Peer-to-Peer-Technologie in Teilnehmerzugangsnetzen

Dissertation
zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

vorgelegt von
Danielis, Peter, geb. am 19.05.1982 in Wismar,
aus Rostock

urn:nbn:de:gbv:28-diss2013-0158-0

Gutachter:

- Prof. Dr. Dirk Timmermann (Universität Rostock, Fakultät für Informatik und Elektrotechnik, Institut MD)
- Prof. Dr. Gero Mühl (Universität Rostock, Fakultät für Informatik und Elektrotechnik, Institut für Informatik, Lehrstuhl AVA)
- Prof. Dr. Martin Mauve (Universität Düsseldorf, Institut für Informatik, Lehrstuhl für Rechnernetze)

Tag der Einreichung: 25.06.2012
Tag der Verteidigung: 25.09.2012

Inhaltsverzeichnis

Abbildungsverzeichnis **ix**

Tabellenverzeichnis **xvii**

Abkürzungsverzeichnis **xix**

1. Einleitung **1**

1.1. Die Zielsetzungen dieser Arbeit 1

1.2. Aufbau dieser Arbeit 3

2. Grundlagen **5**

2.1. Netzwerktheoretische Grundlagen 5

2.1.1. Netzwerkstruktur des Internets 6

2.1.2. ISO-OSI-Referenzmodell 8

2.2. Ausgewählte Netzwerkprotokolle 9

2.2.1. Ethernet 9

2.2.2. IP 11

2.2.3. UDP 12

2.2.4. TCP 13

2.3. Ausgewählte Internet-Basisdienste 14

2.3.1. DHCP 14

2.3.2. DNS 15

2.4. P2P-Netzwerke 19

2.4.1. Klassifizierung von P2P-Systemen 22

2.4.2. P2P-Systeme 23

2.4.3. Unstrukturierte, zentralisierte P2P-Systeme (BitTorrent) 23

2.4.4. Unstrukturierte, dezentralisierte P2P-Systeme 26

2.4.5. Unstrukturierte, hybride P2P-Systeme 26
2.4.6. Strukturierte, Distributed Hash Table-basierte P2P-Systeme (Kad) 27
2.4.7. Zusammenfassender Vergleich der P2P-Systeme 32

3. Berücksichtigung netzwerktechnischer Nähe in P2P-Netzen 35
3.1. Motivation 36
3.2. Stand der Technik 38
3.3. Berechnung des Hop Counts 39
3.3.1. Einfügen des initialen TTL-Wertes in BitTorrent-Nachrichten . . . 40
3.3.2. Bereitstellung des TTL-Wertes in der BitTorrent-Applikation . . . 41
3.4. Verbesserung der Peer-Auswahl 42
3.5. Auswertung von Standard- und modifiziertem BitTorrent-Algorithmus . . 44
3.5.1. Auswahl eines geeigneten Simulators 45
3.5.1.1. Vorteile von ns-2 45
3.5.1.2. Arbeitsweise mit ns-2 47
3.5.2. BitTorrent in ns-2 48
3.5.3. Simulationsaufbau 51
3.5.4. Simulationsergebnisse und Auswertung 53
3.6. Zusammenfassung und Ausblick 58

4. Auswahl und Erweiterung des P2P-Protokolls für dezentralisierte Kommunikationsinfrastruktur 61
4.1. Auswahl des DHT-basierten P2P-Protokolls 61
4.2. Behandlung von Firewalls und NAT 64
4.2.1. Motivation 64
4.2.2. Stand der Technik 65
4.2.3. Mechanismen zur Erkennung von Firewalls und NAT für das UDP-Protokoll 66
4.2.4. Der Buddy-Mechanismus zur Umgehung von Firewalls und NAT [Bru06] 68
4.2.5. Zusammenfassung und Ausblick 69
4.3. Dynamische Suchtoleranz 70
4.3.1. Motivation 70
4.3.2. Stand der Technik 71

4.3.3. Globale Peer-Ermittlung mittels KaDis-Algorithmus 72
4.3.4. Algorithmus zur Berechnung der dynamischen Suchtoleranz 74
4.3.5. Beweis der Gültigkeit des DST-Algorithmus 75
4.3.6. Einschränkungen des vorgestellten Ansatzes 79
4.3.7. Zusammenfassung 80

5. Entwurf und Realisierung P2P-basierter Netzwerkinfrastruktur 81
5.1. Die P2P-basierte Speicherplattform *PSP* 83
5.1.1. Motivation 84
5.1.2. Stand der Technik 85
5.1.3. P2P-Technologie für die Realisierung von PSP 87
5.1.4. Sicherstellung 99,999 %-iger Datenverfügbarkeit 88
5.1.4.1. Einfache Datenreplikation vs. Erasure Resilient Codes . . 89
5.1.4.2. Reed-Solomon-Codes 91
5.1.5. Kad-basierte Realisierung von PSP 91
5.1.5.1. Architektur eines PSP-Knotens 92
5.1.5.2. Auslösung von PSP-Knotenaktivitäten 94
5.1.5.3. Weitere Aspekte 97
5.1.6. Zusammenfassung 98
5.2. Das P2P-basierte verteilte Rechensystem *DuDE* 99
5.2.1. Motivation 99
5.2.2. Stand der Technik 101
5.2.3. DuDE im Allgemeinen 103
5.2.3.1. DuDEs Datenbasis: Log-Datensätze und Langzeitstatistiken 104
5.2.3.2. Globale Datenermittlung 106
5.2.3.3. Der DuDE-Algorithmus 107
5.2.3.4. Backup-System 111
5.2.3.5. Ressourcen-Gruppierung 112
5.2.4. Software-Realisierung von DuDE 112
5.2.5. Testszenario 114
5.2.6. Auswertungsergebnisse 116
5.2.7. Zusammenfassung 121
5.3. Das P2P-basierte DNS *P-DONAS* 122

5.3.1. Motivation . . . 122
5.3.2. Stand der Technik . . . 123
5.3.3. Herausforderungen des traditionellen DNS . . . 125
5.3.3.1. Begrenzte Skalierbarkeit . . . 126
5.3.3.2. Niedrige Performance . . . 127
5.3.3.3. Inkonsistenz . . . 128
5.3.3.4. Lame Delegations . . . 128
5.3.4. Konzipierung von P-DONAS . . . 129
5.3.4.1. Anforderungen an ein P2P-basiertes DNS . . . 131
5.3.4.2. Kad-basierter Entwurf von *P-DONAS* . . . 132
5.3.4.3. Architektur eines *P-DONAS*-Knotens . . . 134
5.3.4.4. Algorithmen zur Hashwertberechnung . . . 135
5.3.4.5. Formate der Speichereinträge . . . 136
5.3.4.6. Adressierungsschema . . . 138
5.3.4.7. Auslösung von Knotenaktivitäten . . . 140
5.3.5. Implementierung von P-DONAS . . . 147
5.3.5.1. Socket-Kommunikation . . . 149
5.3.5.2. Threads und Inter-Thread-Kommunikation . . . 151
5.3.5.3. Software-Portierung auf ein Xilinx Evaluation Board . . . 154
5.3.5.4. Erweiterung um ein Sicherheitsprotokoll . . . 158
5.3.6. Auswertung . . . 158
5.3.6.1. Testaufbau . . . 158
5.3.6.2. Aufgezeichnete DNS-Nutzlast . . . 160
5.3.6.3. Ergebnisse . . . 162
5.3.7. Zusammenfassung und Ausblick . . . 173
5.4. Kapitelzusammenfassung und Ausblick . . . 175

6. Zusammenfassung **177**
6.1. Berücksichtigung netzwerktechnischer Nähe in P2P-Netzen . . . 177
6.2. Entwurf und Realisierung P2P-basierter Netzwerkinfrastruktur . . . 178
6.3. Ausblick . . . 179

A. DHCP-Pakete und -Protokoll **I**
A.1. Aufbau von DHCP-Paketen . . . I

A.2. Ablauf des DHCP-Protokolls II

B. DNS-Pakete und -Protokoll V

B.1. Allgemeiner Aufbau V

B.2. Aufbau von DNS-Updates VIII

B.3. Durch P-DONAS unterstützte Teile des DNS-Protokolls X

Literaturverzeichnis XIII

Liste der Veröffentlichungen und Fachvorträge auf Tagungen XXIII

Betreute studentische Arbeiten XXVII

Selbständigkeitserklärung XXXI

Thesen XXXIII

Kurzreferat XXXVII

Abstract XXXIX

Abbildungsverzeichnis

1.1. Kapitelstruktur der Forschungsarbeit. Die wesentlichen eigenen Beträge sind fett hervorgehoben. 4

2.1. Einteilung des Internets in Netzwerksegmente. 7
2.2. (a) OSI-Netzwerkarchitektur und (b) Internet-Protokollstapel [Hal06,PD97]. 10
2.3. Aufbau eines Ethernetframes. 11
2.4. Aufbau eines IPv4-Pakets. 12
2.5. Aufbau eines UDP-Pakets. 13
2.6. Aufbau eines TCP-Pakets. 14
2.7. Schematische Darstellung der DNS-Hierarchie. Als Beispiel ist der Domainname www.yahoo.com. dargestellt. 16
2.8. Weltweite Verteilung der DNS-Root Server [Gib07]. 17
2.9. Ablauf einer DNS-Anfrage. 18
2.10. Traditionelles Client-Server-Netzwerkmodell. 20
2.11. Vollständig dezentralisiertes P2P-Netzwerkmodell. 21
2.12. P2P-Overlay und die Entsprechung auf dem physikalischen Underlay (Beispiel). 22
2.13. Klassifizierung von P2P-Systemen. 23
2.14. Veranschaulichung der BitTorrent-Funktionsweise. 24
2.15. Kontaktaufnahme von Peers bei BitTorrent. 24
2.16. Ablauf des Datenaustauschs zwischen Peers bei BitTorrent. 25
2.17. Beispiel für einen DHT-Ring mit einigen Peers und Daten. Die gewählten Zuständigkeitsbereiche der Peers für Daten dienen nur der Veranschaulichung. 27

2.18. Beispiel für eine Routing-Tabelle eines Peers mit einem 4 Bit-Adressraum im Kad-Netzwerk. Alle Kontakte des Peers ordnen sich entsprechend der XOR-Distanzen zu diesem Peer in entsprechende Bereiche—die sogenannten k-Buckets—ein. Da der Peer zu sich selbst eine XOR-Distanz von Null aufweist, befindet er sich ganz rechts in der Routing-Tabelle. 28
2.19. Rekursiver Lookup-Vorgang, bei dem der suchende Peer (schwarz) für einen Ziel-Hashwert einen Ziel-Peer (rot) sucht, der einen dem Ziel-Hashwert ähnlichen oder gleichen Hashwert aufweist. Der Ziel-Peer ist dem suchenden Peer nicht bekannt, so dass die Suchanfrage durch andere Peers direkt an den Ziel-Peer weitergegeben wird. Nur der Ziel-Peer antwortet. 29
2.20. Iterativer Lookup-Vorgang, bei dem der suchende Peer (schwarz) für einen Ziel-Hashwert einen Ziel-Peer (rot) sucht, der einen dem Ziel-Hashwert ähnlichen oder gleichen Hashwert aufweist. Der Ziel-Peer ist dem suchenden Peer nicht bekannt, so er erst andere Peers befragen muss, um den Ziel-Peer kennenzulernen. Alle Peers auf dem Suchpfad antworten, so dass der suchende Peer sich mit den neu erlernten Kontakten selbst in Verbindung setzen kann. 30
2.21. Erster Schritt eines iterativen Kad-Lookup-Vorgangs. 31
2.22. Zweiter Schritt eines iterativen Kad-Lookup-Vorgangs. 31
2.23. Dritter Schritt eines iterativen Kad-Lookup-Vorgangs. 32
2.24. Vierter Schritt eines iterativen Kad-Lookup-Vorgangs. 32
2.25. Finaler Schritt eines iterativen Kad-Lookup-Vorgangs. 32
2.26. Komplexitätsvergleich der P2P-Systeme in Abhängigkeit von der Knotenanzahl N [SW05]. Die Abbildung legt den Fokus auf die Darstellung strukturierter, DHT-basierter Systeme als bester Kompromiss aus den drei Systemen. Demgegenüber sind die gravierenden Schwachstellen von zentralisierten und dezentralisierten/hybriden Systemen dargestellt. 33

3.1. Anteil des P2P-/BitTorrent-Datenvolumens am gesamten Datenverkehr im Internet in verschiedenen Regionen der Welt 2009 [ipo09]. 35
3.2. Aufbau einer BitTorrent-Handshake-Nachricht. 41
3.3. Neuer BitTorrent-Choking-Algorithmus zur Berechnung der Position eines Nutzers in der Unchoke-Liste. Der Hop Count wird dabei als zusätzliches Auswahlkriterium genutzt. 43

3.4. Komplementäre kumulative Weibull-Verteilungsfunktion für den Eintrittszeitpunkt von BitTorrent-Nutzern in das Netzwerk. Die Y-Achse ist logarithmisch zur Basis 10 skaliert. Zum Zeitpunkt t = 4800 s = 80 min nach Simulationsstart hat die Mehrzahl der BitTorrent-Nutzer (ca. 90 %) das Netzwerk betreten. 49
3.5. Komplementäre kumulative Weibull-Verteilungsfunktion für die Sitzungslänge bzw. die Verweildauer von BitTorrent-Nutzern. Die X-Achse ist logarithmisch zur Basis 2 skaliert; für die Y-Achse wurde eine logarithmische Skalierung zur Basis 10 gewählt. 50
3.6. Telefonicas Netzwerkinfrastruktur in Deutschland [Sko08, Tel12] 52
3.7. Anzahl der Hops für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant. 54
3.8. Anzahl der Paketverwürfe für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant. 55
3.9. Maximaler Durchsatz für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant. 56
3.10. Zeitpunkt des maximalen Durchsatzes für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant. 57
3.11. Zeitpunkt der Fertigstellung des Datei-Downloads für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant. 57

4.1. Einfache Firewall-Erkennung mit Wartezeit [Bru06]. 66
4.2. Verteilte Firewall-Erkennung ohne Wartezeit. 67
4.3. Kontaktaufnahme von Peer B mit Peer A mit Hilfe von Buddy-Peer C [Bru06]. 69
4.4. Ablauf des KaDis-Algorithmus für die globale Peer-Ermittlung. 73
4.5. Kad-Ring mit N=8 Knoten in einem 4 Bit-Adressraum (n=4) für i=0. . . 75
4.6. Kad-Ring für i=1 (erste Iteration). 75
4.7. Kad-Ring für i=2 (zweite Iteration). 76
4.8. Kad-Ring für i=3 (letzte Iteration vor dem Abbruch). 76
4.9. Kad-Ring mit N=8 Knoten (nummeriert) in einem 4 Bit-Adressraum (n=4). 77
4.10. Kad-Routing-Tabelle aus Sicht von Knoten 1. 78
4.11. Kad-Routing-Tabelle aus Sicht von Knoten 2. 78

4.12. Kad-Ring nach Aufteilung in Ringsegmente $M_{a_1,...,a_d}$ 79

5.1. TZN mit Zugangsknoten wie z. B. PowerQUICC II Pro (MPC8378). Ersichtlich sind der persistente Flash-Speicher, der flüchtige RAM-Speicher in Höhe von 1 GByte RAM sowie die 1234 Dhrystone-MIPS-Rechenkapazität. 40 % der RAM-Speicherressourcen sowie der Rechenkapazität stehen zur freien Verfügung [Fre11, Nin11]. 82
5.2. Aufteilung der Sitzungsdaten in m=9 Datenstücke und Generierung von k=3 Kodierungstücken mittels ERCs. 90
5.3. Dekodierung der Sitzungsdaten aus beliebigen m=9 Stücken mittels ERCs. 90
5.4. Kad-basierter DHT-Ring für die strukturierte Speicherung von Sitzungsdaten- und Kodierungsstücken auf Zugangsknoten (Access Nodes (ANs)). 92
5.5. Blockdiagramm eines PSP-Knotens. Der eigene Sitzungsdatenbestand, auf den ein DHCP-Server zur Bereitstellung von IP-Adressen zugreift, wird über das Kad-Protokoll auf anderen Knoten gesichert. Externe Steuerung durch einen Administrator ist möglich. 93
5.6. Externe und interne Auslösung von Knotenaktivität und Interaktion mit anderen PSP-Knoten. 94
5.7. Entwicklung der Internet-Nutzerzahlen weltweit und des anfallenden Datenverkehrs pro Monat am Amsterdam Internet Exchange (AMS-IX) [Exc11, Int11]. 100
5.8. Kad-basierter DHT-Ring für die verteilte Statistikberechnung und verteilte Datenhaltung. Jeder Knoten hält seine eigenen Log-Daten, erzeugt fortwährend daraus LZS und verteilt beide Datensätze. 105
5.9. KaDis-Algorithmus für die globale Log-Datenermittlung. Bei der Suche nach Langzeitstatistiken (LZS) wird entsprechend nach *Knoten i_ cpu.lzs*, *Knoten i_ ram.lzs* etc. gesucht. 106
5.10. DuDE-Algorithmus für die verteilte Statistikberechnung (Phase 1-4). . . . 109
5.11. DuDE-Algorithmus für die verteilte Statistikberechnung (Phase 5-7). . . . 111
5.12. Systemarchitektur eines DuDE-Knotens. 113
5.13. Zeit für die Erledigung des Jobs für eine ansteigende Anzahl von Tasks und einer unterschiedlichen Anzahl von Log-Datensätzen (Rechenlast = 0). 117
5.14. Zeit für die Erledigung des Jobs für eine ansteigende Anzahl von Tasks und unterschiedlichen Rechenlasten (Log-Datensätze = 7). 118

5.15. Maximale Speicherauslastung der Knoten für eine ansteigende Anzahl von Tasks (Rechenlast = 10, Log-Datensätze = 7). 119
5.16. Maximale Speicherauslastung der Knoten für eine ansteigende Rechenlast (Tasks = 6, Log-Datensätze = 7). 120
5.17. Maximale Speicherauslastung der Knoten für eine ansteigende Anzahl von Log-Datensätzen (Rechenlast = 10). 121
5.18. Minimale Anzahl der zu kompromittierenden Nameserver für alle Domains (Delegation Bottlenecks) [RS04]. 127
5.19. Minimale Anzahl der zu kompromittierenden Nameserver für die Top 500-Webseiten (Delegation Bottlenecks) [RS04]. 128
5.20. Minimale Anzahl der zu kompromittierenden Gateways/Router zwischen Nutzer und Nameserver für alle Domains (Physikalische Bottleneck) [RS04]. 129
5.21. Beispiel für das Auftreten einer Lame Delegation. 130
5.22. AS (enthält mehrere Zugangsknoten), die mit einer Domain Name System (DNS)-Server-Farm verbunden ist. 131
5.23. Mittels Kad verbundene ASs mit DNS-Einträgen (Domainname (DN)-IP-Adresse (IP)-Paar). 132
5.24. Beispiel für den Aufbau einer DHT-Ring-Struktur mit Master-P2P-Knoten zur Verbindung zweier ISP. Aus Übersichtlichkeitsgründen sind die DNS-Einträge nicht dargestellt. 133
5.25. Blockschaltbild eines P-DONAS-Knotens. 135
5.26. Aufbau eines Knoten-Eintrags. 136
5.27. Aufbau eines Cache-Eintrags. 137
5.28. Adressierungsschema für Knoten-Einträge. 139
5.29. Adressierungsschema für Cache-Einträge. 140
5.30. Auslöser für Knotenaktivitäten: Nutzer, Kad-Netzwerk oder traditionelles DNS bzw. vertrauenswürdige Instanz. 141
5.31. Verarbeitung eines DNS-Requests vom Nutzer. 142
5.32. Lookup-Vorgang auf dem Kad-Ring. Bei der Aktion handelt es sich um die Suche nach Knoten, dem Anfordern von RRs oder einer Speicher- oder Löschoperation. 144
5.33. Verarbeitung eines DNS-Response vom Nameserver. Operationen im Kad-Netzwerk sind fett hervorgehoben. 147

5.34. Verarbeitung eines DNS-Update (Delete) Requests von einer vertrauenswürdigen Instanz. Die Speicher-/Löschoperation betrifft nur die Knoten-Einträge. Der Cache dient nur als temporärer Speicher; Cache-Einträge verschwinden nur durch den Ablauf ihrer TTL und werden nicht explizit gelöscht. 148
5.35. Implementierung der Sockets. 150
5.36. Inter-Thread-Kommunikation. 152
5.37. Schaltplan der Hardware-Spezifikation. Aus Übersichtlichkeitsgründen sind die Taktleitungen des Clock-Generators zum Betreiben des Boards und der Busnetze nicht dargestellt. 156
5.38. P-DONAS-Testaufbau. 159
5.39. Anzahl der DNS-Requests pro Minute über der Testzeit. 161
5.40. Durchschnittliche Latenz pro Minute über der Testzeit. 161
5.41. KVF der Domainnamen-Längen. 162
5.42. Leerlauftests: Durchschnittliche Speicher- und CPU-Auslastung pro Minute (alle Knoten). 164
5.43. Durchschnittliche Latenz über dem Time-out-Wert im Vergleich mit der Nameserver-Latenz und dem Anteil der Kad-Antworten auf verarbeitbare DNS-Requests. 165
5.44. Anteil der Cache-, Knoten-, Kad- und Nameserver-Antworten auf verarbeitbare DNS-Requests über dem Time-out-Wert. 166
5.45. Durchschnittliche Anzahl der Cache-Einträge pro Minute über dem Time-out-Wert. 167
5.46. Durchschnittliche Anzahl der Knoten-Einträge pro Minute über dem Time-out-Wert. 167
5.47. Anzahl der verarbeiteten Kad-Pakete über dem Time-out-Wert. 168
5.48. Volumen des verarbeiteten Kad-Datenverkehrs über dem Time-out-Wert. . 168
5.49. Maximale Füllstände der Client- und Nameserver-Warteschlangen (Laptop_H54) über dem Time-out-Wert. 169
5.50. Maximale Füllstände der Kad-Empfangs-Warteschlangen (alle Knoten) über dem Time-out-Wert. 170
5.51. Maximale Füllstände der Kad-Sende-Warteschlangen (alle Knoten) über dem Time-out-Wert. 170

