten-Paare

Zur Erhöhung der Verfügbarkeit der darauf aufbauenden Dienste erscheint es sinnvoll, die Funktion der Overlay-Netzwerkknoten für die Router-Appliances `r1` und `r2` paarweise redundant zu implementieren. Auf Router `r1` wird die Funktionalität von zwei Overlay-Netzwerkknoten implementiert, auf Router `r2` wird ebenso die Funktionalität von zwei Overlay-Netzwerkknoten implementiert.

Die Implementierung der Funktionalität eines zweiten Overlay-Netzwerkknotens für das Tor-Netzwerk auf der Router-Appliance lässt sich untergliedern in drei Bereiche: zum Einen in die Anpassung der genutzten Ressourcen wie z.B. der TCP-Ports und *Unix domain sockets* für die zweite Instanz des Overlay-Netzwerkknotens, zum Anderen in die Integration der zweiten Instanz des Overlay-Netzwerkknotens in das Unix-Laufzeitsystem (*BSD runlevel*) und des Weiteren in Erweiterungen des Kernel-Sicherheitssystems (*AppArmor*)[55], welches für Prozesse, Dienste und Anwendungen auf dem eingesetzten Betriebssystem *Debian Linux* die Zugriffsmöglichkeiten (*capabilities*) steuert.

Die integrierte Funktion des SOCKS-Proxy-Servers wird von der zweiten Instanz des Overlay-Netzwerkknotens über den lokalen TCP-Port 9051 für SOCKS-Clients zur Verfügung gestellt.

Die Implementierung der Funktion *eines zweiten* Overlay-Netzwerkknotens für Router `r1` mit der Weiterleitung von TCP-Verbindungen ist auszugsweise in Anhang A.7.2 dargestellt.

6.4.2 Testen

Eine Möglichkeit, die Funktionalität der beiden Overlay-Netzwerkknoten-Paare auf Router `r1` und auf Router `r2` zu testen, besteht darin, jeweils die Funktionalität der TCP-Weiterleitung durch das Overlay-Netzwerk hindurch mit dem SSH-Dienst zu testen:

```
root@r1:~# cat /etc/tor/torrc | grep -e "^HiddenService"
HiddenServiceDir /var/lib/tor/hidden_service_10022/
HiddenServicePort 10022 127.0.0.1:22
```

[55] AppArmor security project: AppArmor - Linux application security system, URL: `http://apparmor.net/` [214]

```
root@r1:~# cat /var/lib/tor/hidden_service_10022/hostname
ryqoqnzfmomxora2.onion
root@r1:~#

root@r2:~# cat /etc/tor/torrc | grep -e "^SocksPort"
SocksPort 127.0.0.1:9050 IsolateClientProtocol IsolateDestPort PreferSOCKSNoAuth
root@r2:~# # ssh client connect to r1 via tor node1
root@r2:~# ssh \
  -o'ProxyCommand socat STDIO SOCKS4A:127.0.0.1:%h:%p,socksport=9050' \
  -p 10022 -l root ryqoqnzfmomxora2.onion \
  cat /etc/hostname
r1
root@r2:~#


root@r1:~# cat /etc/tor/torrc2 | grep -e "^HiddenService"
HiddenServiceDir /var/lib/tor2/hidden_service_10022/
HiddenServicePort 10022 127.0.0.1:22
root@r1:~# cat /var/lib/tor2/hidden_service_10022/hostname
2vvfivuimbvnjav5.onion
root@r1:~#

root@r2:~# cat /etc/tor/torrc2 | grep -e "^SocksPort"
SocksPort 127.0.0.1:9051 IsolateClientProtocol IsolateDestPort PreferSOCKSNoAuth
root@r2:~# # ssh client connect to r1 via tor node2
root@r2:~# ssh \
  -o'ProxyCommand socat STDIO SOCKS4A:127.0.0.1:%h:%p,socksport=9051' \
  -p 10022 -l root 2vvfivuimbvnjav5.onion \
  cat /etc/hostname
r1
root@r2:~#
```

6.5 Integration in eine Plattform

Die Integration in eine Plattform gliedert sich in mehrere Teilbereiche: zum Einen in die Integration der *Overlay-Netzwerkknoten*, der *VPN-Komponente* und der *Komponenten für das dynamische Routing*, zum Anderen in die Entwicklung der steuernden Laufzeitumgebung in Form eines erweiterbaren Frameworks und des Weiteren in die Vervollständigung der Laufzeitumgebung zu einer produktiv nutzbaren Router-Plattform.

6.5.1 Integration in einen ersten Prototyp

Die Integration in einen ersten Prototyp gliedert sich in mehrere Phasen: in die Integration der *Overlay-Netzwerkknoten* mit der *VPN-Komponente* und sich daran anschliessend in die darauf aufbauende Integration mit den *Komponenten für das dynamische Routing*.

6.5.1.1 Integration der Overlay-Netzwerkknoten mit der VPN-Komponente

Die gemeinsame Schnittstelle der Overlay-Netzwerkknoten und der VPN-Software ist das SOCKS-Protokoll[56,57].

[56] *SOCKS* ist ein Protokoll für *Circuit Level Proxies*; eine Anwendung nutzt einen *SOCKS*-Server als Stellvertreter für den Pakettransport, die nutzende Anwendung wird *SOCKS*-Client genannt.[57]
[57] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "SOCKS" [8, S. 779]

Die Overlay-Netzwerkknoten stellen mit ihrer Funktionalität als SOCKS-Proxy-Server der VPN-Software in der Rolle als SOCKS-Client redundante Datenpfade für den Pakettransport zur Verfügung. Die Overlay-Netzwerkknoten stellen die Funktionalität als SOCKS-Proxy-Server über den lokalen TCP-Port 9050 beim ersten Overlay-Netzwerkknoten und über den lokalen TCP-Port 9051 beim zweiten Overlay-Netzwerkknoten für die VPN-Software in der Rolle als SOCKS-Client zur Nutzung zur Verfügung.

Die Implementationsdetails der Integration des jeweils ersten Overlay-Netzwerkknotens von Router `r1` und von Router `r2` mit einem darauf aufbauenden VPN-Tunnel sind auszugsweise in Anhang A.8.1 dargestellt.

Abbildung 6.6 veranschaulicht das Zusammenwirken eines der Overlay-Netzwerkknoten-Paare mit einem darauf aufbauendem VPN-Tunnel.

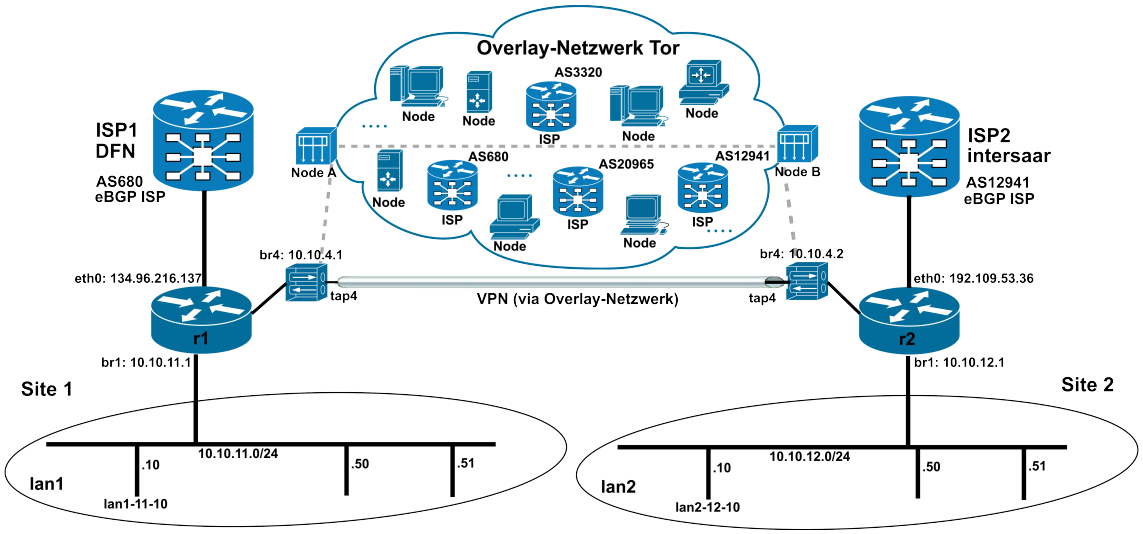

Abbildung 6.6: Veranschaulichung der Integration der Overlay-Netzwerkknoten mit einem darauf aufbauendem VPN-Tunnel

6.5.1.2 Integration mit den Komponenten für das dynamische Routing mit OSPF

Das Motiv, dynamisches Routing mit OSPF zu nutzen, ist es, bei Ausfall oder bei Störung der präferierten primären nativen VPN-Tunnel den Datenverkehr hilfsweise durch die auf Overlay-Netzwerkknoten aufbauenden VPN-Tunnel zu leiten.

Um eine Präferenz bei den Leitungswegen auszudrücken, existiert beim dynamischen Routing mit OSPF neben anderen diesbezüglichen Optionen die Möglichkeit, jedem Leitungsweg einen Gewichtungsfaktor zuzuordnen, der *cost* genannt wird.[58] Um die höhere Präferenz auf die primären nativen VPN-Tunnel vor den auf Overlay-Netzwerkknoten aufbauenden VPN-Tunneln auszudrücken, wird den primären nativen VPN-Tunneln der Wert `cost = 50` zugeordnet und den auf Overlay-Netzwerkknoten aufbauenden VPN-Tunneln der Wert `cost = 250` zugeordnet.

Für das Routing mit dem dynamischen Routing-Protokoll OSPF zwischen den beiden Routern `r1` und `r2` werden periodisch Informationen über die Verbindungszustände ermittelt; dies geschieht durch das Versenden von IP-Datagrammen als *Multicast*[59] über die definierten Leitungswege zwischen den beteiligten Routern. Ein solches IP-Datagramm

[58] Kunihiro Ishiguro et al.: Quagga - A routing software package for TCP/IP networks - Quagga 1.2.0, 2017, URL: `http://www.nongnu.org/quagga/docs/quagga.pdf` [213, S. 43]

[59] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Multicast" [8, S. 553]

wird beim Routing-Protokoll OSPF als Hello-Paket (*Hello Packet*) bezeichnet (siehe Abschnitt 2.1.2.2). Die Periode für das Versenden dieser Hello-Pakete zur Ermittlung von Informationen über die Verbindungszustände ist einstellbar, der übliche Wert für das zeitliche Intervall zwischen solchen Hello-Paketen beträgt 10 Sekunden.[60]

Der Wert für das zeitliche Intervall zwischen Hello-Paketen wird für die primären nativen VPN-Tunnel auf 2 Sekunden festgelegt, um Ausfälle oder Störungen dieser Tunnel rasch erkennen zu können; der Wert für das zeitliche Intervall zwischen Hello-Paketen wird hingegen für die auf Overlay-Netzwerkknoten aufbauenden VPN-Tunnel auf 5 Sekunden festgelegt.

Ein Auszug aus dem Setup von Router `r1` für das dynamische Routing mit OSPF mit Präferenzen bei den dargestellten Leitungswegen ist in Anhang A.8.2 gelistet.

Abbildung 6.7 veranschaulicht die Integration von nativen VPN-Tunneln über das Internet und von auf Overlay-Netzwerkknoten aufbauenden VPN-Tunneln mit dynamischem Routing.

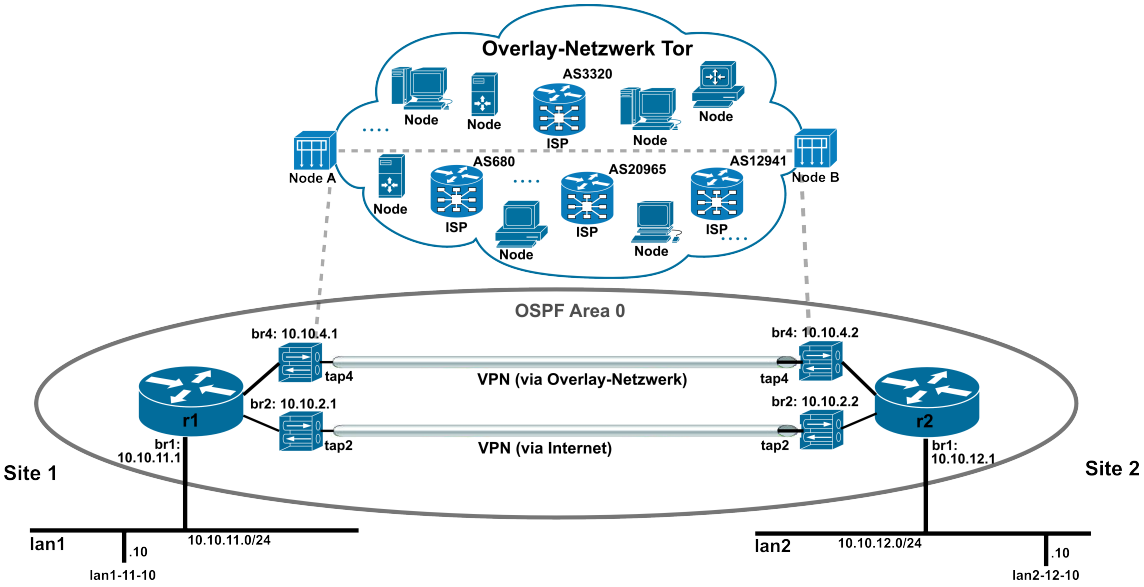

Abbildung 6.7: Veranschaulichung der Integration nativer VPN-Tunnel und auf Overlay-Netzwerkknoten aufbauender VPN-Tunnel mit dynamischem Routing

6.5.2 Entwicklung der Laufzeitumgebung

Auf die Integration der vorgenannten Komponenten in einen ersten Prototyp für die zugrunde liegende Systemtechnik und -architektur (siehe Abschnitt 6.5.1) baut die Entwicklung des Codes der Laufzeitumgebung auf.

Die hauptsächliche Anforderung an die Laufzeitumgebung ist die Überwachung und Steuerung der Zustände der Komponenten, das Anstossen von Zustandsübergängen (*Transitionen*) und die Überwachung des Gesamtsystemzustands; des Weiteren ist eine Anforderung an die Laufzeitumgebung das revisionssichere Protokollieren (*loggen*) der Zustände und Zustandsübergänge. Die Überwachung und Steuerung der Zustände der Komponenten bedingt Funktionen, relevante Daten aus den Teilsystemen zu erfassen, um

[60] Kunihiro Ishiguro et al.: Quagga - A routing software package for TCP/IP networks - Quagga 1.2.0, 2017, URL: `http://www.nongnu.org/quagga/docs/quagga.pdf` [213, S. 46]

die Zustände der Teilsysteme zu bestimmen.

Die Entwicklung der Laufzeitumgebung folgt vom Vorgehen her dem Entwicklungsparadigma des *Rapid Prototyping*.
Als Programmiersprache zur Entwicklung der Laufzeitumgebung wird die Skriptsprache der klassischen *Bourne shell* (*AT&T Shell*, *sh*) gewählt; die Wahl einer Skriptsprache als Programmiersprache ist durch den zu erwartenden raschen Entwicklungsfortschritt motiviert, die Wahl der Skriptsprache der klassischen *Bourne shell* ist durch deren breite Verfügbarkeit für eine Vielzahl unterschiedlicher Betriebssysteme begründet. Auf die Verwendung von Sprachelementen der auf Betriebssystemen wie z.B. *Linux* verbreiteten *Bourne again shell*[61] (*GNU bash*) wird verzichtet, um möglichst hohe Portabilität und grösstmögliche Kompatibilität zu gewährleisten.
Die Skriptsprache der klassischen *Bourne shell* als Programmiersprache für die Laufzeitumgebung impliziert teilweise die Verwendung von Code-Konstrukten, deren Einsatz der Nichtverfügbarkeit von Sprachelementen bei der *Bourne shell* gegenüber der *Bourne again shell* geschuldet ist.

Die Entwicklung der Laufzeitumgebung gestaltet sich vom Ablauf her in drei Phasen: in der ersten Phase werden notwendige Funktionen entwickelt, um relevante Daten aus den Komponenten zu erfassen, mit denen Zustände bestimmt werden können; in der sich daran anschliessenden zweiten Phase werden notwendige Funktionen entwickelt, um Zustandsübergänge anzustossen; in der dritten Phase werden die Funktionen mit Steuerungslogik zusammengefügt. Dieses Vorgehen entspricht dem *Bottom up*[62]-Ansatz in der Systementwicklung.
Ein Testen der Funktionen und der Steuerungslogik schliesst sich bei der Entwicklung jeweils zeitlich an.

Als Struktur für die Laufzeitumgebung wird ein Framework gewählt; die Wahl fusst auf der Anforderung an die Laufzeitentwicklung, erweitert werden zu können und anpassungsfähig bezüglich zukünftiger Einsatzzwecke zu sein.

Das Framework ist gegliedert in die Laufzeitumgebung mit den Kernfunktionen für Ablauflogik, Überwachung und Steuerung als Hauptbestandteil (`router-framework-runtime`, siehe Listing A.6) und in einen Bibliotheksbestandteil (`router-framework-lib`). Der Bibliotheksbestandteil ist weiter untergliedert nach Funktion und Zweck und nach Zuordenbarkeit zu den Komponenten.

Für die Laufzeitumgebung werden zwei verschiedene Betriebsarten für unterschiedliche Anforderungen gewählt: zum Einen die Betriebsart mit Ausgabe auf der Konsole - diese Betriebsart ist die bevorzugte Betriebsart während der Entwicklung; zum Anderen die Betriebsart als *Daemon* mit Protokollierung der Ausgabe in das *syslog*[63] eines Syslog-Servers - diese Betriebsart ist die Betriebsart in der Produktivumgebung.

In der Betriebsart mit Ausgabe auf der Konsole wird der Laufzeitumgebungsbestandteil `router-framework-runtime` ausgeführt, in der Betriebsart für die Produktivumgebung mit Protokollierung der Ausgabe auf einen Syslog-Server wird hingegen der Laufzeitumgebungsbestandteil `router-framework-runtime-daemon` ausgeführt; in beiden Be-

[61] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Bourne again shell" [8, S. 117]
[62] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Bottom up" [8, S. 117]
[63] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "syslog" [8, S. 822]

triebsarten wird der gleiche Bibliotheksbestandteil `router-framework-lib` der Laufzeitumgebung genutzt.

Listing 6.1 zeigt die Ausgaben der Laufzeitumgebung auf Router `r1` in der Betriebsart mit Ausgabe auf der Konsole; die Ausgabe ist verbos.

Listing 6.1: Transkript der Laufzeitumgebung auf Router `r1` in der Betriebsart mit Ausgabe auf der Konsole

```
root@r1:~# /usr/local/sbin/router-framework-runtime.sh
do_init_qos_traffic_shaping()
do_init_qos_traffic_shaping() stop traffic shaping
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh stop br4
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh stop br5
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh stop br6
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh stop br7
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh stop br8
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh stop br9
do_init_qos_traffic_shaping() start traffic shaping
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh start br4 4000 1000
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh start br5 4000 1000
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh start br6 4000 1000
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh start br7 4000 1000
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh start br8 4000 1000
/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh start br9 4000 1000
do_check_internet_connectivity()
do_check_internet_connectivity() online
do_check_remote_router_external()
do_check_remote_router_external() remote router r2 ext-if online
do_check_native_vpn_tunnels()
do_check_native_vpn_tunnel() peer r2-br2 online
do_check_native_vpn_tunnel() peer r2-br10 online
do_check_native_vpn_tunnels() native vpn tunnels count 2
do_check_overlay_nodes()
do_check_overlay_node1() online
do_check_overlay_node2() online
do_check_overlay_nodes() overlay nodes count 2
do_check_overlay_vpn_tunnels()
do_check_overlay_vpn_tunnel() peer r2-br4 online
do_check_overlay_vpn_tunnel() peer r2-br5 online
do_check_overlay_vpn_tunnel() peer r2-br6 online
do_check_overlay_vpn_tunnel() peer r2-br7 online
do_check_overlay_vpn_tunnel() peer r2-br8 online
do_check_overlay_vpn_tunnel() peer r2-br9 online
do_check_overlay_vpn_tunnels() overlay vpn tunnels count 6
do_check_remote_router_lan()
do_check_remote_router_lan() remote router r2 lan online
do_adjust_qos_traffic_shaping()
do_wait() 5s
^C
```

6.5.3 Vervollständigung der Laufzeitumgebung

Das Framework ist gegliedert in die Laufzeitumgebung mit den Kernfunktionen für Ablauflogik, Überwachung und Steuerung als Hauptbestandteil (`router-framework-runtime`, siehe Listing A.6) und in einen Bibliotheksbestandteil (`router-framework-lib`, siehe Listing A.8). Der Bibliotheksbestandteil ist weiter untergliedert nach Funktion und Zweck und nach Zuordenbarkeit zu Komponenten.

Der Bibliotheksbestandteil der Laufzeitumgebung untergliedert sich wie folgt:

- `router-framework-init`
  enthält Routinen für die Initialisierung und Standardwerte (siehe Listing A.7)
- `router-framework-lib-util`
  enthält von anderen Teilen der Laufzeitumgebung genutzte Funktionen (siehe Listing A.9)
- `router-framework-lib-connectivity`
  enthält die Internet-Connectivity betreffende Funktionen (siehe Listing A.10)
- `router-framework-lib-overlay`
  enthält das Overlay-Netzwerk betreffende Funktionen (siehe Listing A.11)
- `router-framework-lib-vpn`
  enthält das VPN betreffende Funktionen (siehe Listing A.12).
- `router-framework-lib-router`
  enthält die beiden Router betreffende Funktionen (siehe Listing A.13)
- `router-framework-lib-qos`
  enthält die Dienstgüte[64] und Dienstqualität (QoS[65], Quality of Service[66]) betreffende Funktionen (siehe Listing A.14)

Im Teil für Dienstgüte und Dienstqualität (`router-framework-lib-qos`) sind Funktionen enthalten, die das Priorisieren von IP-Paketdaten nach Paketgrösse und nach Pakettyp feingranuliert regeln, indem die IP-Paketdaten in unterschiedliche Klassen mit unterschiedlichen Charakteristiken eingeteilt werden (`do_init_qos_traffic_shaping()`); dies wird als *Traffic Shaping*[67,68] bezeichnet.

Das implementierte *Traffic Shaping* besteht aus drei ausgehenden Warteschlangen und einer eingehenden Warteschlange und drei Klassen für die Klassifizierung der IP-Pakete.

[64] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Dienstgüte" [8, S. 226]
[65] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "QoS" [8, S. 669]
[66] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 404]
[67] Andrew S. Tanenbaum und David J. Wetherall: Computer networks, 5. Aufl., 2011 [9, S. 407]
[68] Gary A. Donahue: Network Warrior, 2. Aufl., 2011 [215, S. 582]

Listing 6.2 zeigt Statistiken über das *Traffic Shaping* an der Netzwerkbrücke `br4` auf Router `r1`; die nämliche Netzwerkbrücke `br4` ist Teil eines auf dem Overlay-Netzwerk aufbauendem VPN-Tunnels zwischen Router `r1` und Router `r2`.

Listing 6.2: Statistiken über das *Traffic Shaping* an der Netzwerkbrücke `br4` auf Router `r1`

```
root@r1:~# tc -s qdisc ls dev br4
qdisc cbq 1: root refcnt 2 rate 10Mbit (bounded,isolated) prio no-transmit
 Sent 29148589 bytes 53360 pkt (dropped 0, overlimits 6375 requeues 0)
 backlog 0b 0p requeues 0
  borrowed 0 overactions 0 avgidle 12500 undertime 0
qdisc sfq 10: parent 1:10 limit 127p quantum 1514b depth 127 divisor 1024 perturb 10sec
 Sent 1357889 bytes 14381 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 20: parent 1:20 limit 127p quantum 1514b depth 127 divisor 1024 perturb 10sec
 Sent 27636911 bytes 36341 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc sfq 30: parent 1:30 limit 127p quantum 1514b depth 127 divisor 1024 perturb 10sec
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
qdisc ingress ffff: parent ffff:fff1 ----------------
 Sent 8864628 bytes 40083 pkt (dropped 14, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
root@r1:~#

root@r1:~# tc -s class ls dev br4
class cbq 1: root rate 10Mbit (bounded,isolated) prio no-transmit
 Sent 153831 bytes 2639 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
  borrowed 0 overactions 0 avgidle 12500 undertime 0
class cbq 1:1 parent 1: rate 1Mbit (bounded,isolated) prio 5
 Sent 28995388 bytes 50123 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
  borrowed 5329 overactions 0 avgidle 12500 undertime 0
class cbq 1:10 parent 1:1 leaf 10: rate 1Mbit prio 1
 Sent 1358085 bytes 14383 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
  borrowed 0 overactions 0 avgidle 12500 undertime 0
class cbq 1:20 parent 1:1 leaf 20: rate 900Kbit prio 2
 Sent 27637303 bytes 35740 pkt (dropped 0, overlimits 6931 requeues 0)
 backlog 0b 0p requeues 0
  borrowed 5329 overactions 3061 avgidle 12500 undertime 0
class cbq 1:30 parent 1:1 leaf 30: rate 800Kbit prio 2
 Sent 0 bytes 0 pkt (dropped 0, overlimits 0 requeues 0)
 backlog 0b 0p requeues 0
  borrowed 0 overactions 0 avgidle 12500 undertime 0
root@r1:~#
```

6.5.4 Testen

Die Entwicklung und das Testen einzelner Funktionen geschieht primär auf den Systemen der Laborumgebung; nach dem *Deployment* in die Produktivumgebung geschieht primär das Testen der Steuerungslogik und des Laufzeitverhaltens.

Ein Testfall für das Laufzeitverhalten ist der spontane Wegfall der Leitungswege der direkten nativen VPN-Verbindungen.