5.52. Durchschnittliche CPU-Auslastung pro Minute (alle Knoten) über dem Time-out-Wert. 171
5.53. Durchschnittliche Speicherauslastung pro Minute (alle Knoten) über dem Time-out-Wert. 171

A.1. Aufbau eines DHCP-Pakets. I
A.2. Ablauf des DHCP-Protokolls. IV

B.1. Aufbau eines DNS-Pakets (außer DNS-Updates) [Moc87]. V
B.2. Aufbau einer Questions-Struktur. VI
B.3. Aufbau einer Query-Struktur, die in einer Questions-Struktur enthalten sein kann. VII
B.4. Aufbau eines RRs. VII
B.5. Struktur zur DNS-Nachrichtenkomprimierung. VII
B.6. Aufbau eines DNS-Updates [Vix97]. VIII
B.7. Aufbau einer Zone Section. VIII

Tabellenverzeichnis

2.1. Klassifizierung des Internets in Abhängigkeit von der geographischen Ausdehnung [Tan03]. 6

3.1. Simulatoren und ihre Eigenschaften [FV08, BHK07, GPM+04, SGR+11, MJ09]. 45
3.2. Aufgezeichnete Daten in einem ns-2 Trace File [FV08] 47
3.3. Beispieldaten in einem ns-2 Trace File [FV08] 48
3.4. Aspekte des dynamischen Nutzerverhaltens im BitTorrent-Netzwerk [SR06b] 50
3.5. Datenmenge im Kernnetz für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant. 54

4.1. Suchkomplexitäten der DHT-basierten P2P-Netze mit deterministischer Suchkomplexität in Abhängigkeit von der Anzahl der Knoten N und Parameter b [SW05, SR06a, MM02]. 63
4.2. Beispielszenario für die Funktionsweise des KaDis-Algorithmus. 74

5.1. Grundlegende Parameter der verteilten Speicherung 89
5.2. Beispielszenario für die Funktionsweise des für DuDE angepassten KaDis-Algorithmus bei der Suche nach Log-Daten. 107
5.3. Beispielszenario für die Veranschaulichung des DuDE-Algorithmus mit vier Knoten. Freie Rechen- und Speicherkapazitäten sind aufgeführt. . . . 108
5.4. Einstellbare Parameter des Testsystems 115
5.5. Bedeutung des Valid-Felds eines Speichereintrags 137
5.6. Hardware-Spezifikation des ML405-Boards. 155
5.7. Ausstattung der für den P-DONAS-Testaufbau eingesetzten Rechner bzw. Xilinx ML405 Boards. 160

5.8. Zusammensetzung der durch P2P-based Domain Name System (P-DONAS) nicht verarbeitbaren Pakete. 163
5.9. Zusammenfassung der wichtigsten Ergebnisse. Für die untersuchten Timeout-Werte für die Suche im Kad-Netz sind die Antwortlatenz und der Anteil aller Antworten bzw. der Kad-Antworten von P-DONAS auf verarbeitbare DNS-Requests ersichtlich. 173

A.1. Nachrichten des DHCP-Protokolls. II

B.1. Opcode-Feld von DNS-Paketen. VI
B.2. Unterstützte Teile des DNS-Protokolls. X

Abkürzungsverzeichnis

ALTO	Application-Layer Traffic Optimization
AMS-IX	Amsterdam Internet Exchange
AN	Access Node
A RR	Address RR
API	Application Programming Interface
AS	Access Site
BBP	Bailey-Borwein-Plouffe
BCH	Bose-Chaudhuri-Hocquenghem
BRAM	Block RAM
BRAS	Broadband Remote Access Server
BSD	Berkeley Software Distribution
BTA	BitTorrent-Algorithmus
ccTLD	Country Code TLD
CD	Collision Detection
CDN	Content Distribution Network
CSMA	Carrier Sense Multiple Access
CMTS	Cable Modem Termination System
CoDiP2P	Computing Distributed architecture using the P2P paradigm
CoDoNS	Cooperative Domain Name System
CPU	Central Processor Unit
CRC	Cyclic Redundancy Check
DDNS	Distributed DNS
DDoS	Distributed Denial of Service
DDR	Double Data Rate
DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Table
DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
DoS	Denial of Service
DR	Datenreplikation
DS	Data Spreader
DSLAM	Digital Subscriber Line Access Multiplexer
DST	Dynamische Suchtoleranz
DuDE	Distributed Computing System using a Decentralized P2P Environment
DVB	Digital Video Broadcasting
DVD	Digital Versatile Disc
EDK	Embedded Development Kit
ENTROPIA	Encompassing Technological Reuse of P2P in Access Networks
ERC	Erasure Resilient Code

FCS	Frame Check Sequence
FPGA	Field-Programmable Gate Array
FTP	File Transfer Protocol
GAN	Global Area Network
GPL	General Public License
GPON	Gigabit Passive Optical Network
gTLD	Generic TLD
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPclip	Internet Protocol Calling Line Identification Presentation
IPsec	Internet Protocol Security
IPTV	Internet Protocol Television
ISO	International Organization for Standardization
ISP	Internet Service Provider
JTAG	Joint Test Action Group
JXTA	Juxtapose
KaDis	Kademlia Discovery
KVF	Kumulative Verteilungsfunktion
KZS	Kurzzeitstatistiken
LAN	Local Area Network
LGPL	Lesser General Public License
lwIP	LightWeight IP
LZS	Langzeitstatistiken
MAC	Medium Access Control
MAN	Metropolitan Area Network
MD4	Message-Digest Algorithm 4
MD5	Message-Digest Algorithm 5
MIPS	Millionen Instruktionen pro Sekunde
MPMC	Multi-Port Memory Controller
MTU	Maximum Transmission Unit
NAPA-WINE	Network-Aware P2P-TV Application over Wise Networks
NAT	Network Address Translation
NS RR	Nameserver RR
OLT	Optical Line Termination
OSI	Open Systems Interconnection Model
P2P	Peer-to-Peer
P4P	Portal for P2P Applications
PA	Periodic Accumulator
PAN	Personal Area Network
PAT	Port Address Translation

PCAP	Packet Capture
PCI	Protocol Control Information
P-DONAS	P2P-based Domain Name System
PDU	Protocol Data Unit
PLB	Processor Local Bus
PSP	P2P-based Storage Platform
QoE	Quality of Experience
RAM	Random Access Memory
RG	Residential Gateway
RR	Resource Record
SDK	Software Development Kit
SOA RR	Start of Authority RR
SPoF	Single Point of Failure
SDRAM	Synchronous Dynamic RAM
SRAM	Static RAM
ST	Suchtoleranz
TCL	Tool Command Language
TCP	Transmission Control Protocol
TLD	Top-Level Domain
TOE	Time of Expiry
TSIG	Transaction Signature
TTL	Time-to-Live
TZN	Teilnehmerzugangsnetz
UDP	User Datagram Protocol
URL	Uniform Ressource Locator
VI	Vertrauensintervall
VLAN	Virtual LAN
WAN	Wide Area Network
WF	Hop Count-Wichtungsfaktor

Danksagung

An dieser Stelle möchte ich die Gelegenheit ergreifen, mich bei meiner Familie zu bedanken, die mich stets aufopferungsvoll bei meinem nun bereits 24-jährigem Bildungsweg unterstützt hat. Ohne diese Unterstützung wäre die Verfassung dieser Forschungsarbeit nicht möglich gewesen.

Mein großer Dank gilt weiterhin meinem Doktorvater Professor Timmermann, der durch meine Anstellung am Institut für Angewandte Mikroelektronik und Datentechnik die Grundlage für die Erstellung dieser Forschungsarbeit schuf. Er half mir Tiefs und Schwierigkeiten zu meistern, motivierte mich stets und trug damit maßgeblich zu dem Gelingen dieser Forschungsarbeit bei.

Darüber hinaus danke ich Jens Rohrbeck, Jan Skodzik und Vlado Altmann, die diese Arbeit fleißig Korrektur gelesen haben sowie allen Kollegen und Studenten, die an dieser Arbeit beteiligt sind. Dabei möchte ich besonders Stephan Kubisch, Harald Widiger, Jens Rohrbeck, Jan Skodzik, Thomas Bahls und Daniel Duchow sowie Matthias Böhm und Tobias Strauß nennen.

Schlussendlich richte ich ein großes Dankeschön an alle Mitarbeiter des Instituts für Angewandte Mikroelektronik und Datentechnik, die mir ein Arbeiten in freundlicher und netter Atmosphäre ermöglichten.

Kapitel 1.

Einleitung

Durch eine wachsende Anzahl von Nutzern und die steigende Komplexität der angebotenen Dienste nimmt der Datenverkehr im Internet stetig zu. Der Bedarf an Bandbreite vergrößert sich fortwährend, so dass der Zusammenbruch des Internets immer wieder vorhergesagt wurde - so geschehen für 1995 und 2006. 2006 waren der Grund vor allen Dingen Audio- und Video-Streams - es war deshalb auch das Jahr des „Internet-Videos“. Dabei stellen die Video-Server den Flaschenhals dar, da sie unter Umständen nicht alle Anfragen von den Clients der Nutzer bedienen können und überlastet werden. Netzbetreiber reagieren heutzutage auf diese Herausforderungen mit zusätzlicher leistungsfähiger und oft zentralisierter Infrastruktur und mehr Bandbreite.

Um zentrale Server zu sparen und Flaschenhälse zu vermeiden, wurde das Peer-to-Peer (P2P)-Netzwerkmodell entwickelt. Die Idee dabei ist es, Daten nicht auf zentralen Servern zu lagern, sondern bereits existente Kopien auf Festplatten von Nutzern zum Download heranzuziehen. Im Gegensatz zum traditionellen Client-Server-Netzwerkmodell bieten Peers sowohl Server- als auch Client-Funktionalität an und teilen ihre Speicher- und Rechenressourcen. P2P-Netze bieten damit eine Lösung, immer höheren Bandbreitenanforderungen gerecht zu werden und Up- und Download besser zu verteilen.

1.1. Die Zielsetzungen dieser Arbeit

Berücksichtigung netzwerktechnischer Nähe in P2P-Netzen [DT10]: Insbesondere durch File-Sharing-Clients, die 1999 aufkamen, erlangten P2P-Netze innerhalb weniger Monate bei einem breiten Publikum große Popularität. Im Gegensatz zu der Entwick-

lung des Medienhypes ist P2P-Datenverkehr durchgehend gewachsen. Die Verhaltensmuster haben sich verändert und die Beliebtheit der einzelnen Netzwerke hat sich dramatisch verschoben, aber der Verbrauch von Netzwerkressourcen insbesondere durch P2P-File-Sharing ist weiter gewachsen [ipo09]. Heutzutage machen P2P-Daten einen großen Teil des Datenverkehrs im Internet aus. So lag der Anteil 2009 bei 43 bis 70 % in einigen Regionen der Welt. Dies wird hauptsächlich durch File-Sharing-Anwendungen wie eMule oder BitTorrent verursacht. Der Anteil des BitTorrent-Datenverkehr machte bis zu 80 % des P2P-Datenverkehrs aus, d.h. 56 % des gesamten Datenverkehrs im Internet. Neuere Quellen belegen, dass P2P-Daten nach wie vor einen großen Anteil am gesamten Datenverkehr von 10 bis 20 % ausmachen, obwohl der Hauptanteil des Datenverkehrs mittlerweile auf Echtzeitunterhaltungsanwendungen wie z. B. Video-Streaming entfällt [san11b, san11a]. BitTorrent ist nach wie vor das weltweit vorherrschende P2P-Protokoll mit einem Anteil von ungefähr 90 % am P2P-Datenaufkommen.

Daraus ergibt sich die Notwendigkeit, P2P-Datenverkehr in geordnete Bahnen zu lenken, um zu vermeiden, dass die Kernnetze der Netzbetreiber überflutet werden. Da P2P-Daten ohne zusätzliche Maßnahmen unabhängig vom physikalischen Underlay auf dem logischen P2P-Overlay geroutet werden, widmet sich ein Aspekt dieser Arbeit damit, das Routing auf dem logischen Overlay besser an das physikalische Underlay anzupassen. Hierfür wurde ein Algorithmus zur verbesserten Auswahl von Peers zum Datenaustausch entwickelt.

Entwurf P2P-basierter Netzwerkinfrastruktur [DT10]: Auf Grund der populären File-Sharing-Clients wie BitTorrent oder eMule werden P2P-Netze oft (jedoch zu Unrecht) nur mit illegalen Aktivitäten wie dem Herunterladen von nicht lizenzierten Filmen oder Software in Verbindung gebracht. Allerdings stellt P2P abgesehen von seinen ihm vorgeworfenen Anwendungen wie z. B. dem File-Sharing ein neues Netzwerkparadigma dar. Es existieren viele legale Anwendungen wie unlängst entwickelte P2P-basierte Internet Protocol Television (IPTV)-Lösungen wie Zattoo und Joost, durch die sich das P2P-Konzept bereits in der Praxis bewährt hat [Zat07, Adc07]. Für die Realisierung effizienter dezentralisierter Netzwerkinfrastruktur bietet P2P-Technologie somit offensichtlich eine exzellente technische Basis und war somit ebenfalls Gegenstand der Untersuchungen dieser Forschungsarbeit. Als Ergebnis der Untersuchungen werden Prototypen für eine P2P-basierte verteilte Speicher- (P2P-based Storage Platform (PSP)) und Rechenplattform (Distributed Computing System using a Decentralized P2P Environment (DuDE))

für Netzbetreiber und ein P2P-basiertes Domain Name System (P-DONAS) vorgestellt.

1.2. Aufbau dieser Arbeit

Diese Arbeit gliedert sich in sechs Kapitel. Nach der Einleitung in Kapitel 1 folgt in Kapitel 2 eine Darstellung der Grundlagen. Es schließen sich konkrete Anwendungsfälle an. In Kapitel 3 wird ein Algorithmus zur Berücksichtigung netzwerktechnischer Nähe von P2P-Datenverkehr beschrieben. Bevor in Kapitel 5 der Entwurf P2P-basierter Netzwerkinfrastruktur vorgestellt wird, widmet sich Kapitel 4 der Auswahl und Erweiterung des eingesetzten P2P-Protokolls. Schlussendlich wird in Kapitel 6 eine Zusammenfassung der Ergebnisse gegeben, auf die ein Ausblick folgt.

In Abbildung 1.1 ist eine etwas genauere Darstellung der Hauptkapitel dieser Forschungsarbeit ersichtlich. Die wesentlichen eigenen Beiträge sind fett hervorgehoben.

Abbildung 1.1.: Kapitelstruktur der Forschungsarbeit. Die wesentlichen eigenen Beträge sind fett hervorgehoben.

Kapitel 2.

Grundlagen

Dieses Kapitel gibt Aufschluss über für den weiteren Verlauf dieser Arbeit notwendige netzwerktheoretische Grundlagen, erläutert ausgewählte Internet-Basisdienste (grundlegende Netzwerkdienste) und widmet sich P2P-Netzwerken. Als heutzutage wichtigstes weltumspannendes Netz mitsamt seiner angebotenen Dienste wird das Internet für alle Betrachtungen herangezogen. Der Forschungsschwerpunkt liegt dabei auf der Kommunikation in kabelgebundenen Netzen, insbesondere dem Ethernet-basierten **Teilnehmerzugangsnetz (TZN)** (da Ethernet einen weit verbreiteten Standard darstellt) und den **Kernnetzen** der Netzbetreiber. Auf diesem Forschungsgebiet lag der Fokus des Autors während seiner Forschungstätigkeit sowie dem Wirken innerhalb der Kooperation mit dem Industriepartner Nokia Siemens Networks GmbH & Co. KG (Broadband Access Division) in Greifswald mit dem Thema „Ethernetbasierte Zugangsarchitekturen“ [KWD+07, DKT07, DKW+08c, KWD+08a, DKW+08b, KWD+08b, KWD+08c, WKD+08, DKW+09, TDT11, RAP+11].

2.1. Netzwerktheoretische Grundlagen

Nachfolgende netzwerktheoretische Grundlagen vermitteln einen Überblick über die Netzwerkstruktur des Internets und das International Organization for Standardization (ISO)-Open Systems Interconnection Model (OSI)-Referenzmodell. Sie bilden die Basis für alle nachfolgenden Betrachtungen, klären Begrifflichkeiten und erleichtern das Verständnis und die Einordnung der Vielzahl von anschließend erläuterten Protokollen und Diensten.

2.1.1. Netzwerkstruktur des Internets

Das Internet kann in unterschiedlich große Rechnernetze abhängig von deren geographischer Ausdehnung unterteilt werden. Distanzen als Klassifizierungsmerkmal sind von hoher Bedeutung, da abhängig von der Ausdehnung des Netzwerksegments verschiedene Techniken zum Einsatz kommen. Eine entsprechende Klassifizierung ist in Tabelle 2.1 ersichtlich [Tan03].

Geographische Ausdehnung	Einzugsgebiet	Bezeichnung
1 m	Quadratmeter	Personal Area Network (PAN)
10 m	Raum	Local Area Network (LAN)
100 m	Gebäude	
1 km	Campus	
10 km	Stadt	Metropolitan Area Network (MAN)
100 km	Land	Wide Area Network (WAN)
1000 km	Kontinent	
10.000 km	Planet	Global Area Network (GAN)

Tabelle 2.1.: Klassifizierung des Internets in Abhängigkeit von der geographischen Ausdehnung [Tan03].

PANs sind Netze für die Belange einer Person und vernetzen etwa einen Rechner mit seiner Maus, Tastatur und seinem Drucker. Traditionelle LANs haben ein Einzugsgebiet von bis zu einigen Kilometern und vernetzen Personalcomputer oder Workstations mehrerer Unternehmen. Bei der Charakterisierung von LANs sind insbesondere die geringe Fehlerrate, die Beschränkung auf eine einzelne Übertragungstechnologie sowie eine Datenrate von bis zu einem Bereich von 1 GBit/s ausschlaggebend. Das MAN verbindet und bündelt den Datenverkehr aus dem Einzugsgebiet einer Stadt. Es fallen hier üblicherweise Datenraten von bis zu 10 GBit/s an. WANs sind länder- oder Kontinent umfassende Netzwerke und verfügen über Übertragungskapazitäten von mehreren 10 GBit/s. Die weltweite Vernetzung der Kernnetze erfolgt über GANs.

Eine im Vergleich zu Tabelle 2.1 andere Einteilung in Netzsegmente ist in Abbildung 2.1

dargestellt. Ersichtlich ist der Zusammenhang zwischen dem Kundennetz, Zugangsnetz (oft auch „First Mile“ oder „Last Mile“) und Kernnetz. Im Kundennetz befinden sich Endnutzer, deren Netzwerkverkehr durch Residential Gateways (RGs) aggregiert wird. Den Übergang vom Kundennetz (PAN und LAN) in das Kernnetz (MAN, WAN und GAN) bildet das sogenannte **TZN** oder auch **Access Network**. Zugangsknoten (auch ANs) bündeln den Datenverkehr des Kundesnetzes und leiten ihn weiter an Broadband Remote Access Server (BRAS), die wiederum Eintrittspunkte ins **Kernnetz** darstellen. ANs können je nach Art der Übertragungstechnologie z. B. Digital Subscriber Line Access Multiplexers (DSLAMs), Cable Modem Termination Systems (CMTSs), Gigabit Passive Optical Network (GPON) oder Optical Line Terminations (OLTs) sein. Im Kernnetz sorgen über Backbones verbundene Router für die Verteilung von Daten.

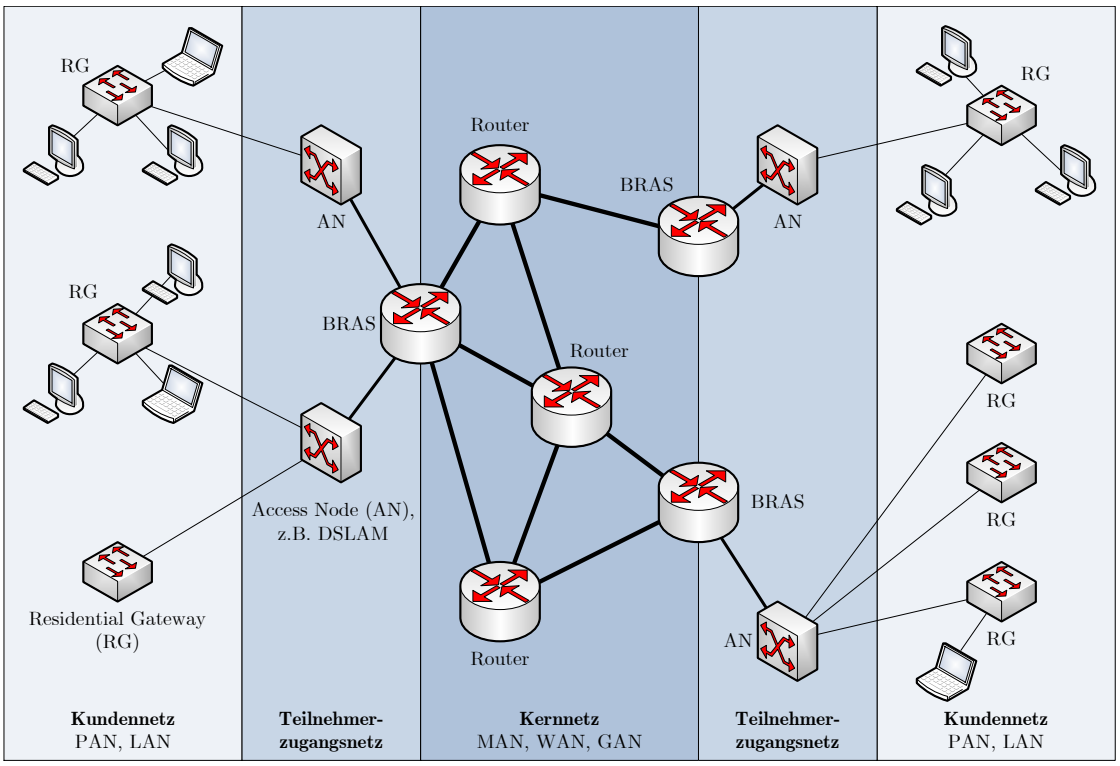


Abbildung 2.1.: Einteilung des Internets in Netzwerksegmente.

2.1.2. ISO-OSI-Referenzmodell

Um Rechnern im weltweiten Internet eine Verbindung und damit Kommunikation zu ermöglichen, hat die ISO als erste Organisation einen formalen Standard entwickelt. Diese Architektur wird OSI-Netzwerkarchitektur genannt und ist in Abbildung 2.2 (a) dargestellt. Sie besteht aus sieben Schichten, auf denen von oben nach unten gesehen der Protocol Data Unit (PDU) - den Anwendungsdaten - der vorherigen Schicht ein Header (auch Protocol Control Information (PCI)) vorangestellt wird. Dieser Header enthält Steuer- und Kontrollinformationen, um Kommunikation zu ermöglichen und abzusichern.

Die Aufgaben der einzelnen Schichten sind im Folgenden kurz zusammengefasst:

1. Auf der *Bitübertragungsschicht* wird die Art der physikalischen Datenübertragung festgelegt. Bei den Daten spricht man auf dieser Schicht von *Bits*. Es werden Schnittstellen zwischen dem Übertragungsmedium und Netzwerkgeräten bereitgestellt sowie optische, elektrische und mechanische Eigenschaften des Übertragungsmediums bestimmt.

2. Die *Sicherungsschicht* ist für die Gewährleistung der Betriebsfähigkeit von Kommunikationsverbindungen zuständig. Dabei sollen auf dieser Schicht Übertragungsfehler erkannt und korrigiert werden. Übertragene Datenstücke werden *Frames* (Rahmen) genannt.

3. Auf der *Vermittlungsschicht* erfolgt die Bestimmung der Übermittlungsart von Daten zwischen Netzwerkgeräten. Die sogenannten *Pakete* werden entsprechend eindeutiger Netzwerkadressen geroutet. Fluss- und Überlastkontrollmechanismen ermöglichen ein bestimmtes Maß an Dienstgüte-Zusicherung bei der Datenübertragung.

4. Die Ende-zu-Ende-Auslieferung von *Segmenten* wird auf der *Transportschicht* verwaltet. Mittels Fehlerbehebungs- und Flusskontrollmaßnahmen kann eine zuverlässige und fortlaufende Übertragung sichergestellt werden. Eine verbindungslose, nicht zuverlässige Kommunikation wird allerdings ebenso unterstützt. In diesem Fall spricht man anstatt von Segmenten auch von *Datagrammen*.

5. Für die Verwaltung von Nutzersitzungen und -dialogen ist die *Sitzungsschicht* zuständig. Hier erfolgt die Kontrolle des Verbindungsaufbaus und -abbaus logischer Verbindungen zwischen Nutzern sowie die Anzeige von Fehlern höherer Schichten.

Ausgetauschte Daten heißen auf dieser Schicht sowie auch auf Darstellungs- und Anwendungsschicht *Nachrichten*.

6. Die *Darstellungsschicht* passt Datenformate an, die auf Grund der Nutzung verschiedener Systeme Unterschiede aufweisen und legt Architektur-unabhängige Datentranferformate fest. Die Kodierung, Dekodierung, Verschlüsselung, Entschlüsselung, Kompression und Dekompression von Daten fällt ebenso in die Zuständigkeit dieser Schicht.
7. Schnittstellen zwischen Nutzerprozessen werden auf der *Anwendungsschicht* definiert, um Kommunikation und Datentransfer im Netzwerk zu ermöglichen. Standardisierte Dienste wie z. B. die virtuelle Datenübertragung werden bereitgestellt.

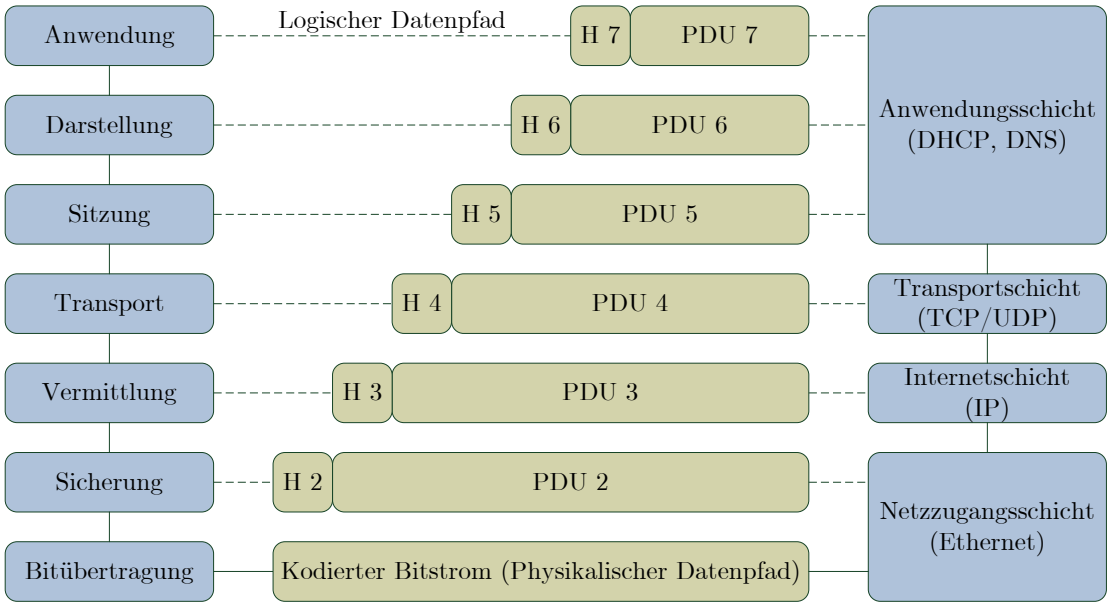
Die jeweiligen Schichten zweier Endknoten kommunizieren logisch direkt miteinander; die eigentliche physikalische Kommunikation erfolgt jedoch stets auf der Bitübertragungsschicht. Abbildung 2.2 (b) zeigt als Gegenüberstellung den in der Praxis verwendeten Internet-Protokollstapel (auch Transmission Control Protocol (TCP)/Internet Protocol (IP)-Protokollstapel) [Hal06,PD97]. Hier werden die sieben Schichten des OSI-Modells in vier Schichten zusammengefasst. Für jede Schicht wurden jeweils ein oder mehrere wichtige Protokolle als Beispiel angegeben.

2.2. Ausgewählte Netzwerkprotokolle

2.2.1. Ethernet

Ethernet ist mit Abstand das meist genutzte Netzwerkprotokoll in der Bitübertragungs- und Sicherungsschicht und wird vorwiegend in LANs verwendet [JT07]. Es ist in der Gruppe der Institute of Electrical and Electronics Engineers (IEEE) 802.3-Standards spezifiziert. Das Ethernet-System besteht aus drei grundlegenden Elementen:

1. Das physikalische Medium zur Übertragung von Ethernet-Signalen zwischen Rechnern.
2. Ein Regelsatz für den Medienzugriff, der in jede Ethernet-Schnittstelle eingebettet ist. Dadurch wird der faire Zugriff auf das geteilte Ethernet-Medium geregelt.
3. Ein Ethernetframe, der aus einer standardisierten Anzahl von Bits besteht, die zur Datenübertragung über das physikalische Medium genutzt werden.



a) OSI-Netzarchitektur, Kapselung von Daten (PDU 2 - 7) und Headern (H 2 - 7)

b) Internet-Protokollstapel

Abbildung 2.2.: (a) OSI-Netzwerkarchitektur und (b) Internet-Protokollstapel [Hal06, PD97].

Als Übertragungsmedium werden Glasfaser- oder Kupferkabel (Twisted Pair, Koaxial) genutzt. Unterstützte Duplex-Operationsmodi sind sowohl Halbduplex als auch Vollduplex. Das Ethernet-Protokoll unterstützt momentan vier verschiedene Datenraten [JT07]:

- 10 MBit/s — Ethernet 10Base-T (802.3i), 10Base2 und 10Base5 (802.3) und 10BaseF (802.3j)
- 100 MBit/s — Fast Ethernet 100BaseX: 100 Base-T, 100Base-F (802.3u)
- 1 GBit/s — Gigabit Ethernet 1000BaseX: 100Base-T (802.3ab), 1000Base-F, 1000Base-LX, LH, SX, CX (802.3z)
- 10 GBit/s — 10 Gigabit Ethernet 10GBaseE, 10GBaseL und 10 GBaseS (802.3ae); 10 GBaseCX4 (802.3ak) und 10GBaseT (802.3an)

Der geteilte Medienzugriff basiert bei der Halbduplex-Übertragung auf dem Carrier Sense Multiple Access (CSMA)/Collision Detection (CD)-Prinzip der Kollisionserkennung.

In der Praxis spielt dieses Prinzip allerdings eine immer geringere Rolle, da meist im Vollduplex-Verfahren über Glasfaser- oder Twisted Pair-Kabel mit separaten Empfangs- und Sendekanälen kommuniziert wird.Dabei übernehmen Switches als Netzwerkelemente der Sicherungsschicht die Zugriffsauflösung und so entstehen keine Kollisionen mehr.

Ein Ethernetframe hat den in Abbildung 2.3 dargestellten Aufbau und besitzt eine Länge von 64 - 1518 Byte. Zur Adressierung dienen Ziel- und Quell-Medium Access Con-

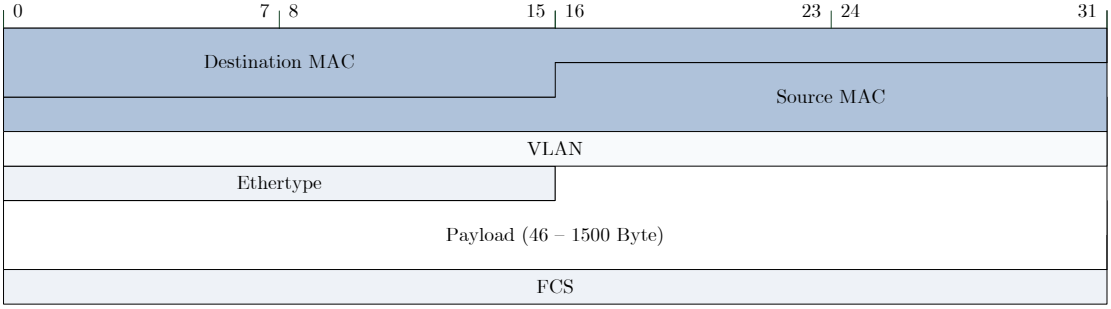


Abbildung 2.3.: Aufbau eines Ethernetframes.

trol (MAC)-Adresse (Destination und Source MAC), die jeweils 6 Byte lang sind. Um die Bildung logischer (virtueller) Netze zu ermöglichen, können darüber hinaus nach der Quell-MAC-Adresse Virtual LAN (VLAN)-Tags (definiert im IEEE 802.1Q-Standard) eingefügt werden. Jedes VLAN-Tag erhöht die Länge eines Ethernetframes um 4 Byte. Der Nutzdatenteil (Payload) ist 46-1500 Byte lang. Um die Minimallänge nicht zu unterschreiten, sind ggf. Padding-Bytes enthalten. Den Abschluss eines Ethernetframes bildet das Frame Check Sequence (FCS)-Feld, das eine Cyclic Redundancy Check (CRC)-Prüfsumme enthält und der Fehlererkennung dient.

2.2.2. IP

IP ist ein Protokoll der Vermittlungsschicht, das Adress- und Kontrollinformationen enthält [JT07]. Es ermöglicht das Routing eines Pakets im Internet. IP stellt das grundlegende Protokoll der Vermittlungsschicht im TCP/IP-Protokollstapel dar. Zusammen mit TCP ist es das wichtigste aller Internet-Protokolle und sowohl für LAN- als auch WAN-Kommunikation geeignet.

IP sorgt für die verbindungslose, sogenannte „Best-Effort“-Übertragung von Paketen durch das Internet. Außerdem werden Mechanismen zur Fragmentierung und Reassemblierung von Paketen bereitgestellt, um Datenleitungen mit unterschiedlichen maximalen Paketgrößen (auch Maximum Transmission Unit (MTU)) zu unterstützen. Das IP-Adressierungsschema bildet die Grundlage für das Routing eines IP-Pakets durch ein Netzwerk. Wie in Abbildung 2.4 ersichtlich ist, besteht eine IP-Adresse der Version 4 (IPv4) aus 4 Byte. Wenn eine Nachricht (z. B. eine Email oder Web-Seite) gesendet oder

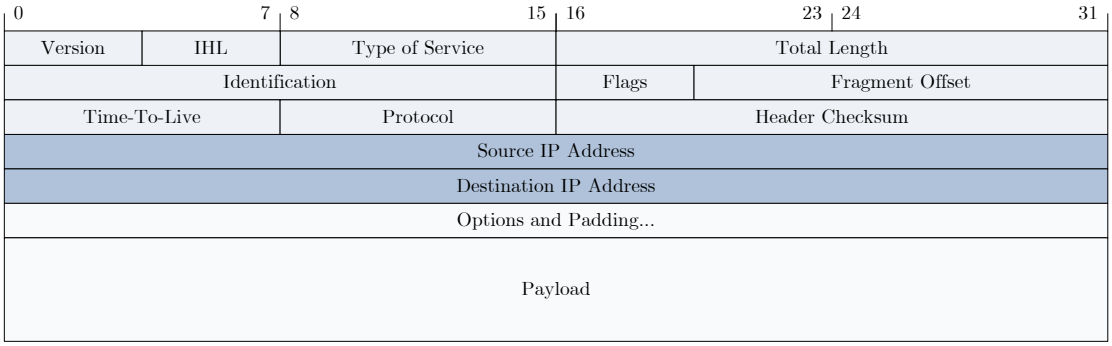


Abbildung 2.4.: Aufbau eines IPv4-Pakets.

empfangen wird, erfolgt eine Aufteilung der Nachricht in IP-Pakete. Ein IPv4-Paket ist 46 - 1500 Byte lang, wobei der IPv4-Header 20 - 60 Byte umfasst. Jedes Paket enthält sowohl Absender- (Source IP Address) als auch Empfängeradresse (Destination IP Address). Nachdem alle Pakete beim Empfänger angekommen sind, werden sie wieder zur Original-Nachricht zusammengesetzt.

Da eine IPv4-Adresse auf 4 Byte begrenzt ist, reicht die Anzahl von ca. 4,3 Milliarden möglichen Adressen nicht mehr für eine eindeutige Adressierung im Internet aus. Deshalb wurde die IP-Version 6 (IPv6) entwickelt, bei welcher eine IP-Adresse 16 Byte lang ist und somit ca. 340 Sextillionen Adressen möglich sind.

2.2.3. UDP

Das User Datagram Protocol (UDP) ist ein verbindungsloses Protokoll der Transportschicht und ermöglicht einen einfachen aber unzuverlässigen Nachrichtendienst mit Hilfe des Austausches von Datagrammen. UDP stellt damit eine Schnittstelle zwischen der

Vermittlungsschicht und höheren Schichten zur Verfügung. Da eine Vielzahl von Netzwerkapplikationen auf dem selben Rechner laufen können, müssen Computer in der Lage sein, die richtige Anwendung auf dem Zielcomputer anzusprechen und ihnen die anfragende Anwendung mitzuteilen. Dies wird über UDP-Protokoll-Ports realisiert („Source Port“ für die anfragende Anwendung und „Destination Port“ für die Anwendung auf dem Zielcomputer, siehe Abbildung 2.5). UDP enthält keine Mechanismen für die Bereitstel-

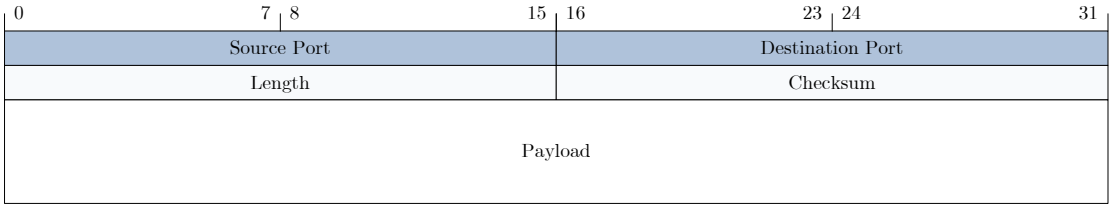


Abbildung 2.5.: Aufbau eines UDP-Pakets.

lung von zuverlässiger Kommunikation, Flusskontrolle oder Fehlerbehebungsmaßnahmen für IP-Pakete. Durch die Einfachheit von UDP wird allerdings weniger Overhead erzeugt als z. B. bei TCP. UDP kommt somit vorwiegend zum Einsatz, wenn zuverlässige Kommunikation nicht notwendig ist und z. B. höhere Schichten für Fehlerbehebung und Flusskontrolle sorgen.

2.2.4. TCP

TCP ist das Protokoll der Transportschicht, welches eine zuverlässige Datenübertragung und einen betriebssicheren virtuellen Verbindungsdienst für Anwendungen zur Verfügung stellt. Dabei arbeitet TCP mit aufeinander folgenden Empfangsbestätigungen für übertragene Bytes. Bytes, die nicht innerhalb eines spezifizierten Zeitraums bestätigt wurden, werden erneut übertragen. Wie bei UDP dienen auch bei TCP Ports („Source Port“ und „Destination Port“ in Abbildung 2.6) zur Kommunikation von zwei Anwendungen. Die Kombination von der IP-Adresse eines Geräts und seiner Port-Nummer wird als Endpunkt oder Socket bezeichnet. Um zuverlässige Kommunikation und Datenübertragung zu ermöglichen, baut TCP Verbindungen zwischen zwei Endpunkten bzw. Sockets auf.

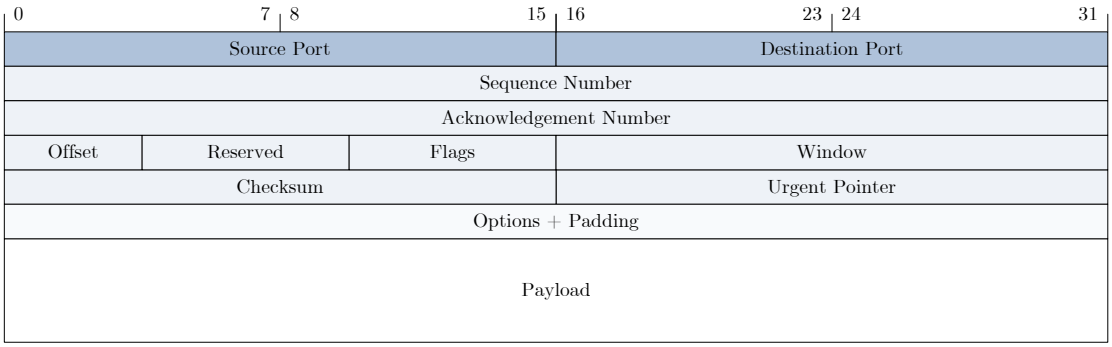


Abbildung 2.6.: Aufbau eines TCP-Pakets.

2.3. Ausgewählte Internet-Basisdienste

2.3.1. DHCP

Dynamic Host Configuration Protocol (DHCP) ist ein Kommunikationsprotokoll, das es Systemadministratoren ermöglicht, die IP-Adressvergabe zentral zu handhaben und zu automatisieren [JT07]. In einem IP-Netzwerk braucht jedes Gerät eine eindeutige IP-Adresse, wenn es sich mit dem Internet verbinden möchte. Mittels DHCP bekommt ein Computer automatisch eine IP-Adresse, wenn er sich mit dem Netzwerk verbindet.