Das nachfolgende Transkript zeigt die Übertragung von IP-Datagrammen vom Typ *ICMP Echo Request*[69,70,71] von einer am LAN `lan1` über Router `r1` angebundenen Arbeitsstation an eine Arbeitsstation, die am LAN `lan2` über Router `r2` angebunden ist; zeitlich damit einhergehend werden die direkten nativen VPN-Verbindungen getrennt[72]:

```
Sep 15 00:09:21 r2-ext logger: Stopping virtual private network daemon: r1-r2-tcp-1195-tap10-server.
Sep 15 00:09:21 r2-ext logger: Stopping virtual private network daemon: r1-r2-tcp-1194-tap2-server.

Sep 15 00:09:18 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=2 ttl=62 time=22.0 ms
Sep 15 00:09:19 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=3 ttl=62 time=22.2 ms
Sep 15 00:09:20 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=4 ttl=62 time=22.1 ms
Sep 15 00:09:24 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=7 ttl=62 time=538 ms
Sep 15 00:09:25 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=8 ttl=62 time=540 ms
Sep 15 00:09:26 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=9 ttl=62 time=440 ms
```

Das nachfolgende Transkript zeigt die Übertragung von IP-Datagrammen vom Typ *ICMP Echo Request* während zeitlich damit einhergehend die direkten nativen VPN-Verbindungen wiederhergestellt werden:

```
Sep 15 00:14:57 r2-ext logger: Starting virtual private network daemon: r1-r2-tcp-1195-tap10-server.
Sep 15 00:14:57 r2-ext logger: Starting virtual private network daemon: r1-r2-tcp-1194-tap2-server.

Sep 15 00:14:55 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=2 ttl=62 time=505 ms
Sep 15 00:14:56 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=3 ttl=62 time=166 ms
Sep 15 00:14:57 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=4 ttl=62 time=818 ms
Sep 15 00:14:58 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=5 ttl=62 time=457 ms
Sep 15 00:14:59 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=6 ttl=62 time=456 ms
Sep 15 00:15:00 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=7 ttl=62 time=389 ms
Sep 15 00:15:01 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=8 ttl=62 time=695 ms
Sep 15 00:15:02 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=9 ttl=62 time=239 ms
Sep 15 00:15:03 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=10 ttl=62 time=750 ms
Sep 15 00:15:04 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=11 ttl=62 time=443 ms
Sep 15 00:15:05 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=12 ttl=62 time=945 ms
Sep 15 00:15:06 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=13 ttl=62 time=343 ms
Sep 15 00:15:07 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=14 ttl=62 time=443 ms
Sep 15 00:15:08 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=15 ttl=62 time=447 ms
Sep 15 00:15:09 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=16 ttl=62 time=504 ms
Sep 15 00:15:10 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=17 ttl=62 time=384 ms
Sep 15 00:15:11 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=18 ttl=62 time=376 ms
Sep 15 00:15:12 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=19 ttl=62 time=880 ms
Sep 15 00:15:13 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=20 ttl=62 time=631 ms
Sep 15 00:15:14 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=21 ttl=62 time=402 ms
Sep 15 00:15:15 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=22 ttl=62 time=150 ms
Sep 15 00:15:16 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=23 ttl=62 time=260 ms
Sep 15 00:15:17 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=24 ttl=62 time=28.1 ms
Sep 15 00:15:18 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=25 ttl=62 time=27.0 ms
Sep 15 00:15:19 lan1-11-10 logger: 64 bytes from lan2-12-10 (10.10.12.10): icmp_req=26 ttl=62 time=26.7 ms
```

[69] Bruce Hartpence: Packet Guide to Core Network Protocols, 2011 [209, S. 97]
[70] RFC 792: Internet Control Message Protocol, 1981, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc792.txt` [210]
[71] RFC 4884: Extended ICMP to Support Multi-Part Messages, 2007, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4884.txt` [211]
[72] Dies wird als *Failover* bezeichet.

6.6 Zusammenfassung

Abbildung 6.8 gibt einen Überblick über die Systemarchitektur der Routerumgebung mit dynamischem Routing mit OSPF, nativen VPN-Tunneln und mit auf Overlay-Netzwerkknoten aufbauenden VPN-Tunneln.

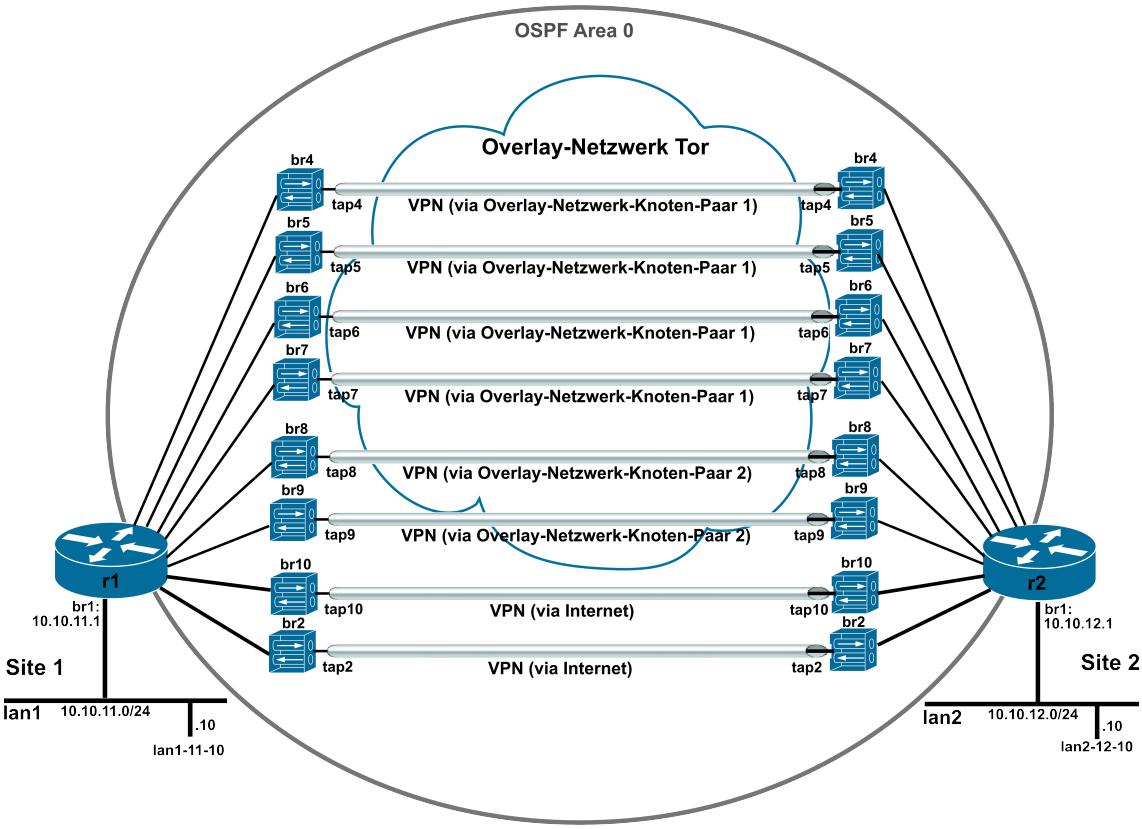

Abbildung 6.8: Überblick über die Systemarchitektur der Routerumgebung

Die das dynamische Routing mit OSPF betreffende Konfiguration von Router `r1` mit mehreren nativen VPN-Tunneln und mehreren auf Overlay-Netzwerkknoten aufbauenden VPN-Tunneln ist in Anhang A.8.3 dargestellt.

In der Produktivumgebung ist die Betriebsart der Routerumgebung die Betriebsart mit Protokollierung der Ausgabe in das *syslog*[73,74] eines Syslog-Servers. Oftmals wird ein zentraler Syslog-Server verwendet; die übliche Bezeichnung für einen solchen zentralen Syslog-Server ist *loghost*. Ein Vorteil eines Syslog-Servers für die Protokollierung ist, dass eine hohe Revisionssicherheit der Protokollierung gegeben ist.

Listing 6.3 zeigt die Syslog-Einträge der Laufzeitumgebung auf Router `r1` in der Betriebsart als Daemon mit Ausgabe auf den Syslog-Server mit dem *Syslog-Facility-Name* `local5` und dem *Syslog-Level-Name* `info`; die Ausgabe ist verbos.

[73] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "syslog" [8, S. 822]
[74] Bob Smith, John Hardin, Graham Phillips und Bill Pierce: Linux appliance design: A hands-on guide to building linux appliances, 2007 [216, S. 83]

Listing 6.3: Transkript der Syslog-Einträge der Laufzeitumgebung auf Router `r1` in der Betriebsart mit Ausgabe auf den Syslog-Server

```
Sep 15 01:57:14 r1 router-framework-runtime: do_check_remote_router_external()
Sep 15 01:57:14 r1 router-framework-runtime: do_check_remote_router_external() remote router r2 ext-if online
Sep 15 01:57:14 r1 router-framework-runtime: do_check_native_vpn_tunnels()
Sep 15 01:57:14 r1 router-framework-runtime: do_check_native_vpn_tunnel() peer r2-br2 online
Sep 15 01:57:14 r1 router-framework-runtime: do_check_native_vpn_tunnel() peer r2-br10 online
Sep 15 01:57:14 r1 router-framework-runtime: do_check_native_vpn_tunnels() native vpn tunnels count 2
Sep 15 01:57:14 r1 router-framework-runtime: do_check_overlay_nodes()
Sep 15 01:57:14 r1 router-framework-runtime: do_check_overlay_node1() offline
Sep 15 01:57:14 r1 router-framework-runtime: do_check_overlay_node2() offline
Sep 15 01:57:14 r1 router-framework-runtime: do_check_overlay_nodes() overlay nodes count 0
Sep 15 01:57:14 r1 router-framework-runtime: do_check_overlay_nodes() overlay nodes offline
Sep 15 01:57:14 r1 router-framework-runtime: do_restart_overlay_nodes()
Sep 15 01:57:14 r1 router-framework-runtime: Stopping tor (via systemctl): tor.service.
Sep 15 01:57:14 r1 router-framework-runtime: Stopping tor2 (via systemctl): tor2.service.
Sep 15 01:57:15 r1 router-framework-runtime: Starting tor (via systemctl): tor.service.
Sep 15 01:57:15 r1 router-framework-runtime: Starting tor2 (via systemctl): tor2.service.
Sep 15 01:57:15 r1 router-framework-runtime: do_delay_after_service_restart() 30s
Sep 15 01:57:45 r1 router-framework-runtime: do_check_overlay_vpn_tunnels()
Sep 15 01:57:46 r1 router-framework-runtime: do_check_overlay_vpn_tunnel() peer r2-br4 online
Sep 15 01:57:47 r1 router-framework-runtime: do_check_overlay_vpn_tunnel() peer r2-br5 offline
Sep 15 01:57:48 r1 router-framework-runtime: do_check_overlay_vpn_tunnel() peer r2-br6 offline
Sep 15 01:57:49 r1 router-framework-runtime: do_check_overlay_vpn_tunnel() peer r2-br7 offline
Sep 15 01:57:49 r1 router-framework-runtime: do_check_overlay_vpn_tunnel() peer r2-br8 online
Sep 15 01:57:50 r1 router-framework-runtime: do_check_overlay_vpn_tunnel() peer r2-br9 offline
Sep 15 01:57:50 r1 router-framework-runtime: do_check_overlay_vpn_tunnels() overlay vpn tunnels count 2
Sep 15 01:57:50 r1 router-framework-runtime: do_check_remote_router_lan()
Sep 15 01:57:50 r1 router-framework-runtime: do_check_remote_router_lan() remote router r2 lan online
Sep 15 01:57:50 r1 router-framework-runtime: do_adjust_qos_traffic_shaping()
Sep 15 01:57:50 r1 router-framework-runtime: do_wait() 5s
Sep 15 01:57:55 r1 router-framework-runtime: do_check_internet_connectivity()
Sep 15 01:57:55 r1 router-framework-runtime: do_check_internet_connectivity() online
Sep 15 01:57:55 r1 router-framework-runtime: do_check_remote_router_external()
Sep 15 01:57:55 r1 router-framework-runtime: do_check_remote_router_external() remote router r2 ext-if online
Sep 15 01:57:55 r1 router-framework-runtime: do_check_native_vpn_tunnels()
Sep 15 01:57:55 r1 router-framework-runtime: do_check_native_vpn_tunnel() peer r2-br2 online
Sep 15 01:57:55 r1 router-framework-runtime: do_check_native_vpn_tunnel() peer r2-br10 online
Sep 15 01:57:55 r1 router-framework-runtime: do_check_native_vpn_tunnels() native vpn tunnels count 2
Sep 15 01:57:55 r1 router-framework-runtime: do_check_overlay_nodes()
Sep 15 01:57:58 r1 router-framework-runtime: do_check_overlay_node1() online
Sep 15 01:58:29 r1 router-framework-runtime: do_check_overlay_node2() online
Sep 15 01:58:29 r1 router-framework-runtime: do_check_overlay_nodes() overlay nodes count 2
```

7 Evaluation

Die folgenden Abschnitte legen die Evaluation der Konzeption und Implementierung der Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks dar - es werden Indikatoren und Werkzeuge in Form Mess-Software vorgestellt und ausgewählt; daran schliesst sich die Erfassung und Auswertung der erhobenen Messdaten an.

7.1 Auswahl der Indikatoren

Die Indikatoren für die Güte der Konzeption und Implementierung der Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung sind zum Einen qualitative Indikatoren und zum Anderen quantitative Indikatoren.

7.1.1 Verfügbarkeit

Der Indikator *Verfügbarkeit* lässt sich im Kontext einer Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung den qualitativen Indikatoren zuordnen.

Eine Möglichkeit, den Indikator *Verfügbarkeit* zu messen und in Form eines Werts auszudrücken, besteht darin, periodisch - z.B. minütlich oder zehnsekündlich - durch das Versenden von IP-Datagrammen vom Typ *ICMP Echo Request*[1,2,3] die Erreichbarkeit der LANs untereinander zu prüfen und den Grad der Erreichbarkeit über einen Bemessungszeitraum - z.B. über den Zeitraum eines Tages oder über den Zeitraum eines Jahres - in Prozent darzustellen.

7.1.2 Rekonvergenz

Der Indikator *Rekonvergenz*[4] (siehe Kapitel 2 Grundlagen, Seite 11) lässt sich im Kontext einer Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung den qualitativen Indikatoren zuordnen - *Rekonvergenz* bemisst die Zeitspanne in Sekunden ausgedrückt, nach der bei Veränderungen der Netzwerktopologie durch z.B. den Wegfall von Transportpfaden alternative Transportpfade genutzt werden können.

Eine Möglichkeit, den Indikator *Rekonvergenz* zu messen, besteht darin, die nativen VPN-Tunnel - welche die primären zu Grunde liegenden Transportpfade für die LAN-LAN-Kopplung sind - zu unterbrechen, und zeitlich damit einhergehend das Rekonvergenzverhalten zu beobachten.

[1] Bruce Hartpence: Packet Guide to Core Network Protocols, 2011 [209, S. 97]
[2] RFC 792: Internet Control Message Protocol, 1981, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc792.txt` [210]
[3] RFC 4884: Extended ICMP to Support Multi-Part Messages, 2007, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4884.txt` [211]
[4] Prof. Dr. Norbert Pohlmann und Stefan Dierichs: So funktioniert Internet-Routing: Wie Routing dem Netz seine Selbstheilungskräfte verleiht, 2008 [2, S. 4]

7.1.3 Latenz

Der Indikator *Latenz*[5] lässt sich im Kontext einer Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung den qualitativen Indikatoren zuordnen - *Latenz* ist die Paketlaufzeit von IP-Datagrammen vom einem absendenden Computer zu einem anderen empfangenden Computer, angegeben in Millisekunden; eine übliche Bezeichnung für die *Latenz* von IP-Datagrammen ist der Begriff *packet delay*.[6]

Eine Möglichkeit, den Indikator *Latenz* zu messen, besteht darin, durch das Versenden von IP-Datagrammen vom Typ *ICMP Echo Request*[7,8,9] mit dem Kommandozeilen-Netzwerkhilfsprogramm `ping`[10] die Umlaufzeit (*Round Trip Time - RTT*) der versandten IP-Datagramme zu messen; die Latenz - die Paketlaufzeit - beträgt in ausreichender Präzision[11] die Hälfte der Umlaufzeit (*Round Trip Time*).

7.1.4 Datendurchsatz

Der Indikator *Datendurchsatz* lässt sich im Kontext einer Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung den quantitativen Indikatoren zuordnen - *Datendurchsatz* (*throughput*) bemisst die nutzbare Bandbreite, die für Datenübertragungen von IP-Datagrammen verfügbar ist, angegeben in Bit/s[12].

Eine Möglichkeit, den Indikator *Datendurchsatz* (*throughput*) zu messen, besteht darin, durch das Versenden von IP-Datagrammen von einem absendenden Computer und dem damit einhergehenden Empfang der IP-Datagramme durch einen anderen empfangenden Computer mit Hilfe geeigneter Mess-Software die Anzahl der übertragenen Bits pro Sekunde zu messen.

7.2 Auswahl der Erfassungs-Tools

Die folgenden Abschnitte stellen Erfassungs-Tools - Werkzeuge in Form von Mess-Software - vor, es werden für die Auswahl relevante Eigenschaften der Softwares dargestellt.

7.2.1 iperf3

iperf3[13] ist eine frei zugängliche und frei verwendbare Mess-Software, die es ermöglicht, *aktive* Bandbreitenmessungen zwischen zwei Computern durchzuführen. Gemessen werden kann die maximale Bandbreite, die für Datenübertragungen zur Verfügung steht

[5] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "Latenz; -zeit" [8, S. 476]
[6] RFC 2330: Framework for IP Performance Metrics, 1998, Informational, URL: `http://www.rfc-editor.org/rfc/rfc2330.txt` [217, S. 10]
[7] Bruce Hartpence: Packet Guide to Core Network Protocols, 2011 [209, S. 97]
[8] RFC 792: Internet Control Message Protocol, 1981, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc792.txt` [210]
[9] RFC 4884: Extended ICMP to Support Multi-Part Messages, 2007, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4884.txt` [211]
[10] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "ping" [8, S. 629]
[11] RFC 2330: Framework for IP Performance Metrics, 1998, Informational, URL: `http://www.rfc-editor.org/rfc/rfc2330.txt` [217, S. 7]
[12] RFC 2330: Framework for IP Performance Metrics, 1998, Informational, URL: `http://www.rfc-editor.org/rfc/rfc2330.txt` [217, S. 9]
[13] Energy Sciences Network (ESnet) und Lawrence Berkeley National Laboratory: iperf3 Project, URL: `http://software.es.net/iperf/` [218]

(*throughput*), sowie Übertragungsparameter, die mit der Datenübertragung in Zusammenhang stehen, wie beispielsweise die Anzahl verlorengegangener Pakete (*packet loss*).[13]

Zum Realisieren der Bandbreitenmessungen kommuniziert eine Client-Instanz von *iperf3* auf einem Computer mit einer Server-Instanz vom *iperf3* auf einem anderem Computer und überträgt Daten; in festlegbaren zeitlichen Intervallen werden statistische Daten über den Datendurchsatz im betreffenden Intervall ausgegeben.
Unter den Möglichkeiten der Ausgabe ist die Ausgabe Zeile für Zeile mit den Spalten "ID der Messung", "transferierte Daten im Intervall" und "Datendurchsatz im Intervall"; über die Möglichkeit der zeilenweisen Ausgabe hinaus ist die Möglichkeit gegeben, die statistischen Daten der Datenübertragung in maschinenlesbaren Format im JSON-Format auszugeben.[13]

Als Übertragungsprotokoll für die Messungen wird das TCP-Protokoll oder das UDP-Protokoll genutzt, möglich ist auch die Nutzung des SCTP-Protokolls (Stream Control Transmission Protocol)[14,15,16] für die Durchführung der Messungen.

iperf3 ist in der Programmiersprache *C++* geschrieben und ist primär für die Betriebssysteme *Linux*, *FreeBSD* und Apple *MacOS/X* verfügbar,[13] d.h. der Quellcode ist - ohne dass weitere Anpassungen notwendig sind - auf diesen Betriebssystemen übersetzbar und funktional ausführbar; diese Betriebssysteme werden von den Entwicklern von *iperf3* offiziell unterstützt.[13] Unter den Betriebssystemen *OpenBSD*, *NetBSD* und Google *Android* ist *iperf3* jedoch ebenfalls übersetzbar und funktional ausführbar.[13]
Die aktuelle Version von *iperf3* ist Version 3.2 vom 26.06.2017 (Stand 27.06.2017).[13]
Der Quellcode der Software ist unter der URL `http://downloads.es.net/pub/iperf/` abrufbar;[13,17] des Weiteren ist der Quellcode des aktuellen Entwicklungszweiges der Software auf der Internetseite "GitHub - esnet/iperf: iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool" unter der URL `https://github.com/esnet/iperf` abrufbar.[13,18]

Weiterführende Informationen zur Funktionalität von *iperf3* und Dokumentation zu *iperf3* sind auf der Internetseite "iperf3 – iperf3 3.2 documentation" unter der URL `http://software.es.net/iperf/` abrufbar.[13]

Es existieren öffentlich zugängliche *iperf3*-Server, um Bandbreitenmessungen - z.B. zur Erstellung von *Baselines* - zu bewerkstelligen; eine Auflistung öffentlich zugänglicher *iperf3*-Server für den zu messenden Bandbreitenbereich bis 10 Gbit/s ist unter der URL `https://iperf.fr/iperf-servers.php` abrufbar.[19]

Eine hervorhebenswerte Eigenschaft der Software *iperf3* ist die Möglichkeit der Ausgabe der statistischen Daten der Bandbreitenmessungen im JSON-Format.

[14] RFC 2960: Stream Control Transmission Protocol, 2000, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc2960.txt` [219]
[15] RFC 3286: An Introduction to the Stream Control Transmission Protocol (SCTP), 2002, Informational, URL: `http://www.rfc-editor.org/rfc/rfc3286.txt` [220]
[16] RFC 4960: Stream Control Transmission Protocol, 2007, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4960.txt` [221]
[17] Energy Sciences Network (ESnet) und Lawrence Berkeley National Laboratory: iperf3 Project - Source Code downloads, URL: `http://downloads.es.net/pub/iperf/` [222]
[18] GitHub Inc.: GitHub - esnet/iperf: iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool, URL: `https://github.com/esnet/iperf` [223]
[19] Vivien Gueant: Iperf.fr - Public iPerf3 servers, URL: `https://iperf.fr/iperf-servers.php` [224]

7.2.2 vnStat

vnStat[20] ist ein Netzwerk-Monitoring-Werkzeug, das es ermöglicht, Statistiken über den Netzwerkverkehr von Servern und Routern aufbereitet in Textform oder in Diagramm-Form darzustellen; *vnStat* ist eine frei zugängliche und frei verwendbare Software.[20]

Zum Erheben der Daten über den Netzwerkverkehr verwendet *vnStat* die internen Statistiken des Betriebssystems über die Netzwerkschnittstellen,[20] z.B. über die Netzwerkschnittstellen mit den Bezeichnern *eth0*, *eth1* oder *ppp0* unter dem Betriebssystem *Linux*; die erhobenen Daten werden von *vnStat* in eine interne Datenbank geschrieben und bei Bedarf aus dieser Datenbank ausgelesen, um diese aufbereitet darzustellen.

vnStat ist für die Betriebssysteme *Linux* und Betriebssysteme der *BSD*-Familie (*FreeBSD*, *NetBSD*, *OpenBSD*) verfügbar; die Unterstützung des Betriebssystems Apple *MacOS/X* ist in experimentellem Stadium. *vnStat* ist in der Programmiersprache *C* geschrieben.[21]

Die aktuelle Version von *vnStat* ist Version 1.17 vom Februar 2017 (Stand 28.06.2017).[20] Der Quellcode der Software ist unter der URL `http://humdi.net/vnstat/vnstat-1.17.tar.gz` abrufbar;[20] des Weiteren ist der Quellcode des aktuellen Entwicklungszweiges der Software unter der URL `https://github.com/vergoh/vnstat` sowie unter der URL `https://github.com/vergoh/vnstat/archive/master.zip` abrufbar.[20,21]

Weiterführende Informationen zur Funktionalität von *vnStat* und Dokumentation zu *vnStat* sind auf der Internetseite "vnStat - a network traffic monitor for Linux and BSD" unter der URL `http://humdi.net/vnstat/` abrufbar.[20]

Eine suboptimale Eigenschaft der Software *vnStat* in Hinblick auf Bandbreitenmessungen mit feiner zeitlicher Granulierung ist die kleinste zeitliche Auflösung von zwei Sekunden - dies ist das kleinstmögliche definierbare zeitliche Intervall zum periodischen Abfragen der internen Statistiken des Betriebssystems über die Netzwerkschnittstellen durch *vnStat*.[22]

7.2.3 MRTG

MRTG[23] ist ein Netzwerk-Monitoring-Werkzeug, das es ermöglicht, Statistiken über den Netzwerkverkehr von Servern und Routern aufbereitet in Diagramm-Form darzustellen; *MRTG* ist eine frei zugängliche und frei verwendbare Software.[23,24]

[20] Teemu Toivola: vnStat - a network traffic monitor for Linux and BSD, URL: `http://humdi.net/vnstat/` [225]
[21] GitHub Inc.: GitHub - vergoh/vnstat: vnStat - a network traffic monitor for Linux and BSD, URL: `https://github.com/vergoh/vnstat` [226]
[22] Teemu Toivola: vnStat - Documentation - Man page of vnstat.conf, URL: `http://humdi.net/vnstat/man/vnstat.conf.html` [227]
[23] Tobias Oetiker, Oetiker+Partner AG: MRTG - Multi Router Traffic Grapher, URL: `https://oss.oetiker.ch/mrtg/` [228]
[24] Tobias Oetiker, Oetiker+Partner AG: MRTG - Documentation - What is MRTG?, URL: `https://oss.oetiker.ch/mrtg/doc/mrtg.en.html` [229]

Zum Erheben der Daten über den Netzwerkverkehr verwendet *MRTG* das Protokoll *SNMP*[25,26,27,28,29] (*Simple Network Management Protocol*);[23] mit Hilfe des Protokolls *SNMP*[30,31,32] werden die internen Statistiken über die Netzwerkschnittstellen von Routern periodisch abgefragt, die erhobenen Daten werden von *MRTG* in Dateien fortgeschrieben und bei Bedarf aus diesen Dateien ausgelesen, um diese aufbereitet in Form von Diagramm-Grafiken im *PNG*-Format darzustellen.[23,24]

MRTG ist für das Betriebssystem *Linux* und für viele Betriebssysteme der *UNIX*-Betriebssystemfamilie sowie auch für das Betriebssystem Microsoft *Windows* verfügbar.[23,24,33,34] *MRTG* ist in der Programmiersprache *Perl* geschrieben.[23,24]

Die aktuelle Version von *MRTG* ist Version 2.17.4 vom 12.01.2012 (Stand 30.09.2017).[33,35] Der Quellcode der Software ist unter der URL `https://oss.oetiker.ch/mrtg/pub/mrtg-2.17.4.tar.gz` abrufbar.[35]

Weiterführende Informationen zur Funktionalität von *MRTG* und Dokumentation zu *MRTG* sind auf der Internetseite "MRTG - Tobi Oetiker's MRTG - The Multi Router Traffic Grapher" unter der URL `https://oss.oetiker.ch/mrtg/` sowie unter der URL `https://oss.oetiker.ch/mrtg/pub/` abrufbar.[23]

Eine suboptimale Eigenschaft der Software *MRTG* in Hinblick auf Bandbreitenmessungen mit feiner zeitlicher Granulierung ist die kleinste zeitliche Auflösung von fünf Minuten - dies ist das kleinstmögliche definierbare zeitliche Intervall zum periodischen Abfragen der Statistiken über den Netzwerkverkehr durch *MRTG*.[34,36]

Eine hervorhebenswerte Eigenschaft der Software *MRTG* ist die Funktionalität der aufbereiteten Ausgabe der statistischen Daten in Form von Diagramm-Grafiken im *PNG*-Format.
Eine weitere hervorhebenswerte Eigenschaft der Software *MRTG* ist die Möglichkeit der Einbindung externer dynamischer Datenquellen mit der damit verbundenen Möglichkeit,

[25] RFC 1901: Introduction to Community-based SNMPv2, 1996, Historic, URL: `http://www.rfc-editor.org/rfc/rfc1901.txt` [230]
[26] RFC 1908: Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework, 1996, Draft Standard, URL: `http://www.rfc-editor.org/rfc/rfc1908.txt` [231]
[27] RFC 2578: Structure of Management Information Version 2 (SMIv2), 1999, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc2578.txt` [232]
[28] RFC 3416: Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP), 2002, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc3416.txt` [233]
[29] RFC 3418: Management Information Base (MIB) for the Simple Network Management Protocol (SNMP), 2002, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc3418.txt` [234]
[30] RFC 1213: Management Information Base for Network Management of TCP/IP-based internets: MIB-II, 1991, Internet Standard, URL: `http://www.rfc-editor.org/rfc/rfc1213.txt` [235]
[31] RFC 2863: The Interfaces Group MIB, 2000, Draft Standard, URL: `http://www.rfc-editor.org/rfc/rfc2863.txt` [236]
[32] RFC 4293: Management Information Base for the Internet Protocol (IP), 2006, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc4293.txt` [237]
[33] Tobias Oetiker, Oetiker+Partner AG: MRTG - Documentation, URL: `https://oss.oetiker.ch/mrtg/doc/index.en.html` [238]
[34] Tobias Oetiker, Oetiker+Partner AG: MRTG - Documentation - The MRTG 2.17.4 Linux/Unix Installation Guide, URL: `https://oss.oetiker.ch/mrtg/doc/mrtg-unix-guide.en.html` [239]
[35] Tobias Oetiker, Oetiker+Partner AG: MRTG - Download, URL: `https://oss.oetiker.ch/mrtg/download.en.html` [240]
[36] Tobias Oetiker, Oetiker+Partner AG: MRTG - Documentation - MRTG 2.17.4 configuration reference, URL: `https://oss.oetiker.ch/mrtg/doc/mrtg-reference.en.html` [241]

statistische Daten über die Datenreihen dieser externen dynamischen Datenquellen in Form von Diagramm-Grafiken im *PNG*-Format auszugeben - diese Funktionalität der Software *MRTG* wird unter dem Begriff *External Monitoring Scripts* in der Dokumentation beschrieben.[36]

7.2.4 Wget, Nginx

Eine weitere Möglichkeit, aktive Bandbreitenmessungen zwischen zwei Computern durchzuführen, besteht darin, grössere Dateien mit einem *HTTP*-Client von einem *HTTP*-Server (Web-Server) zu transferieren.