Dabei bedient sich DHCP des Konzepts der „Lease Time“, d.h. eine IP-Adresse besitzt eine zeitlich begrenzte Gültigkeit. Der Wert für die „Lease Time“ kann abhängig davon variieren, wie lange ein Nutzer voraussichtlich eine Internet-Verbindung benötigt. Dieser Mechanismus ist besonders nützlich in Bildungs- und anderen Einrichtungen, da Nutzer in diesen Bereichen oft wechseln. Mit Hilfe kurzer „Lease Times“ kann DHCP dynamisch Netzwerke rekonfigurieren, in denen mehr Computer als verfügbare IP-Adressen sind. DHCP-Pakete werden über UDP übertragen. Ein DHCP-Server verwendet dafür den Port 67; ein Client den Port 68. Für Aufbau eines DHCP-Pakets sowie weitere Details bezüglich der Bedeutung der einzelnen Felder und dem Ablauf des Protokolls sei an dieser Stelle auf [JT07] sowie Anhang A verwiesen.

2.3.2. DNS

Das DNS ist ein spezieller Dienst, der im Internet verfügbar ist. Die meisten Internet-Anwendungen nutzen DNS. Die hauptsächliche Aufgabe von DNS ist es, für den Menschen lesbare Domainnamen auf IP-Adressen abzubilden, die Computer verstehen.

Das DNS ist als eine verteilte Datenbank auf weltweit verstreuten Nameservern zu verstehen, wobei diese untereinander über Verweise (sogenannte Delegierungen) verknüpft sind. Jeder Nameserver besitzt Zonendateien, die sogenannten Resource Records (RRs). Ein grundlegendes DNS kann dabei bereits durch Address RRs (A RRs) und Nameserver RRs (NS RRs) gebildet werden. A RRs dienen der Zuordnung einer IP-Adresse zu einem angefragten Domainnamen. Über NS RRs wird die Verknüpfung der Nameserver untereinander vorgenommen. Um eine DNS-Anfrage (DNS-Request) zu beantworten, müssen oft mehrere Nameserver kontaktiert werden, bis das angefragte RR gefunden wird. Dabei sind DNS-Nameserver sehr ungleichmäßig verteilt [Gib07].

Obwohl es möglich ist, Internet-Ressourcen direkt ohne DNS anzusprechen, indem die IP-Adresse direkt eingegeben ist, wird diese Art von Zugriff allgemein nicht vorgenommen. IP-Adressen sind schwer zu merken und ändern sich zudem oft ohne Ankündigung.

DNS-Komponenten: Das DNS besteht aus drei Komponenten. Diese sind der Domain-Namensraum, die Nameserver sowie der Resolver. In Abbildung 2.7 ist der hierarchisch aufgebaute DNS-Namensraum anhand des Beispiels www.yahoo.com. dargestellt. Die Hierarchie beginnt mit dem sogenannten root ".". Dieser wird oft nicht mit dargestellt. Es schließt sich die Top-Level Domain (TLD) bzw. First-Level Domain an (im Beispiel com). Es folgt die Second-Level Domain (yahoo). www ist im Beispiel die Third-Level Domain. Bei den TLDs unterscheidet man zwischen Generic TLD (gTLD) (z. B. .com, .net und .org) und Country Code TLD (ccTLD) wie .de, .se und .ru.

Nameserver sind Programme bzw. Rechner zur Beantwortung von DNS-Anfragen. Es wird zwischen autoritativen Nameservern, die in einem Start of Authority RR (SOA RR) angegeben sind und nicht-autoritativen Nameservern unterschieden. Autoritative Nameserver sind verantwortlich für eine bestimmte Zone (Teil des Domainnamensraums) und enthalten gesicherte Informationen. Nicht-autoritative Nameserver hingegen beziehen Infos von anderen Nameservern. Sie enthalten deshalb nicht gesicherte Informationen. Zudem cachen sie angefragte Informationen, die ggf. nicht auf dem neuesten Stand sind.

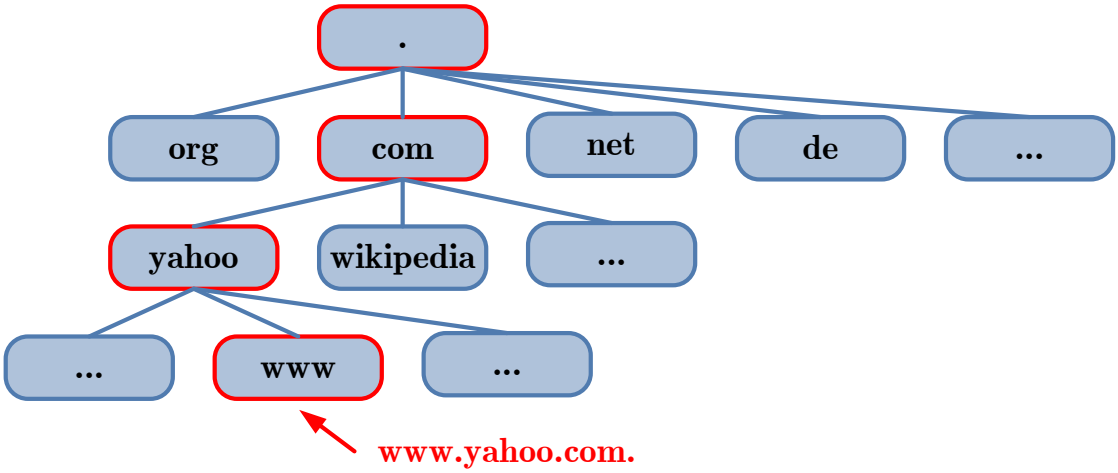


Abbildung 2.7.: Schematische Darstellung der DNS-Hierarchie. Als Beispiel ist der Domainname www.yahoo.com. dargestellt.

Es gibt verschieden Strategien, die nicht-autoritative Nameserver anwenden, um Anfragen zu beantworten. Zum einen ist dies die Delegierung. Dabei wird eine Anfrage an den zuständigen Nameserver für eine Subdomain weitergereicht. Weiterhin gibt es die Weiterleitungsstrategie (Forwarding). Dabei wird eine Anfrage an einen fest konfigurierten Nameserver weitergeleitet. Außerdem kann eine Anfrage über die sogenannten Root-Server aufgelöst werden. Dies passiert, wenn kein Weiterleitungsserver bekannt ist oder keine Antwort für eine Anfrage erhalten wurde. Die weltweit verteilten DNS-Root-Server sind die höchste Instanz, die kontaktiert werden kann, um eine DNS-Anfrage zu beantworten. Sie speichern Delegierungen zu den zuständigen Nameservern, um eine Anfrage zu beantworten. Im Mai 2006 gab es 117 Root-Server (siehe Abbildung 2.8) [Gib07]. Allerdings stehen nur 13 IP-Adressen für diese Root-Server zur Verfügung. Mittels Anycast wird deshalb der lokal nächste Root-Server ausgewählt.

Resolver sind Software-Module der DNS-Teilnehmer und -Nameserver. Sie bilden die Schnittstelle für die Bearbeitung einer Anfrage. Eine Anfrage kann iterativ oder rekursiv ablaufen (siehe Abbildung 2.9). Zunächst möchte eine Anwendung (z. B. ein Web-Browser) eine DNS-Anfrage stellen und kontaktiert dafür per System Call den lokalen Resolver (1.). Dieser reicht die Anfrage rekursiv an einen vorkonfigurierten DNS-Nameserver weiter (2.), wenn kein Eintrag für die Anfrage im Cache vorliegt. Der vorkonfigurierte

Abbildung 2.8.: Weltweite Verteilung der DNS-Root Server [Gib07].

Nameserver bekommt die Antwort erst, nachdem er iterativ zwei weitere Nameserver befragt hat (3.-5.), von denen der zweite die Antwort auf die Anfrage kennt (6.). Die Antwort wird im Cache des konfigurierten Nameservers gespeichert, um die gleiche Anfrage selbst beantworten zu können. Diese Antwort wird über den Resolver an die Anwendung weitergereicht (7.-8.).

DNS-Protokoll: DNS-Pakete sind zumeist in UDP eingebettet, können aber auch über TCP übertragen werden. Die Übertragung erfolgt über Port 53.

Für den Aufbau von DNS-Paketen und DNS-Updates sowie weitere Details sei an dieser Stelle auf Anhang B sowie [Moc87] verwiesen.

Sicherheitsprotokolle für das DNS-Protokoll: Zur Sicherstellung der Authentizität und Integrität von DNS-Transaktionen wurden Domain Name System Security Extensions (DNSSEC) und Transaction Signature (TSIG) zur Erweiterung von DNS entwickelt, die an dieser Stelle nur kurz angeführt werden. Für weiterführende Information über diese Sicherheitsprotokolle sei auf [Eas97, Vix00] verwiesen.

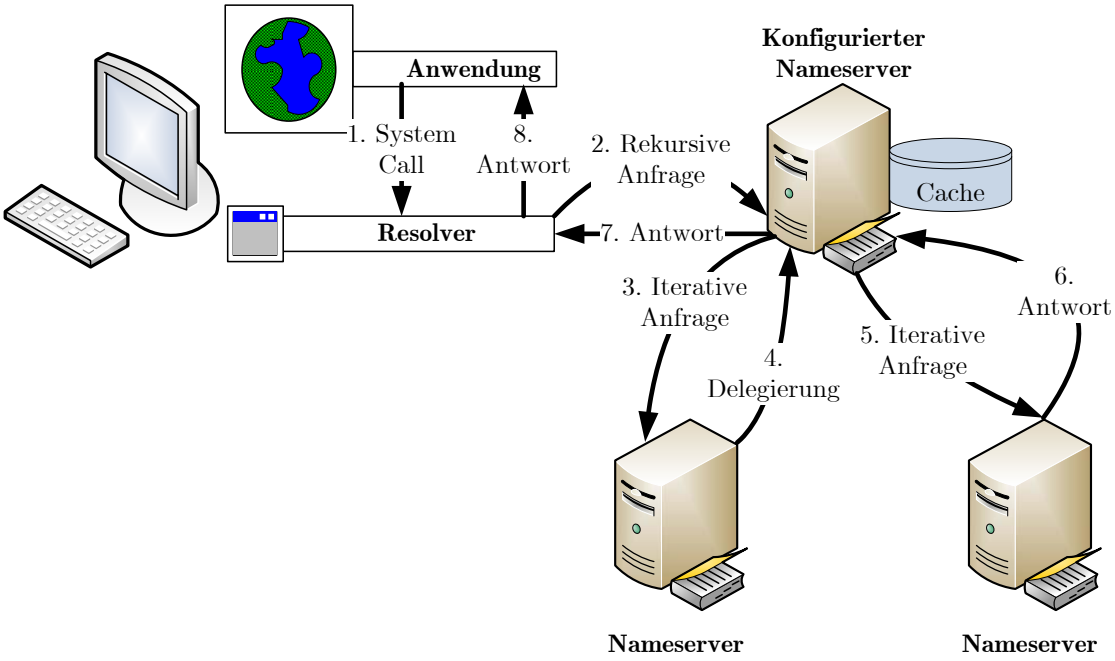


Abbildung 2.9.: Ablauf einer DNS-Anfrage.

Bei DNSSEC werden Daten weiterhin im Klartext übertragen. Mittels Validierung von Signaturen wird die Authentizität und Integrität von DNS-Transaktionen gewährleistet. Dabei wird für die Signierung von RRs ein asymmetrisches Kryptographieverfahren verwendet und Signaturen werden fest in Konfigurationsdateien gespeichert. Jedes durch einen DNS-Client angefragte RR einer Zone wird vom Master-Nameserver mittels seines Private Key signiert und über das sogenannte RRSIG RR an den DNS-Client übertragen. Dabei muss der anfragende DNS-Client seine Anfrage nicht signieren. Der DNS-Client kann mittels Public Key des Master-Nameservers die Signatur eines RR und damit das RR selbst validieren. Der Public Key wird über das DNSKEY RR an den DNS-Client übertragen und kann ebenfalls validiert werden. Zur Validierung eines Public Key, das in einem DNSKEY RR empfangen wurde, wird das sogenannte DS RR genutzt. Angefangen von der höchsten Nameserver-Instanz, der Root-Zone, enthält jede Instanz ein DS RR, das die Signatur des Public Key einer Subzone enthält. So verfügt z. B. die Root-Zone über ein DS RR mit der Signatur des Public Keys der TLD "de". Der Public Key der Root-Zone selbst kann nicht validiert werden, da sie die höchste Instanz darstellt.

Auch bei TSIG ist keine Verschlüsselung des Datenverkehrs vorgesehen. Authentizität und Integrität von DNS-Transaktionen wird bei TSIG ebenfalls über die Validierung von Signaturen vorgenommen. Die Signaturen werden im Gegensatz zu DNSSEC allerdings mit Hilfe eines symmetrischen Kryptographieverfahrens dynamisch zur Laufzeit erstellt. Jedes durch einen DNS-Client angefragte RR einer Zone wird vom Master-Nameserver mittels seines Private Key signiert und über das sogenannte TSIG RR an den DNS-Client übertragen. Dabei muss der anfragende DNS-Client seine Anfrage ebenfalls signieren. Der DNS-Client kann mittels Private Key des Master-Nameservers die Signatur eines RR und damit das RR selbst validieren. Der Private Key muss somit sowohl Master-Nameserver als auch DNS-Client bekannt sein.

Vergleicht man beide Mechanismen ist festzustellen, dass bei DNSSEC die Schlüsselverteilung einfacher als bei TSIG ist. Allerdings werden Denial of Service (DoS)-Attacken durch DNSSEC erleichtert, da ein höherer Rechenaufwand anfällt. Die Verteilung der Schlüssel zur Bildung von Vertrauensketten (Chains of Trust) erfolgt manuell und bietet damit Angriffspunkte. Mittels DNSSEC-Walking können Zoneninhalte vollständig ausgelesen werden. Da Stubresolver nicht selbst die DNS-Antworten validieren, kann ein Angriff auf der Kommunikationsstrecke zu ihrem rekursiven Nameserver erfolgen. Auch kann der rekursive Nameserver selbst kompromittiert sein. Stubresolver sind herkömmliche DNS-Resolver, die normalerweise nicht in der Lage sind, signierte RRs zu validieren. Stattdessen werden die DNSSEC-Funktionen in einem zentralen rekursiven Nameserver gehalten. Ein Client, der einen Namen auflösen möchte, sendet eine entsprechende Anfrage an diesen zentralen Server. Durch Setzen des DO-Bits im DNS-Header teilt er mit, dass authentifiziert werden soll. Bei erfolgreicher Authentifizierung setzt der zentrale Nameserver in der Antwort das AD-Bit (Authenticated Data). Die Handhabung von TSIG ist einfacher als von DNSSEC. Da die Schlüsselverteilung allerdings aufwändig ist, bietet sich TSIG eher in Umgebungen mit wenigen Servern bzw. Clients an.

2.4. P2P-Netzwerke

P2P ist ein Netzwerkparadigma, das einen Gegensatz zum traditionellen Client-Server-Modell darstellt und mit dem Aufkommen von Napster 1999 große Berühmtheit erlangte. Das Client-Server-Modell beruht auf der Trennung von Server und Clients (siehe Abbildung 2.10). Der Server (rot umrandet in Abbildung 2.10) ist die zentrale Instanz mit

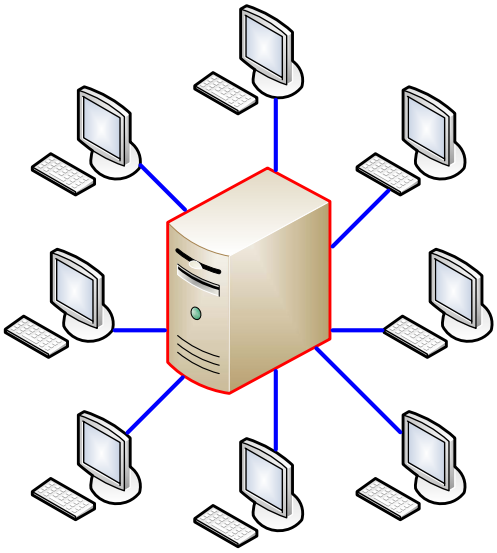


Abbildung 2.10.: Traditionelles Client-Server-Netzwerkmodell.

hoher Performance und stellt Dienste und Inhalte bereit. Clients (in Abbildung 2.10 schwarz umrandet) sind in der Regel Rechner mit vergleichsweise niedriger Performance und nehmen vom Server angebotene Dienste und Inhalte in Anspruch.

Das P2P-Netzwerkmodell umfasst hingegen keine Server, sondern kennt nur Peers. Abbildung 2.11 zeigt ein vollständig dezentralisiertes P2P-Netz ohne zentrale Instanz, bei dem alle Peers miteinander verbunden sind. Alle Peers besitzen sowohl Client- als auch Server-Funktionalität und werden deshalb auch Servents genannt. Sie teilen ihre Speicher- und Rechenressourcen und weisen in der Regel eine heterogene Performance auf. Dabei sind hohe Skalierbarkeit und hohe Ausfallsicherheit intrinsische Eigenschaften von P2P-Netzen und heben sie deutlich von einem reinen Client-Server-Modell ab. In einem vollständig dezentralisierten P2P-Netz gibt es keinen Single Point of Failure (SPoF), den in einem Client-Server-Modell der Server darstellt. Deshalb bietet es eine hohe Widerstandsfähigkeit gegenüber DoS-Attacken und Netzwerkfehlern. Diese Eigenschaften qualifizieren P2P-Netzwerke für die Realisierung effizienter dezentralisierter Netzwerkinfrastruktur, um existierende zentralisierte und kostenintensive Lösungen zu ersetzen oder zumindest zu ergänzen. Dieser Aspekt bildet demzufolge einen Hauptpunkt dieser Forschungsarbeit.

Aus Sicht des ISO-OSI-Referenzmodells ist P2P eine Overlay-Technologie in der Anwendungsschicht, die ein logisches Overlay auf die existierende Internet-Topologie (physi-

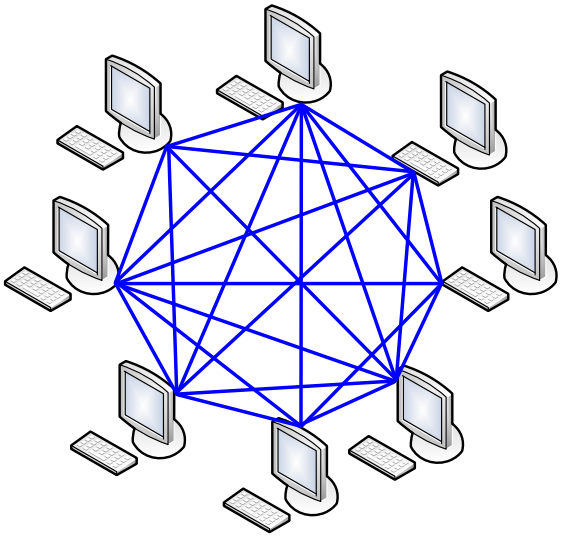


Abbildung 2.11.: Vollständig dezentralisiertes P2P-Netzwerkmodell.

kalisches Underlay) aufsetzt (siehe Abbildung 2.12). Damit erbt P2P die Eigenschaften des Mediums Internet mit all seinen Vor- und Nachteilen. Auf der Vermittlungsschicht sind - verbunden mit dem IP-Protokoll - verschiedene Eigenschaften und Maßnahmen zu nennen, die Einfluss auf P2P haben und für diese Arbeit von Relevanz sind. Dies sind z. B. - aufsetzend auf das IP-Protokoll - Network Address Translation (NAT), Port Address Translation (PAT) und Firewalls, die direkte P2P-Verbindungen be- oder verhindern können, da Rechner nicht kontaktiert werden können, ohne dass sie vorher von sich aus Kontakt aufnehmen [Sch06]. Um betroffenen Peers dennoch eine Kommunikation im P2P-Netz zu ermöglichen, sind geeignete Methoden zu wählen und wurden deshalb in dieser Forschungsarbeit untersucht.

Für das Routing auf dem logischen P2P-Overlay wird das physikalische Underlay ohne zusätzliche Mechanismen nicht berücksichtigt. Logische Nachbarn auf dem P2P-Overlay, die miteinander kommunizieren, sind nicht unbedingt auch auch auf dem Underlay dicht beieinander (siehe z. B. Peer 1 und 2 in Abbildung 2.12). Somit kann es zu unnötiger Kommunikation zwischen Peers über lange Distanzen kommen, die viele Netzwerkressourcen verbraucht. Um diese Fehlanpassung zwischen logischem Overlay und physikalischem Underlay zu vermindern und somit Netzwerkressourcen zu sparen, wurde in dieser Forschungsarbeit ein Algorithmus für eine verbesserte Peer-Auswahl entwickelt.

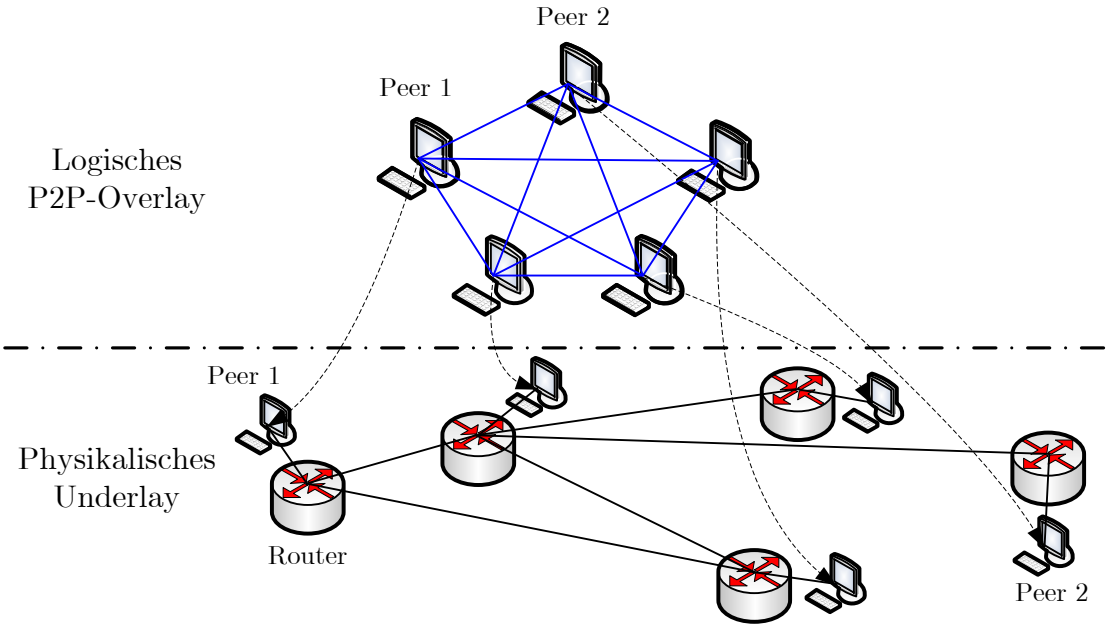


Abbildung 2.12.: P2P-Overlay und die Entsprechung auf dem physikalischen Underlay (Beispiel).

P2P-Protokolle setzen - je nach Anwendungsfall - auf TCP oder UDP auf. TCP ist zuverlässig, hat aber einen größeren Overhead als z. B. UDP. UDP ist hingegen schnell, aber unzuverlässig.

2.4.1. Klassifizierung von P2P-Systemen

P2P-Systeme werden im Wesentlichen in unstrukturierte und strukturierte System unterschieden (siehe Abbildung 2.13) [SW05]. Unstrukturierte Systeme umfassen zentralisierte Netzwerke, dezentralisierte Netzwerke und hybride Lösungen. Bei diesen Systemen haben Daten und Peers keinen Bezug zueinander. Daten sind also unstrukturiert im P2P-Netzwerk verteilt.

Strukturierte Systeme basieren auf den sogenannten Distributed Hash Tables (DHTs). Sowohl Peers als auch Daten erhalten einen Hashwert aus dem selben Adressraum. Peers verwalten Daten mit einem Hashwert ähnlich dem ihren und somit liegen Daten strukturiert im P2P-Netzwerk. Die Definition der Ähnlichkeit hängt dabei vom eingesetzten

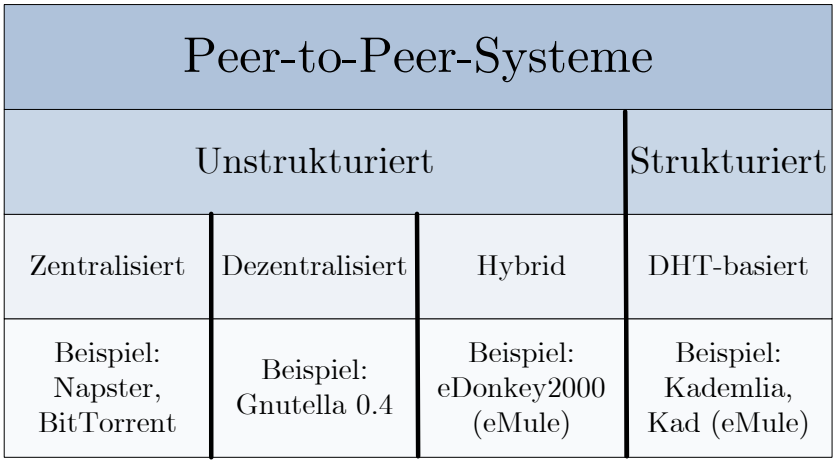


Abbildung 2.13.: Klassifizierung von P2P-Systemen.

DHT-Protokoll ab.

2.4.2. P2P-Systeme

2.4.3. Unstrukturierte, zentralisierte P2P-Systeme (BitTorrent)

In einem unstrukturierten, zentralisierten P2P-System existiert ein zentraler Server, der alle Peers kennt. Suchanfragen nach Daten werden an den zentralen Server gestellt, der Peers zurück liefert, die diese Daten haben. Der P2P-Aspekt bei diesem Netzwerk ist die direkte Datenübertragung zwischen den Peers. Ein Vertreter für ein solches P2P-System ist der File-Sharing-Client Napster, der einer der ersten Dienste dieser Art war. Auch das sehr populäre File-Sharing-Protokoll **BitTorrent** in seiner ursprünglichen Tracker-basierten, unstrukturierten Form ist in diese Kategorie einzuordnen [Coh03].

Das BitTorrent-Netzwerk: Bei BitTorrent wird für jede Datei ein temporäres P2P-Netz gebildet. Jede Datei wird in 256 KByte große Stücke (Pieces) unterteilt. Für jedes Stück existiert eine Prüfsumme, um eine Verifikation zu ermöglichen. Die Stücke werden weiter in Teilstücke (Sub-Pieces) unterteilt. In Abbildung 2.14 ist die Funktionsweise des BitTorrent-Protokolls dargestellt. Während des ersten Schritts lädt Peer k eine .torrent-Datei von einem Web-Server herunter. Diese Datei enthält die Adresse eines sogenannten

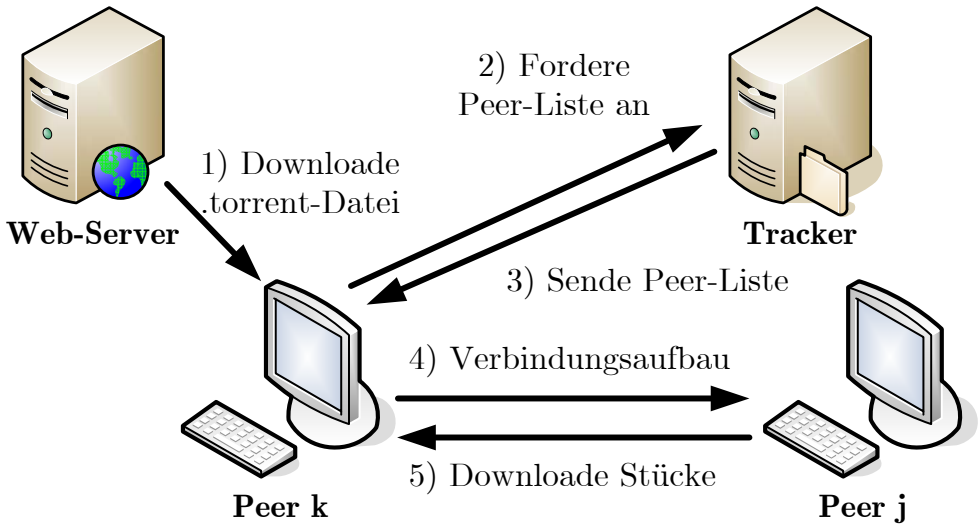


Abbildung 2.14.: Veranschaulichung der BitTorrent-Funktionsweise.

Trackers, Dateiname und -größe sowie die Namen der Stücke und Prüfsummen. Anschließend wird der angegebene Tracker kontaktiert, um eine Liste mit Peers anzufordern, die über die gewünschte Datei verfügen. Im dritten Schritt liefert der Tracker eine solche Liste, die zufällig ausgewählte Peers und Statusinformationen enthält. Mit diesen Peers (im Beispiel Peer j) wird Kontakt aufgenommen und verfügbare Stücke werden heruntergeladen.

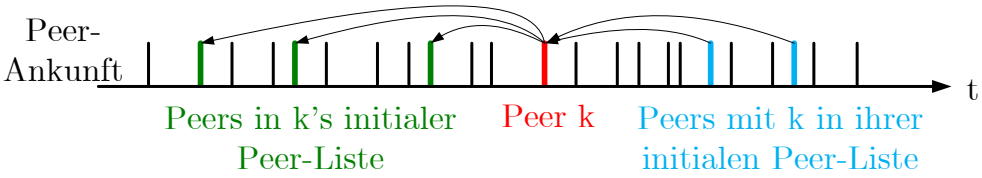


Abbildung 2.15.: Kontaktaufnahme von Peers bei BitTorrent.

Peer k kann zunächst nur mit Peers kommunizieren, die in seiner Peer-Liste enthalten waren. Allerdings ist er auch in folgenden Peer-Listen enthalten, die später eintreffende Peers bekommen (siehe Abbildung 2.15). Somit kann Peer k sowohl mit Peers kommunizieren, die in seiner Liste enthalten waren als auch mit Peers, die ihn später kontaktieren.

In Abbildung 2.16 ist der Ablauf des Datenaustauschs zwischen Peers ersichtlich. Als

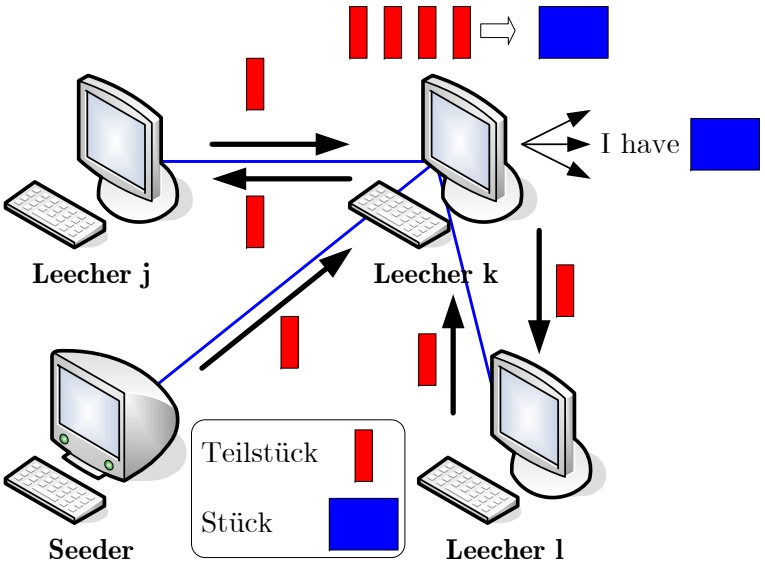


Abbildung 2.16.: Ablauf des Datenaustauschs zwischen Peers bei BitTorrent.

Leecher werden Peers bezeichnet, die die gewünschte Datei noch nicht komplett besitzen und somit noch downloaden. Ein Seeder ist ein Peer mit der kompletten Datei. Alle an einer Datei interessierten Peers nennt man Schwarm. Während des Datei-Downloads erfolgt der parallele Bezug von Teilstücken. Ist ein Stück vollständig, wird es über die Prüfsumme verifiziert. Komplett heruntergeladene Pieces werden anderen Peers als zum Download bereit angezeigt („I have“).

Peers beginnen das Herunterladen von Stücken in zufälliger Reihenfolge (Random) und laden nach der Fertigstellung des ersten Stückes anschließend am seltensten vertretende Stücke (Rarest First) herunter. Dabei verfolgen sie die „Strict Priority“-Strategie, bei der nur Teilstücke eines bestimmten Stückes heruntergeladen werden, bevor Teilstücke des nächsten Stückes angefordert werden. Außerdem gibt es den sogenannten Endgame-Modus, der der schnellen Beendigung eines Downloads dient. Dieser wird aktiviert, sobald ein Download fast vollständig ist. In diesem Modus fordern Peers gleichzeitig alle noch fehlenden Teilstücke von allen bekannten Peers an.

Die BitTorrent-Strategie für einen schnellen Datei-Download basiert auf dem Aufbau mehrerer gleichzeitiger bidirektionaler Verbindungen. Um dabei ein faires Geben und Nehmen unter den Peers zu gewährleisten, werden egoistische Peers, die nur downloaden,

bestraft: Sie werden ge“choke“t [Coh03]. „Choken“ bedeutet, einem anderen Peer den Download zu verweigern. Umgekehrt heißt „unchoke“, dass der Download gewährt wird. Der Choking-Algorithmus funktioniert so, dass nur zu Peers hochgeladen wird, die selbst bereits Daten gegeben haben. Dabei wird fortwährend die Upload-Performance gemessen, um die besten Peers auszuwählen; die Entscheidung erfolgt alle 10 s neu. Um aber auch ungenutzten Verbindungen eine Chance zu geben, gibt es das Konzept der „Optimistic Unchokes“, das alle 30 s angewendet wird. Dabei erhalten zufällig ausgewählte Peers unabhängig von ihrer Upload-Performance eine Chance, Daten herunterzuladen. Darüber hinaus findet als Teil des Choking-Algorithmus ein Anti-Snubbing-Verfahren Anwendung. Peers, die kein einziges Stück innerhalb einer Minute zur Verfügung gestellt haben, wird dabei der Download mit Ausnahme von Optimistic Unchokes verwehrt.

Nachdem ein Peer seinen Download beendet hat, kann dieser eine Weile im Netzwerk verbleiben (Lingering). Während dieses Zeitraums werden von diesem Peer nur Stücke zu anderen Peers hochgeladen, wobei Peers bevorzugt werden, zu denen die beste Upload-Rate besteht.

2.4.4. Unstrukturierte, dezentralisierte P2P-Systeme

In unstrukturierten, dezentralisierten P2P-Systemen gibt es keinen zentralen Server. Peers besitzen Nachbarschaftslisten mit angrenzenden Peers, aber keinerlei Routing-Informationen. Für Suchanfragen nach Daten wird das Netzwerk deshalb geflutet. Damit Anfragen nicht unendlich lange zirkulieren, wird der Suchraum begrenzt, indem die Suchanfrage nach einer bestimmten Anzahl von Hops (passierten Peers) verworfen wird. Ist ein Ziel-Peer für das gewünschte Datum gefunden, erfolgt auch hier die Datenübertragung direkt zwischen den Peers. Ein Beispiel für dieses System ist das Gnutella-Protokoll der Version 0.4.

2.4.5. Unstrukturierte, hybride P2P-Systeme

Bei unstrukturierten, hybriden P2P-Systemen erfolgt die Einführung von Super-Peers, die angeschlossene Peers kennen und wiederum andere Super-Peers. Suchanfragen werden an einen Super-Peer gestellt, der diese ggf. an andere Super-Peers weiterleitet. Die Datenübertragung erfolgt direkt zwischen den Peers. Das im populären File-Sharing-Client eMule integrierte Protokoll eDonkey2000 ist ein Vertreter dieses Systems.

2.4.6. Strukturierte, Distributed Hash Table-basierte P2P-Systeme (Kad)

Für die strukturierte Datenhaltung werden DHTs eingesetzt, die sowohl Peers als auch Daten einen Hashwert aus dem gleichen Adressraum zuordnen. Der Hashwert eines Peers wird z. B. aus der IP-Adresse des Peers gebildet; der Hashwert eines Datums z. B. aus dem Dateinamen. Peers und Daten werden entsprechend ihres Hashwertes auf einem Ring angeordnet, der durch die Menge aller möglichen Hashwerte 0 ... 2^{n-1} gebildet wird (siehe Beispiel in Abbildung 2.17). Peers sind für Daten verantwortlich, die einen ähnlichen Hashwert wie sie selbst aufweisen (gestrichelt umrandeter Bereich in Abbildung 2.17). Die Funktion, die zur Hashwertberechnung eingesetzt wird (z. B. Message-Digest

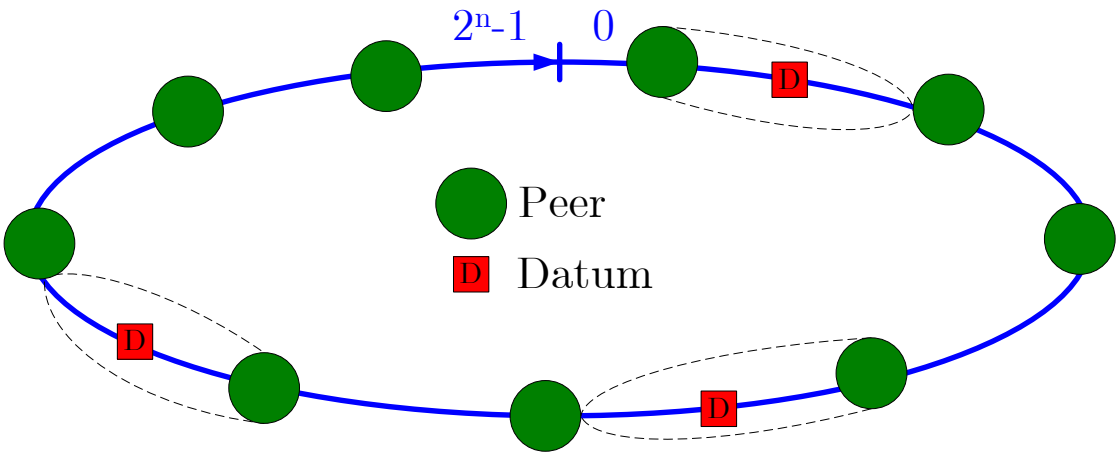


Abbildung 2.17.: Beispiel für einen DHT-Ring mit einigen Peers und Daten. Die gewählten Zuständigkeitsbereiche der Peers für Daten dienen nur der Veranschaulichung.

Algorithm 4 (MD4) oder Message-Digest Algorithm 5 (MD5)) sowie die Bitbreite n des Hashwerts sind abhängig vom eingesetzten Protokoll. Ebenso verhält es sich mit der Definition der Ähnlichkeit, das heißt, ab welcher Ähnlichkeit von Hashwerten ein Peer für ein bestimmtes Datum zuständig ist.

Jeder Peer enthält eine Routing-Tabelle, die Kontaktinformationen über *einige* (nicht alle) andere Peers enthält. Abhängig vom gesuchten Hashwert des Datums wird ein Peer mit einem möglichst ähnlichen Hashwert aus der Routing-Tabelle kontaktiert. Suchanfra-

gen werden so (möglicher Weise über mehrere Peers) an Peers weitergegeben, die einen ausreichend ähnlichen Hashwert wie das gewünschte Datum haben, also in einer vordefinierten Suchtoleranz liegen. Ein Vertreter für dieses System ist Kademlia bzw. die Kademlia-Implementierungsvariante **Kad**, die im File-Sharing-Client eMule eingesetzt wird [MM02, Bru06].

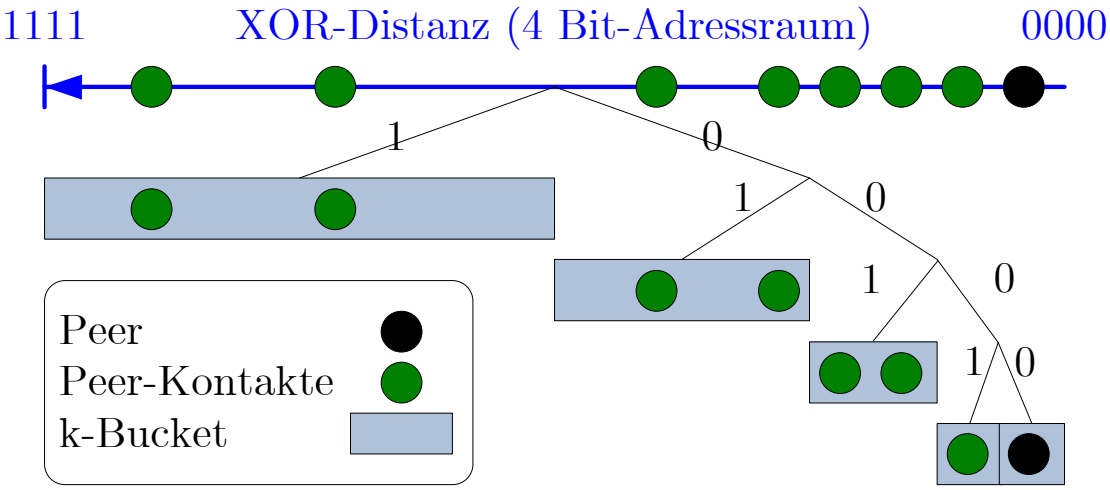


Abbildung 2.18.: Beispiel für eine Routing-Tabelle eines Peers mit einem 4 Bit-Adressraum im Kad-Netzwerk. Alle Kontakte des Peers ordnen sich entsprechend der XOR-Distanzen zu diesem Peer in entsprechende Bereiche—die sogenannten k-Buckets—ein. Da der Peer zu sich selbst eine XOR-Distanz von Null aufweist, befindet er sich ganz rechts in der Routing-Tabelle.