Zum Realisieren der Bandbreitenmessungen kommuniziert ein *HTTP*-Client - z.B. der *HTTP*-Client *Wget* - auf einem Computer, mit einem *HTTP*-Server - z.B. dem *HTTP*-Server *Nginx* - auf einem anderen Computer; es wird eine auf dem *HTTP*-Server bereitgestellte grössere Datei transferiert und die Übertragungsrate des Datentransfers bestimmt.

Als Übertragungsprotokoll für die Messungen wird das *HTTP*-Protokoll[37,38,39] genutzt, das *HTTP*-Protokoll baut auf dem *TCP*-Protokoll auf.

Zur Realisieren der Bandbreitenmessungen mit Hilfe eines *HTTP*-Clients und eines *HTTP*-Servers erscheint es sinnvoll, jeweils Implementierungen eines *HTTP*-Clients und eines *HTTP*-Servers zu nutzen, welche einfache Funktionalität mit möglichst geringer Prozessorauslastung zur Verfügung stellen - dies fusst auf dem Motiv, einen möglichst geringen Anteil der Verarbeitung des *HTTP*-Protokolls in die Messungen einfliessen zu lassen.

Die Software *Wget*[40] ist ein *HTTP*-Client, der als nicht-interaktives Kommandozeilenprogramm implementiert ist; *Wget* ist eine frei zugängliche und frei verwendbare Software.[40]
Wget ist in der Programmiersprache *C* geschrieben und für das Betriebssystem *Linux*, für viele Betriebssysteme der *UNIX*-Betriebssystemfamilie, für das Betriebssystem *Solaris* sowie auch für das Betriebssystem Microsoft *Windows* und das Betriebssystem Apple *MacOS/X* verfügbar.[40,41]
Die aktuelle Version von *Wget* ist Version 1.19.1 vom 11.02.2017 (Stand 01.10.2017).[42]
Der Quellcode der Software ist unter der URL `https://ftp.gnu.org/gnu/wget/wget-1.19.1.tar.gz` abrufbar.[42]
Weiterführende Informationen zur Funktionalität von *Wget* und Dokumentation zu *Wget* sind auf der Internetseite "Wget - GNU Project - Free Software Foundation" unter der URL `https://www.gnu.org/software/wget/` abrufbar.[40]

---

[37] RFC 1945: Hypertext Transfer Protocol – HTTP/1.0, 1996, Informational,
URL: `http://www.rfc-editor.org/rfc/rfc1945.txt` [179]
[38] RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing, 2014, Proposed Standard,
URL: `http://www.rfc-editor.org/rfc/rfc7230.txt` [180]
[39] RFC 7540: Hypertext Transfer Protocol Version 2 (HTTP/2), 2015, Proposed Standard,
URL: `http://www.rfc-editor.org/rfc/rfc7540.txt` [181]
[40] Free Software Foundation (FSF): Wget - GNU Project - Free Software Foundation,
URL: `https://www.gnu.org/software/wget/` [242]
[41] Free Software Foundation (FSF): Wget - FAQ - Frequently Asked Questions About GNU Wget,
URL: `https://www.gnu.org/software/wget/faq.html` [243]
[42] Free Software Foundation (FSF): Wget - Downloading GNU Wget - source code,
URL: `https://ftp.gnu.org/gnu/wget/` [244]

Die Software *Nginx*[43] ist ein *HTTP*-Server, der als Hochleistungs-Webserver mit geringem Ressourcenverbrauch implementiert ist;[44] *Nginx* ist eine frei zugängliche und frei verwendbare Software.[43,45]
Nginx ist in der Programmiersprache *C* geschrieben[46] und für das Betriebssystem *Linux*, für das Betriebssystem *FreeBSD* und für andere Betriebssysteme der *UNIX*-Betriebssystemfamilie, für das Betriebssystem *Solaris* sowie auch für das Betriebssystem Microsoft *Windows* verfügbar [43,46]
Die aktuelle stabile - für Produktivumgebungen geeignete - Version von *Nginx* ist Version 1.12.1 vom 11.07.2017, die aktuelle Version der Hauptentwicklungslinie von *Nginx* ist Version 1.13.5 vom 05.09.2017 (Stand 01.10.2017).[47]
Der Quellcode der Software ist unter der URL `https://nginx.org/download/nginx-1.12.1.tar.gz` abrufbar.[47]
Weiterführende Informationen zur Funktionalität von *Nginx* und Dokumentation zu *Nginx* sind unter der URL `https://nginx.org/en/` abrufbar.[43]

Hervorhebenswert bezüglich aktiver Bandbreitenmessungen mit Hilfe des *HTTP*-Clients *Wget* und mit Hilfe des *HTTP*-Servers *Nginx* ist die Verfügbarkeit der Softwares *Wget* und *Nginx* auf vielen Betriebssystemplattformen.
Suboptimal bezüglich aktiver Bandbreitenmessungen mit Hilfe eines *HTTP*-Clients und mit Hilfe eines *HTTP*-Servers ist der zusätzliche *Protocol Overhead* des *HTTP*-Protokolls gegenüber der ausschliesslichen Verwendung des *TCP*-Protokolls für aktive Bandbreitenmessungen.

Das Netzwerk-Monitoring-Werkzeug *MRTG*[48] (*Multi Router Traffic Grapher*) (siehe Abschnitt 7.2.3) erscheint in Verbindung mit dem Kommandozeilen-Netzwerkhilfsprogramm `ping`[49] für den Zweck geeignet, die Indikatoren *Verfügbarkeit* (siehe Abschnitt 7.1.1) und *Latenz* (siehe Abschnitt 7.1.3) zu messen; dies ist zum Einen begründet durch die Verbreitung der Software *MRTG*[50] und dies ist zum Anderen begründet durch die Möglichkeit der Einbindung externer dynamischer Datenquellen in *MRTG*.[51]

Die Mess-Software *iperf3*[52] (siehe Abschnitt 7.2.1) erscheint für den Zweck geeignet, den quantitativen Indikator *Datendurchsatz* (*throughput*) (siehe Abschnitt 7.1.4) zu messen; dies ist zum Einen begründet durch die Verbreitung der Software *iperf3*[53,54] und dies ist

[43] Igor Sysoev: nginx [engine x] - HTTP and reverse proxy server, URL: `https://nginx.org/en/` [245]
[44] Dipankar Sarkar: Nginx 1 Web Server Implementation Cookbook [246, S. 1]
[45] Igor Sysoev: nginx - Documentation - FAQ, URL: `https://nginx.org/en/docs/faq.html` [247]
[46] Igor Sysoev: nginx - Documentation, URL: `https://nginx.org/en/docs/` [248]
[47] Igor Sysoev: nginx - Download, URL: `https://nginx.org/en/download.html` [249]
[48] Tobias Oetiker, Oetiker+Partner AG: MRTG - Multi Router Traffic Grapher, URL: `https://oss.oetiker.ch/mrtg/` [228]
[49] Prof. Peter Fischer und Peter Hofer: Lexikon der Informatik, Eintrag "ping" [8, S. 629]
[50] Ein Hinweis auf die Verbreitung der Software *MRTG* ist die Verfügbarkeit von *MRTG* als vorbereitetes bereitgestelltes Packet für das Betriebssystem *Debian Linux*; die bereitgestellte Version der Software *MRTG* ist beim Betriebssystem *Debian Linux* durch das Kommando `apt-cache showpkg mrtg | head -3` abrufbar.
[51] Tobias Oetiker, Oetiker+Partner AG: MRTG - Documentation - MRTG 2.17.4 configuration reference, URL: `https://oss.oetiker.ch/mrtg/doc/mrtg-reference.en.html` [241]
[52] Energy Sciences Network (ESnet) und Lawrence Berkeley National Laboratory: iperf3 Project, URL: `http://software.es.net/iperf/` [218]
[53] Ein Hinweis auf die Verbreitung der Software *iperf3* ist die Existenz öffentlich zugänglicher *iperf3*-Server, um Bandbreitenmessungen zu bewerkstelligen.[54]
[54] Vivien Gueant: Iperf.fr - Public iPerf3 servers, URL: `https://iperf.fr/iperf-servers.php` [224]

zum Anderen begründet durch die Möglichkeit der Ausgabe der statistischen Daten der Bandbreitenmessungen durch *iperf3* in maschinenlesbarem Format.[52]

7.3 Erfassung der Messdaten

Die kontinuierliche Erfassung der Messdaten für die Indikatoren *Verfügbarkeit* und *Latenz* mit Hilfe des Netzwerk-Monitoring-Werkzeugs *MRTG* (*Multi Router Traffic Grapher*) in Verbindung mit dem Kommandozeilen-Netzwerkhilfsprogramm `ping` wird zweifach zeitlich parallel durchgeführt: zum Einen zwischen der Arbeitsstation `lan1-11-10`, welche über Router `r1` angebunden ist, und der Arbeitsstation `lan2-12-10`, welche über Router `r2` angebunden ist; zum Anderen zwischen der LAN-seitigen Netzwerkschnittstelle `br1` von Router `r1` und der LAN-seitigen Netzwerkschnittstelle `br1` von Router `r2` (siehe Abbildung 6.8 Überblick über die Systemarchitektur der Routerumgebung, Seite 90).
Die zweifach zeitlich parallele kontinuierliche Messung an unterschiedlichen Messpunkten fusst auf dem Motiv, die erhobenen Messdaten auf Validität prüfen zu können.

Die kontinuierliche Erfassung der Messdaten für die Indikatoren *Verfügbarkeit* und *Latenz* mit Hilfe des Netzwerk-Monitoring-Werkzeugs *MRTG* (*Multi Router Traffic Grapher*) wird durch die Einbindung des Kommandozeilen-Netzwerkhilfsprogramms `ping` realisiert, das Setup für die Erfassung der Messdaten zwischen der Arbeitsstation `lan1-11-10` und der Arbeitsstation `lan2-12-10` ist auszugsweise in Anhang B.1.1 dargestellt.
Ein Auszug aus den Datenreihen der Messdaten für die Indikatoren *Verfügbarkeit* und *Latenz* ist in Anhang B.1.2 dargestellt; gegenübergestellt sind Datenreihen aus kontinuierlichen Messungen zwischen der Arbeitsstation `lan1-11-10` und der Arbeitsstation `lan2-12-10` und Datenreihen aus kontinuierlichen Messungen zwischen der LAN-seitigen Netzwerkschnittstelle von Router `r1` und der LAN-seitigen Netzwerkschnittstelle von Router `r2`.

Die Erfassung der Messdaten für den Indikator *Datendurchsatz* mit Hilfe der Mess-Software *iperf3* wird zwischen der LAN-seitigen Netzwerkschnittstelle `br1` von Router `r1` und der LAN-seitigen Netzwerkschnittstelle `br1` von Router `r2` durchgeführt.
Die Erfassung der Messdaten für den Indikator *Datendurchsatz* wird in zwei Modi durchgeführt: zum Einen bei Verfügbarkeit der nativen VPN-Verbindungen (*native VPN-Tunnel*), zum Anderen bei Nichtverfügbarkeit der nativen VPN-Verbindungen und der damit verbundenen Nutzung der alternativen Transportpfade über Overlay-VPN-Verbindungen (*Overlay-VPN-Tunnel*).

Die Erfassung der Messdaten für den Indikator *Datendurchsatz* mit Hilfe der Mess-Software *iperf3* vollzieht sich mittels einer Server-Instanz der Software *iperf3* auf Router `r2` und einer Client-Instanz der Software *iperf3* auf Router `r1`. Die Client-Instanz der Software *iperf3* auf Router `r1` sendet über die Zeitdauer von 120 Sekunden einen TCP-Protokoll-basierenden Datenstrom zur Server-Instanz der Software *iperf3* auf Router `r2`, um die maximal zur Verfügung stehende Bandbreite zwischen den beiden Kommunikationsendpunkten zu ermitteln - über die Zeitdauer von 120 Sekunden wird sekündlich die zur Verfügung stehende Bandbreite ausgegeben.
Das Setup für die Erfassung der Messdaten zwischen dem Router `r1` und Router `r2` ist auszugsweise in Anhang B.1.3 dargestellt; ein Auszug aus den Datenreihen der Messdaten für den Indikator *Datendurchsatz* ist in Anhang B.1.4 dargestellt.

7.4 Auswertung der Messdaten

Die Auswertung der Messdaten für die Indikatoren *Verfügbarkeit* und *Latenz* aus der kontinuierlichen Erfassung mit Hilfe des Netzwerk-Monitoring-Werkzeugs *MRTG* (*Multi Router Traffic Grapher*) wird durch in die Software *MRTG* integrierte Funktionalität realisiert (siehe Abschnitt 7.2.3 MRTG).

Die Auswertung der Messdaten für den Indikator *Datendurchsatz* aus der Erfassung mit Hilfe der Mess-Software *iperf3* wird mit der Software *gnuplot*[55] bewerkstelligt.
Die Software *gnuplot* ist eine frei zugängliche und frei verwendbare Software, die es ermöglicht, Messdaten graphisch darzustellen; *gnuplot* ist für viele Betriebssystemplattformen verfügbar.[55]
Das Setup und das Transkript für die Auswertung der Messdaten für den Indikator *Datendurchsatz* mit Hilfe der Software *gnuplot* ist in Anhang B.2.1 auszugsweise dargestellt.

7.5 Ergebnisse der Auswertung

Das Ergebnis der Auswertung der Messdaten für die Indikatoren *Verfügbarkeit* und *Latenz* stellt sich wie folgt dar:

Abbildung 7.1 stellt als Diagramm die Umlaufzeiten von IP-Datagrammen zwischen LAN `lan1` und LAN `lan2` über den Tagesverlauf des 11.10.2017 dar - das Diagramm visualisiert die *Round Trip Time* (*RTT*) von IP-Datagrammen vom Typ *ICMP Echo Request*, gemessen in Millisekunden; im Diagramm farblich grün visualisiert ist die maximale *Round Trip Time*, im Diagramm farblich blau visualisiert ist die minimale *Round Trip Time*. Die maximale *Round Trip Time* über den Tagesverlauf des 11.10.2017 ist 31 Millisekunden; der arithmetische Mittelwert der maximalen *Round Trip Time* ist 22 Millisekunden, der arithmetische Mittelwert der minimalen *Round Trip Time* ist 21 Millisekunden über den Tagesverlauf des 11.10.2017; das Zeitfenster für die Klasseneinteilung in Maxima und Minima ist 5 Minuten.

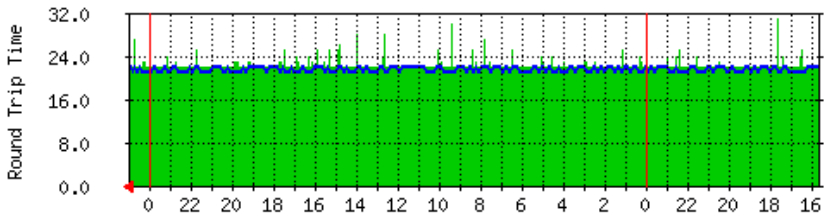

Abbildung 7.1: Diagramm über Paket-Umlaufzeiten zwischen LAN `lan1` und LAN `lan2` am 11.10.2017, gemessen in Millisekunden

Korrespondierend mit Abbildung 7.1 stellt Abbildung 7.2 als Diagramm die Umlaufzeiten von IP-Datagrammen zwischen Router `r1` und Router `r2` über den Tagesverlauf des 11.10.2017 dar.

Abbildung 7.3 stellt als Diagramm die Umlaufzeiten von IP-Datagrammen zwischen LAN `lan1` und LAN `lan2` über den Zeitraum von 04.10.2017 bis 11.10.2017 dar - das Diagramm visualisiert die *Round Trip Time* (*RTT*) von IP-Datagrammen vom Typ *ICMP*

[55] Gnuplot Project: gnuplot - a portable command-line driven graphing utility,
URL: `http://www.gnuplot.info/` [250]

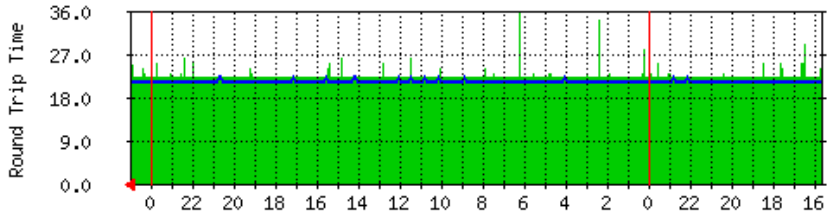

Abbildung 7.2: Diagramm über Paket-Umlaufzeiten zwischen Router `r1` und Router `r2` am 11.10.2017, gemessen in Millisekunden

Echo Request, gemessen in Millisekunden; im Diagramm farblich grün visualisiert ist die maximale *Round Trip Time*, im Diagramm farblich blau visualisiert ist die minimale *Round Trip Time*.
Die maximale *Round Trip Time* über den Zeitraum von 04.10.2017 bis 11.10.2017 ist 999 Millisekunden; der arithmetische Mittelwert der maximalen *Round Trip Time* ist 131 Millisekunden, der arithmetische Mittelwert der minimalen *Round Trip Time* ist 70 Millisekunden über den Zeitraum von 04.10.2017 bis 11.10.2017; das Zeitfenster für die Klasseneinteilung in Maxima und Minima ist 5 Minuten - im Diagramm farblich dunkelgrün visualisiert sind die Maxima der Maxima über 5 Minuten, im Diagramm magentafarben visualisiert sind die Maxima der Minima über 5 Minuten.

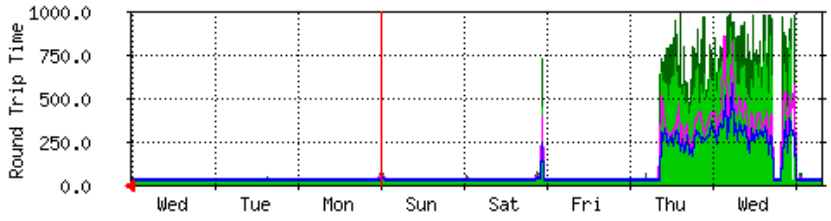

Abbildung 7.3: Diagramm über Paket-Umlaufzeiten zwischen LAN `lan1` und LAN `lan2` von 04.10.2017 bis 11.10.2017, gemessen in Millisekunden

Das Diagramm zeigt Zeiträume mit erhöhten *Round Trip Time*-Werten; in diese Zeiträume fallen aktive Messungen mit der Mess-Software *iperf3* zum Zweck der Messung des Datendurchsatzes über Overlay-Netzwerk-VPN-Verbindungen. Zum Einen verändern aktive Messungen des Datendurchsatzes die *Round Trip Time*; dies fusst darauf, dass die Sende- und Empfangswarteschlangen der TCP-Stacks der beteiligten Computer bei aktiven Messungen des Datendurchsatzes einen hohen Füllgrad aufweisen. Zum Anderen ist die erhöhte *Round Trip Time* durch die höhere Latenz beim Transport über Overlay-Netzwerk-VPN-Verbindungen begründet.

Das Ergebnis der Auswertung für den Indikator *Rekonvergenz* stellt sich wie folgt dar:

Für den Indikator *Rekonvergenz* zeigt sich ein Wert von unter drei Sekunden (siehe Abschnitt 6.5.4, Seite 88, Seite 89), d.h. nach dem kurzzeitigen Wegfall der nativen VPN-Verbindungen realisieren sich in weniger als drei Sekunden nutzbare alternative Transportpfade für IP-Datagramme über Overlay-VPN-Verbindungen.

Das Ergebnis der Auswertung der Messdaten für den Indikator *Datendurchsatz* stellt sich wie folgt dar:

Abbildung 7.4 stellt als Diagramm die mit der Software *iperf3* gemessene verfügbare Übertragungsbandbreite zwischen LAN `lan1` und LAN `lan2` bei Verfügbarkeit der nativen VPN-Tunnel dar - das Diagramm visualisiert die Übertragungsbandbreite in MBit/s, das Zeitintervall der Messung ist 120 Sekunden.

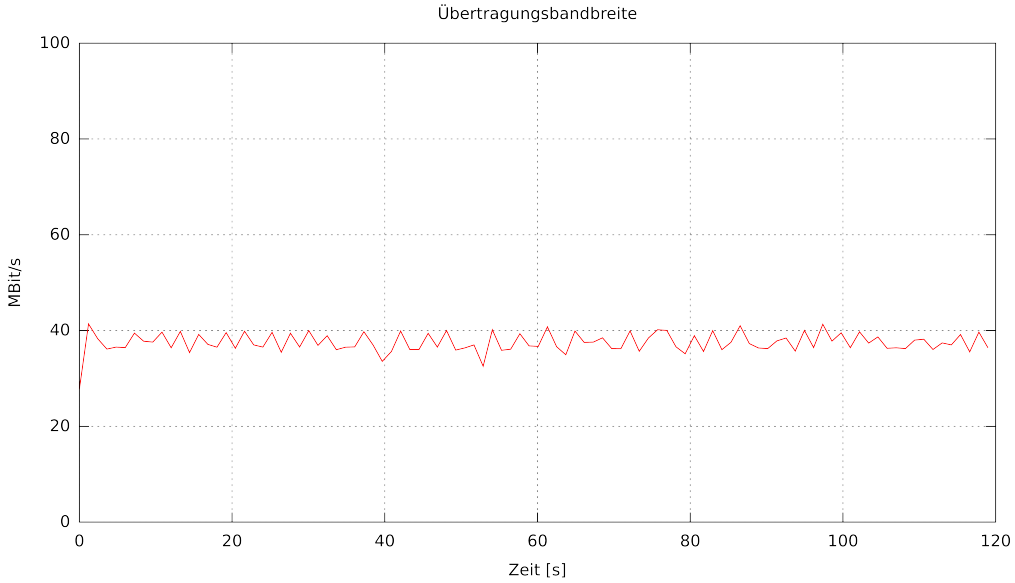

Abbildung 7.4: Diagramm über die gemessene verfügbare Übertragungsbandbreite zwischen LAN `lan1` und LAN `lan2` bei Verfügbarkeit der nativen VPN-Tunnel

Korrespondierend mit Abbildung 7.4 stellt Abbildung 7.5 als Diagramm die Anzahl der während der Messung der verfügbaren Übertragungsbandbreite erneut übertragenen IP-Pakete bei Verfügbarkeit der nativen VPN-Tunnel dar.

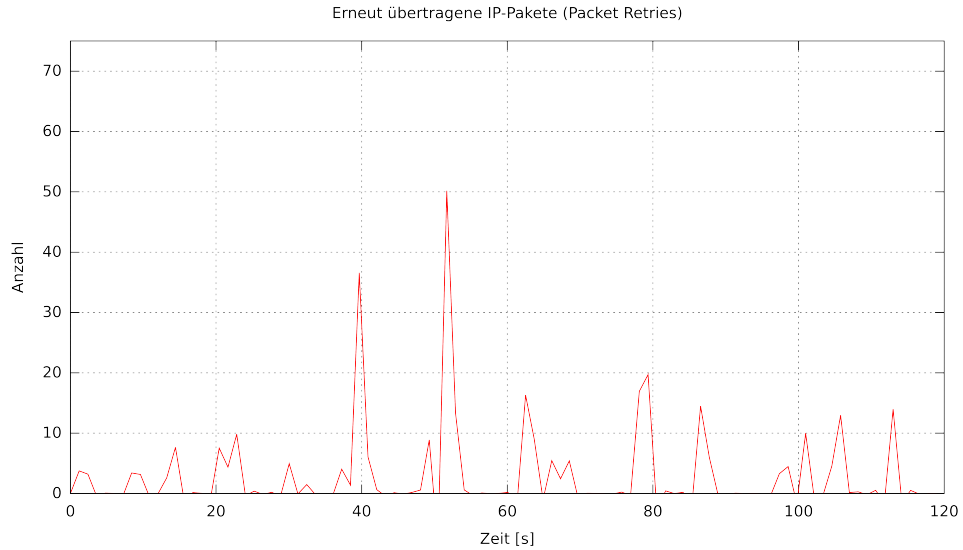

Abbildung 7.5: Diagramm über erneut übertragene IP-Pakete während der Messung bei Verfügbarkeit der nativen VPN-Tunnel

Abbildung 7.6 stellt als Diagramm die mit der Software *iperf3* gemessene verfügbare Übertragungsbandbreite zwischen LAN `lan1` und LAN `lan2` bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel dar - das Diagramm visualisiert die Übertragungsbandbreite in MBit/s, das Zeitintervall der Messung ist 120 Sekunden.

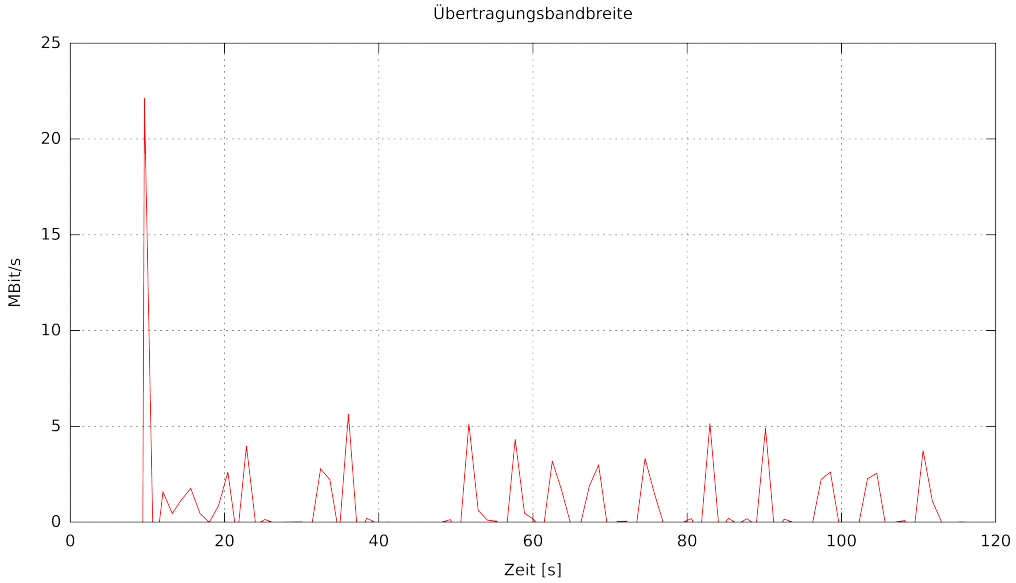

Abbildung 7.6: Diagramm über die gemessene verfügbare Übertragungsbandbreite zwischen LAN `lan1` und LAN `lan2` bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel

Korrespondierend mit Abbildung 7.6 stellt Abbildung 7.7 als Diagramm die Anzahl der während der Messung der verfügbaren Übertragungsbandbreite erneut übertragenen IP-Pakete bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel dar.

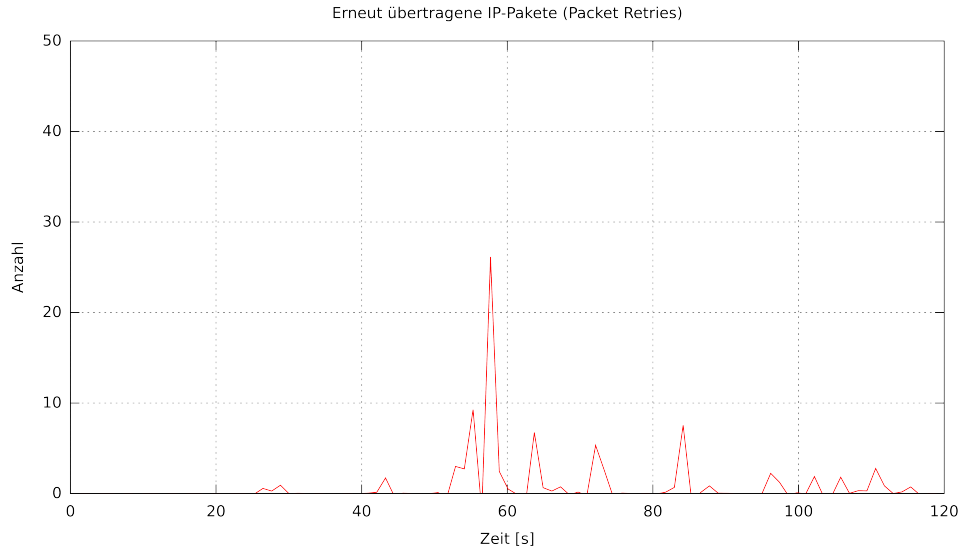

Abbildung 7.7: Diagramm über erneut übertragene IP-Pakete während der Messung bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel

Abbildung 7.8 stellt die Übertragungsbandbreite bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel und einer Bandbreiten-Zielvorgabe von 4 MBit/s dar.

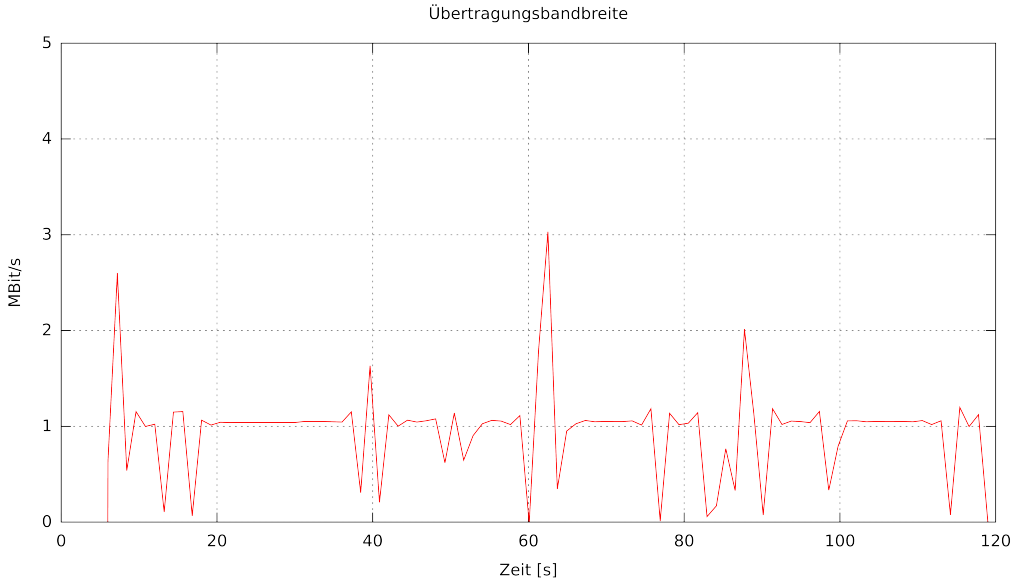

Abbildung 7.8: Diagramm über die Übertragungsbandbreite zwischen den LANs bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel und einer Bandbreiten-Zielvorgabe von 4 MBit/s

Korrespondierend mit Abbildung 7.8 stellt Abbildung 7.9 die Anzahl der erneut übertragenen IP-Pakete bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel und einer Bandbreiten-Zielvorgabe von 4 MBit/s dar.

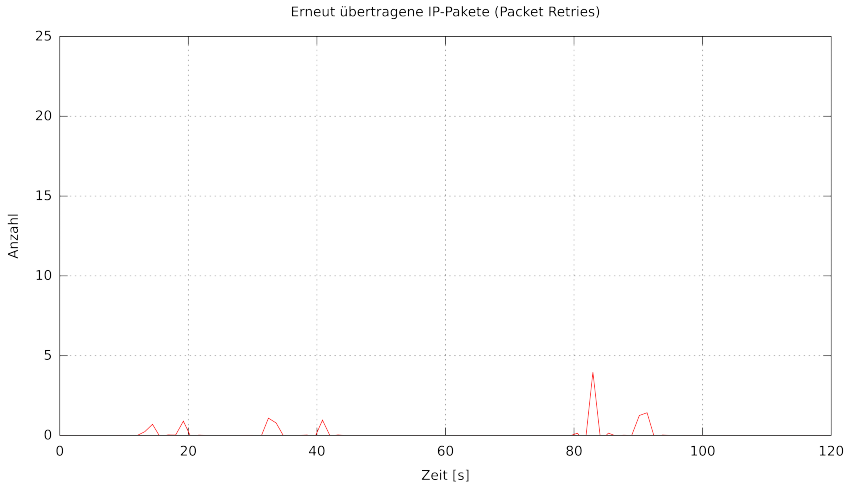

Abbildung 7.9: Diagramm über erneut übertragene IP-Pakete bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel und einer Bandbreiten-Zielvorgabe von 4 MBit/s

In Abbildung 7.6 zeigt sich deutlich ein periodisches Einbrechen des Niveaus des Datendurchsatzes; in Abbildung 7.8 zeigt sich das periodische Einbrechen des Niveaus des Datendurchsatzes ebenso, jedoch in abgeschwächter, akzeptabler Form.

Der Grund für diese Beobachtung ist der Umstand, dass bei der Tunnelung von TCP-

Verbindungen in TCP-Verbindungen - z.B. bei VPN-Verbindungen, die auf dem TCP-Protokoll basieren - unter ungünstigen Umständen Interferenzen zwischen den inneren und den äusseren TCP-Verbindungen auftreten; dies wird als *TCP meltdown problem* bezeichnet[56].

In Abbildung 7.4 (siehe Seite 103) zeigt sich bei der Messung der verfügbaren Übertragungsbandbreite bei Verfügbarkeit der nativen VPN-Tunnel zu Beginn der Messung ein kurzes Zeitfenster, in dem das Niveau der Übertragungsbandbreite niedrig ist - der Grund für diese Beobachtung ist der Umstand, dass der TCP-Stack des sendenden Computers bei einer neu aufgebauten TCP-Verbindung initial langsam mit dem Senden von Daten beginnt; dies wird als *slow start*[57,58,59] bezeichnet. In Abbildung 7.6 (siehe Seite 104) und in Abbildung 7.8 (siehe Seite 105) - bei der Messung der verfügbaren Übertragungsbandbreite bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel - zeigt sich dieses initale Zeitfenster mit niedrigem Niveau der Übertragungsbandbreite ebenfalls, jedoch ausgeprägter in der Länge, da sowohl die TCP-Verbindungen des zu Grunde liegenden Overlay-Netzes als auch die TCP-Verbindungen innerhalb der VPN-Verbindung über das Overlay-Netz als Transportweg initial den Durchsatz langsam beim Senden steigern.

In Abbildung 7.5 (siehe Seite 103), Abbildung 7.7 (siehe Seite 104) und in Abbildung 7.9 (siehe Seite 105) zeigen sich bei den erneut übertragenen IP-Paketen (*packet retries*) gute - in Bezug gesetzt zu den auf den Übertragungswegen aufbauenden IP-Verbindungen gute und akzeptable - Werte.

[56] Osamu Honda, Hiroyuki Ohsaki, Makoto Imase, Mika Ishizuka und Junichi Murayama: Understanding TCP over TCP: effects of TCP tunneling on end-to-end throughput and latency [251, S. 1]

[57] RFC 5681: TCP Congestion Control, 2009, Draft Standard, URL: `http://www.rfc-editor.org/rfc/rfc5681.txt` [252, S. 4]

[58] RFC 3390: Increasing TCP's Initial Window, 2002, Proposed Standard, URL: `http://www.rfc-editor.org/rfc/rfc3390.txt` [253, S. 2]

[59] RFC 6077: Open Research Issues in Internet Congestion Control, 2011, Informational, URL: `http://www.rfc-editor.org/rfc/rfc6077.txt` [254, S. 24,36]

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

In Kapitel 5 und Kapitel 6 wurde die Konzeption und die Implementierung einer Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks beschrieben; es wurden die Hintergründe der Konzeption erläutert und es wurde die Machbarkeit der Konzeption in Form einer praktisch nutzbaren Routerumgebung gezeigt.

Die in Kapitel 2 Grundlagen vorgreifend auf die Analyse und die Konzeption vorgestellten Softwares führten in ihrem *derzeitigen* Entwicklungsstand und ihrer *derzeitigen* Funktionalität zur gewählten Kombination der Softwarekomponenten als Grundlage für die vorliegende Routerumgebung.

Zentrale Fragestellungen in der Konzeption waren die Auswahl der Software-Komponenten, die gemeinsamen nutzbaren Schnittstellen der Software-Komponenten und die Möglichkeit der Integration der zur Auswahl stehenden Komponenten mit einer hierzu entwickelten steuernden Laufzeitumgebung.
Die wichtigsten Entscheidungen in der Konzeption bestanden aus der Wahl des *SOCKS*-Protokolls als gemeinsame nutzbare Schnittstelle zwischen der Overlay-Netzwerk-Software-Komponente *Tor* und der VPN-Software-Komponente *OpenVPN* und aus der Wahl des dynamischen Routingprotokolls OSPF mit der damit verbundenen Auswahl der *Routing Software Suite Quagga* zur Realisation redundanter Transportpfade für den IP-Paketransport.
Eine weitere wichtige Entscheidung bestand in der Wahl von Netzwerkbrücken (*network bridges*) als Bindeglied zwischen der VPN-Software und der Software für das dynamische Routing; die Wahl von Netzwerkbrücken verringerte die Notwendigkeit von Systemneustarts (*reboots*) bei Änderungen signifikant, ein Vorteil von Netzwerkbrücken gegenüber der direkten Nutzung von Netzwerkschnittstellen zeigte sich durch die Möglichkeit der einfachen Erweiterbarkeit der Netzwerkinfrastruktur im laufenden Betrieb ohne Einschränkung der Verfügbarkeit.

Die entwickelte steuernde Laufzeitumgebung wurde als modulares Framework konzipiert; zur Implementierung dieser Laufzeitumgebung wurde die Skriptsprache der klassischen *Bourne shell* (*AT&T Shell*, *sh*) gewählt und verwendet. Während der Entwicklung zeigte sich die Wahl einer Skriptsprache zur Implementierung der steuernden Laufzeitumgebung als vorteilhaft bezüglich eines raschen Entwicklungsfortschritts.

Während der Entwicklung zeigte sich, dass ein Managementsystem für das *Deployment* der Laufzeitumgebung, der kryptographischen Schlüssel und Zertifikate auf die Router wünschenswert ist; das *Bundling* auf den Entwicklungssystemen und das *Deployment* auf die Produktivsysteme vollzog sich händisch.

Während der Tests auf den Produktivsystemen und während der Evaluation an Hand

der Indikatoren zeigte sich eine sehr hohe Gesamtverfügbarkeit der VPNs mit dem Overlay-Netzwerk *Tor* als Transportweg - als Transportpfade über das Overlay-Netzwerk standen insgesamt sechs Transportpfade mit Hilfe von zwei Overlay-Netzwerkknoten-Paaren zur Verfügung.

Abschliessend lässt sich feststellen, dass sich das Design und die Implementierung der Routerumgebung zur redundanten und hochverfügbaren LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und Virtual Private Networks bewährt hat. Für die Indikatoren *Verfügbarkeit* und *Rekonvergenz* zeigten sich sehr gute Werte, für den Indikator *Datendurchsatz* zeigten sich bei Verfügbarkeit und Nutzung der primären nativen VPN-Tunnel sehr gute Werte und bei der Nutzung der alternativen VPN-Tunnel mit Overlay-Netzwerk als Transportweg gute - in Hinblick auf *Use Cases* akzeptable - Werte.

8.2 Ausblick

Die Implementierung der Routerumgebung hat gezeigt, dass sich die Arbeitsparameter der auf dem TCP-Protokoll basierenden VPNs mit dem Overlay-Netzwerk *Tor* als Transportschicht noch weiter optimieren liessen[1], spezifisch in Hinblick auf den Indikator *Datendurchsatz*.

Eine erwägenswerte Möglichkeit, den Datendurchsatz der VPNs zu optimieren, bestünde in der Realisierung der in Software durch die Softwarebibliothek OpenSSL[2] bereitgestellten kryptographischen Funktionen mit Hardwareunterstützung[3].

Eine sinnvolle Erweiterung der Routerumgebung wäre die signifikante Erhöhung der Anzahl der für die Virtual Private Networks genutzten Overlay-Netzwerkknoten durch Auslagerung der Funktionaliät der Overlay-Netzwerkknoten auf hierfür ausschliesslich bereitgestellte separate Maschinen; auch die Anbindung dieser Maschinen mit Internet-Connectivity über unterschiedliche Internet Service Provider erschiene erwägenswert.

Ein Aspekt, der einen Anreiz zur Erweiterung der Routerumgebung darstellen könnte, besteht in der präventiven periodischen Überwachung der genutzten Overlay-Netzwerkknoten des *Tor*-Netzwerks bezüglich der Parameter Zuverlässigkeit und Datendurchsatz.

Eine weitere erwägbare Erweiterung bestünde aus der Realisierung von alternativen Transportpfaden für die VPNs mit Hilfe von DNS-Tunneling, d.h. durch die Realisierung des Pakettransports durch DNS-Anfragen bzw. DNS-Antworten[4].

Es werden derzeit Routing-Protokolle entwickelt, welche ein verbessertes Routing bei der Nutzung von VPNs mit Overlay-Netzwerken als Transportschicht ermöglichen, in-

[1] Berry Hoekstra und Damir Musulin: Comparing TCP Performance of tunneled and non-tunneled Traffic using OpenVPN, 2011, URL: `http://www.delaat.net/rp/2010-2011/p09/report.pdf` [255, S. 53]
[2] OpenSSL Software Foundation: OpenSSL Cryptography and SSL/TLS Toolkit, URL: `https://www.openssl.org/` [104]
[3] OpenSSL Software Foundation: OpenSSL - Documentation - ENGINE cryptographic module support, URL: `https://www.openssl.org/docs/man1.0.2/crypto/engine.html` [256]
[4] Hervé Schauer Consultants (HSC): Dns2tcp - a network tool designed to relay TCP connections through DNS traffic, 2012, URL: `http://www.hsc.fr/ressources/outils/dns2tcp/` [257]

dem beim Routing als Metrik die Latenz (*delay*) der Transportpfade berücksichtigt wird.[5]

Eingedenk der Ergebnisse dieser Arbeit, der weiteren Perspektiven von Virtual Private Networks über Overlay-Netzwerke als Transport-Layer und einiger weiterer damit verbundener interessanter Fragen erscheint es sicher anzunehmen, dass dieses Themengebiet auch in Zukunft in den Bereichen Forschung und Praxis bearbeitet werden wird.

[5] Baptiste Jonglez, Matthieu Boutier und Juliusz Chroboczek: A delay-based routing metric, 2014, URL: `https://arxiv.org/pdf/1403.3488.pdf` [258]

Literatur

[1] Richard P. Feynman. *Feynman lectures on computation: Edited by Anthony J.G. Hey, Robin W. Allen*. Reading, Massachusetts, USA: Addison-Wesley, 1996. ISBN: 0-201-48991-0.

[2] Prof. Dr. Norbert Pohlmann und Stefan Dierichs. *So funktioniert Internet-Routing: Wie Routing dem Netz seine Selbstheilungskräfte verleiht: Rubrik -Praxis-*. Hrsg. von Heise Medien GmbH & Co. KG. Hannover, 2008. URL: https://www.heise.de/ct/artikel/So-funktioniert-Internet-Routing-221495.html.

[3] FreeMind Project. *FreeMind - free mind-mapping software*. 2016. URL: http://freemind.sourceforge.net/.

[4] GanttProject Team. *GanttProject*. 2017. URL: http://www.ganttproject.biz/.

[5] IANA. *Autonomous System (AS) Numbers*. Hrsg. von IANA. Los Angeles, CA, USA, 2016. URL: https://www.iana.org/assignments/as-numbers/as-numbers.xhtml.

[6] CAIDA. *AS Rank: AS Ranking*. Center for Applied Internet Data Analysis. San Diego, CA, USA, 2016. URL: http://as-rank.caida.org/.

[7] Measurement Lab Consortium. *ISP Interconnection and its Impact on Consumer Internet Performance*. 2014. URL: https://www.measurementlab.net/publications/M-Lab_Interconnection_Study_US.pdf.

[8] Prof. Peter Fischer und Peter Hofer, Hrsg. *Lexikon der Informatik*. 14. Aufl. Berlin: Springer, 2008. ISBN: 978-3-540-72549-7.

[9] Andrew S. Tanenbaum und David J. Wetherall. *Computer networks*. 5. Aufl. Boston, MA, USA: Pearson Prentice Hall, 2011. ISBN: 978-0-13-212695-3.

[10] Y. Rekhter, T. Li und S. Hares. *RFC 4271: A Border Gateway Protocol 4 (BGP-4)*. Internet Requests for Comments. 2006. URL: http://www.rfc-editor.org/rfc/rfc4271.txt.

[11] BGP Reports. *BGP Routing Table Analysis Reports: Growth of the BGP Table - 1994 to Present*. 2017. URL: http://bgp.potaroo.net/.

[12] T. Bates, R. Chandra, D. Katz und Y. Rekhter. *RFC 4760: Multiprotocol Extensions for BGP-4*. Internet Requests for Comments. 2007. URL: http://www.rfc-editor.org/rfc/rfc4760.txt.

[13] S. Murphy. *RFC 4272: BGP Security Vulnerabilities Analysis*. Internet Requests for Comments. 2006. URL: http://www.rfc-editor.org/rfc/rfc4272.txt.

[14] K. Lougheed und Y. Rekhter. *RFC 1105: A Border Gateway Protocol (BGP)*. Internet Requests for Comments. 1989. URL: http://www.rfc-editor.org/rfc/rfc1105.txt.

[15] K. Lougheed und Y. Rekhter. *RFC 1163: A Border Gateway Protocol (BGP)*. Internet Requests for Comments. 1990. URL: http://www.rfc-editor.org/rfc/rfc1163.txt.

[16] J. Honig, D. Katz, M. Mathis, Y. Rekhter und J. Yu. *RFC 1164: Application of the Border Gateway Protocol in the Internet*. Internet Requests for Comments. 1990. URL: http://www.rfc-editor.org/rfc/rfc1164.txt.

[17] K. Lougheed und Y. Rekhter. *RFC 1267: A Border Gateway Protocol 3 (BGP-3)*. Internet Requests for Comments. 1991. URL: http://www.rfc-editor.org/rfc/rfc1267.txt.

[18] Y. Rekhter und P. Gross. *RFC 1268: Application of the Border Gateway Protocol in the Internet*. Internet Requests for Comments. 1991. URL: http://www.rfc-editor.org/rfc/rfc1268.txt.

[19] Y. Rekhter und T. Li. *RFC 1654: A Border Gateway Protocol 4 (BGP-4)*. Internet Requests for Comments. 1994. URL: http://www.rfc-editor.org/rfc/rfc1654.txt.

[20] Y. Rekhter und P. Gross. *RFC 1655: Application of the Border Gateway Protocol in the Internet*. Internet Requests for Comments. 1994. URL: http://www.rfc-editor.org/rfc/rfc1655.txt.

[21] A. Heffernan. *RFC 2385: Protection of BGP Sessions via the TCP MD5 Signature Option*. Internet Requests for Comments. 1998. URL: http://www.rfc-editor.org/rfc/rfc2385.txt.

[22] J. Touch, A. Mankin und R. Bonica. *RFC 5925: The TCP Authentication Option*. Internet Requests for Comments. 2010. URL: http://www.rfc-editor.org/rfc/rfc5925.txt.

[23] G. Lebovitz und E. Rescorla. *RFC 5926: Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)*. Internet Requests for Comments. 2010. URL: http://www.rfc-editor.org/rfc/rfc5926.txt.

[24] Niels Ferguson, Bruce Schneier und Tadayoshi Kohno. *Cryptography engineering: Design principles and practical applications*. Indianapolis, IN, USA: Wiley Publishing, 2010. ISBN: 978-0-470-47424-2.

[25] John Moy. *RFC 2328: OSPF Version 2*. Internet Requests for Comments. 1998. URL: http://www.rfc-editor.org/rfc/rfc2328.txt.

[26] R. Coltun, D. Ferguson, J. Moy und A. Lindem. *RFC 5340: OSPF for IPv6*. Internet Requests for Comments. 2008. URL: http://www.rfc-editor.org/rfc/rfc5340.txt.

[27] M. Bhatia, V. Manral, M. Fanto, R. White, M. Barnes, T. Li und R. Atkinson. *RFC 5709: OSPFv2 HMAC-SHA Cryptographic Authentication*. Internet Requests for Comments. 2009. URL: http://www.rfc-editor.org/rfc/rfc5709.txt.

[28] J. Moy. *RFC 1247: OSPF Version 2*. Internet Requests for Comments. 1991. URL: http://www.rfc-editor.org/rfc/rfc1247.txt.

[29] C. Hedrick. *RFC 1058: Routing Information Protocol*. Internet Requests for Comments. 1988. URL: http://www.rfc-editor.org/rfc/rfc1058.txt.

[30] G. Malkin. *RFC 2453: RIP Version 2*. Internet Requests for Comments. 1998. URL: http://www.rfc-editor.org/rfc/rfc2453.txt.

[31] G. Malkin und R. Minnear. *RFC 2080: RIPng for IPv6*. Internet Requests for Comments. 1997. URL: http://www.rfc-editor.org/rfc/rfc2080.txt.

[32] D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch und R. White. *RFC 7868: Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)*. Internet Requests for Comments. 2016. URL: http://www.rfc-editor.org/rfc/rfc7868.txt.

[33] Bijal Sanghani. *Euro-IX IXP Report 2011*. Euro-IX (European Internet Exchange Association). London, UK, 2012. URL: https://www.euro-ix.net/m/cms_page_media/82/Euro-IX%20IXP%20Report%202011.pdf.

[34] *OpenBGPD*. URL: http://openbgpd.org/.

[35] Free Software Foundation (FSF). *Zebra - GNU Project - Free Software Foundation*. Boston, MA, USA. URL: https://www.gnu.org/software/zebra/.

[36] *XORP: eXtensible Open Router Platform*. URL: http://www.xorp.org/.

[37] Cisco Systems. *Cross Platform Release Notes for Cisco IOS Release 15.6M&T - Cisco IOS Software Release 15.6M&T*. URL: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/15-6m-and-t/release/notes/15-6m-and-t-book/156TREQS.html.

[38] Cisco Systems. *Cisco IOS IP Routing: OSPF Command Reference*. URL: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_ospf/command/iro-cr-book/ospf-a1.html.

[39] D. Farinacci, V. Fuller, D. Meyer und D. Lewis. *RFC 6830: The Locator/ID Separation Protocol (LISP)*. Internet Requests for Comments. 2013. URL: http://www.rfc-editor.org/rfc/rfc6830.txt.

[40] Cisco Systems. *Cisco IOS Software Release 15.6(3)M Documentation*. URL: http://www.cisco.com/c/en/us/support/ios-nx-os-software/ios-software-release-15-6-3-m/model.html.

[41] Cisco Systems. *Cisco IOS XR Routing Command Reference for the Cisco CRS Router, Release 5.1.x - OSPF Commands*. URL: http://www.cisco.com/c/en/us/td/docs/routers/crs/software/crs_r5-1/routing/command/reference/b_routing_cr51xcrs/b_routing_cr51xcrs_chapter_0100.html.

[42] Cisco Systems. *Cisco IOS XR Routing Command Reference for the Cisco CRS Router, Release 5.1.x - OSPFv3 Commands*. URL: http://www.cisco.com/c/en/us/td/docs/routers/crs/software/crs_r5-1/routing/command/reference/b_routing_cr51xcrs/b_routing_cr51xcrs_chapter_0101.html.

[43] Cisco Systems. *Cisco IOS XR Routing Command Reference for the Cisco CRS Router*. URL: http://www.cisco.com/c/en/us/td/docs/routers/crs/software/crs_r5-1/routing/command/reference/b_routing_cr51xcrs.html.

[44] Cisco Systems. *Cisco IOS XE*. URL: http://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-xe/index.html.

[45] Cisco Systems. *Cisco Integrated Services Virtual Router Data Sheet*. URL: http://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/enterprise-network-functions-virtualization-nfv/datasheet-c78-736768.html.

[46] Cisco Systems. *IP Routing: OSPF Configuration Guide*. URL: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_ospf/configuration/xe-16/iro-xe-16-book.html.

[47] Cisco Systems. *Cisco Cloud Services Router 1000V Series - Configuration Guides*. URL: http://www.cisco.com/c/en/us/support/routers/cloud-services-router-1000v-series/products-installation-and-configuration-guides-list.html.

[48] Cisco Systems. *Netzwerksoftware (IOS und NX-OS)*. URL: http://www.cisco.com/c/de_de/products/ios-nx-os-software/index.html.

[49] Juniper Networks. *Routing-Produkte*. URL: http://www.juniper.net/de/de/products-services/routing/.

[50] Juniper Networks. *Junos OS - Technical Documentation - Support*. URL: https://www.juniper.net/documentation/en_US/release-independent/junos/information-products/pathway-pages/junos/product/index.html.

[51] Juniper Networks. *OSPF Feature Guide - Technical Documentation - Support*. URL: https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/config-guide-routing/config-guide-ospf.html.

[52] Juniper Networks. *Junos OS Routing Protocols Library - Technical Documentation - Support*. URL: https://www.juniper.net/documentation/en_US/junos/information-products/pathway-pages/config-guide-routing/index.html.