Das Kad-Netzwerk: Kad weist einen n = 128 Bit breiten Adressraum auf. Das heißt, jedem Peer bzw. Datum wird ein 128 Bit breiter Hashwert zugewiesen, der in der Originalimplementierung mit MD4 berechnet wird. Die Distanzen zwischen zwei Peers werden durch die XOR-Metrik berechnet. Am Beispiel von zwei 4 Bit breiten Hashwerten h_1 = 0100 und h_2 = 1101 ergibt sich h_1 XOR h_2 = 0100 XOR 1101 = 1001. Das Ergebnis 1001 = 9 ist in diesem Beispiel die logische Distanz zwischen den beiden Hashwerten h_1 und h_2. Man spricht in diesem Zusammenhang auch von einem Prefix-Matching-Verfahren, da die logische Distanz umso geringer ist, je mehr führende Bits zweier Hashwerte gleich

sind. Ebenso werden auch die Distanzen zwischen Daten und Peers berechnet. Sei der Hashwert einer Datei $d_1 = 0101$ (beispielsweise berechnet aus dem Dateinamen). Somit ergibt sich der logische Abstand zu einem Peer mit dem Hashwert $h_1 = 0100$ mit h_1 XOR $d_1 = 0100$ XOR $0101 = 0001 = 1$. Dieser Peer gilt in diesem Beispiel als Ziel für die Datei, wenn die Suchtoleranz auf einen Wert größer oder gleich 1 eingestellt ist.

Die Routing-Tabelle eines Peers ist als Binärbaum organisiert, in die ein Peer die XOR-Distanzen anderer Peers zu seinem eigenen Hashwert einträgt (siehe Abbildung 2.18). Somit befindet sich der Peer selbst ganz rechts in seiner Routing-Tabelle, da er zu sich selbst eine XOR-Distanz von Null aufweist. Der Binärbaum ist in Bereiche, die sogenannten k-Buckets, eingeteilt, in denen jeweils bis zu k Peers Platz finden (in der ursprünglichen Kad-Implementierung ist k=10). Die Einteilung wird so vorgenommen, dass ein Peer viele nahe aber nur wenige entfernte Peers kennt. Darüber hinaus besitzt jeder Peer eine Lebenszeit. Um dem Kad-Netzwerk beizutreten, führt ein Peer den Vorgang des Bootstrappings aus. Dazu kontaktiert er einen bekannten Peer, wird dadurch in dessen Routing-Tabelle eingefügt und lernt in der Antwort weitere Peers kennen. Ein Wartungsmechanismus sorgt dafür, dass Peers mit abgelaufener Lebenszeit kontaktiert werden und bei einem Ausbleiben der Antwort aus der Routing-Tabelle entfernt werden. Außerdem werden periodisch andere Peers kontaktiert, um neue Kontakte zu lernen (Self- und Random-Lookup).

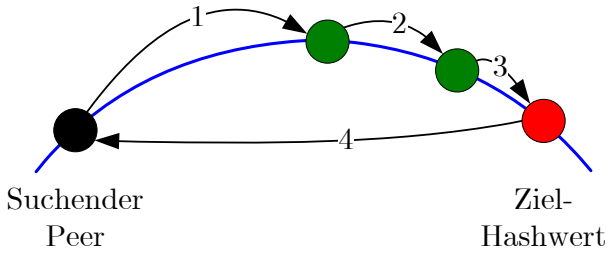


Abbildung 2.19.: Rekursiver Lookup-Vorgang, bei dem der suchende Peer (schwarz) für einen Ziel-Hashwert einen Ziel-Peer (rot) sucht, der einen dem Ziel-Hashwert ähnlichen oder gleichen Hashwert aufweist. Der Ziel-Peer ist dem suchenden Peer nicht bekannt, so dass die Suchanfrage durch andere Peers direkt an den Ziel-Peer weitergegeben wird. Nur der Ziel-Peer antwortet.

Für den Suchvorgang nach einem Datum (Ziel-Hashwert) gibt es die rekursive und die

iterative Strategie (siehe Abbildung 2.19 und 2.20).

Während eines rekursiven Lookup-Vorgangs sucht der suchende Peer (schwarz in Abbildung 2.19) für einen Ziel-Hashwert einen Ziel-Peer (rot in Abbildung 2.19) mit einem dem Ziel-Hashwert ähnlichen oder gleichen Hashwert. Der Ziel-Peer ist dem suchenden Peer ggf. nicht bekannt, so dass die Suchanfrage durch andere Peers direkt an den Ziel-Peer weitergegeben werden muss. Die Peers auf dem Suchpfad antworten dabei dem suchenden Peer nicht, sondern nur der Ziel-Peer antwortet. Eine rekursive Suche kann daher eine niedrigere Lookup-Latenz aufweisen, da nicht auf Antworten entlang des Suchpfads gewartet werden muss. Allerdings besteht das Risiko, dass Peers entlang des Suchpfads das Netzwerk verlassen oder ausfallen und die Suche somit nie zum Ziel führt.

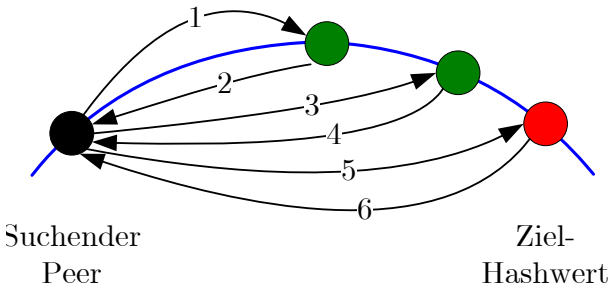


Abbildung 2.20.: Iterativer Lookup-Vorgang, bei dem der suchende Peer (schwarz) für einen Ziel-Hashwert einen Ziel-Peer (rot) sucht, der einen dem Ziel-Hashwert ähnlichen oder gleichen Hashwert aufweist. Der Ziel-Peer ist dem suchenden Peer nicht bekannt, so er erst andere Peers befragen muss, um den Ziel-Peer kennenzulernen. Alle Peers auf dem Suchpfad antworten, so dass der suchende Peer sich mit den neu erlernten Kontakten selbst in Verbindung setzen kann.

Während eines iterativen Lookup-Vorgangs sucht der suchende Peer (schwarz in Abbildung 2.20) für einen Ziel-Hashwert einen Ziel-Peer (rot in Abbildung 2.20) mit einem dem Ziel-Hashwert ähnlichen oder gleichen Hashwert. Der Ziel-Peer ist dem suchenden Peer ggf. nicht bekannt, so dass er erst andere Peers befragen muss, um den Ziel-Peer kennenzulernen. Alle Peers entlang des Suchpfads antworten dabei dem suchenden Peer mit neuen Kontakten, so dass der suchende Peer sich mit den neu erlernten Kontakten selbst in Verbindung setzen kann. Die iterative Suche ist einfacher zu implementieren,

besser zu debuggen und zu warten und kompensiert Peer-Ausfälle besser, indem jeder Peer entlang des Suchpfads die Kontaktaufnahme bestätigen muss.

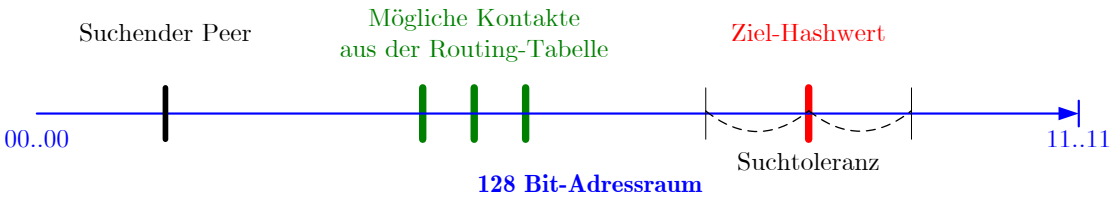


Abbildung 2.21.: Erster Schritt eines iterativen Kad-Lookup-Vorgangs.

Anhand der iterativen Strategie, die auch im Kad-Netzwerk implementiert ist, wird nachfolgend ein Lookup-Vorgang erläutert, bei dem ein Peer nach Peers sucht, die dem Ziel-Hashwert eines Datums möglichst ähnlich sind. Dabei kann es sich durchaus um mehr als einen Ziel-Peer handeln, da mehrere Peers Hashwerte besitzen können, die dem Ziel-Hashwert ähnlich genug sind. Ähnlich genug bedeutet, dass sie innerhalb einer vorde-

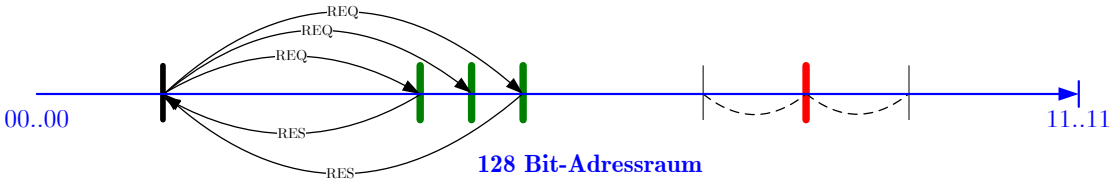


Abbildung 2.22.: Zweiter Schritt eines iterativen Kad-Lookup-Vorgangs.

finierten Suchtoleranz um den Ziel-Hashwert liegen. Zunächst wählt der suchende Peer dazu Peers aus seiner Routing-Tabelle aus, die die geringste XOR-Distanz zu dem Ziel-Hashwert aufweisen (siehe Abbildung 2.21). Diese Peers werden mit einem Request (REQ, siehe Abbildung 2.22) angesprochen und einige antworten (RES) mit neuen Kontakten (siehe Abbildung 2.23). Einige dieser neuen Peers werden kontaktiert und wiederum einige antworten (siehe Abbildung 2.24). Peers, die geantwortet haben und innerhalb der definierten Suchtoleranz liegen, werden angewiesen, eine Aktion auszuführen (ACTION REQ). Wenn sie diese Aktion ausgeführt haben und antworten (ACTION RES), wird ein Zähler (answers) erhöht (siehe Abbildung 2.25). Sobald dieser Zähler einen definierten Wert erreicht hat, wird der Lookup-Vorgang terminiert. Alternativ bricht der Lookup-

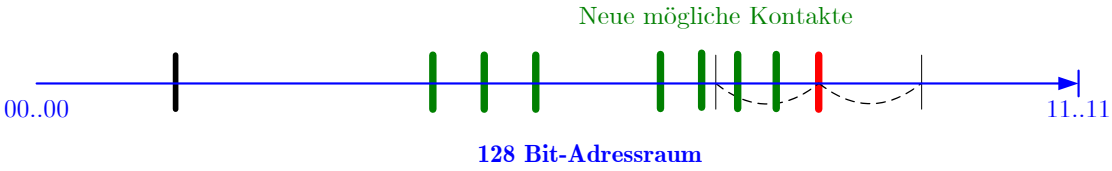


Abbildung 2.23.: Dritter Schritt eines iterativen Kad-Lookup-Vorgangs.

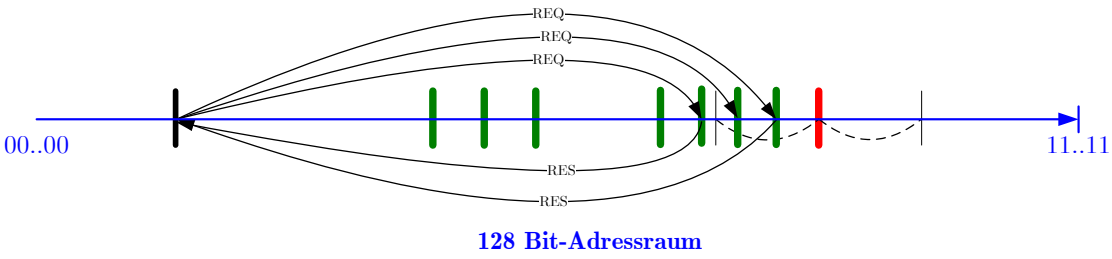


Abbildung 2.24.: Vierter Schritt eines iterativen Kad-Lookup-Vorgangs.

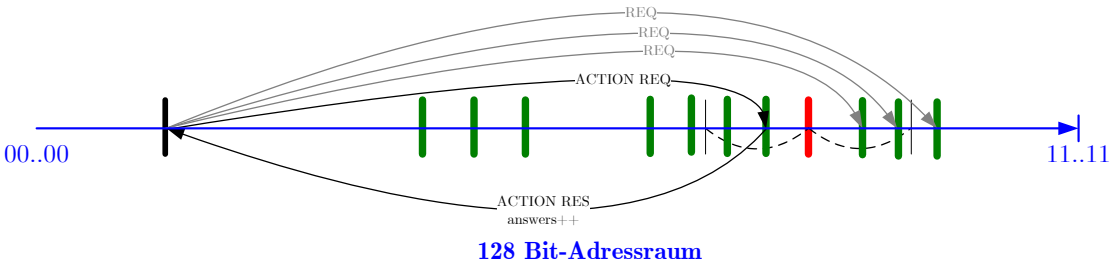


Abbildung 2.25.: Finaler Schritt eines iterativen Kad-Lookup-Vorgangs.

Vorgang auch nach einem Time-out ab.

2.4.7. Zusammenfassender Vergleich der P2P-Systeme

Stellt man die erläuterten P2P-Systeme vergleichend gegenüber, ist festzustellen, dass unstrukturierte, zentralisierte Systeme sich nur gut für kleine Netzwerke eignen, da ein zentraler Server als Flaschenhals bezüglich Central Processor Unit (CPU)- und Netzwerkbelastung existiert (niedrige Skalierbarkeit). Ohne zusätzliche Maßnahmen stellt der

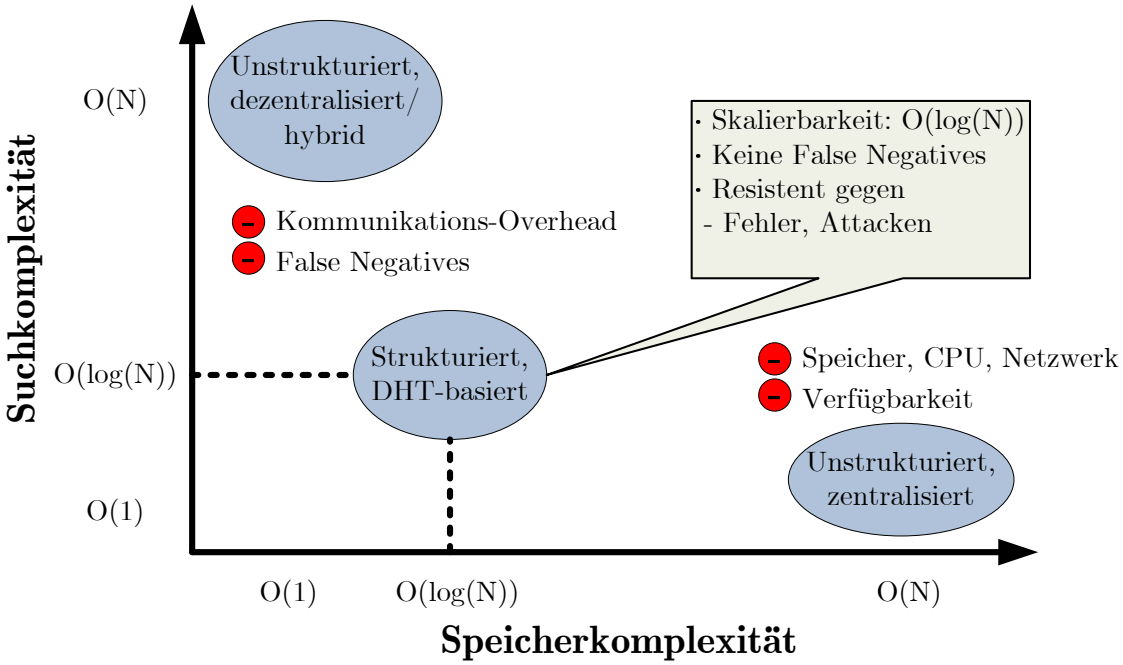


Abbildung 2.26.: Komplexitätsvergleich der P2P-Systeme in Abhängigkeit von der Knotenanzahl N [SW05]. Die Abbildung legt den Fokus auf die Darstellung strukturierter, DHT-basierter Systeme als bester Kompromiss aus den drei Systemen. Demgegenüber sind die gravierenden Schwachstellen von zentralisierten und dezentralisierten/hybriden Systemen dargestellt.

Server zudem einen SPoF dar, was das ganze Netz anfällig für DoS-Angriffe macht und bei einem Ausfall des Servers Daten nicht mehr verfügbar sind. Er muss die Kontaktdaten aller Peers speichern und weist somit eine sehr hohe Speicherkomplexität O(N) auf (N = Anzahl aller Peers), wobei Suchanfragen sofort beantwortet werden können (Suchkomplexität O(1)). Unstrukturierte, dezentralisierte Systeme weisen hingegen eine höhere Skalierbarkeit auf, da keine zentrale Instanz und somit auch kein SPoF existiert. Allerdings wird ein hoher Traffic-Overhead bei der Suche erzeugt (Suchkomplexität bis zu $O(N^2)$, da dazu das Netzwerk geflutet werden muss. Darüber hinaus wird eine Suche nach einer festgelegten Anzahl von passierten Peers abgebrochen, so dass einige Daten unter Umständen nicht gefunden werden (False Negatives). Die Speicherkomplexität ist mit O(1) gering, da nur Nachbarschaftslisten mit den Kontaktdaten einiger Peers und

keine Routing-Information gespeichert werden. Der Traffic-Overhead für die Suche wird bei hybriden Systemen etwas verringert; allerdings müssen dafür zusätzliche Super-Peers eingeführt werden. Strukturierte, DHT-basierte Systeme bieten den besten Kompromiss bezüglich Such- und Speicherkomplexität (O(log(N)), siehe Abbildung 2.26). Strukturierte, DHT-basierte Systeme sind auch robuster gegenüber Netzwerkfehlern und -attacken, da sie keinen SPoF aufweisen. Durch die strukturierte Datenhaltung kommt es zudem bei der Suche nach einem Datum zu keinen False Negatives, wie dies bei dezentralisierten und hybriden Systemen der Fall ist. Das heißt, die Suche ist deterministisch und jedes Datum wird wiedergefunden. Dies gilt unter der Voraussetzung, dass die Suchtoleranz so eingestellt ist, dass es für jeden Hashwert des Adressraums mindestens einen verantwortlichen Peer gibt.

Kapitel 3.

Berücksichtigung netzwerktechnischer Nähe in P2P-Netzen

P2P-Daten machen heutzutage einen großen Teil des Internet-Datenverkehrs aus. 2009 lag der Anteil in verschiedenen Regionen der Welt bei 43 % bis 70 % (siehe Abbildung 3.1).

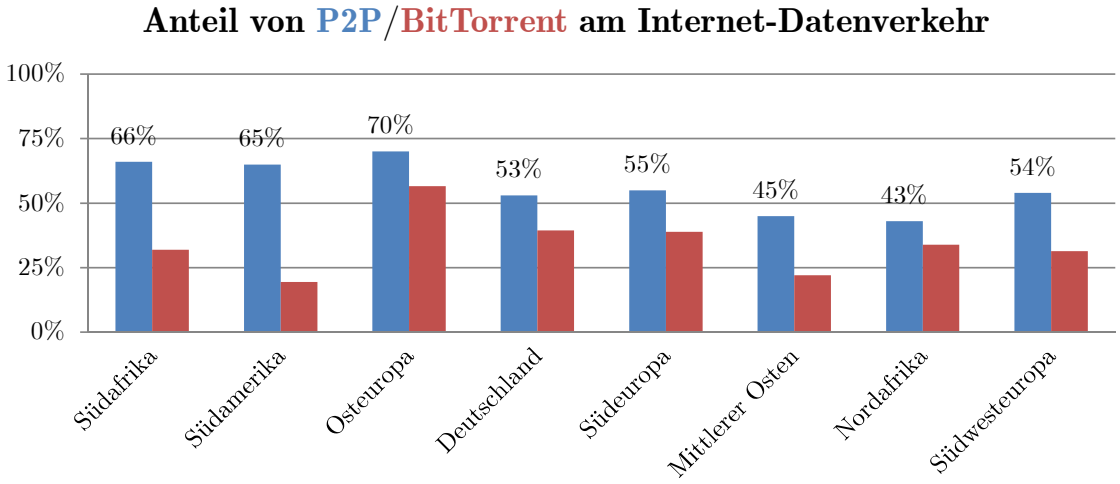


Abbildung 3.1.: Anteil des P2P-/BitTorrent-Datenvolumens am gesamten Datenverkehr im Internet in verschiedenen Regionen der Welt 2009 [ipo09].

Hauptsächlich wird dies durch File-Sharing-Applikationen wie eMule oder BitTorrent hervorgerufen. Insbesondere der BitTorrent-Datenverkehr betrug 2009 bis zu 80 % des P2P-Datenaufkommens, d.h., 56 % des gesamten Internet-Datenverkehrs [ipo09].

Neuere Quellen bescheinigen P2P-Daten nach wie vor einen großen Anteil am gesamten Datenverkehr von 10 bis 20 %, obwohl Echtzeitunterhaltungsanwendungen wie z. B. Video-Streaming stark zulegen und mittlerweile den Hauptanteil des Datenverkehrs ausmachen [san11b,san11a]. BitTorrent ist weiterhin das weltweit dominante P2P-Protokoll mit einem Anteil von ungefähr 90 % am P2P-Datenaufkommen.

BitTorrent-Technologie erhält derzeit zudem Einzug in neue Anwendungsgebiete wie z. B. Netzwerkdateisysteme für die Datensynchronisation [gol12]. Diese neuesten Entwicklungen lassen eine anhaltende hohe Relevanz von BitTorrent erwarten, da der zentralisierte BitTorrent-Ansatz u. a. Geschäftsmodellen entgegenkommt, bei dem der Anbieter eine hohe Priorität auf die Kontrolle der Kommunikation und des Netzes legt.