[53] Juniper Networks. *Virtual Router Software*. URL: http://www.juniper.net/us/en/products-services/routing/mx-series/vmx/.

[54] Juniper Networks. *Junos - Netzwerkbetriebssystem*. URL: http://www.juniper.net/de/de/products-services/nos/junos/.

[55] Brocade Communications Systems. *Brocade vRouter (formerly Vyatta vRouter)*. URL: http://www.brocade.com/en/products-services/software-networking/network-functions-virtualization/vrouter.html.

[56] Brocade Communications Systems. *Brocade vRouter Architecture*. URL: http://www.brocade.com/content/html/en/technical-documentation-library/5600-vrouter-41R1-library/GUID-B07BD5C1-5E8A-497C-9FE6-DA9B31D27369.html.

[57] Brocade Communications Systems. *Brocade vRouter Data Sheet*. URL: http://www.brocade.com/content/dam/common/documents/content-types/datasheet/brocade-vrouter-ds.pdf.

[58] *Quagga Software Routing Suite*. URL: http://www.nongnu.org/quagga/.

[59] *Quagga Software Routing Suite: Mailing Lists*. URL: http://www.nongnu.org/quagga/lists.html.

[60] *Quagga Software Routing Suite: News*. URL: http://savannah.nongnu.org/news/?group=quagga.

[61] *Quagga Software Routing Suite: Download*. URL: http://download.savannah.gnu.org/releases/quagga/.

[62] *Quagga Software Routing Suite: Development*. URL: http://www.nongnu.org/quagga/devel.html.

[63] *Quagga Software Routing Suite: Documentation - Quagga Info pages - Command Index*. URL: http://www.nongnu.org/quagga/docs/docs-info.html#Command-Index.

[64] *The BIRD Internet Routing Daemon Project: BIRD Programmer's Documentation - BIRD Design*. URL: http://bird.network.cz/?get_doc&f=prog-1.html.

[65] *The BIRD Internet Routing Daemon Project*. URL: http://bird.network.cz/.

[66] *The BIRD Internet Routing Daemon Project: Mailing List*. URL: http://bird.network.cz/?m_list.

[67] *The BIRD Internet Routing Daemon Project: News Archive*. URL: http://bird.network.cz/?o_news.

[68] J. Chroboczek. *RFC 6126: The Babel Routing Protocol*. Internet Requests for Comments. 2011. URL: http://www.rfc-editor.org/rfc/rfc6126.txt.

[69] D. Ovsienko. *RFC 7298: Babel Hashed Message Authentication Code (HMAC) Cryptographic Authentication*. Internet Requests for Comments. 2014. URL: http://www.rfc-editor.org/rfc/rfc7298.txt.

[70] J. Chroboczek. *RFC 7557: Extension Mechanism for the Babel Routing Protocol*. Internet Requests for Comments. 2015. URL: http://www.rfc-editor.org/rfc/rfc7557.txt.

[71] *The BIRD Internet Routing Daemon Project: BIRD User's Guide - Protocols*. URL: http://bird.network.cz/?get_doc&f=bird-6.html.

[72] *The BIRD Internet Routing Daemon Project: Download Current releases*. URL: http://bird.network.cz/?download.

[73] *The BIRD Internet Routing Daemon Project: BIRD User's Guide - Filters*. URL: http://bird.network.cz/?get_doc&f=bird-5.html.

[74] Andrew S. Tanenbaum und Maarten Van Steen. *Distributed systems: Principles and Paradigms: Second Edition*. New York, NY, USA: Pearson Prentice Hall, 2006. ISBN: 0-13-239227-5.

[75] B. Gleeson, A. Lin, J. Heinanen, G. Armitage und A. Malis. *RFC 2764: A Framework for IP Based Virtual Private Networks*. Internet Requests for Comments. 2000. URL: http://www.rfc-editor.org/rfc/rfc2764.txt.

[76] L. Andersson und T. Madsen. *RFC 4026: Provider Provisioned Virtual Private Network (VPN) Terminology*. Internet Requests for Comments. 2005. URL: http://www.rfc-editor.org/rfc/rfc4026.txt.

[77] Loa Andersson und Eric C. Rosen. *Framework for Layer 2 Virtual Private Networks (L2VPNs)*. Working Draft. Internet-Draft. 2004. URL: http://www.ietf.org/internet-drafts/draft-ietf-l2vpn-l2-framework-05.txt.

[78] E. Rosen, A. Viswanathan und R. Callon. *RFC 3031: Multiprotocol Label Switching Architecture*. Internet Requests for Comments. 2001. URL: http://www.rfc-editor.org/rfc/rfc3031.txt.

[79] Thomas D. Nadeau und Ken Gray. *SDN: Software Defined Networks*. Sebastopol, CA, USA: O'Reilly, 2013. ISBN: 978-1-449-34230-2.

[80] E. Rosen, P. Psenak und P. Pillay-Esnault. *RFC 4577: OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)*. Internet Requests for Comments. 2006. URL: http://www.rfc-editor.org/rfc/rfc4577.txt.

[81] E. Rosen und Y. Rekhter. *RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs)*. Internet Requests for Comments. 2006. URL: http://www.rfc-editor.org/rfc/rfc4364.txt.

[82] D. Smith, J. Mullooly, W. Jaeger und T. Scholl. *RFC 6178: Label Edge Router Forwarding of IPv4 Option Packets*. Internet Requests for Comments. 2011. URL: http://www.rfc-editor.org/rfc/rfc6178.txt.

[83] S. Kent und R. Atkinson. *RFC 2401: Security Architecture for the Internet Protocol*. Internet Requests for Comments. 1998. URL: http://www.rfc-editor.org/rfc/rfc2401.txt.

[84] S. Kent und R. Atkinson. *RFC 2402: IP Authentication Header*. Internet Requests for Comments. 1998. URL: http://www.rfc-editor.org/rfc/rfc2402.txt.

[85] S. Kent und R. Atkinson. *RFC 2406: IP Encapsulating Security Payload (ESP)*. Internet Requests for Comments. 1998. URL: http://www.rfc-editor.org/rfc/rfc2406.txt.

[86] C. Kaufman, P. Hoffman, Y. Nir, P. Eronen und T. Kivinen. *RFC 7296: Internet Key Exchange Protocol Version 2 (IKEv2)*. Internet Requests for Comments. http://www.rfc-editor.org/rfc/rfc7296.txt. 2014. URL: http://www.rfc-editor.org/rfc/rfc7296.txt.

[87] D. McGrew und P. Hoffman. *RFC 7321: Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH)*. Internet Requests for Comments. 2014. URL: http://www.rfc-editor.org/rfc/rfc7321.txt.

[88] S. Kent und K. Seo. *RFC 4301: Security Architecture for the Internet Protocol*. Internet Requests for Comments. 2005. URL: http://www.rfc-editor.org/rfc/rfc4301.txt.

[89] S. Kent. *RFC 4303: IP Encapsulating Security Payload (ESP)*. Internet Requests for Comments. 2005. URL: http://www.rfc-editor.org/rfc/rfc4303.txt.

[90] D. Eastlake. *RFC 4305: Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)*. Internet Requests for Comments. 2005. URL: http://www.rfc-editor.org/rfc/rfc4305.txt.

[91] V. Smyslov und P. Wouters. *RFC 7619: The NULL Authentication Method in the Internet Key Exchange Protocol Version 2 (IKEv2)*. Internet Requests for Comments. 2015. URL: http://www.rfc-editor.org/rfc/rfc7619.txt.

[92] T. Kivinen, P. Wouters und H. Tschofenig. *RFC 7670: Generic Raw Public-Key Support for IKEv2*. Internet Requests for Comments. 2016. URL: http://www.rfc-editor.org/rfc/rfc7670.txt.

[93] R. Aggarwal, M. Townsley und M. Dos Santos. *RFC 4719: Transport of Ethernet Frames over Layer 2 Tunneling Protocol Version 3 (L2TPv3)*. Internet Requests for Comments. 2006. URL: http://www.rfc-editor.org/rfc/rfc4719.txt.

[94] J. Lau, M. Townsley und I. Goyret. *RFC 3931: Layer Two Tunneling Protocol - Version 3 (L2TPv3)*. Internet Requests for Comments. 2005. URL: http://www.rfc-editor.org/rfc/rfc3931.txt.

[95] B. Patel, B. Aboba, W. Dixon, G. Zorn und S. Booth. *RFC 3193: Securing L2TP using IPsec*. Internet Requests for Comments. 2001. URL: http://www.rfc-editor.org/rfc/rfc3193.txt.

[96] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little und G. Zorn. *RFC 2637: Point-to-Point Tunneling Protocol (PPTP)*. Internet Requests for Comments. 1999. URL: http://www.rfc-editor.org/rfc/rfc2637.txt.

[97] W. Simpson. *RFC 1661: The Point-to-Point Protocol (PPP)*. Internet Requests for Comments. 1994. URL: http://www.rfc-editor.org/rfc/rfc1661.txt.

[98] D. Rand. *RFC 1663: PPP Reliable Transmission*. Internet Requests for Comments. 1994. URL: http://www.rfc-editor.org/rfc/rfc1663.txt.

[99] W. Simpson. *RFC 2153: PPP Vendor Extensions*. Internet Requests for Comments. 1997. URL: http://www.rfc-editor.org/rfc/rfc2153.txt.

[100] Daniel J. Barrett und Richard E. Silverman. *SSH, The Secure Shell: The Definitive Guide*. Sebastopol, CA, USA: O'Reilly, 2001. ISBN: 0-596-00011-1.

[101] Henk C.A. van Tilborg, Hrsg. *Encyclopedia of Cryptography and Security: Editor-in-chief Henk C. A. van Tilborg*. New York, NY, USA: Springer, 2005. ISBN: 978-0387-23473-1.

[102] *OpenSSH*. URL: https://www.openssh.com/.

[103] Scott Bronson. *VPN PPP-SSH Mini-HOWTO, Revision v1.00*. The Linux Documentation Project. 2002. URL: http://www.tldp.org/HOWTO/pdf/ppp-ssh.pdf.

[104] OpenSSL Software Foundation. *OpenSSL Cryptography and SSL/TLS Toolkit*. URL: https://www.openssl.org/.

[105] Maxim Krasnyansky. *Universal TUN/TAP driver - Virtual Point-to-Point(TUN) and Ethernet(TAP) devices*. URL: http://vtun.sourceforge.net/tun/index.html.

[106] Maxim Krasnyansky. *VTun - Virtual Tunnels over TCP/IP networks*.

[107] Maxim Krasnyansky. *VTun - Virtual Tunnels over TCP/IP networks - Download VTun*. URL: http://vtun.sourceforge.net/download.html.

[108] University of Kent, School of Computing. *UK Mirror Service*. Kent, UK. URL: http://www.mirrorservice.org/.

[109] Internet Archive. *Wayback Machine*. URL: https://web.archive.org/.

[110] Luca Deri und Richard Andrews. *N2N: A Layer Two Peer-to-Peer VPN*. 2008. URL: http://luca.ntop.org/n2n.pdf.

[111] ARM Limited. *SSL Library mbed TLS / PolarSSL*. URL: http://polarssl.org/.

[112] OpenVPN Inc. *OpenVPN - Open Source VPN*. URL: https://openvpn.net/.

[113] Jan Just Keijser. *OpenVPN 2 Cookbook*. Birmingham, UK: Packt Publishing, 2011. ISBN: 978-1-849510-10-3.

[114] OpenVPN Inc. *OpenVPN - HOWTO - Determining whether to use a routed or bridged VPN*. URL: https://openvpn.net/index.php/open-source/documentation/howto.html#vpntype.

[115] T. Dierks und E. Rescorla. *RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2*. Internet Requests for Comments. 2008. URL: http://www.rfc-editor.org/rfc/rfc5246.txt.

[116] OpenVPN Inc. *OpenVPN - Installation Notes*. URL: https://openvpn.net/index.php/open-source/documentation/install.html.

[117] Markus F.X.J. Oberhumer, oberhumer.com GmbH. *LZO real-time data compression library*. Linz, Österreich. URL: http://www.oberhumer.com/opensource/lzo/.

[118] OpenVPN Inc. *OpenVPN - What is OpenVPN?* URL: https://openvpn.net/index.php/open-source/333-what-is-openvpn.html.

[119] OpenVPN Inc. *OpenVPN - HOWTO - Hardening OpenVPN Security*. URL: https://openvpn.net/index.php/open-source/documentation/howto.html#security.

[120] OpenVPN Inc. *OpenVPN - Release Notes*. URL: https://openvpn.net/index.php/open-source/documentation/release-notes.html.

[121] OpenVPN Inc. *OpenVPN - HOWTO - Creating configuration files for server and clients*. URL: https://openvpn.net/index.php/open-source/documentation/howto.html#config.

[122] OpenVPN Inc. *OpenVPN - Community Project Overview*. URL: https://openvpn.net/index.php/open-source/245-community-open-source-software-overview.html.

[123] GitHub Inc. *GitHub - OpenVPN/tap-windows: Windows TAP driver*. URL: https://github.com/OpenVPN/tap-windows.

[124] OpenVPN Inc. *OpenVPN - Downloads*. URL: https://openvpn.net/index.php/open-source/downloads.html.

[125] *Tinc VPN - tinc*. URL: http://www.tinc-vpn.org/.

[126] *Tinc VPN - Example: bridging Ethernet segments using tinc under Linux*. URL: http://www.tinc-vpn.org/examples/bridging/.

[127] *LibreSSL*. URL: https://www.libressl.org/.

[128] Ivo Timmermans und Guus Sliepen. *tinc Manual*. 2016. URL: http://www.tinc-vpn.org/documentation/tinc.pdf.

[129] *Tinc VPN - Supported platforms*. URL: http://www.tinc-vpn.org/platforms/.

[130] *Tinc VPN - Download*. URL: http://www.tinc-vpn.org/download/.

[131] *PeerVPN - the open source peer-to-peer VPN*. URL: https://peervpn.net/.

[132] *PeerVPN - Tutorial*. URL: https://peervpn.net/tutorial/.

[133] *PeerVPN - Download source code*. URL: https://peervpn.net/files/peervpn-0-044.tar.gz.

[134] Charlie Hosner. *OpenVPN and the SSL VPN Revolution*. SANS Institute. 2004. URL: https://www.sans.org/reading-room/whitepapers/vpns/openvpn-ssl-vpn-revolution-1459.

[135] B. Cain, S. Deering, I. Kouvelas, B. Fenner und A. Thyagarajan. *RFC 3376: Internet Group Management Protocol, Version 3*. Internet Requests for Comments. 2002. URL: http://www.rfc-editor.org/rfc/rfc3376.txt.

[136] Kamill Panitzek und Thorsten Strufe. „A Formal Model for Peer-to-Peer Systems“. In: *LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, Understanding Quality of Service in Large-Scale Distributed Systems*. Hrsg. von Wolfgang Effelsberg, Ralf Steinmetz und Thorsten Strufe. Bd. 7847. SL 5 - Computer Communication Networks and Telecommunications. Berlin: Springer, 2013, S. 15–18. ISBN: 978-3-642-38673-2.

[137] Wolfgang Effelsberg, Ralf Steinmetz und Thorsten Strufe, Hrsg. *LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems: Understanding Quality of Service in Large-Scale Distributed Systems*. Bd. 7847. SL 5 - Computer Communication Networks and Telecommunications. Berlin: Springer, 2013. ISBN: 978-3-642-38673-2.

[138] Christian Groß, Björn Richerzhagen und Max Lehn. „Structured Search Overlays“. In: *LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, Understanding Quality of Service in Large-Scale Distributed Systems*. Hrsg. von Wolfgang Effelsberg, Ralf Steinmetz und Thorsten Strufe. Bd. 7847. SL 5 - Computer Communication Networks and Telecommunications. Berlin: Springer, 2013, S. 49–67. ISBN: 978-3-642-38673-2.

[139] Wolfgang Ertel. *Angewandte Kryptographie: 3., aktualisierte Auflage*. München: Carl Hanser Verlag, 2007. ISBN: 978-3-446-41195-1.

[140] Chord Project. *Overview*. URL: https://pdos.csail.mit.edu/chord/.

[141] Petar Maymounkov und David Mazières. „Kademlia: A Peer-to-Peer Information System Based on the XOR Metric“. In: *LNCS (Lecture Notes in Computer Science): Peer-to-Peer Systems: IPTPS 2002 (International Workshop on Peer-to-Peer Systems)*. Hrsg. von Peter Druschel, Frans Kaashoek und Antony Rowstron. Bd. 2429. Berlin: Springer, 2002, S. 53–65. ISBN: 978-3-540-44179-3.

[142] Peter Druschel, Frans Kaashoek und Antony Rowstron, Hrsg. *LNCS (Lecture Notes in Computer Science): Peer-to-Peer Systems: IPTPS 2002 (International Workshop on Peer-to-Peer Systems)*. Bd. 2429. Berlin: Springer, 2002. ISBN: 978-3-540-44179-3.

[143] Pastry Project. *Pastry - A scalable, decentralized, self-organizing and fault-tolerant substrate for peer-to-peer applications*. URL: http://www.freepastry.org/.

[144] Sylvia Ratnasamy. „A Scalable Content-Addressable Network“. Ph.D. Thesis. Berkeley, CA, USA: University of California at Berkeley, 2002. URL: http://www.icir.org/sylvia/thesis.ps.

[145] Freenet Project Inc. *Freenet*. URL: https://freenetproject.org/.

[146] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph und John D. Kubiatowicz. „Tapestry: A Resilient Global-Scale Overlay for Service Deployment“. In: *IEEE Journal on Selected Areas in Communications* 22.1 (2004), S. 41–53. ISSN: 0733-8716. DOI: 10.1109/JSAC.2003.818784.

[147] Kamill Panitzek, Muhammad Ikram und Michael Stein. „Service Overlays“. In: *LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, Understanding Quality of Service in Large-Scale Distributed Systems*. Hrsg. von Wolfgang Effelsberg, Ralf Steinmetz und Thorsten Strufe. Bd. 7847. SL 5 - Computer Communication Networks and Telecommunications. Berlin: Springer, 2013, S. 113–140. ISBN: 978-3-642-38673-2.

[148] Elias Weingärtner, René Glebke und Alexander Hocks. „Content Delivery Overlays“. In: *LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, Understanding Quality of Service in Large-Scale Distributed Systems*. Hrsg. von Wolfgang Effelsberg, Ralf Steinmetz und Thorsten Strufe. Bd. 7847. SL 5 - Computer Communication Networks and Telecommunications. Berlin: Springer, 2013, S. 69–79. ISBN: 978-3-642-38673-2.

[149] Osama Abboud und Julius Rückert. „Video-Streaming Overlays“. In: *LNCS (Lecture Notes in Computer Science): Benchmarking Peer-to-Peer Systems, Understanding Quality of Service in Large-Scale Distributed Systems*. Hrsg. von Wolfgang Effelsberg, Ralf Steinmetz und Thorsten Strufe. Bd. 7847. SL 5 - Computer Communication Networks and Telecommunications. Berlin: Springer, 2013, S. 169–193. ISBN: 978-3-642-38673-2.

[150] The Tor Project Inc. *Tor Project: Anonymity Online*. Seattle, WA, USA. URL: https://www.torproject.org/.

[151] Roger Dingledine, Nick Mathewson und Paul F. Syverson. „Tor: The Second-Generation Onion Router“. In: *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*. Hrsg. von Matt Blaze. USENIX, 2004, S. 303–320. URL: http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html.

[152] Matt Blaze, Hrsg. *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*. USENIX, 2004.

[153] George Danezis, Claudia Diaz und Paul Syverson. *Systems for Anonymous Communication*. 2009. DOI: 10.1.1.225.2880. URL: https://www.esat.kuleuven.be/cosic/publications/article-1335.pdf.

[154] Michael J. Freedman. „A Peer-to-Peer Anonymizing Network Layer“. Master Thesis. Cambridge, MA, USA: Massachusetts Institute of Technology, 2002. URL: https://www.cs.princeton.edu/~mfreed/docs/tarzan-thesis.pdf.

[155] I2P Project. *The Invisible Internet Project: I2P Anonymous Network*. URL: https://geti2p.net/.

[156] Roger R. Dingledine. „The Free Haven Project: Design and Deployment of an Anonymous Secure Data Haven“. Master Thesis. Cambridge, MA, USA: Massachusetts Institute of Technology, 2000. URL: https://dspace.mit.edu/bitstream/handle/1721.1/86475/46861930-MIT.pdf?sequence=2.

[157] I2P Project. *The Invisible Internet Project: I2P Anonymous Network - Documentation - Garlic Routing*. URL: https://geti2p.net/en/docs/how/garlic-routing.

[158] Google Inc. *Google Search*. Mountain View, CA, USA. URL: https://www.google.com/.

[159] The Tor Project Inc. *Tor Project: Documentation - FAQ - Running a Tor relay*. URL: https://www.torproject.org/docs/faq.html.en#RunningATorRelay.

[160] The Tor Project Inc. *Tor Project: Documentation - FAQ - Running a Tor relay: I'm behind a NAT/Firewall*. URL: https://www.torproject.org/docs/faq.html.en#BehindANAT.

[161] The Tor Project Inc. *Tor Project: Documentation - FAQ - Running a Tor relay: How do offline ed25519 identity keys work? What do I need to know?* URL: https://www.torproject.org/docs/faq.html.en#OfflineED25519.

[162] The Tor Project Inc. *Tor Project: Download Tor - Source Code*. URL: https://www.torproject.org/download/download.html.en#source.

[163] *Libevent: libevent - an event notification library*. URL: http://libevent.org/.

[164] W. Richard Stevens und Gary R. Wright. *TCP/IP Illustrated, Volume 2: The Implementation*. Addison-Wesley professional computing series. Upper Saddle River, NJ, USA: Addison-Wesley, 1995. ISBN: 0-201-63354-X.

[165] The Tor Project Inc. *Tor Project: Documentation - FAQ - Development: What is Libevent?* URL: https://www.torproject.org/docs/faq.html.en#WhatIsLibevent.

[166] The Tor Project Inc. *Tor Project: Download for Linux/Unix*. URL: https://www.torproject.org/download/download-unix.html.en.

[167] The Tor Project Inc. *Tor Project: Download Tor - Tor for Smartphones*. URL: https://www.torproject.org/download/download.html.en#smartphone.

[168] I2P Project. *The Invisible Internet Project: I2P Anonymous Network - Introduction - I2P: A scalable framework for anonymous communication*. URL: https://geti2p.net/en/docs/how/tech-intro.

[169] I2P Project. *The Invisible Internet Project: I2P Anonymous Network - Documentation - API - I2PTunnel*. URL: https://geti2p.net/en/docs/api/i2ptunnel.

[170] I2P Project. *The Invisible Internet Project: I2P Anonymous Network - Index to Technical Documentation*. URL: https://geti2p.net/en/docs.

[171] I2P Project. *The Invisible Internet Project: I2P Anonymous Network - Documentation - API - SOCKS*. URL: https://geti2p.net/en/docs/api/socks.

[172] Khalid Shahbar und A. Nur Zincir-Heywood. „Effects of Shared Bandwidth on Anonymity of the I2P Network Users“. In: *38th IEEE Symposium on Security and Privacy Workshops, 2nd International Workshop on Traffic Measurements for Cybersecurity (WTMC 2017)*. 2017. URL: https://www.cs.dal.ca/sites/default/files/technical_reports/cs-2017-04.pdf.

[173] I2P Project. *The Invisible Internet Project: I2P Anonymous Network - Specifications - Low-level Cryptography Specification*. URL: https://geti2p.net/spec/cryptography.

[174] T. Kivinen und M. Kojo. *RFC 3526: More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)*. Internet Requests for Comments. 2003. URL: http://www.rfc-editor.org/rfc/rfc3526.txt.

[175] I2P Project. *The Invisible Internet Project: I2P Anonymous Network - Download*. URL: https://geti2p.net/en/download.

[176] I2P Project. *The Invisible Internet Project: I2P Anonymous Network - New Developer's Guide*. URL: https://geti2p.net/en/get-involved/guides/new-developers.

[177] Telekom Deutschland GmbH. *Leistungsbeschreibung Standard-Festverbindungen Digital (dSFV)*. Bonn. URL: http://www.telekom.de/dlp/agb/pdf/43602.pdf.

[178] Matthias Wählisch und Thomas C. Schmidt. „Peer the Peers: An Overlay ID Assignment Service at Internet Exchange Points". In: *Proceedings of the 5th International Student Workshop on Emerging Networking Experiments and Technologies*. Co-Next Student Workshop '09. Rome, Italy: ACM, 2009, S. 45–46. ISBN: 978-1-60558-751-6. DOI: 10.1145/1658997.1659022. URL: http://conferences.sigcomm.org/co-next/2009/workshops/student/papers/Wahlisch.pdf.

[179] T. Berners-Lee, R. Fielding und H. Frystyk. *RFC 1945: Hypertext Transfer Protocol – HTTP/1.0*. Internet Requests for Comments. 1996. URL: http://www.rfc-editor.org/rfc/rfc1945.txt.

[180] R. Fielding und J. Reschke. *RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. Internet Requests for Comments. 2014. URL: http://www.rfc-editor.org/rfc/rfc7230.txt.

[181] M. Belshe, R. Peon und M. Thomson. *RFC 7540: Hypertext Transfer Protocol Version 2 (HTTP/2)*. Internet Requests for Comments. 2015. URL: http://www.rfc-editor.org/rfc/rfc7540.txt.

[182] CAIDA. *IPv4 and IPv6 AS Core: Visualizing IPv4 and IPv6 Internet Topology at a Macroscopic Scale in 2013*. Center for Applied Internet Data Analysis. San Diego, CA, USA, 2013. URL: http://www.caida.org/research/topology/as_core_network/2013/.

[183] Alex Biryukov, Ivan Pustogarov und Ralf-Philipp Weinmann. „TorScan: Deanonymizing Connections Using Topology Leaks". In: *ERCIM News 90: Special theme: Cybercrime and Privacy Issues*. Hrsg. von ERCIM EEIG. Bd. Number 90, July 2012. Sophia Antipolis Cedex, France: ERCIM EEIG, 2012, S. 29–30. URL: https://ercim-news.ercim.eu/en90/special/torscan-deanonymizing-connections-using-topology-leaks.

[184] ERCIM EEIG, Hrsg. *ERCIM News 90: Special theme: Cybercrime and Privacy Issues*. Bd. Number 90, July 2012. Sophia Antipolis Cedex, France: ERCIM EEIG, 2012.

[185] Chen Chen, Daniele Enrico Asoni, David Barrera, George Danezis und Adrian Perrig. *HORNET: High-speed Onion Routing at the Network Layer*. 2016. URL: https://arxiv.org/pdf/1507.05724v3.

[186] Google Inc. *Google Code Archive: curve25519-donna*. Mountain View, CA, USA. URL: https://code.google.com/archive/p/curve25519-donna/.

[187] Jeffrey Rott, Intel Corporation. *Intel AESNI Sample Library*. Santa Clara, CA, USA, 2011. URL: https://software.intel.com/en-us/articles/download-the-intel-aesni-sample-library.

[188] DPDK Project. *DPDK - Data Plane Development Kit*. URL: http://dpdk.org/.

[189] Bernhard R. Fischer. *The OnionCat Project*. URL: http://www.cypherpunk.at/onioncat/.

[190] The Tor Project Inc. *Tor Project: Documentation - Hidden Service Protocol*. URL: https://www.torproject.org/docs/hidden-services.html.en.

[191] The Tor Project Inc. *Tor Project: Documentation - Hidden Service Configuration Instructions*. URL: https://www.torproject.org/docs/tor-hidden-service.html.en.

[192] Cygwin Project. *The Cygwin Project Page*. URL: http://www.cygwin.com/.

[193] The Open Group. *ISO/IEC 9945*. Portable Operating System Interface (POSIX). URL: http://www.unix.org/version3/iso_std.html.

[194] The Open Group. *IEEE Portable Applications Standards Committee (PASC Committee)*. URL: https://collaboration.opengroup.org/external/pasc.org/plato/#POSIX.

[195] Bernhard R. Fischer und Ferdinand Haselbacher. *The OnionCat Download Page*. URL: http://www.cypherpunk.at/ocat/download.

[196] SPI Inc. *Debian Project - About Debian*. Software in the Public Interest, Inc. URL: https://www.debian.org/intro/about.

[197] SPI Inc. *Debian Project - Debian Installation Guide*. Software in the Public Interest, Inc. URL: https://www.debian.org/releases/stable/installmanual.

[198] B. S. Everitt und A. Skrondal. *The Cambridge Dictionary of Statistics*. 4. Aufl. Cambridge, UK: Cambridge University Press, 2010. ISBN: 978-0-521-76699-9.

[199] Christopher Clapham und James Nicholson. *The Concise Oxford Dictionary of Mathematics*. 4. Aufl. Oxford: Oxford Univ. Press, 2009. ISBN: 978-0-19-923594-0.

[200] *LVM2 Resource Page*. Logical Volume Manager (Linux). URL: https://www.sourceware.org/lvm2/.

[201] Oracle Corporation. *Oracle VM VirtualBox*. Redwood Shores, CA. URL: https://www.virtualbox.org/.

[202] *DFN-Verein e. V. - Deutsches Forschungsnetz*. DFN Verein zur Förderung eines Deutschen Forschungsnetzes e.V. Berlin. URL: https://www.dfn.de/.

[203] F. Baker. *RFC 1812: Requirements for IP Version 4 Routers*. Internet Requests for Comments. 1995. URL: http://www.rfc-editor.org/rfc/rfc1812.txt.

[204] D. Mills, J. Martin, J. Burbank und W. Kasch. *RFC 5905: Network Time Protocol Version 4: Protocol and Algorithms Specification*. Internet Requests for Comments. 2010. URL: http://www.rfc-editor.org/rfc/rfc5905.txt.

[205] P. Mockapetris. *RFC 1034: Domain names - concepts and facilities*. Internet Requests for Comments. 1987. URL: http://www.rfc-editor.org/rfc/rfc1034.txt.

[206] P. Mockapetris. *RFC 1035: Domain names - implementation and specification*. Internet Requests for Comments. 1987. URL: http://www.rfc-editor.org/rfc/rfc1035.txt.

[207] R. Arends, R. Austein, M. Larson, D. Massey und S. Rose. *RFC 4033: DNS Security Introduction and Requirements*. Internet Requests for Comments. 2005. URL: http://www.rfc-editor.org/rfc/rfc4033.txt.

[208] R. Arends, R. Austein, M. Larson, D. Massey und S. Rose. *RFC 4035: Protocol Modifications for the DNS Security Extensions*. Internet Requests for Comments. 2005. URL: http://www.rfc-editor.org/rfc/rfc4035.txt.

[209] Bruce Hartpence. *Packet Guide to Core Network Protocols*. Sebastopol, CA, USA: O'Reilly, 2011. ISBN: 978-1-449-30653-3.

[210] J. Postel. *RFC 792: Internet Control Message Protocol*. Internet Requests for Comments. 1981. URL: http://www.rfc-editor.org/rfc/rfc792.txt.

[211] R. Bonica, D. Gan, D. Tappan und C. Pignataro. *RFC 4884: Extended ICMP to Support Multi-Part Messages*. Internet Requests for Comments. 2007. URL: http://www.rfc-editor.org/rfc/rfc4884.txt.

[212] OpenVPN Inc. *OpenVPN - Documentation - RSA Key Management*. URL: https://openvpn.net/index.php/open-source/documentation/miscellaneous/77-rsa-key-management.html.

[213] Kunihiro Ishiguro, et al. *Quagga Software Routing Suite: Documentation*. Quagga - A routing software package for TCP/IP networks - Quagga 1.2.0. März 2017. URL: http://www.nongnu.org/quagga/docs/quagga.pdf.

[214] AppArmor security project. *AppArmor - Linux application security system*. URL: http://apparmor.net/.

[215] Gary A. Donahue. *Network Warrior*. 2. Aufl. Sebastopol, CA, USA: O'Reilly, 2011. ISBN: 978-1-449-38786-0.

[216] Bob Smith, John Hardin, Graham Phillips und Bill Pierce. *Linux appliance design: A hands-on guide to building linux appliances*. San Francisco, CA: No Starch Press, 2007. ISBN: 978-1-59327-140-4.

[217] V. Paxson, G. Almes, J. Mahdavi und M. Mathis. *RFC 2330: Framework for IP Performance Metrics*. Internet Requests for Comments. 1998. URL: http://www.rfc-editor.org/rfc/rfc2330.txt.

[218] Energy Sciences Network (ESnet) und Lawrence Berkeley National Laboratory. *iperf3 Project*. Berkeley, CA, USA, 2017. URL: http://software.es.net/iperf/.

[219] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang und V. Paxson. *RFC 2960: Stream Control Transmission Protocol*. Internet Requests for Comments. 2000. URL: http://www.rfc-editor.org/rfc/rfc2960.txt.

[220] L. Ong und J. Yoakum. *RFC 3286: An Introduction to the Stream Control Transmission Protocol (SCTP)*. Internet Requests for Comments. 2002. URL: http://www.rfc-editor.org/rfc/rfc3286.txt.

[221] R. Stewart. *RFC 4960: Stream Control Transmission Protocol*. Internet Requests for Comments. 2007. URL: http://www.rfc-editor.org/rfc/rfc4960.txt.

[222] Energy Sciences Network (ESnet) und Lawrence Berkeley National Laboratory. *iperf3 Project - Source Code downloads*. Berkeley, CA, USA, 2017. URL: http://downloads.es.net/pub/iperf/.

[223] GitHub Inc. *GitHub - esnet/iperf: iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool*. 2017. URL: https://github.com/esnet/iperf.

[224] Vivien Gueant. *Iperf.fr - Public iPerf3 servers*. URL: https://iperf.fr/iperf-servers.php.

[225] Teemu Toivola. *vnStat - a network traffic monitor for Linux and BSD*. 2017. URL: http://humdi.net/vnstat/.

[226] GitHub Inc. *GitHub - vergoh/vnstat: vnStat - a network traffic monitor for Linux and BSD*. 2017. URL: https://github.com/vergoh/vnstat.

[227] Teemu Toivola. *vnStat - Documentation - Man page of vnstat.conf*. 2017. URL: http://humdi.net/vnstat/man/vnstat.conf.html.

[228] Tobias Oetiker, Oetiker+Partner AG. *MRTG - Multi Router Traffic Grapher*. Olten, Schweiz, 2012. URL: https://oss.oetiker.ch/mrtg/.

[229] Tobias Oetiker, Oetiker+Partner AG. *MRTG - Documentation - What is MRTG?* Olten, Schweiz, 2012. URL: https://oss.oetiker.ch/mrtg/doc/mrtg.en.html.

[230] Jeffrey D. Case, Keith McCloghrie, Keith McCloghrie, Marshall T. Rose und Steve Waldbusser. *RFC 1901: Introduction to Community-based SNMPv2*. Internet Requests for Comments. 1996. URL: http://www.rfc-editor.org/rfc/rfc1901.txt.

[231] J. Case, K. McCloghrie, M. Rose und S. Waldbusser. *RFC 1908: Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework*. Internet Requests for Comments. 1996. URL: http://www.rfc-editor.org/rfc/rfc1908.txt.

[232] Keith McCloghrie, David Perkins und Juergen Schoenwaelder. *RFC 2578: Structure of Management Information Version 2 (SMIv2)*. Internet Requests for Comments. 1999. URL: http://www.rfc-editor.org/rfc/rfc2578.txt.

[233] R. Presuhn. *RFC 3416: Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)*. Internet Requests for Comments. 2002. URL: http://www.rfc-editor.org/rfc/rfc3416.txt.

[234] R. Presuhn. *RFC 3418: Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*. Internet Requests for Comments. 2002. URL: http://www.rfc-editor.org/rfc/rfc3418.txt.

[235] Keith McCloghrie und Marshall T. Rose. *RFC 1213: Management Information Base for Network Management of TCP/IP-based internets: MIB-II*. Internet Requests for Comments. 1991. URL: http://www.rfc-editor.org/rfc/rfc1213.txt.

[236] K. McCloghrie und F. Kastenholz. *RFC 2863: The Interfaces Group MIB*. Internet Requests for Comments. 2000. URL: http://www.rfc-editor.org/rfc/rfc2863.txt.

[237] S. Routhier. *RFC 4293: Management Information Base for the Internet Protocol (IP)*. Internet Requests for Comments. 2006. URL: http://www.rfc-editor.org/rfc/rfc4293.txt.

[238] Tobias Oetiker, Oetiker+Partner AG. *MRTG - Documentation*. Olten, Schweiz, 2012. URL: https://oss.oetiker.ch/mrtg/doc/index.en.html.

[239] Tobias Oetiker, Oetiker+Partner AG. *MRTG - Documentation - The MRTG 2.17.4 Linux/Unix Installation Guide*. Olten, Schweiz, 2012. URL: https://oss.oetiker.ch/mrtg/doc/mrtg-unix-guide.en.html.

[240] Tobias Oetiker, Oetiker+Partner AG. *MRTG - Download*. Olten, Schweiz, 2012. URL: https://oss.oetiker.ch/mrtg/download.en.html.

[241] Tobias Oetiker, Oetiker+Partner AG. *MRTG - Documentation - MRTG 2.17.4 configuration reference*. Olten, Schweiz, 2012. URL: https://oss.oetiker.ch/mrtg/doc/mrtg-reference.en.html.

[242] Free Software Foundation (FSF). *Wget - GNU Project - Free Software Foundation*. Boston, MA, USA, 2017. URL: https://www.gnu.org/software/wget/.

[243] Free Software Foundation (FSF). *Wget - FAQ - Frequently Asked Questions About GNU Wget*. Boston, MA, USA, 2017. URL: https://www.gnu.org/software/wget/faq.html.

[244] Free Software Foundation (FSF). *Wget - Downloading GNU Wget - source code*. Boston, MA, USA, 2017. URL: https://ftp.gnu.org/gnu/wget/.

[245] Igor Sysoev. *nginx [engine x] - HTTP and reverse proxy server*. 2017. URL: https://nginx.org/en/.

[246] Dipankar Sarkar. *Nginx 1 Web Server Implementation Cookbook*. Birmingham, UK: Packt Publishing, 2011. ISBN: 978-1-849514-96-5.

[247] Igor Sysoev. *nginx - Documentation - FAQ*. 2017. URL: https://nginx.org/en/docs/faq.html.

[248] Igor Sysoev. *nginx - Documentation*. 2017. URL: https://nginx.org/en/docs/.

[249] Igor Sysoev. *nginx - Download*. 2017. URL: https://nginx.org/en/download.html.

[250] Gnuplot Project. *gnuplot - a portable command-line driven graphing utility*. 2017. URL: http://www.gnuplot.info/.

[251] Osamu Honda, Hiroyuki Ohsaki, Makoto Imase, Mika Ishizuka und Junichi Murayama. *Understanding TCP over TCP: effects of TCP tunneling on end-to-end throughput and latency*. Osaka und Tokyo, Japan, 2005. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.78.5815&rep=rep1&type=pdf.

[252] M. Allman, V. Paxson und E. Blanton. *RFC 5681: TCP Congestion Control*. Internet Requests for Comments. 2009. URL: http://www.rfc-editor.org/rfc/rfc5681.txt.

[253] M. Allman, S. Floyd und C. Partridge. *RFC 3390: Increasing TCP's Initial Window*. Internet Requests for Comments. 2002. URL: http://www.rfc-editor.org/rfc/rfc3390.txt.

[254] D. Papadimitriou, M. Welzl, M. Scharf und B. Briscoe. *RFC 6077: Open Research Issues in Internet Congestion Control*. Internet Requests for Comments. 2011. URL: http://www.rfc-editor.org/rfc/rfc6077.txt.

[255] Berry Hoekstra und Damir Musulin. *Comparing TCP Performance of tunneled and non-tunneled Traffic using OpenVPN*. 2011. URL: http://www.delaat.net/rp/2010-2011/p09/report.pdf.

[256] OpenSSL Software Foundation. *OpenSSL - Documentation - ENGINE cryptographic module support*. URL: https://www.openssl.org/docs/man1.0.2/crypto/engine.html.

[257] Hervé Schauer Consultants (HSC). *Dns2tcp - a network tool designed to relay TCP connections through DNS traffic*. Neuilly-sur-Seine, France, 2012. URL: http://www.hsc.fr/ressources/outils/dns2tcp/.

[258] Baptiste Jonglez, Matthieu Boutier und Juliusz Chroboczek. *A delay-based routing metric*. 2014. URL: https://arxiv.org/pdf/1403.3488.pdf.

Abbildungsverzeichnis

2.1 Grössere Autonome Systeme im Internet - ein Teil des Internet-Backbones 8
2.2 Beispiel für eine Netzwerk-Topologie mit OSPF als Routing-Protokoll . . 13

3.1 Kern Autonome Systeme (IPv4 und IPv6), Januar 2013 47
3.2 Connectivity der 105 schnellsten Tor-Relays am 14.03.2012, 13:59 GMT . . 48

5.1 LAN-LAN-Kopplung mit mehrfacher Auslegung der VPNs 55
5.2 Übersicht der Konzeption der LAN-LAN-Kopplung mit Hilfe von Overlay-Netzwerken und VPNs 57
5.3 Übersicht über die definierten und erwünschten Systemzustände 66
5.4 Übersicht der Konzeption der Laufzeitumgebung mit Software-Komponenten 68

6.1 Struktur der Laborumgebung mit zwei Routern und zwei Arbeitsstationen an LAN-Segmenten 70
6.2 Struktur der Produktivumgebung mit zwei Routern und zwei Arbeitsstationen an LAN-Segmenten 71
6.3 Muster-Appliances mit VPN-Kopplung 76
6.4 Muster-Appliances mit dynamischem Routing 78
6.5 Weitergeleitete TCP-Verbindungen über das Overlay-Netzwerk Tor 81
6.6 Veranschaulichung der Integration der Overlay-Netzwerkknoten mit einem darauf aufbauendem VPN-Tunnel 83
6.7 Veranschaulichung der Integration nativer VPN-Tunnel und auf Overlay-Netzwerkknoten aufbauender VPN-Tunnel mit dynamischem Routing . . 84
6.8 Überblick über die Systemarchitektur der Routerumgebung 90

7.1 Diagramm über Paket-Umlaufzeiten zwischen LAN `lan1` und LAN `lan2` am 11.10.2017, gemessen in Millisekunden 101
7.2 Diagramm über Paket-Umlaufzeiten zwischen Router `r1` und Router `r2` am 11.10.2017, gemessen in Millisekunden 102
7.3 Diagramm über Paket-Umlaufzeiten zwischen LAN `lan1` und LAN `lan2` von 04.10.2017 bis 11.10.2017, gemessen in Millisekunden 102
7.4 Diagramm über die gemessene verfügbare Übertragungsbandbreite zwischen LAN `lan1` und LAN `lan2` bei Verfügbarkeit der nativen VPN-Tunnel 103
7.5 Diagramm über erneut übertragene IP-Pakete während der Messung bei Verfügbarkeit der nativen VPN-Tunnel 103
7.6 Diagramm über die gemessene verfügbare Übertragungsbandbreite zwischen LAN `lan1` und LAN `lan2` bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel 104
7.7 Diagramm über erneut übertragene IP-Pakete während der Messung bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel 104
7.8 Diagramm über die Übertragungsbandbreite zwischen den LANs bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel und einer Bandbreiten-Zielvorgabe von 4 MBit/s 105

7.9 Diagramm über erneut übertragene IP-Pakete bei ausschliesslicher Nutzung der Overlay-VPN-Tunnel und einer Bandbreiten-Zielvorgabe von 4 MBit/s 105

Tabellenverzeichnis

2.1 Regionale Internet Registries weltweit mit ihren Zuständigkeiten nach Region 5
2.2 20 wichtige Autonome Systeme mit Topologie-Kriterien 7
2.3 Hersteller von genutzten Switches und Routern bei den Europäischen Internet-Austauschpunkten nach Marktanteil 15
2.4 Betriebssysteme der genutzten Route Server bei den Europäischen Internet-Austauschpunkten nach Anteil 16
2.5 Routing-Softwares der genutzten Route Server bei den Europäischen Internet-Austauschpunkten nach Anteil 16
2.6 Anzahl der Ergebnisse der Suchmaschine *Google Search* bezüglich verschiedener Overlay-Netzwerke 38

5.1 Übersicht über die definierten und erwünschten Systemzustände 66
5.2 Übersicht über die wichtigsten Zustandsübergänge der Routerumgebung 67

Listings

6.1 Transkript der Laufzeitumgebung auf Router `r1` in der Betriebsart mit Ausgabe auf der Konsole 86
6.2 Statistiken über das *Traffic Shaping* an der Netzwerkbrücke `br4` auf Router `r1` 88
6.3 Transkript der Syslog-Einträge der Laufzeitumgebung auf Router `r1` in der Betriebsart mit Ausgabe auf den Syslog-Server 91

A.1 Datei backup-pull-r1-configs-logs-to-backuphost.sh 140
A.2 Dynamisches Routing mit OSPF - Datei `/etc/quagga/ospfd.conf` 145
A.3 Dynamisches Routing mit OSPF mit Präferenzen bei den Leitungswegen - Datei `/etc/quagga/ospfd.conf` 150
A.4 Dynamisches Routing mit OSPF mit mehrfacher Redundanz - Routing-Tabellen-Manager - Datei `/etc/quagga/zebra.conf` 151
A.5 Dynamisches Routing mit OSPF mit mehrfacher Redundanz - OSPF-Daemon (*ospfd*) - Datei `/etc/quagga/ospfd.conf` 153
A.6 Datei /usr/local/sbin/router-framework-runtime.sh 156
A.7 Datei /usr/local/lib/router-framework/router-framework-init 157
A.8 Datei /usr/local/lib/router-framework/router-framework-lib 158
A.9 Datei /usr/local/lib/router-framework/router-framework-lib-util 158
A.10 Datei /usr/local/lib/router-framework/router-framework-lib-connectivity 159

A.11 Datei /usr/local/lib/router-framework/router-framework-lib-overlay . . 159
A.12 Datei /usr/local/lib/router-framework/router-framework-lib-vpn 161
A.13 Datei /usr/local/lib/router-framework/router-framework-lib-router . . 164
A.14 Datei /usr/local/lib/router-framework/router-framework-lib-qos 165

Abkürzungsverzeichnis

AS Autonomous System

ASIC Application Specific Integrated Circuit

ASN Autonomous System Number

AES Advanced Encryption Standard

API Application Programming Interface

BGP Border Gateway Protocol

CA Certificate Authority

CDO Content Delivery Overlay

CLI Command Line Interface

DES Data Encryption Standard

DHT Distributed Hash Table

DNS Domain Name System

DSA Digital Signature Algorithm

eBGP external BGP

ECDSA Elliptic Curve Digital Signature Algorithm

ECMP Equal Cost MultiPath

EGP Exterior Gateway Protocol

EIGRP Enhanced Interior Gateway Routing Protocol

ERC European Research Council

Euro-IX European Internet Exchange Association

FSM Finite State Machine

HMAC Hashed Message Authentication Code

HTTP Hypertext Transfer Protocol

IANA Internet Assigned Numbers Authority

IETF Internet Engineering Task Force

iBGP internal BGP

ICMP Internet Control Message Protocol

IGMP Internet Group Management Protocol

IGP Interior Gateway Protocol

IGRP Interior Gateway Routing Protocol

iOS iPhone OS

IOS Internetwork Operating System

IPv4 Internet Protocol Version 4

IPv6 Internet Protocol version 6

IP Internet Protocol

IRC Internet Relay Chat

IS-IS Intermediate-System to Intermediate-System

ISP Internet Service Provider

JRE Java Runtime Environment

JSON JavaScript Object Notation

LAN Local Area Network

L2TP Layer Two Tunneling Protocol

LISP Locator/ID Separation Protocol

LVM Logical Volume Manager

MD5 Message Digest 5

MPLS Multiprotocol Label Switching

NTP Network Time Protocol

OSI Open Systems Interconnection

OSPF Open Shortest Path First

PKI Public-Key-Infrastruktur

PNG Portable Network Graphics

POSIX Portable Operating System Interface

PSK Pre-Shared Key

RFC Request For Comments

RIP Routing Information Protocol

RIR Regional Internet Registry

SCTP Stream Control Transmission Protocol

SHA Secure Hash Algorithm

SNMP Simple Network Management Protocol

SSL Secure Sockets Layer

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

VPN Virtual Private Network

XML Extensible Markup Language

XSLT XSL (Extensible Stylesheet Language) Transformations

Anhang

A Weiterführende Informationen zur Implementierung

A.1 Weiterführende Informationen zur Muster-Appliance als Basissystem

A.1.1 Liste der verfügbaren Pakete des Basissystems

```
root@base:~# dpkg --get-selections | sort | \
  awk '{print $1" "}' | pr -ts' ' -a --column 3 | column -t
acpid                       acpi                            acpi-support-base
adduser                     apt                             aptitude-common
aptitude                    apt-utils                       base-files
base-passwd                 bash                            binutils
bsdmainutils                bsdutils                        busybox
console-setup               console-setup-linux             coreutils
cpio                        cpp-4.6                         cpp-4.7
cpp                         cron                            dash
debconf-i18n                debconf                         debian-archive-keyring
debianutils                 dictionaries-common             diffutils
discover-data               discover                        dkms
dmidecode                   dmsetup                         dpkg
e2fslibs:i386               e2fsprogs                       eject
fakeroot                    findutils                       gcc-4.6-base:i386
gcc-4.6                     gcc-4.7-base:i386               gcc-4.7
gcc                         gettext-base                    gnupg
gpgv                        grep                            groff-base
grub2-common                grub-common                     grub-pc-bin
grub-pc                     gzip                            hostname
iamerican                   ibritish                        ienglish-common
ifupdown                    info                            initramfs-tools
initscripts                 insserv                         installation-report
install-info                iproute                         iptables
iputils-ping                isc-dhcp-client                 isc-dhcp-common
ispell                      kbd                             keyboard-configuration
klibc-utils                 kmod                            krb5-locales
laptop-detect               libacl1:i386                    libapt-inst1.5:i386
libapt-pkg4.12:i386         libasprintf0c2:i386             libattr1:i386
libaudit0                   libblkid1:i386                  libboost-iostreams1.49.0
libbsd0:i386                libbz2-1.0:i386                 libc6-dev:i386
libc6:i386                  libc6-i686:i386                 libc-bin
libc-dev-bin                libcomerr2:i386                 libcwidget3
libdb5.1:i386               libdevmapper1.02.1:i386         libdiscover2
libdrm2:i386                libdrm-intel1:i386              libdrm-nouveau1a:i386
libdrm-radeon1:i386         libedit2:i386                   libept1.4.12
libexpat1:i386              libffi5:i386                    libfontenc1:i386
libfreetype6:i386           libfuse2:i386                   libgcc1:i386
libgcrypt11:i386            libgdbm3:i386                   libgl1-mesa-dri:i386
libgmp10:i386               libgnutls26:i386                libgomp1:i386
libgpg-error0:i386          libgssapi-krb5-2:i386           libice6:i386
```

```
libidn11:i386                libitm1:i386                 libk5crypto3:i386
libkeyutils1:i386            libklibc                     libkmod2:i386
libkrb5-3:i386               libkrb5support0:i386         liblocale-gettext-perl
liblzma5:i386                libmount1                    libmpc2:i386
libmpfr4:i386                libncurses5:i386             libncursesw5:i386
libnewt0.52                  libnfnetlink0                libp11-kit0:i386
libpam0g:i386                libpam-modules-bin           libpam-modules:i386
libpam-runtime               libpci3:i386                 libpciaccess0:i386
libpipeline1:i386            libpixman-1-0:i386           libpopt0:i386
libprocps0:i386              libquadmath0:i386            libreadline6:i386
libselinux1:i386             libsemanage1:i386            libsemanage-common
libsepol1:i386               libsigc++-2.0-0c2a:i386      libslang2:i386
libsm6:i386                  libsqlite3-0:i386            libss2:i386
libssl1.0.0:i386             libstdc++6:i386              libtasn1-3:i386
libtext-charwidth-perl       libtext-iconv-perl           libtext-wrapi18n-perl
libtinfo5:i386               libudev0:i386                libusb-0.1-4:i386
libusb-1.0-0:i386            libustr-1.0-1:i386           libuuid1:i386
libuuid-perl                 libwrap0:i386                libx11-6:i386
libx11-data                  libxapian22                  libxau6:i386
libxaw7:i386                 libxcb1:i386                 libxcomposite1:i386
libxdamage1:i386             libxdmcp6:i386               libxext6:i386
libxfixes3:i386              libxfont1                    libxkbfile1:i386
libxmu6:i386                 libxmuu1:i386                libxpm4:i386
libxrandr2:i386              libxrender1:i386             libxt6:i386
linux-base                   linux-headers-3.2.0-4-686-pae linux-headers-3.2.0-4-common
linux-headers-686-pae        linux-image-3.2.0-4-486      linux-image-486
linux-kbuild-3.2             linux-libc-dev:i386          locales
login                        logrotate                    lsb-base
make                         man-db                       manpages-dev
manpages                     mawk                         menu
module-init-tools            mount                        multiarch-support
nano                         ncurses-base                 ncurses-bin
ncurses-term                 netbase                      netcat-traditional
net-tools                    openssh-blacklist-extra      openssh-blacklist
openssh-client               openssh-server               os-prober
passwd                       patch                        pciutils
perl-base                    procps                       readline-common
rsyslog                      sed                          sensible-utils
sysvinit                     sysvinit-utils               sysv-rc
tar                          task-english                 tasksel-data
tasksel                      task-ssh-server              tcpd
traceroute                   tzdata                       ucf
udev                         usbutils                     util-linux
util-linux-locales           vim-common                   vim-tiny
virtualbox-guest-dkms        virtualbox-guest-utils       virtualbox-guest-x11
virtualbox-ose-guest-x11     wamerican                    wget
whiptail                     x11-common                   x11-xkb-utils
xauth                        xfonts-base                  xfonts-encodings
xfonts-utils                 xkb-data                     xserver-common
xserver-xorg-core            xz-utils                     zlib1g:i386
root@base:~#
```