3.1. Motivation

Einerseits profitieren Internet Service Provider (ISP) von der großen Beliebtheit von P2P-Anwendungen durch eine Erhöhung ihres Betriebseinkommens. Das liegt daran, dass P2P-Anwendungen eine der Hauptgründe für Internet-Nutzer sind, sich einen Breitbandzugang zuzulegen [Men05]. Andererseits stellen die enormen P2P-Datenmassen eine signifikante Herausforderung bezüglich der Lenkung der Datenflüsse durch das Internet (Traffic Engineering) dar [XN99]. Dieser Zwiespalt zwischen Betriebseinkommen und Traffic Engineering bringt Netzbetreiber in eine schwierige Situation. Anderer Datenverkehr wie Hypertext Transfer Protocol (HTTP) wird unterdrückt, da die Netzwerkinfrastruktur mit P2P-Daten überflutet wird. Der Hauptgrund dafür ist, dass das Routing in P2P-Netzwerken die darunter liegende physikalische Internet-Topologie (Underlay) nicht berücksichtigt [ABFW04].

Üblicherweise wird ein unstrukturiertes P2P-Overlay-Netzwerk—auf dem der Fokus dieser Forschungsarbeit liegt—durch die Auswahl zufälliger Peers konstruiert [SW05]. Auf Grund dieses zufälligen Vorgehens impliziert eine Nachbarschaft auf dem P2P-Overlay in keinster Weise Nähe auf der darunter liegenden Internet-Topologie. Dieses Problem wird als Fehlanpassung zwischen logischem P2P-Overlay und physikalischer Netzwerkinfrastruktur bezeichnet [WIH05]. Es ist daher möglich, dass zwei miteinander kommunizierende Nachbarn (auf dem P2P-Overlay) physikalisch weit voneinander entfernt sind, obwohl der gewünschte Inhalt oft auf physikalisch näheren Peers zu finden wäre [RSR06].

Kommunikation zwischen physikalisch entfernten Peers erfolgt über lange Datenpfade; insbesondere bezüglich des Hop Counts. Es wird somit mehr Bandbreite beansprucht, was hochgradig ineffizient ist, wenn die Netzwerklast ohnehin schon hoch ist. Dadurch wird das Auftreten von Verkehrsstauungen begünstigt [XN99]. Im Gegensatz dazu erfordert die Kommunikation mit physikalisch nahen Peers weniger Bandbreite und der Datenverkehr in den Kernnetzen nimmt ab. ISP profitieren von einer Verkehrsverringerung in ihren Kernnetzen, da mehr Bandbreite für andere Applikationen verfügbar wird. Darüber hinaus können Verkehrsstauungen ebenso reduziert werden.

Daraus ergibt sich die Motivation für den in diesem Kapitel vorgestellten Ansatz. Sie besteht darin, die Kernnetze der ISP zu entlasten, indem die physikalische Pfadlänge, d.h. der Hop Count reduziert wird. Um dieses Ziel zu realisieren, wurde ein neuer Algorithmus für das BitTorrent-Netzwerk entwickelt, um den Hop Count als zusätzliches Auswahlkriterium für Peers zu nutzen. BitTorrent wurde gewählt, da es das weltweit vorherrschende P2P-Protokoll mit einem Anteil von ungefähr 90 % am P2P-Datenaufkommen ist. Zudem wird die Peer-Auswahl in diesem unstrukturierten P2P-Netzwerk über einen Belohnungsalgorithmus (Choking-Algorithmus, siehe Abschnitt 2.4.3) gesteuert, welchen die Neuentwicklungen betreffen.

Der Hop Count, der der Ermittlung der physikalischen Nähe von zwei Peers dient, ist allerdings nicht direkt verfügbar. Deshalb wird zunächst erläutert, wie P2P-Nutzer—im Speziellen BitTorrent-Nutzer—mit Informationen über den physikalischen Hop Count zu anderen BitTorrent-Nutzern ausgestattet werden.

Ein BitTorrent-Nutzer ist sich allerdings üblicherweise nicht bewusst bzw. nicht daran interessiert, wie der zu Grunde legende Transportmechanismus für seine Pakete funktioniert. Er möchte in erste Linie gewünschte Inhalte so schnell wie möglich herunterladen. Daher würde er ohne Weiteres nicht den physikalisch nächsten BitTorrent-Nutzer unter allen Nutzern auswählen; vorausgesetzt, dass diese ihm gewünschte Inhalte alle nahezu gleich schnell zur Verfügung stellen könnten. Es wird allerdings angenommen, dass sich BitTorrent-Nutzer kooperativ verhalten und nahe Nutzer als Kommunikationspartner auswählen, wenn dadurch ihre Download-Performance nicht leidet. Download-Performance sei hierbei als diejenige Zeit definiert, die für das Herunterladen gewünschter Inhalte anfällt und wird nachfolgend auch als Quality of Experience (QoE) bezeichnet. Im besten Fall ist der vorgestellte Ansatz also sowohl für Nutzer als auch ISP vorteilhaft. Im schlechtesten Fall soll die Performance des Nutzers durch Berücksichtigung des Hop

Counts jedoch zumindest nicht verschlechtert werden, wobei ISP nach wie vor profitieren.

3.2. Stand der Technik

Die Fehlanpassung zwischen logischem P2P-Overlay und der sich darunter befindlichen physikalischen Topologie entwickelt sich immer stärker zu einem ernsthaften Hindernis für die Entwicklung von P2P-Systemen. Im Bewusstsein dieses Problems widmen sich viele wissenschaftliche Arbeiten dem Lokalitätsproblem in DHT-basierten P2P-Netzwerken, d.h. strukturierten P2P-Netzwerken [ZDH+02, HJS+03]. Grundsätzlich gibt es drei Ansätze, die zur Ausbeutung netzwerktechnischer Nähe in DHT-basierten P2P-Netzwerken vorgeschlagen werden. Für eine detaillierte Betrachtung dieser Ideen sei an dieser Stelle auf [CDHR02] verwiesen.

Allerdings leiden nicht nur strukturierte, sondern auch unstrukturierte P2P-Systeme unter der Fehlanpassung. Im Gegensatz zu strukturierten P2P-Netzwerken wie z. B. BitTorrent sind Peers zufällig im Netzwerk organisiert. Es gibt daher diverse Bestrebungen wie z. B. [MZ05] und [LLX+04], um Overlay-Netzwerke von vornherein so zu *konstruieren*, dass die zu Grunde liegenden Topologie Berücksichtigung findet. Diese Verfahren verbessern die Performance der Netzwerke signifikant und vermeiden unnötigen Datenverkehr, indem netzwerktechnische Nähe ausgenutzt wird. Allerdings wird dies dadurch erreicht, dass Struktur in unstrukturierte Netzwerke eingebracht wird, die den physikalischen Netzwerkeigenschaften entspricht. Darüber hinaus entsteht Traffic-Overhead, um die eingebrachte Struktur in Stand zu halten. Im Gegensatz zu diesen Ansätzen greift die in dieser Forschungsarbeit vorgeschlagene Herangehensweise nicht in die *Konstruktion* von unstrukturierten P2P-Netzwerken ein. Vielmehr wird im BitTorrent-Netzwerk der Hop Count bereitgestellt, um mit Hilfe eines veränderten BitTorrent-Choking-Algorithmus (Choking-Algorithmus siehe Abschnitt 2.4.3) diese Zusatzinformation für die Auswahl naher Peers zu nutzen. Hierbei ist keine Modifikation des Konstruktionsalgorithmus notwendig und es müssen keine zusätzlichen Pakete gesendet werden, um die physikalische Distanz, d.h. den Hop Count zu bestimmen.

Weiterhin existieren Techniken, um P2P-Datenverkehr mit Hilfe von Netzbetreiber-seitigen Maßnahmen besser zu *steuern*. Die Internet Engineering Task Force (IETF) entwickelt dafür ein Protokoll für die Optimierung von Datenverkehr auf der Anwendungs-

schicht (Application-Layer Traffic Optimization (ALTO)). Unter Einsatz zusätzlicher ALTO-Server können Peers Informationen beziehen, um eine bessere als zufällige initiale Peer-Auswahl vorzunehmen [IET09]. Das Portal for P2P Applications (P4P)-Projekt strebt die Ermöglichung eines effektiveren Zusammenspiels zwischen P2P-Applikationen und Netzbetreibern an, indem dedizierte Tracker zur örtlichen Eingrenzung von P2P-Datenverkehr eingesetzt werden [XYKS08]. Das Network-Aware P2P-TV Application over Wise Networks (NAPA-WINE)-Projekt widmet sich der Aufgabe, P2P-IPTV-Datenströme so effizient zu verteilen, dass Auswirkungen auf die darunter liegenden Transportnetzwerke minimiert werden [LMH+08]. Dabei ist ein sogenannter „Network-Peer“ in der Lage, Maßnahmen zur Optimierung des P2P-Overlays zu ergreifen, so dass Kapazität und Auslastung der Transportnetzwerke Berücksichtigung finden. In der Arbeit von [LCC+10] wird die Nutzung des Autonomous System-Hop Count vorgeschlagen, um BitTorrent-ähnlichen P2P-Applikationen die Berücksichtigung netzwerktechnischer Nähe zu ermöglichen. Allerdings wird dafür ein BitTorrent-Tracker mit Kenntnis über die Internet-Topologie vorausgesetzt und Peers müssen dynamische Distanzinformationen über P4P-Tracker oder Content Distribution Networks (CDNs) beziehen. Im Unterschied zu diesen Konzepten arbeitet der Ansatz dieser Forschungsarbeit völlig eigenständig und unabhängig von Netzbetreiber-seitigen Maßnahmen und die notwendigen Modifikationen betreffen ausschließlich die P2P-Applikation.

3.3. Berechnung des Hop Counts

Da der Hop Count weder im IP-Header noch in einem anderen Protokoll direkt verfügbar ist, ist es notwendig, ihn zu ermitteln. Dafür gibt es zwei grundsätzliche Methoden für die Hop Count-Bestimmung [FG00]: Entweder kann er aktiv *gemessen* oder mit Hilfe des Time-to-Live (TTL)-Wertes *errechnet* werden. Für die aktive Messung werden Internet Control Message Protocol (ICMP) ECHO-Pakete gesendet. Obwohl bei dieser Methode der Hop Count meist exakt ermittelt wird, ist die Vorgehensweise in einem P2P-Netzwerk mit vielen Peers unpraktisch, da enormer Traffic-Overhead durch das Senden der zusätzlichen Pakete entsteht. Im Gegensatz dazu bedeutet die Berechnung des Hop Counts lediglich die Subtraktion des finalen TTL-Wertes eines empfangenen IP-Paketes von seinem initialen TTL-Wert. Dazu werden IP-Pakete genutzt, die ohnehin bei der Kommunikation zwischen BitTorrent-Nutzern ausgetauscht werden. Dieses Verfahren ist

geeignet, um den Hop Count zwischen vielen Peers zu berechnen, da keine zusätzlichen Pakete gesendet werden müssen. Folglich wird diese Vorgehensweise für die Ermittlung des Hop Counts gewählt. Gleichwohl ist es so, dass der initiale TTL-Wert in IP-Paketen nicht verfügbar ist und daher zur Verfügung gestellt werden muss.

Der TTL-Wert ist Bestandteil des IPv4-Headers (im Folgenden nunmehr als IP-Header bezeichnet) [ISI81]. Dieser Wert wird faktisch als Hop Counter genutzt. Jeder Router, der ein IP-Paket verarbeitet, dekrementiert deshalb den TTL-Wert um Eins. Um den Hop Count auf der Grundlage des TTL-Wertes zu berechnen, wird also der initiale TTL-Wert eines ausgehenden IP-Paketes benötigt. Wie in [Swi02] dargestellt ist, gibt es auf Grund der Heterogenität der eingesetzten Betriebssysteme im Internet *keinen* einheitlichen initialen TTL-Wert.

Daher stellt sich die Frage, wie der initiale TTL-Wert zur Verfügung gestellt werden kann. Die Antwort hierauf bietet das direkte Einfügen des initialen TTL-Wertes in BitTorrent-Nachrichten durch einen modifizierten BitTorrent-Algorithmus.

3.3.1. Einfügen des initialen TTL-Wertes in BitTorrent-Nachrichten

Um so wenig wie möglich Overhead für den BitTorrent-Algorithmus und -Datenverkehr zu verursachen, wird der initiale TTL-Wert nur in BitTorrent-Nachrichten eingefügt, wenn dies notwendig ist. Es gibt zwei Arten von BitTorrent-Nachrichten: Tracker-Requests und -Responses sowie die Nachrichten, die zwischen BitTorrent-Nutzern ausgetauscht werden. Zwischen BitTorrent-Nutzern erfolgt der Nachrichtenaustausch ausschließlich über TCP-Sockets. Eine Nachricht, die für die Interaktion zwischen Nutzern notwendig ist, ist die *Handshake*-Nachricht (siehe Abbildung 3.2) [The09]. Per BitTorrent-Spezifikation 1.0, die gängigen BitTorrent-Protokoll-Implementierungen zu Grunde liegt, werden die Felder *Pstrlen* auf den Wert 19 und *Protocol String* auf "BitTorrent protocol" gesetzt.

Ein BitTorrent-Handshake wird durch den Initiator einer Verbindung zwischen zwei Nutzern eines Schwarms gesendet. Zur Erwiderung muss der Empfänger dieser Handshake-Nachricht selbst mit einer Handshake-Nachricht antworten. Sowohl in der Handshake-Nachricht als auch in der Antwort auf diese fügt der modifizierte BitTorrent-Algorithmus den initialen TTL-Wert ein. In dieser Forschungsarbeit wird vorgeschlagen, das erste Byte der acht *Reserved*-Bytes dafür zu nutzen. Diese Bytes sind speziell dafür vorgesehen, das Verhalten des BitTorrent-Protokolls zu verändern.

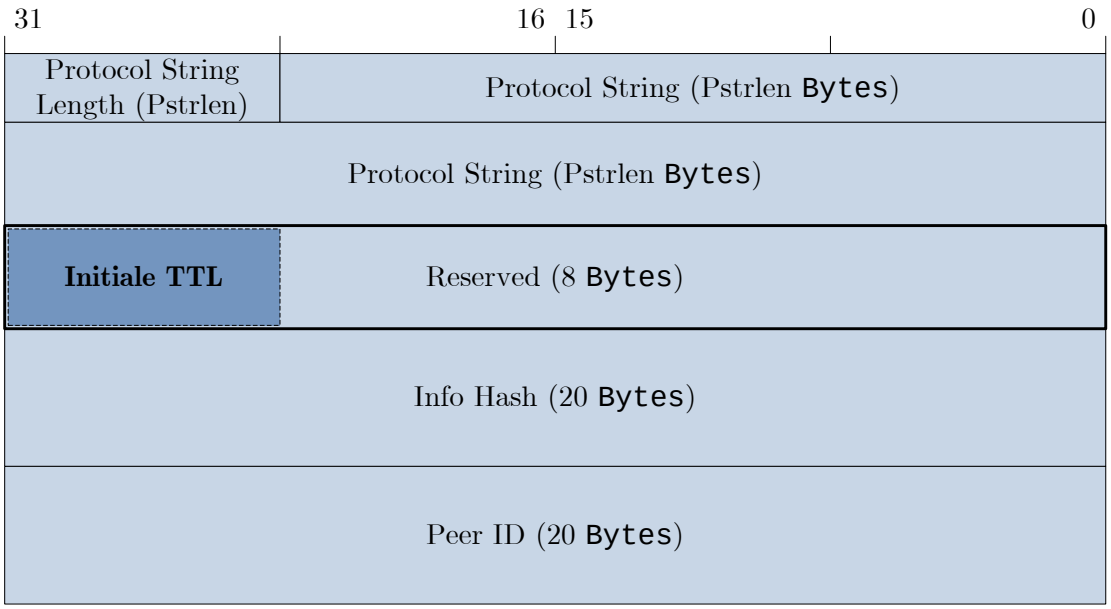


Abbildung 3.2.: Aufbau einer BitTorrent-Handshake-Nachricht.

3.3.2. Bereitstellung des TTL-Wertes in der BitTorrent-Applikation

Da TCP-Sockets für die Nutzerkommunikation verwendet werden, sind Felder des IP-Headers wie der TTL-Wert per se nicht in Applikationen verfügbar. Deshalb wird nachfolgend erläutert, wie dieser Wert verfügbar gemacht werden kann:

1) **Wie erfolgt der Zugriff auf den initialen TTL-Wert von ausgehenden BitTorrent-Handshake-Nachrichten?** Der initiale TTL-Wert kann über die Berkeley Software Distribution (BSD)-kompatible Funktion *getsockopt* bezogen werden, die sowohl Windows- als auch Unix-kompatibel ist. Dabei wird die spezifische Socket-Option *IP_TTL* genutzt, die von TCP-Sockets für ausgehende Pakete unterstützt wird.

2) **Wie kann der finale TTL-Wert von eingehenden BitTorrent-Handshakes bezogen werden?** Da TCP-Sockets das Auslesen des TTL-Wertes von eingehenden Paketen nicht unterstützen, wird zusätzlich die Packet Capture (PCAP)-Schnittstelle genutzt. Unix-basierte Betriebssysteme machen Gebrauch von der PCAP-Implementierung namens *libpcap*, wohingegen Windows-Betriebssysteme eine libpcap-Portierung namens *WinPcap* nutzen. Indem außerdem Filter wie der Berkeley Packet Filter zur Anwen-

dung kommen, kann der Kernel instruiert werden, der BitTorrent-Anwendung nur Pakete zu übergeben, die dem Aufbau eines BitTorrent-Handshakes entsprechen. So wird der Kernel-Puffer nicht mit Paketen überfüllt, was zu hohen Paketverlusten führen könnte. Mögliche Alternativen sind *Raw Sockets*, die ebenfalls direkten Zugriff auf die Vermittlungsschicht zulassen. Allerdings haben sich Raw Sockets als fehlerbehaftet und unportierbar herausgestellt oder sie leiten TCP-Pakete überhaupt nicht weiter (abhängig von Implementierung des Betriebssystems) [AH99]. Sie kommen damit als Alternative nicht in Frage.

3.4. Verbesserung der Peer-Auswahl

Die Modifikation zur Ermöglichung einer verbesserten Peer-Auswahl betrifft den BitTorrent-Choking-Algorithmus (siehe Abschnitt 2.4.3) [Sko08, DSTB11, DSR+11]. Der Standard-Choking-Algorithmus wählt BitTorrent-Nutzer, die Stücke herunterladen dürfen, ausschließlich anhand ihrer angebotenen Upload-Performance aus (mit Ausnahme von sogenannten *Optimistic Unchokes*). Nutzer werden so in eine Unchoke-Liste eingeordnet, dass der Nutzer mit der höchsten Upload-Performance (d.h. seiner *Service Rate*) an der Spitze steht. Üblicherweise darf eine feste Anzahl von Nutzern an der Spitze dieser Liste (z.B vier in der Standard-BitTorrent-Implementierung) gleichzeitig Stücke herunterladen. In der modifizierten Version hängt die Einordnung von Nutzern in die Liste nicht länger nur noch von ihrer Upload-Performance ab. Stattdessen werden für zwei Nutzer A und B *in der Unchoke-Liste eines anderen Nutzers* Quotienten wie folgt berechnet:

$$Quotient \; = \; Service \; Rate / Hop \; Count.$$

Der Quotient (in dem Algorithmus aus Abbildung 3.3 als *Quot* bezeichnet) bestimmt die Position eines Nutzers in der Unchoke-Liste. Wenn A's Quotient größer als der von B ist, d.h. Bedingung *Cond1 = True* und A's Hop Count kleiner als oder gleich B's (Cond2 = True) ist, ordnet sich A vor B in die Unchoke-Liste ein.