A.2 Weiterführende Informationen zu spezifischen Anpassungen für Router

A.2.1 Spezifische Betriebssystemkern-Einstellungen für Router

```
root@r1:~# cat /etc/sysctl.conf \
  /etc/sysctl.d/network-security.conf /etc/sysctl.d/router*.conf | \
  grep -v -e "^#" -e "^$" | sort
net.core.optmem_max = 65536
net.core.rmem_default = 262144
net.core.rmem_max = 262144
net.core.wmem_default = 266240
net.core.wmem_max = 8388608
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.arp_announce = 2
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_notify = 1
net.ipv4.conf.all.forwarding = 1
net.ipv4.conf.all.ignore_routes_with_linkdown = 1
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.arp_announce = 2
net.ipv4.conf.default.arp_ignore = 1
net.ipv4.conf.default.arp_notify = 1
net.ipv4.conf.default.forwarding = 1
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.lo.rp_filter = 0
net.ipv4.fib_multipath_hash_policy = 1
net.ipv4.fib_multipath_use_neigh = 1
net.ipv4.icmp_errors_use_inbound_ifaddr = 1
net.ipv4.igmp_max_memberships = 1000
net.ipv4.ip_forward = 1
net.ipv4.neigh.default.base_reachable_time_ms = 14400000
net.ipv4.neigh.default.gc_thresh2 = 7168
net.ipv4.neigh.default.gc_thresh3 = 8192
net.ipv4.neigh.default.mcast_solicit = 10
net.ipv4.tcp_syncookies = 1
net.ipv6.conf.all.ignore_routes_with_linkdown = 1
net.ipv6.conf.all.keep_addr_on_down = 1
net.ipv6.mld_max_msf = 512
net.ipv6.neigh.default.base_reachable_time_ms = 14400000
net.ipv6.neigh.default.gc_thresh2 = 3584
net.ipv6.neigh.default.gc_thresh3 = 4096
net.ipv6.route.max_size = 131072
root@r1:~#

root@r1:~# cat /proc/sys/net/ipv4/ip_forward
1
root@r1:~#
```

A.3 Weiterführende Informationen zur Archivierung relevanter Dateien der Router-Appliance

A.3.1 Datei backup-pull-r1-configs-logs-to-backuphost.sh

Listing A.1: Datei backup-pull-r1-configs-logs-to-backuphost.sh

```
#!/bin/sh

DATE_PATH_FRAGMENT=`/bin/date +%Y-%m-%d-%H%M%S`

ORIGIN_HOST=r1
ORIGIN_HOST_USERNAME=root
ORIGIN_HOST_RSA_ID=${HOME}/.ssh/id_rsa_root-backup-host-for-root-r1
    -backup-pull

DESTINATION_PATH=/backup-ext/routerumgebung/r1
DESTINATION_FILENAME_PREFIX=r1-configs-logs-
DESTINATION_FILENAME_POSTFIX=.tgz

DESTINATION_PATH_FILENAME=${DESTINATION_PATH}/${
    DESTINATION_FILENAME_PREFIX}${DATE_PATH_FRAGMENT}${
    DESTINATION_FILENAME_POSTFIX}

echo "remote tar via ssh to local destination..."
ssh \
  -i ${ORIGIN_HOST_RSA_ID} ${ORIGIN_HOST_USERNAME}@${ORIGIN_HOST} \
  "tar -c -v -p -C / -z -f - \
  /boot/grub/grub.cfg /boot/config-* /boot/initrd.* \
  /boot/System.* /boot/vmlinuz-* /etc/debian_version \
  /etc/apt/sources.list /var/lib/dpkg/status /etc/hostname \
  /etc/motd /etc/fstab /etc/passwd* /etc/shadow* /etc/group* \
  /etc/gshadow* /etc/hosts /etc/networks \
  /etc/network/interfaces /etc/environment /etc/profile \
  /etc/bash.bashrc /etc/ssh /etc/rc.local /etc/rc*.d/*tor \
  /etc/rc*.d/*tor2 /etc/apparmor.d/local/system_tor \
  /etc/profile /etc/timezone /etc/localtime \
  /etc/default/locale /etc/default/ntpdate /etc/quagga \
  /etc/default/openvpn /etc/openvpn/* /etc/tor/torrc* \
  /var/lib/tor/hidden_service_* /var/lib/tor2/hidden_service_* \
  /usr/local/sbin/* /usr/local/bin/* \
  /usr/local/lib/router-framework /root/.profile \
  /root/.bashrc /root/.ssh /root/WORK /root/.screenrc \
  /var/lib/lxc/*/config /root/.bash_history /var/run/utmp \
  /var/run/openvpn-status-*.txt /var/log/btmp /var/log/wtmp \
  /var/log/lastlog /var/log/dmesg /var/log/kern.log \
  /var/log/syslog /var/log/messages /var/log/daemon.log \
  /var/log/user.log /var/log/auth.log /var/log/dpkg.log \
  /var/log/apt/* /var/log/aptitude /var/log/tor/log* \
  /var/log/openvpn-log-*.log /var/log/quagga " | \
dd of=${DESTINATION_PATH_FILENAME}
if [ $? -ne 0 ]
then
        echo "remote tar via ssh to local destination failed" 1>&2
        exit 1
fi
```

A.4 Weiterführende Informationen zur Public-Key-Infrastruktur für die VPNs

A.4.1 Transkript der Zertifizierungsstelle *Router r2 CA*

```
root@ca:/etc/openvpn/easy-rsa# cat >> /etc/openvpn/easy-rsa/vars
export KEY_SIZE=2048
export KEY_COUNTRY="DE"
export KEY_PROVINCE="SL"
export KEY_CITY="Saarbruecken"
export KEY_ORG="Peter Schmidt"
export KEY_EMAIL="peter.schmidt@saar.de"
root@ca:/etc/openvpn/easy-rsa#

root@ca:/etc/openvpn/easy-rsa# mkdir keys
root@ca:/etc/openvpn/easy-rsa# touch keys/index.txt
root@ca:/etc/openvpn/easy-rsa# echo 01 > keys/serial
root@ca:/etc/openvpn/easy-rsa# . ./vars
NOTE: If you run ./clean-all, I will be doing a rm -rf on /etc/openvpn/easy-rsa/keys
root@ca:/etc/openvpn/easy-rsa#

root@ca:/etc/openvpn/easy-rsa# ./build-ca
Generating a 2048 bit RSA private key
............+++
..+++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:
State or Province Name (full name) [SL]:
Locality Name (eg, city) [Saarbruecken]:
Organization Name (eg, company) [Peter Schmidt]:
Organizational Unit Name (eg, section) [changeme]:Routerumgebung
Common Name (eg, your name or your server's hostname) [changeme]:r2
Name [changeme]:Router r2 CA
Email Address [peter.schmidt@saar.de]:
root@ca:/etc/openvpn/easy-rsa#

root@ca:/etc/openvpn/easy-rsa# ./build-key-server server
Generating a 2048 bit RSA private key
.....................................................................+++
.....................................................................+++
writing new private key to 'server.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

```
Country Name (2 letter code) [DE]:
State or Province Name (full name) [SL]:
Locality Name (eg, city) [Saarbruecken]:
Organization Name (eg, company) [Peter Schmidt]:
Organizational Unit Name (eg, section) [changeme]:Routerumgebung
Common Name (eg, your name or your server's hostname) [server]:r2
Name [changeme]:Router r2
Email Address [peter.schmidt@saar.de]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /etc/openvpn/easy-rsa/openssl-1.0.0.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'SL'
localityName          :PRINTABLE:'Saarbruecken'
organizationName      :PRINTABLE:'Peter Schmidt'
organizationalUnitName:PRINTABLE:'Routerumgebung'
commonName            :PRINTABLE:'r2'
name                  :PRINTABLE:'Router r2'
emailAddress          :IA5STRING:'peter.schmidt@saar.de'
Certificate is to be certified until Aug 24 00:43:43 2027 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
root@ca:/etc/openvpn/easy-rsa#

root@ca:/etc/openvpn/easy-rsa# ./build-key r1
Generating a 2048 bit RSA private key
......+++
.....................................................................+++
writing new private key to 'r1.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]:
State or Province Name (full name) [SL]:
Locality Name (eg, city) [Saarbruecken]:
Organization Name (eg, company) [Peter Schmidt]:
Organizational Unit Name (eg, section) [changeme]:Routerumgebung
Common Name (eg, your name or your server's hostname) [r1]:
Name [changeme]:Router r1
Email Address [peter.schmidt@saar.de]:

Please enter the following 'extra' attributes
to be sent with your certificate request
```

```
A challenge password []:
An optional company name []:
Using configuration from /etc/openvpn/easy-rsa/openssl-1.0.0.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'SL'
localityName          :PRINTABLE:'Saarbruecken'
organizationName      :PRINTABLE:'Peter Schmidt'
organizationalUnitName:PRINTABLE:'Routerumgebung'
commonName            :PRINTABLE:'r1'
name                  :PRINTABLE:'Router r1'
emailAddress          :IA5STRING:'peter.schmidt@saar.de'
Certificate is to be certified until Aug 24 00:46:26 2027 GMT (3650 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
root@ca:/etc/openvpn/easy-rsa#
```

A.5 Weiterführende Informationen zu den Muster-Appliances mit VPN-Kopplung

A.5.1 Setup der VPN-Endpunkte mit Code-Einsprungpunkten (*hooks*) für die Laufzeitumgebung

```
root@r1:~# cat /etc/openvpn/r1-r2-tcp-1194-tap2-client.conf | \
  grep -v -e "^#" -e "^$"
tls-client
nobind
resolv-retry infinite
remote 192.109.53.36 1194
proto tcp-client
socket-flags TCP_NODELAY
mtu-disc maybe
tun-mtu 1500
mssfix 1400
dev tap2
ca   /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/r1.crt
key  /etc/openvpn/easy-rsa/keys/r1.key
ns-cert-type server
remote-cert-tls server
tls-auth /etc/openvpn/easy-rsa/keys/ta.key 1
reneg-sec 3600
ping 5
ping-restart 45
txqueuelen 64
sndbuf 393216
rcvbuf 393216
user nobody
group nogroup
```

```
persist-key
persist-tun
up   "/etc/openvpn/r1-r2-tcp-1194-tap2-client-up.sh br2"
down "/etc/openvpn/r1-r2-tcp-1194-tap2-client-down.sh br2"
script-security 3
log-append /var/log/openvpn-log-r1-r2-tcp-1194-tap2-client.log
status     /var/run/openvpn-status-r1-r2-tcp-1194-tap2-client.txt
verb 3  # verbose mode
root@r1:~#


root@r2:~# cat /etc/openvpn/r1-r2-tcp-1194-tap2-server.conf | \
  grep -v -e "^#" -e "^$"
tls-server
local 192.109.53.36
port 1194
proto tcp-server
socket-flags TCP_NODELAY
mtu-disc maybe
tun-mtu 1500
mssfix 1400
dev tap2
ca      /etc/openvpn/easy-rsa/keys/ca.crt
cert    /etc/openvpn/easy-rsa/keys/server.crt
key     /etc/openvpn/easy-rsa/keys/server.key
dh      /etc/openvpn/easy-rsa/keys/dh2048.pem
tls-auth /etc/openvpn/easy-rsa/keys/ta.key 0
reneg-sec 3600
ping 5
ping-restart 45
txqueuelen 64
sndbuf 393216
rcvbuf 393216
user nobody
group nogroup
persist-key
persist-tun
up   "/etc/openvpn/r1-r2-tcp-1194-tap2-server-up.sh br2"
down "/etc/openvpn/r1-r2-tcp-1194-tap2-server-down.sh br2"
script-security 3
log-append /var/log/openvpn-log-r1-r2-tcp-1194-tap2-server.log
status     /var/run/openvpn-status-r1-r2-tcp-1194-tap2-server.txt
verb 3  # verbose mode
root@r2:~#
```

A.6 Weiterführende Informationen zu den Muster-Appliances mit dynamischem Routing

A.6.1 Muster-Appliances mit dynamischem Routing mit OSPF

Listing A.2: Dynamisches Routing mit OSPF - Datei `/etc/quagga/ospfd.conf`

```
hostname r1
password routerumgebung-htw
enable password routerumgebung-htw
log file /var/log/quagga/ospfd.log
!
interface lo
!
interface eth0
!
interface br1
!
interface tap2
!
interface br2
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 2
 ip ospf dead-interval 5
 ip ospf cost 50
!
interface tap10
!
interface br10
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 2
 ip ospf dead-interval 5
 ip ospf cost 50
!
!
router ospf
 ospf router-id 10.10.11.1
 redistribute connected route-map route-map-connected-ospf
 redistribute static
 passive-interface default
 no passive-interface br2
 no passive-interface br10
 network 10.10.2.0/24 area 0.0.0.0
 network 10.10.10.0/24 area 0.0.0.0
 area 0.0.0.0 authentication message-digest
!
!
ip prefix-list host seq 10 permit 10.10.11.0/24
!
route-map route-map-connected-ospf permit 10
match ip address prefix-list host
!
!
line vty
```

A.7 Weiterführende Informationen zu den Muster-Appliances als Overlay-Netzwerkknoten

A.7.1 Auszug aus den Implementierungsdetails für die Funktion des Overlay-Netzwerkknotens

```
root@r1:~# cat /etc/tor/torrc | grep -v -e "^#" -e "^$"
ConnLimit 128
KeepalivePeriod 55
LearnCircuitBuildTimeout 0
CircuitBuildTimeout 45
CircuitStreamTimeout 45
SocksTimeout 60
SocksPort 127.0.0.1:9050 IsolateClientProtocol IsolateDestPort PreferSOCKSNoAuth
SocksPolicy accept 127.0.0.0/8
SocksPolicy accept 134.96.216.137/32
SocksPolicy reject *
DataDirectory /var/lib/tor
Log notice file /var/log/tor/log
HiddenServiceDir /var/lib/tor/hidden_service_10022/
HiddenServicePort 10022 127.0.0.1:22
PidFile /var/run/tor/tor.pid
User debian-tor
ControlSocket /var/run/tor/control
ControlSocketsGroupWritable 1
CookieAuthentication 1
CookieAuthFileGroupReadable 1
CookieAuthFile /var/run/tor/control.authcookie
OutboundBindAddress 134.96.216.137
root@r1:~#

root@r1:~# cat /etc/tor/torrc | grep -e "^SocksPolicy"
SocksPolicy accept 127.0.0.0/8
SocksPolicy accept 134.96.216.137/32
SocksPolicy reject *
root@r1:~#

root@r1:~# cat /etc/tor/torrc | grep -e "^HiddenService"
HiddenServiceDir /var/lib/tor/hidden_service_10022/
HiddenServicePort 10022 127.0.0.1:22
root@r1:~#

root@r1:~# cat /var/lib/tor/hidden_service_10022/hostname
ryqoqnzfmomxora2.onion
root@r1:~#
```

A.7.2 Auszug aus den Implementierungsdetails für die zweite Instanz des Overlay-Netzwerkknotens

```
root@r1:~# cat /etc/tor/torrc2 | grep -v -e "^#" -e "^$"
ConnLimit 128
KeepalivePeriod 55
LearnCircuitBuildTimeout 0
CircuitBuildTimeout 45
CircuitStreamTimeout 45
SocksTimeout 60
SocksPort 127.0.0.1:9051 IsolateClientProtocol IsolateDestPort PreferSOCKSNoAuth
SocksPolicy accept 127.0.0.0/8
SocksPolicy accept 134.96.216.137/32
SocksPolicy reject *
DataDirectory /var/lib/tor2
Log notice file /var/log/tor/log2
HiddenServiceDir /var/lib/tor2/hidden_service_10022/
HiddenServicePort 10022 127.0.0.1:22
PidFile /var/run/tor/tor2.pid
User debian-tor
ControlSocket /var/run/tor/control2
ControlSocketsGroupWritable 1
CookieAuthentication 1
CookieAuthFileGroupReadable 1
CookieAuthFile /var/run/tor/control2.authcookie
OutboundBindAddress 134.96.216.137

root@r1:~# cat /var/lib/tor2/hidden_service_10022/hostname
2vvfivuimbvnjav5.onion
root@r1:~#

root@r1:~# cat /etc/apparmor.d/local/system_tor | grep -v -e "^#" -e "^$"
owner /var/lib/tor2/** rwk,
/{,var/}run/tor/tor2.pid w,
/{,var/}run/tor/control2 w,
/{,var/}run/tor/control2.authcookie w,
/{,var/}run/tor/control2.authcookie.tmp rw,
root@r1:~#

root@r1:~# diff --left-column /etc/init.d/tor2 /etc/init.d/tor | grep -e "^<"
< # Provides:          tor2
< DESC="tor daemon custom node 2"
< TORPID=$TORPIDDIR/tor2.pid
< ARGS="-f /etc/tor/torrc2"
root@r1:~#

root@r1:~# ls -n /etc/rc?.d/*tor2
lrwxrwxrwx 1 0 0 14 Aug 28 19:04 /etc/rc0.d/K01tor2 -> ../init.d/tor2
lrwxrwxrwx 1 0 0 14 Aug 28 19:04 /etc/rc1.d/K01tor2 -> ../init.d/tor2
lrwxrwxrwx 1 0 0 14 Aug 28 19:05 /etc/rc2.d/S02tor2 -> ../init.d/tor2
lrwxrwxrwx 1 0 0 14 Aug 28 19:05 /etc/rc3.d/S02tor2 -> ../init.d/tor2
lrwxrwxrwx 1 0 0 14 Aug 28 19:05 /etc/rc4.d/S02tor2 -> ../init.d/tor2
lrwxrwxrwx 1 0 0 14 Aug 28 19:05 /etc/rc5.d/S02tor2 -> ../init.d/tor2
lrwxrwxrwx 1 0 0 14 Aug 28 19:04 /etc/rc6.d/K01tor2 -> ../init.d/tor2
root@r1:~#
```

A.8 Weiterführende Informationen zur Integration in eine Plattform

A.8.1 Auszug aus den Implementierungsdetails der Integration eines Overlay-Netzwerkknotens mit einem darauf aufbauenden VPN-Tunnel

```
root@r2:~# cat /etc/tor/torrc | grep -e "^SocksPort" -e "^HiddenService"
SocksPort 127.0.0.1:9050 IsolateClientProtocol IsolateDestPort PreferSOCKSNoAuth
HiddenServiceDir /var/lib/tor/hidden_service_11194/
HiddenServicePort 11194 127.0.0.1:1194
root@r2:~# cat /var/lib/tor/hidden_service_11194/hostname
y4tha7ojk5bqrtxz.onion
root@r2:~#

root@r2:~# cat /etc/openvpn/r1-r2-tcp-1194-onion-11194-tap4-server.conf | \
  grep -v -e "^#" -e "^$"
tls-server
local 127.0.0.1
port 1194
proto tcp-server
socket-flags TCP_NODELAY
mtu-disc no
tun-mtu 1500
mssfix 1400
dev tap4
ca      /etc/openvpn/easy-rsa/keys/ca.crt
cert    /etc/openvpn/easy-rsa/keys/server.crt
key     /etc/openvpn/easy-rsa/keys/server.key
dh      /etc/openvpn/easy-rsa/keys/dh2048.pem
tls-auth /etc/openvpn/easy-rsa/keys/ta.key 0
reneg-sec 3600
ping 5
ping-restart 45
txqueuelen 64
sndbuf 65536
rcvbuf 65536
user nobody
group nogroup
persist-key
persist-tun
up   "/etc/openvpn/r1-r2-tcp-1194-onion-11194-tap4-server-up.sh br4"
down "/etc/openvpn/r1-r2-tcp-1194-onion-11194-tap4-server-down.sh br4"
script-security 3
log-append /var/log/openvpn-log-r1-r2-tcp-1194-onion-11194-tap4-server.log
status     /var/run/openvpn-status-r1-r2-tcp-1194-onion-11194-tap4-server.txt
verb 3  # verbose mode
root@r2:~#



root@r1:~# cat /etc/tor/torrc | grep -e "^SocksPort"
SocksPort 127.0.0.1:9050 IsolateClientProtocol IsolateDestPort PreferSOCKSNoAuth
root@r1:~#

root@r1:~# cat /etc/openvpn/r1-r2-tcp-1194-onion-11194-tap4-client.conf | \
  grep -v -e "^#" -e "^$"
tls-client
```

```
nobind
socks-proxy 127.0.0.1 9050
socks-proxy-retry
resolv-retry infinite
remote y4tha7ojk5bqrtxz.onion 11194
proto tcp-client
socket-flags TCP_NODELAY
mtu-disc no
tun-mtu 1500
mssfix 1400
dev tap4
ca   /etc/openvpn/easy-rsa/keys/ca.crt
cert /etc/openvpn/easy-rsa/keys/r1.crt
key  /etc/openvpn/easy-rsa/keys/r1.key
ns-cert-type server
remote-cert-tls server
tls-auth /etc/openvpn/easy-rsa/keys/ta.key 1
reneg-sec 3600
ping 5
ping-restart 45
txqueuelen 64
sndbuf 16384
rcvbuf 16384
user nobody
group nogroup
persist-key
persist-tun
up   "/etc/openvpn/r1-r2-tcp-1194-onion-11194-tap4-client-up.sh br4"
down "/etc/openvpn/r1-r2-tcp-1194-onion-11194-tap4-client-down.sh br4"
script-security 3
log-append /var/log/openvpn-log-r1-r2-tcp-1194-onion-11194-tap4-client.log
status     /var/run/openvpn-status-r1-r2-tcp-1194-onion-11194-tap4-client.txt
verb 3  # verbose mode
root@r1:~#
```

A.8.2 Integration mit den Komponenten für das dynamische Routing mit OSPF

Listing A.3: Dynamisches Routing mit OSPF mit Präferenzen bei den Leitungswegen - Datei /etc/quagga/ospfd.conf

```
hostname r1
password routerumgebung-htw
enable password routerumgebung-htw
log file /var/log/quagga/ospfd.log
!
interface lo
!
interface eth0
!
interface br1
!
interface tap2
!
interface br2
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 2
 ip ospf dead-interval 5
 ip ospf cost 50
!
interface tap4
!
interface br4
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 5
 ip ospf dead-interval 10
 ip ospf cost 250
!
!
router ospf
 ospf router-id 10.10.11.1
 redistribute connected route-map route-map-connected-ospf
 redistribute static
 passive-interface default
 no passive-interface br2
 no passive-interface br4
 network 10.10.2.0/24 area 0.0.0.0
 network 10.10.4.0/24 area 0.0.0.0
 area 0.0.0.0 authentication message-digest
!
!
ip prefix-list host seq 10 permit 10.10.11.0/24
!
route-map route-map-connected-ospf permit 10
match ip address prefix-list host
!
!
line vty
```

A.8.3 Dynamisches Routing mit OSPF mit mehrfacher Redundanz

Listing A.4: Dynamisches Routing mit OSPF mit mehrfacher Redundanz - Routing-Tabellen-Manager - Datei `/etc/quagga/zebra.conf`

```
hostname r1
password routerumgebung-htw
enable password routerumgebung-htw
log file /var/log/quagga/zebra.log
!
interface lo
 description loopback
!
interface eth0
 description eth0
 ipv6 nd suppress-ra
!
interface br1
 description lan1
 link-detect
 ip address 10.10.11.1/24
 multicast
 ipv6 nd suppress-ra
!
interface tap2
 ipv6 nd suppress-ra
!
interface br2
 description net-trans-vpn-2
 link-detect
 ip address 10.10.2.1/24
 multicast
 ipv6 nd suppress-ra
!
interface tap4
 ipv6 nd suppress-ra
!
interface br4
 description net-trans-vpn-4
 link-detect
 ip address 10.10.4.1/24
 multicast
 ipv6 nd suppress-ra
!
interface tap5
 ipv6 nd suppress-ra
!
interface br5
 description net-trans-vpn-5
 link-detect
 ip address 10.10.5.1/24
 multicast
 ipv6 nd suppress-ra
!
interface tap6
 ipv6 nd suppress-ra
!
```

```
interface br6
 description net-trans-vpn-6
 link-detect
 ip address 10.10.6.1/24
 multicast
 ipv6 nd suppress-ra
!
interface tap7
 ipv6 nd suppress-ra
!
interface br7
 description net-trans-vpn-7
 link-detect
 ip address 10.10.7.1/24
 multicast
 ipv6 nd suppress-ra
!
interface tap8
 ipv6 nd suppress-ra
!
interface br8
 description net-trans-vpn-8
 link-detect
 ip address 10.10.8.1/24
 multicast
 ipv6 nd suppress-ra
!
interface tap9
 ipv6 nd suppress-ra
!
interface br9
 description net-trans-vpn-9
 link-detect
 ip address 10.10.9.1/24
 multicast
 ipv6 nd suppress-ra
!
interface tap10
 ipv6 nd suppress-ra
!
interface br10
 description net-trans-vpn-10
 link-detect
 ip address 10.10.10.1/24
 multicast
 ipv6 nd suppress-ra
!
!
ip forwarding
!
access-list vtylist permit 127.0.0.1/32
access-list vtylist deny any
!
!
line vty
 access-class vtylist
```

Listing A.5: Dynamisches Routing mit OSPF mit mehrfacher Redundanz - OSPF-Daemon (*ospfd*) - Datei `/etc/quagga/ospfd.conf`

```
hostname r1
password routerumgebung-htw
enable password routerumgebung-htw
log file /var/log/quagga/ospfd.log
!
interface lo
!
interface eth0
!
interface br1
!
interface tap2
!
interface br2
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 2
 ip ospf dead-interval 5
 ip ospf cost 50
!
interface tap4
!
interface br4
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 5
 ip ospf dead-interval 10
 ip ospf cost 250
!
interface tap5
!
interface br5
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 2
 ip ospf dead-interval 5
 ip ospf cost 250
!
interface tap6
!
interface br6
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 2
 ip ospf dead-interval 5
 ip ospf cost 250
!
interface tap7
!
interface br7
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 2
 ip ospf dead-interval 5
```

```
 ip ospf cost 250
!
interface tap8
!
interface br8
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 2
 ip ospf dead-interval 5
 ip ospf cost 250
!
interface tap9
!
interface br9
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 2
 ip ospf dead-interval 5
 ip ospf cost 250
!
interface tap10
!
interface br10
 ip ospf authentication message-digest
 ip ospf message-digest-key 1 md5 routerumgebung-h
 ip ospf hello-interval 2
 ip ospf dead-interval 5
 ip ospf cost 50
!
!
router ospf
 ospf router-id 10.10.11.1
 redistribute connected route-map route-map-connected-ospf
 redistribute static
 passive-interface default
 no passive-interface br2
 no passive-interface br4
 no passive-interface br5
 no passive-interface br6
 no passive-interface br7
 no passive-interface br8
 no passive-interface br9
 no passive-interface br10
 network 10.10.2.0/24 area 0.0.0.0
 network 10.10.4.0/24 area 0.0.0.0
 network 10.10.5.0/24 area 0.0.0.0
 network 10.10.6.0/24 area 0.0.0.0
 network 10.10.7.0/24 area 0.0.0.0
 network 10.10.8.0/24 area 0.0.0.0
 network 10.10.9.0/24 area 0.0.0.0
 network 10.10.10.0/24 area 0.0.0.0
 area 0.0.0.0 authentication message-digest
!
!
ip prefix-list host seq 10 permit 10.10.11.0/24
!
route-map route-map-connected-ospf permit 10
```

```
match ip address prefix-list host
!
!
line vty
```

A.9 Weiterführende Informationen und Code zur Laufzeitumgebung der Router-Appliance

A.9.1 Datei /usr/local/sbin/router-framework-runtime.sh

Listing A.6: Datei /usr/local/sbin/router-framework-runtime.sh

```
#!/bin/sh

################################################
# Routerumgebung Framework                     #
# Peter Schmidt, peter.schmidt@saar.de, 2017-09 #
################################################

#
# router-framework-runtime.sh
#

# router framework runtime settings
REMOTE_ROUTER_ID="r2"
REMOTE_ROUTER_LAN_INTERFACE="br1"

NATIVE_VPN_TUNNEL_ID_LIST="br2 br10"
NATIVE_VPN_TUNNEL_CONF_LIST="r1-r2-tcp-1194-tap2-client r1-r2-tcp-1195-tap10-client"
OVERLAY_VPN_TUNNEL_ID_LIST="br4 br5 br6 br7 br8 br9"
OVERLAY_VPN_TUNNEL_CONF_LIST="r1-r2-tcp-1194-onion-11194-tap4-client \
  r1-r2-tcp-1195-onion-11195-tap5-client r2-r1-tcp-1196-onion-11196-tap6-server \
  r2-r1-tcp-1197-onion-11197-tap7-server r1-r2-tcp-1198-onion-n2-11198-tap8-client \
  r2-r1-tcp-1199-onion-n2-11199-tap9-server"

# router framework runtime
export ROUTER_FRAMEWORK_BASENAME=router-framework
export ROUTER_FRAMEWORK_BASENAME_PREFIX=${ROUTER_FRAMEWORK_BASENAME}-
export ROUTER_FRAMEWORK_LIBPATH=/usr/local/lib/${ROUTER_FRAMEWORK_BASENAME}
export ROUTER_FRAMEWORK_LIBPATH_PREFIX=${ROUTER_FRAMEWORK_LIBPATH}/
export RFPR=${ROUTER_FRAMEWORK_LIBPATH_PREFIX}${ROUTER_FRAMEWORK_BASENAME_PREFIX}

export ROUTER_FRAMEWORK_VERBOSE="yes"


# router framework runtime library functions
. ${RFPR}lib

# router framework runtime init
. ${RFPR}init


do_init_qos_traffic_shaping


# router framework runtime event loop
while [ true ]; do
  do_check_internet_connectivity

  do_check_remote_router_external

  do_check_native_vpn_tunnels
```

```
  do_check_overlay_nodes
  check_overlay_nodes=$?
  if [ $check_overlay_nodes -ne 0 ]; then
    do_restart_overlay_nodes
  fi

  do_check_overlay_vpn_tunnels

  do_check_remote_router_lan

  do_adjust_qos_traffic_shaping

  do_wait
done
```

A.9.2 Datei /usr/local/lib/router-framework/router-framework-init

Listing A.7: Datei /usr/local/lib/router-framework/router-framework-init

```
#!/bin/sh

#
# router-framework-init
#

# environment
export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin

# locales
unset LANG
unset LC_ADDRESS
unset LC_ALL
unset LC_CTYPE
unset LC_IDENTIFICATION
unset LC_MEASUREMENT
unset LC_MONETARY
unset LC_NAME
unset LC_NUMERIC
unset LC_PAPER
unset LC_TELEPHONE
unset LC_TIME
export LANG=C

# defaults
ROUTER_FRAMEWORK_WAIT_TIME=1

ROUTER_FRAMEWORK_DELAY_AFTER_SERVICE_RESTART_TIME=30
```

A.9.3 Datei /usr/local/lib/router-framework/router-framework-lib

Listing A.8: Datei /usr/local/lib/router-framework/router-framework-lib

```
#!/bin/sh

#
# router-framework-lib
#

# router framework library functions

# router framework library functions - util
. ${RFPR}lib-util

# router framework library functions - connectivity
. ${RFPR}lib-connectivity

# router framework library functions - overlay
. ${RFPR}lib-overlay

# router framework library functions - vpn
. ${RFPR}lib-vpn

# router framework library functions - qos
. ${RFPR}lib-qos

# router framework library functions - router
. ${RFPR}lib-router
```

A.9.4 Datei /usr/local/lib/router-framework/router-framework-lib-util

Listing A.9: Datei /usr/local/lib/router-framework/router-framework-lib-util

```
#!/bin/sh

#
# router-framework-lib-util
#

# router framework library functions - util

do_wait() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_wait() ${ROUTER_FRAMEWORK_WAIT_TIME}s"

  sleep ${ROUTER_FRAMEWORK_WAIT_TIME}

  return 0
}

do_delay_after_service_restart() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_delay_after_service_restart() \
          ${ROUTER_FRAMEWORK_DELAY_AFTER_SERVICE_RESTART_TIME}s"
```

```
  sleep ${ROUTER_FRAMEWORK_DELAY_AFTER_SERVICE_RESTART_TIME}

  return 0
}
```

A.9.5 Datei /usr/local/lib/router-framework/router-framework-lib-connectivity

Listing A.10: Datei /usr/local/lib/router-framework/router-framework-lib-connectivity

```
#!/bin/sh

#
# router-framework-lib-connectivity
#

# router framework library functions - connectivity

do_check_internet_connectivity() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_check_internet_connectivity()"

  nc -w 1 -z 8.8.8.8 53 >/dev/null 2>&1
  nc_dns_google_online=$?
  if [ $nc_dns_google_online -eq 0 ]; then
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_internet_connectivity() online"
    return 0
  else
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_internet_connectivity() offline"
    return 1
  fi

  return 0
}
```

A.9.6 Datei /usr/local/lib/router-framework/router-framework-lib-overlay

Listing A.11: Datei /usr/local/lib/router-framework/router-framework-lib-overlay

```
#!/bin/sh

#
# router-framework-lib-overlay
#

# router framework library functions - overlay

do_restart_overlay_nodes() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
```

```
    echo "do_restart_overlay_nodes()"

  /etc/init.d/tor stop
  /etc/init.d/tor2 stop
  /etc/init.d/tor start
  /etc/init.d/tor2 start

  do_delay_after_service_restart

  return 0
}

do_check_overlay_nodes() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_check_overlay_nodes()"

  check_overlay_nodes_count=0

  do_check_overlay_node1
  check_overlay_node=$?
  if [ $check_overlay_node -eq 0 ]; then
    check_overlay_nodes_count=‘expr \
      "$check_overlay_nodes_count" + "1"‘
  fi

  do_check_overlay_node2
  check_overlay_node=$?
  if [ $check_overlay_node -eq 0 ]; then
    check_overlay_nodes_count=‘expr \
      "$check_overlay_nodes_count" + "1"‘
  fi

  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo -n "do_check_overlay_nodes() " ; \
      echo "overlay nodes count $check_overlay_nodes_count"

  if [ $check_overlay_nodes_count -eq 0 ]; then
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_overlay_nodes() overlay nodes offline"
    return 1
  fi

  return 0
}

do_check_overlay_node1() {
  overlay_node1_hostname=‘cat /var/lib/tor/hidden_service_10022/hostname‘

  nc \
    -w 1 -z -X5 -x localhost:9050 \
    ${overlay_node1_hostname} 10022 >/dev/null 2>&1
  nc_ssh_overlay_node1_online=$?
  if [ $nc_ssh_overlay_node1_online -eq 0 ]; then
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_overlay_node1() online"
    return 0
  else
```

```
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_overlay_node1() offline"
    return 1
  fi

  return 0
}

do_check_overlay_node2() {
  overlay_node2_hostname=‘cat /var/lib/tor2/hidden_service_10022/hostname‘

  nc \
    -w 1 -z -X5 -x localhost:9051 \
    ${overlay_node2_hostname} 10022 >/dev/null 2>&1
  nc_ssh_overlay_node2_online=$?
  if [ $nc_ssh_overlay_node2_online -eq 0 ]; then
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_overlay_node2() online"
    return 0
  else
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_overlay_node2() offline"
    return 1
  fi

  return 0
}
```

A.9.7 Datei /usr/local/lib/router-framework/router-framework-lib-vpn

Listing A.12: Datei /usr/local/lib/router-framework/router-framework-lib-vpn

```
#!/bin/sh

#
# router-framework-lib-vpn
#

# router framework library functions - vpn

do_stop_native_vpn_tunnels() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_stop_native_vpn_tunnels()"

  native_vpn_tunnel_conf_list=${NATIVE_VPN_TUNNEL_CONF_LIST}

  for native_vpn_tunnel_conf in ${native_vpn_tunnel_conf_list} ;
  do
    /etc/init.d/openvpn stop ${native_vpn_tunnel_conf}
  done

  return 0
}

do_start_native_vpn_tunnels() {
```

```
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_start_native_vpn_tunnels()"

  native_vpn_tunnel_conf_list=${NATIVE_VPN_TUNNEL_CONF_LIST}

  for native_vpn_tunnel_conf in ${native_vpn_tunnel_conf_list} ;
  do
    /etc/init.d/openvpn start ${native_vpn_tunnel_conf}
  done

  return 0
}

do_check_native_vpn_tunnels() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_check_native_vpn_tunnels()"

  check_native_vpn_tunnels_count=0

  remote_router_id=${REMOTE_ROUTER_ID}
  native_vpn_tunnel_id_list=${NATIVE_VPN_TUNNEL_ID_LIST}

  for native_vpn_tunnel_id in ${native_vpn_tunnel_id_list} ;
  do
    do_check_native_vpn_tunnel \
      ${remote_router_id} ${native_vpn_tunnel_id}
    check_native_vpn_tunnel=$?
    if [ $check_native_vpn_tunnel -eq 0 ]; then
      check_native_vpn_tunnels_count=‘expr \
        "$check_native_vpn_tunnels_count" + "1"‘
    fi
  done

  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo -n "do_check_native_vpn_tunnels() " ; \
      echo "native vpn tunnels count $check_native_vpn_tunnels_count"

  if [ $check_native_vpn_tunnels_count -eq 0 ]; then
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_native_vpn_tunnels() native vpn tunnels offline"
    return 1
  fi

  return 0
}

do_check_native_vpn_tunnel() {
  remote_router_id=${1}
  native_vpn_tunnel_id=${2}

  native_vpn_tunnel_peer="${remote_router_id}-${native_vpn_tunnel_id}"

  ping -c1 -w1 -q ${native_vpn_tunnel_peer} > /dev/null
  ping_native_vpn_tunnel_peer=$?
  if [ $ping_native_vpn_tunnel_peer -eq 0 ]; then
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_native_vpn_tunnel() peer ${native_vpn_tunnel_peer} online"
```

```
    return 0
  else
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_native_vpn_tunnel() peer ${native_vpn_tunnel_peer} offline"
    return 1
  fi

  return 0
}

do_stop_overlay_vpn_tunnels() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_stop_overlay_vpn_tunnels()"

  overlay_vpn_tunnel_conf_list=${OVERLAY_VPN_TUNNEL_CONF_LIST}

  for overlay_vpn_tunnel_conf in ${overlay_vpn_tunnel_conf_list} ;
  do
    /etc/init.d/openvpn stop ${overlay_vpn_tunnel_conf}
  done

  return 0
}

do_start_overlay_vpn_tunnels() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_start_overlay_vpn_tunnels()"

  overlay_vpn_tunnel_conf_list=${OVERLAY_VPN_TUNNEL_CONF_LIST}

  for overlay_vpn_tunnel_conf in ${overlay_vpn_tunnel_conf_list} ;
  do
    /etc/init.d/openvpn start ${overlay_vpn_tunnel_conf}
  done

  return 0
}

do_check_overlay_vpn_tunnels() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_check_overlay_vpn_tunnels()"

  check_overlay_vpn_tunnels_count=0

  remote_router_id=${REMOTE_ROUTER_ID}
  overlay_vpn_tunnel_id_list=${OVERLAY_VPN_TUNNEL_ID_LIST}

  for overlay_vpn_tunnel_id in ${overlay_vpn_tunnel_id_list} ;
  do
    do_check_overlay_vpn_tunnel \
      ${remote_router_id} ${overlay_vpn_tunnel_id}
    check_overlay_vpn_tunnel=$?
    if [ $check_overlay_vpn_tunnel -eq 0 ]; then
      check_overlay_vpn_tunnels_count=‘expr \
        "$check_overlay_vpn_tunnels_count" + "1"‘
    fi
  done
```

```
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo -n "do_check_overlay_vpn_tunnels() " ; \
      echo "overlay vpn tunnels count $check_overlay_vpn_tunnels_count"

  if [ $check_overlay_vpn_tunnels_count -eq 0 ]; then
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_overlay_vpn_tunnels() overlay vpn tunnels offline"
    return 1
  fi

  return 0
}

do_check_overlay_vpn_tunnel() {
  remote_router_id=${1}
  overlay_vpn_tunnel_id=${2}

  overlay_vpn_tunnel_peer="${remote_router_id}-${overlay_vpn_tunnel_id}"

  ping -c1 -w1 -q ${overlay_vpn_tunnel_peer} > /dev/null
  ping_overlay_vpn_tunnel_peer=$?
  if [ $ping_overlay_vpn_tunnel_peer -eq 0 ]; then
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_overlay_vpn_tunnel() peer ${overlay_vpn_tunnel_peer} online"
    return 0
  else
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_overlay_vpn_tunnel() peer ${overlay_vpn_tunnel_peer} offline"
    return 1
  fi

  return 0
}
```

A.9.8 Datei /usr/local/lib/router-framework/router-framework-lib-router

Listing A.13: Datei /usr/local/lib/router-framework/router-framework-lib-router

```
#!/bin/sh

#
# router-framework-lib-router
#

# router framework library functions - router

do_check_remote_router_external() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_check_remote_router_external()"

  r_router_id=${REMOTE_ROUTER_ID}
  r_router_external_postfix="ext"

  r_router_external_interface_adr="${r_router_id}-${r_router_external_postfix}"
```

```
  ping -c1 -w1 -q ${r_router_external_interface_adr} > /dev/null
  ping_r_router_external_interface_adr=$?
  if [ $ping_r_router_external_interface_adr -eq 0 ]; then
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_remote_router_external() remote router ${r_router_id} ext-if online"
    return 0
  else
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_remote_router_external() remote router ${r_router_id} ext-if offline"
    return 1
  fi

  return 0
}

do_check_remote_router_lan() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_check_remote_router_lan()"

  r_router_id=${REMOTE_ROUTER_ID}
  r_router_lan_interface=${REMOTE_ROUTER_LAN_INTERFACE}

  r_router_lan_interface_peer="${r_router_id}-${r_router_lan_interface}"

  ping -c1 -w1 -q ${r_router_lan_interface_peer} > /dev/null
  ping_r_router_lan_interface_peer=$?
  if [ $ping_r_router_lan_interface_peer -eq 0 ]; then
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_remote_router_lan() remote router ${r_router_id} lan online"
    return 0
  else
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_check_remote_router_lan() remote router ${r_router_id} lan offline"
    return 1
  fi

  return 0
}
```

A.9.9 Datei /usr/local/lib/router-framework/router-framework-lib-qos

Listing A.14: Datei /usr/local/lib/router-framework/router-framework-lib-qos

```
#!/bin/sh

#
# router-framework-lib-qos
#

# router framework library functions - qos

TC_DOWNSTREAM=10000
TC_UPSTREAM=5000
```

```
do_init_qos_traffic_shaping() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_init_qos_traffic_shaping()"


    # stop traffic shaping
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_init_qos_traffic_shaping() stop traffic shaping"

    overlay_vpn_tunnel_id_list=${OVERLAY_VPN_TUNNEL_ID_LIST}

    for overlay_vpn_tunnel_id in ${overlay_vpn_tunnel_id_list} ;
    do
      [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
        echo "/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh \
          stop ${overlay_vpn_tunnel_id}"

      /usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh \
          stop ${overlay_vpn_tunnel_id}
    done


    # start traffic shaping
    [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
      echo "do_init_qos_traffic_shaping() start traffic shaping"

    overlay_vpn_tunnel_id_list=${OVERLAY_VPN_TUNNEL_ID_LIST}

    for overlay_vpn_tunnel_id in ${overlay_vpn_tunnel_id_list} ;
    do
      [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
        echo "/usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh \
          start ${overlay_vpn_tunnel_id} ${TC_DOWNSTREAM} ${TC_UPSTREAM}"

      /usr/local/lib/router-framework/router-framework-lib-qos-tc-shaper.sh \
          start ${overlay_vpn_tunnel_id} ${TC_DOWNSTREAM} ${TC_UPSTREAM}
    done


  return 0
}

do_adjust_qos_traffic_shaping() {
  [ "$ROUTER_FRAMEWORK_VERBOSE" != no ] && \
    echo "do_adjust_qos_traffic_shaping()"

  #
  #function stub for custom bandwidth management
  #
  #QoS, traffic shaping of delicate data such as
  #storage traffic or dedicated business applications
  #

  return 0
}
```

B Weiterführende Informationen zur Evaluation

B.1 Weiterführende Informationen zur Erfassung der Messdaten

B.1.1 Auszug aus dem Setup zur Erfassung der Messdaten für die Indikatoren *Verfügbarkeit* und *Latenz* mit der Software *MRTG*

```
root@lan1-11-10:~# cat /etc/mrtg.cfg | grep -v -e "^#" -e "^$"
WriteExpires: Yes
WorkDir: /var/www/mrtg
Interval: 5
Title[ping]: Round Trip Time Analysis - lan1-11-10 -- lan2-12-10
PageTop[ping]: <H1>Round Trip Time Analysis - lan1-11-10 -- lan2-12-10</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>WS lan1-11-10 via Router r1</TD></TR>
   <TR><TD>Remote System:</TD> <TD>WS lan2-12-10 via Router r2</TD></TR>
 </TABLE>
MaxBytes[ping]: 1000
Target[ping]: `/usr/bin/mrtg-ping-probe -s -o "-w 30" -t 30 10.10.12.10`
YLegend[ping]: Round Trip Time
ShortLegend[ping]: ms
Legend1[ping]: Maximum Round Trip Time in ms
Legend2[ping]: Minimum Round Trip Time in ms
Legend3[ping]: Maximal 5 Minute Maximum Round Trip Time in ms
Legend4[ping]: Maximal 5 Minute Minimum Round Trip Time in ms
LegendI[ping]:  Max:
LegendO[ping]:  Min:
WithPeak[ping]: ymwd
Options[ping]: nopercent, integer, transparent, nobanner, gauge
root@lan1-11-10:~#


root@lan1-11-10:~# cat /var/spool/cron/crontabs/root | grep -v -e "^#" -e "^$"
*/5 * * * *  /usr/local/sbin/mrtg-cfg-etc-mrtg-cfg-logging-var-log-mrtg-mrtg-log.sh
root@lan1-11-10:~#


root@lan1-11-10:~# cat \
  /usr/local/sbin/mrtg-cfg-etc-mrtg-cfg-logging-var-log-mrtg-mrtg-log.sh
#!/bin/sh

env LANG=C /usr/bin/mrtg /etc/mrtg.cfg --logging /var/log/mrtg/mrtg.log

root@lan1-11-10:~#
```

B.1.2 Auszug aus den Datenreihen der Messdaten für die Indikatoren *Verfügbarkeit* und *Latenz*

```
root@lan1-11-10:~# date --rfc-2822 --date='@1507340711'
Sat, 07 Oct 2017 03:45:11 +0200
root@lan1-11-10:~#

root@lan1-11-10:~# cat /var/www/mrtg/ping.log | head -25
1507340711 22 22
1507340711 22 22 22 22
1507340410 22 22 22 22
1507340400 22 21 22 22
1507340100 22 21 22 22
1507339800 22 21 22 22
1507339500 22 21 22 22
1507339200 22 22 22 22
1507338900 22 22 22 22
1507338600 23 22 57 38
1507338300 56 37 57 38
1507338000 55 33 56 34
1507337700 51 30 64 42
1507337400 64 42 73 44
1507337100 72 44 73 46
1507336800 61 45 66 46
1507336500 65 44 66 45
1507336200 64 39 65 42
1507335900 62 42 63 46
1507335600 59 45 60 46
1507335300 55 30 58 45
1507335000 58 44 59 45
1507334700 58 40 59 44
1507334400 55 43 56 44
1507334100 56 35 60 43
root@lan1-11-10:~#

root@lan1-11-10:~# cat /var/www/mrtg/ping.log | wc --lines
2537
root@lan1-11-10:~#



root@r1:~# date --rfc-2822 --date='@1507340719'
Sat, 07 Oct 2017 03:45:19 +0200
root@r1:~#

root@r1:~# cat /var/www/mrtg/ping.log | head -25
1507340719 22 21
1507340719 22 21 22 21
1507340419 22 21 22 21
1507340400 22 21 22 21
1507340100 22 21 22 21
1507339800 22 21 22 21
1507339500 22 21 22 21
1507339200 22 21 22 22
1507338900 22 21 22 22
1507338600 24 21 59 21
```

```
1507338300 58 21 59 21
1507338000 52 21 55 21
1507337700 54 21 55 21
1507337400 52 21 62 21
1507337100 61 21 62 21
1507336800 54 21 55 21
1507336500 52 21 61 21
1507336200 60 21 61 21
1507335900 55 21 55 21
1507335600 55 21 59 21
1507335300 59 21 59 21
1507335000 58 21 59 21
1507334700 51 21 54 21
1507334400 54 21 55 21
1507334100 55 21 56 21
root@r1:~#

root@r1:~# cat /var/www/mrtg/ping.log | wc --lines
2537
root@r1:~#
```

B.1.3 Auszug aus dem Setup zur Erfassung der Messdaten für den Indikator *Datendurchsatz* mit der Software *iperf3*

```
root@r2:~# iperf3 -s -p 5002
-----------------------------------------------------------
Server listening on 5002
-----------------------------------------------------------


root@r1:~# iperf3 -t 120 -c 10.10.12.1 -p 5002 | \
  grep -e "Bitrate" -e KBytes | head -4
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  5.36 MBytes  45.0 Mbits/sec   35    348 KBytes
[  5]   1.00-2.00   sec  4.77 MBytes  40.0 Mbits/sec    6    261 KBytes
[  5]   2.00-3.00   sec  5.02 MBytes  42.1 Mbits/sec    0    282 KBytes
root@r1:~#



root@lan2-12-10:~# iperf3 -s -p 5002
-----------------------------------------------------------
Server listening on 5002
-----------------------------------------------------------


root@r1:~# iperf3 -t 120 -c 10.10.12.10 -p 5002 | \
  grep -e Bitrate -e KBytes | head -4
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  4.98 MBytes  41.8 Mbits/sec    0    336 KBytes
[  5]   1.00-2.00   sec  4.52 MBytes  37.9 Mbits/sec   20    211 KBytes
[  5]   2.00-3.00   sec  4.46 MBytes  37.4 Mbits/sec    0    229 KBytes
root@r1:~#

root@r1:~# mkdir ~/WORK-eval
root@r1:~# cd ~/WORK-eval
root@r1:~/WORK-eval#

root@r1:~/WORK-eval# iperf3 -t 120 -c 10.10.12.10 -p 5002 | \
  grep KBytes | awk '{ print $3"\t"$5"\t"$7"\t"$9 }' > results.dat
root@r1:~/WORK-eval#
```

B.1.4 Auszug aus den Datenreihen der Messdaten für den Indikator *Datendurchsatz* bei Verfügbarkeit der nativen VPN-Tunnel

```
root@r1:~# cd WORK-eval/
root@r1:~/WORK-eval# cat results.dat | head -45
0.00-1.00 3.30 27.7 0
1.00-2.00 4.83 40.5 3
2.00-3.00 4.77 40.0 5
3.00-4.00 4.33 36.4 0
4.00-5.00 4.33 36.4 0
5.00-6.00 4.33 36.4 0
6.00-7.00 4.33 36.4 0
7.00-8.00 4.77 40.0 0
8.00-9.00 4.33 36.4 0
9.00-10.00 4.77 40.0 7
10.00-11.00 4.33 36.4 0
11.00-12.00 4.77 40.0 0
12.00-13.00 4.33 36.4 0
13.00-14.00 4.77 40.0 0
14.00-15.00 4.33 36.3 11
15.00-16.00 4.33 36.4 0
16.00-17.00 4.77 40.0 0
17.00-18.00 4.33 36.4 0
18.00-19.00 4.33 36.4 0
19.00-20.00 4.77 40.0 0
20.00-21.00 4.33 36.4 0
21.00-22.00 4.52 37.9 14
22.00-23.00 4.77 40.0 0
23.00-24.00 4.33 36.4 11
24.00-25.00 4.33 36.4 0
25.00-26.00 4.77 40.0 0
26.00-27.00 4.33 36.4 0
27.00-28.00 4.33 36.4 0
28.00-29.00 4.77 40.0 0
29.00-30.00 4.33 36.4 0
30.00-31.00 4.77 40.0 5
31.00-32.00 4.33 36.4 0
32.00-33.00 4.77 40.0 2
33.00-34.00 4.33 36.4 0
34.00-35.00 4.33 36.4 0
35.00-36.00 4.33 36.4 0
36.00-37.00 4.33 36.4 0
37.00-38.00 4.77 40.0 0
38.00-39.00 4.46 37.4 11
39.00-40.00 4.33 36.4 0
40.00-41.00 3.90 32.7 47
41.00-42.00 4.33 36.4 0
42.00-43.00 4.77 40.0 0
43.00-44.00 4.33 36.4 0
44.00-45.00 4.33 36.4 0
root@r1:~/WORK-eval#

root@r1:~/WORK-eval# cat results.dat | wc --lines
120
root@r1:~/WORK-eval#
```

B.2 Weiterführende Informationen zur Auswertung der Messdaten

B.2.1 Auszug aus dem Setup und dem Transkript zur Auswertung der Messdaten für den Indikator *Datendurchsatz* mit der Software *gnuplot*

```
root@r1:~# cd WORK-eval/
root@r1:~/WORK-eval# cat results.dat | head -3
0.00-1.00       3.30    27.7    0
1.00-2.00       4.83    40.5    3
2.00-3.00       4.77    40.0    5
root@r1:~/WORK-eval#


root@r1:~/WORK-eval# cat graph-tcp-data-01.p | grep -v -e "^#" -e "^$"
unset log
unset label
set terminal svg dashed size 1024,600 fontfile 'font-svg.css' font 'rfont,13'
set grid
set key off
set title "Übertragungsbandbreite"
set ylabel "MBit/s"
set yrange [0:100]
set yrange [0:100]
set output "Bandwidth.svg"
plot "results.dat" using 1:3 title 'Bandwidth' smooth csplines
set title "Erneut übertragene IP-Pakete (Packet Retries)"
set ylabel "Anzahl"
set yrange [0:75]
set output "ReTries.svg"
plot "results.dat" using 1:4 title 'ReTries' smooth csplines
root@r1:~/WORK-eval#

root@r1:~/WORK-eval# cat font-svg.css
<style type="text/css">
@font-face {
  font-family: rfont;
  src: url(font.ttf) format("truetype");
}
</style>
root@r1:~/WORK-eval#


root@r1:~/WORK-eval# gnuplot < graph-tcp-data-01.p
Font file 'font-svg.css' contains the font 'font-family: rfont;
'
Font file 'font-svg.css' contains the font 'font-family: rfont;
'
root@r1:~/WORK-eval#

root@r1:~/WORK-eval# file Bandwidth.svg ReTries.svg
Bandwidth.svg: SVG Scalable Vector Graphics image
ReTries.svg:   SVG Scalable Vector Graphics image
root@r1:~/WORK-eval#
```

Kolophon

Dieses Dokument wurde mit der LaTeX-Vorlage für Abschlussarbeiten an der htw saar im Bereich Informatik/Mechatronik-Sensortechnik erstellt (Version 2.1). Die Vorlage wurde von Yves Hary und André Miede entwickelt (mit freundlicher Unterstützung von Thomas Kretschmer, Helmut G. Folz und Martina Lehser). Daten: (F)10 – (B)426.79135pt – (H)688.5567pt