Wenn jedoch A's Hop Count größer als B's (Cond2 = False) ist, wird A's Quotient mit einem variablen Faktor (*1 - Hop Count-Wichtungsfaktor (WF)*) multipliziert, also gewichtet. Die Werte des WF umfassen rationale Zahlen von Null bis Eins. Wenn A's gewichteter Quotient kleiner als oder gleich B'S Quotient (Cond3 = True) ist, ordnet sich

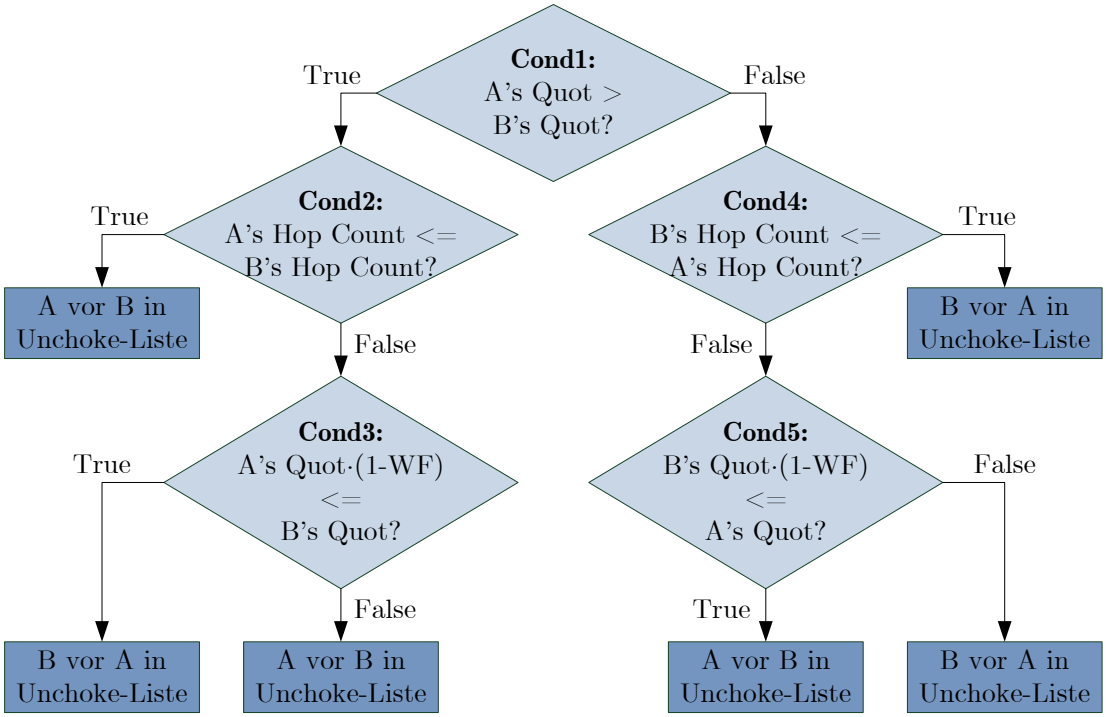


Abbildung 3.3.: Neuer BitTorrent-Choking-Algorithmus zur Berechnung der Position eines Nutzers in der Unchoke-Liste. Der Hop Count wird dabei als zusätzliches Auswahlkriterium genutzt.

B vor A in die Unchoke-Liste ein, da B's Hop Count stärker als A's Upload Performance gewichtet wird. Anderenfalls (Cond3 = False) ordnet sich A vor B ein, da A's Upload Performance höher als B's Hop Count gewichtet wird. In dem Else-Zweig des Algorithmus sind die entsprechenden Bedingungen für den Fall angegeben, dass B's Quotient größer als A's ist.

Als Veranschaulichung des Algorithmus sei ein Nutzer A mit einer hohen Service Rate und einem hohen Hop Count angekommen sowie ein Nutzer B mit einer moderaten Service Rate aber einem sehr niedrigen Hop Count. Gemäß der Rechenvorschrift für den Quotienten eines Nutzers ergibt sich für A ein im Vergleich mit B relativ niedriger Quotient (ungeachtet A's hoher Service Rate). Dennoch sei A's Quotient in diesem Beispiel größer als B's Quotient (Cond1 = True), obwohl B's Hop Count bedeutend kleiner als

A's Hop Count sei (Cond2 = False). Durch die Möglichkeit der Wichtung kann B's Hop Count allerdings ein höherer Stellenwert beigemessen werden, so dass sich durch Wahl eines hohen WF-Wertes B vor A in die Unchoke-List einordnet (Cond3 = True).

Zusammenfassend ist festzustellen, dass der WF genutzt wird, um einen Kompromiss zwischen der Gewichtung der Upload Performance und dem Hop Count eines Nutzers zu schließen. Ein hoher WF-Wert führt dazu, dass BitTorrent-Nutzer mit niedrigem Hop Count fast ungeachtet ihrer Upload Performance an die Spitze der Unchoke-Liste gelangen. Umgekehrt bedeutet ein niedriger WF-Wert eine höhere Wichtung der Upload Performance eines Nutzers. In diesem Fall erreichen Nutzer mit hoher Upload Performance weitestgehend ohne Beachtung ihres Hop Count wahrscheinlicher die Spitze der Unchoke-Liste.

Eine Vorgehensweise für die Auswahl netzwerktechnisch naher BitTorrent-Nutzer ist es, den Nutzer selbst entscheiden zu lassen. Der andere Ansatz, der in dieser Forschungsarbeit verfolgt wird, ist die Integration eines automatischen Selektionsmechanismus in den BitTorrent-Algorithmus.

3.5. Auswertung von Standard- und modifiziertem BitTorrent-Algorithmus

In diesem Abschnitt wird zunächst die Auswahl des Netzwerksimulators ns-2 für die Evaluierung des in dieser Forschungsarbeit vorgeschlagenen Ansatzes motiviert. Nachfolgend wird die Integration des BitTorrent-Protokolls in ns-2 beschrieben. Die Beschreibung schließt dabei eine ausführliche Erläuterung des dynamischen Nutzerverhaltens mit ein. Schlussendlich werden Simulationsergebnisse dargelegt, die durch Simulationen mit dem Netzwerksimulator ns-2 gewonnen wurden. Diese Ergebnisse umfassen einen Vergleich des Standard- mit dem modifizierten BitTorrent-Algorithmus hinsichtlich

- der Anzahl der Hops zwischen Nutzern,
- der Last des Kernnetzes,
- der Anzahl verworfener Pakete,
- dem maximalen Durchsatz im ganzen Netzwerk und
- der QoE der Nutzer.

3.5.1. Auswahl eines geeigneten Simulators

Um realistische und verlässliche Simulationsergebnisse zu erzielen, ist es von entscheidender Bedeutung, einen geeigneten Simulator auszuwählen. Es gibt eine große Anzahl von Netzwerksimulatoren, insbesondere im Bereich der P2P-Simulationen. In Tabelle 3.1 sind die wichtigsten Open-Source-Simulatoren aufgeführt, die für die Evaluierung von Standard- und modifiziertem BitTorrent-Algorithmus genutzt werden können und werden hinsichtlich Skalierbarkeit, Lizenzanforderungen und Betriebssystem verglichen. Die meisten Simulatoren laufen unter Linux/Unix oder sind mit Java implementiert worden und daher plattformunabhängig. Die Simulation von 10^4 bis zu 10^6 Knoten wird unterstützt und somit sind Simulationen von großen P2P-Netzwerken möglich.

Simulator	Skalierbarkeit	Lizenz	Betriebssystem
ns-2	$200 - 10^4$	GNU GPL	Linux/Unix
OverSim	10^5	GNU GPL	Linux, Mac, Windows
PlanetSim	10^5	GNU LGPL	Plattform-unabhängig
PeerfactSim.KOM	$10^5 - 10^6$	GNU GPL	Plattform-unabhängig
PeerSim	10^6	GNU LGPL	Plattform-unabhängig

Tabelle 3.1.: Simulatoren und ihre Eigenschaften [FV08, BHK07, GPM+04, SGR+11, MJ09].

Nachfolgend wird die Wahl von ns-2 motiviert, indem ein Überblick über seine Vorteile hinsichtlich der Berücksichtigung des physikalischen Underlays gegeben wird.

3.5.1.1. Vorteile von ns-2

Die meisten Simulatoren bilden das physikalische Underlay nicht präzise ab, um z. B. Simulationen zu beschleunigen [BL07].

Oversim unterstützt drei verschiedene Underlay-Typen: *Simple*, *Single Host* und *INET-Underlay* [BHK07]. Das INET-Framework bietet z. B. die Möglichkeit, das Zugangs- und Backbone-Netzwerk auf der Grundlage eines kompletten IP-Protokollstapels zu modellieren. Allerdings ist die Präzision dieses Simulators nicht bekannt und daher können realistische Bedingungen nicht ohne Weiteres vorausgesetzt werden. Planetsim ist ein Overlay-Netzwerksimulator, der nur einfache Netzwerkmodelle wie *RingNetwork* oder *CircularNetwork* unterstützt, die Latenzen nicht berücksichtigen [GPM+04]. Somit werden keine realistischen Netzwerkbedingungen gewährleistet. Gleiches gilt für den Simulator PeerfactSim.KOM, der mathematische und stochastische Modelle nutzt, um das Netzwerkverhalten zu emulieren [SGR+11]. Das physikalische Underlay wird hierbei ebenfalls nur ungenau abgebildet. PeerSim umfasst unter Anderem eine Event-basierte Engine, die Simulationen auf der Transportschicht unterstützt [MJ09]. Dabei wird die Transportschicht als ein spezielles Protokoll abgebildet, das einen Nachrichtendienst bereitstellt und deshalb können ebenfalls ohne Weiteres keine realistischen Bedingungen vorausgesetzt werden.

Demgegenüber stellt ns-2 einen komplexen Simulator dar, der den gesamten Netzwerkprotokollstapel abbildet und reale Netzwerkkomponenten wie Netzwerkknoten und Router für die Netzwerkkonstruktion verwendet. Daher ist eine explizite Verbindung zwischen logischem P2P-Overlay und physikalischem Underlay gegeben. Es ist demzufolge möglich, den Einfluss verbesserter Peer-Auswahl auf das Underlay zu untersuchen, was die Voraussetzung für die Evaluierung des vorgeschlagenen Ansatzes darstellt.

ns-2 enthält vollständige Implementierungen der Standardprotokolle TCP, UDP und File Transfer Protocol (FTP). Diese Protokolle wurden verifiziert und durchlaufen fortwährende Validierungen [ver08]. Somit sind diese Protokolle für die Implementierung des BitTorrent-Protokolls vollständig verfügbar. Dies stellt eine entscheidende Voraussetzung dar, da BitTorrent TCP für die Datenübertragung durch das Netzwerk nutzt.

Schließlich ist ns-2 sehr gut dokumentiert und erleichtert dadurch die Entwicklung und Evaluierung des modifizierten BitTorrent-Algorithmus [FV08].

Nach Meinung des Autors ist ns-2 auf Grund der dargelegten Gründe am besten für die beabsichtigten Simulationen geeignet und wurde somit dafür ausgewählt.

3.5.1.2. Arbeitsweise mit ns-2

Es gibt zwei Möglichkeiten, um Simulationsdurchläufe in ns-2 zu konfigurieren und zu beeinflussen.

Zunächst können komplexe Netzwerke mit Hunderten bis zu Tausenden Knoten und ihre Verbindungen in einem Tool Command Language (TCL)-Skript definiert werden. Dabei wird das physikalische Underlay, d.h. die Netzwerktopologie für das BitTorrent-Netzwerk erzeugt. Folgende grundlegende Parameter können für die Simulation eingestellt werden:

- Knotenanzahl
- Netzwerktopologie
- Bandbreite der Verbindungen zwischen Knoten
- Dateigröße

Des Weiteren können Trace Files deklariert werden, die den gesamten Datenverkehr während der Simulation aufzeichnen.

Event	Beschreibung
+	Paket wurde vorbereitet und in die Warteschlange zum Senden eingereiht
-	Paket wurde zum Ziel befördert
h	Paket ist einen Hop weiter gereist
d	Paket wurde verworfen

Tabelle 3.2.: Aufgezeichnete Daten in einem ns-2 Trace File [FV08]

Die zweite und mächtigste Möglichkeit ist die direkte Modifikation der zu simulierenden Protokolle in ihrer C++-Implementierung. Zusätzlich können neue Protokolle erzeugt werden und wesentliche Elemente wie Knoten, Warteschlangen oder Verbindungen können umdefiniert werden. Die Modifikationen des BitTorrent-Protokolls betreffen BitTorrents Choking-Algorithmus, der in Abschnitt 2.4.3 beschrieben wurde. Sobald das TCL-Skript ausgeführt wird, werden Trace Files generiert und die Netzwerkkommunikation zwischen Knoten wird aufgezeichnet. Die aufgezeichneten Daten bestehen aus Events (wichtige Events sind in Tabelle 3.2 aufgeführt) und dienen als Grundlage für die Statistikerstellung.

Ein Beispiel für Trace-Daten ist in Tabelle 3.3 ersichtlich. Als T wird hierbei die simulierte Zeit in Sekunden bezeichnet. SOURCE gibt den Quellknoten an, von dem das Datenpaket gesendet wurde und SINK bezeichnet den Zielknoten. TYPE spezifiziert den Pakettypen; SIZE enthält den Wert für die Paketgröße in Byte.

EVENT	T [s]	SOURCE	SINK	TYPE	SIZE [Byte]
+	0.000233	12	6	tcp	40
-	0.000234	12	6	tcp	40
h	0.000312	12	10	tcp	40
d	0.000333	3	7	tcp	40

Tabelle 3.3.: Beispieldaten in einem ns-2 Trace File [FV08]

Die aufgezeichneten Daten werden schließlich durch Perl-Skripte und Sortieralgorithmen verarbeitet, um gewünschte Statistiken zu erzeugen. Die Verarbeitung wurde automatisiert, um die großen Mengen an Simulationsdaten (einige 10 GByte pro Simulationsdurchlauf) effizient und in kurzer Zeit verarbeiten zu können.

3.5.2. BitTorrent in ns-2

Der BitTorrent-Algorithmus ist im ns-2-Paket nicht enthalten. Für die Implementierung des BitTorrent-Algorithmus in ns-2 wurde daher ein BitTorrent-Patch genutzt, der in der Arbeit von Eger et al. [EHBK07] entwickelt wurde. Diese Implementierung enthält nahezu alle gegebenen Teile des realen BitTorrent-Protokolls, die in der folgenden Liste aufgezählt werden [EHBK07]:

- Einteilung einer Datei in Stücke
- „Strict Priority“-Download-Strategie
- Random und „Rarest-First“-Download-Strategie
- Choking-Algorithmus
- Optimistic-Unchoking-Algorithmus
- Nur Upload: Leave-Optionen von Peers, die fertig geladen haben

Die „Endgame Mode“-Strategie und das „Anti-Snubbing“-Verfahren wurden nicht implementiert. Da das Hauptaugenmerk aber auf der Effizienz bei der Datenübertragung liegt, ist diese Vereinfachung zulässig.

Darüber hinaus wurde der BitTorrent-Algorithmus durch das dynamische Nutzerverhalten ergänzt, da BitTorrent-Nutzer fortwährend das BitTorrent-Netzwerk betreten und verlassen (Parameter der einzelnen Verteilungsfunktionen siehe Tabelle 3.4).

Gemäß den Ausführungen in [SR06b] folgt der Eintrittszeitpunkt von BitTorrent-Nutzern in das Netzwerk einer Weibull-Verteilung (Inter-Arrival Time) (siehe Abbildung 3.4). Abbildung 3.4 zeigt, dass ungefähr zum Zeitpunkt t = 4800 s = 80 min nach Simulationsstart die Mehrzahl der BitTorrent-Nutzer (ca. 90 %) das Netzwerk betreten haben. Eine Weibullfunktion besitzt die kumulative Verteilungsfunktion $Weibull(t, \lambda, k) = 1 - e^{-(t/\lambda)^k}$ mit dem Skalierungsparameter λ und dem Formparameter k. Für die bessere Vergleichbarkeit mit [SR06b] wurde als Darstellung die komplementäre kumulative Verteilungsfunktion $1 - Weibull(t, \lambda, k)$ der einzelnen Verteilungen gewählt.

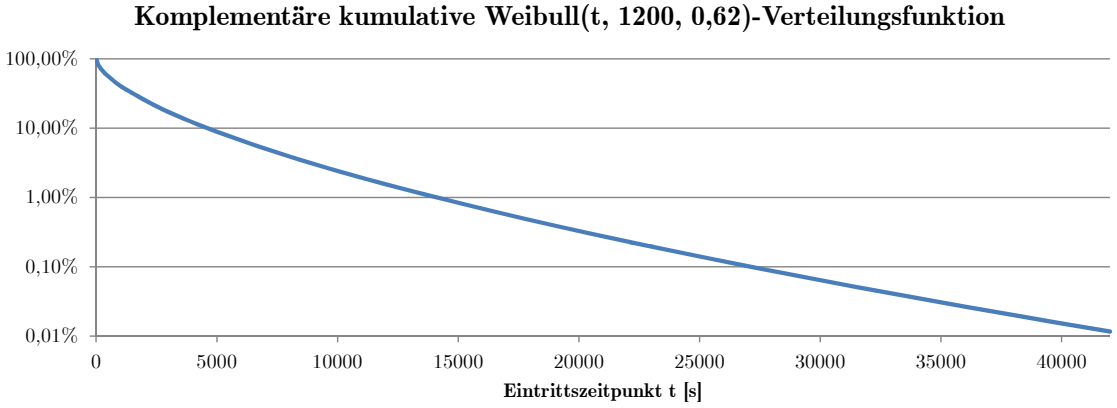


Abbildung 3.4.: Komplementäre kumulative Weibull-Verteilungsfunktion für den Eintrittszeitpunkt von BitTorrent-Nutzern in das Netzwerk. Die Y-Achse ist logarithmisch zur Basis 10 skaliert. Zum Zeitpunkt t = 4800 s = 80 min nach Simulationsstart hat die Mehrzahl der BitTorrent-Nutzer (ca. 90 %) das Netzwerk betreten.

Die Sitzungslängen (Session Lengths) der BitTorrent-Nutzer, d.h. die Zeit, die sie nach jedem Eintreten in das Netzwerk dort verbleiben, folgen ebenfalls einer Weibull-Verteilung

(siehe Abbildung 3.5) [SR06b]. Wenn Nutzer das Netzwerk nach einer Sitzung verlassen, kehren sie nach einer gleichverteilten Zeit (Stillstandzeit, Downtime) zurück [SR06b]. Schließlich bleiben Nutzer eine Zeitlang im Netzwerk, nachdem sie ihren Download vervollständigt haben (Verweildauer, Lingering) [SR06b]. Die Verweildauer wurde mit der gleichen Weibull-Verteilung wie die Sitzungslänge modelliert (siehe Abbildung 3.5).

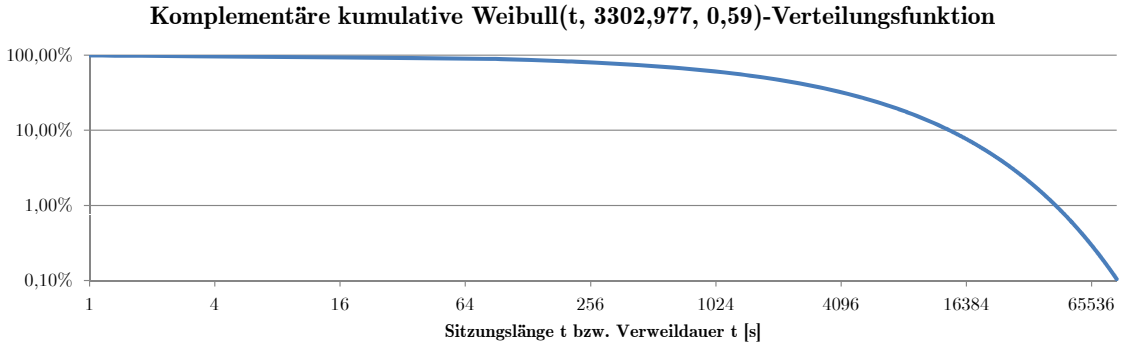


Abbildung 3.5.: Komplementäre kumulative Weibull-Verteilungsfunktion für die Sitzungslänge bzw. die Verweildauer von BitTorrent-Nutzern. Die X-Achse ist logarithmisch zur Basis 2 skaliert; für die Y-Achse wurde eine logarithmische Skalierung zur Basis 10 gewählt.

Nutzerverhalten	Verteilungsfunktion	λ	k	T_{max}
Eintrittszeitpunkt	Weibull	1200	0,62	700 min
Sitzungslänge	Weibull	3302,977	0,59	-
Stillstandzeit	Gleichverteilung	-	-	1/2 d
Verweildauer	Weibull	3302,977	0,59	-

Tabelle 3.4.: Aspekte des dynamischen Nutzerverhaltens im BitTorrent-Netzwerk [SR06b]

Skalierungsparameter λ und Formparameter k wurden jeweils so eingestellt, dass sich realistische Werte für die Simulation des Herunterladens einer 100 MByte großen Datei durch 200 Nutzer ergeben. Darüber hinaus wurde ein Wert T_{max} für die Verteilungsfunktionen von Eintrittszeitpunkt und Stillstandzeit eingeführt, um deren Wertebereiche für

die Simulation realistisch einzugrenzen.

3.5.3. Simulationsaufbau

Für die Simulation wurde eine Topologie entwickelt, die Telefonicas deutsches Backbone-Netzwerk abbildet [Sko08, Tel12]. Telefonica besitzt eine der umfassendsten Netzinfrastrukturen in Deutschland. Der schematische Aufbau der entwickelten Topologie ist in Abbildung 3.6 ersichtlich und besteht aus

- einem Backbone-Netzwerk aus Routern,
- Broadband Remote Access Server (BRAS), die an die Router angeschlossen sind und mit DSLAM verbunden sind,
- und BitTorrent-Nutzern (der Übersichtlichkeit halber ebenso wie DSLAM nicht in Abbildung 3.6 dargestellt), die wiederum an die DSLAM gekoppelt sind.

In Übereinstimmung mit Telefonicas deutscher Netzwerkinfrastruktur umfasst die entwickelte Topologie 16 Router. Die BRAS-Anzahl ist auf 4 pro Router eingestellt (woraus sich 64 BRAS ergeben) und es gibt 6 DSLAM pro BRAS (woraus sich 384 DSLAM ergeben). Die Anzahl der BitTorrent-Nutzer ist auf 200 eingestellt. Die Router bilden eine statische Struktur, wie sie auch in Abbildung 3.6 ersichtlich ist. Die BRAS sind gleichmäßig um die Router angeordnet und ebenso sind die DSLAM gleichverteilt um die BRAS angeordnet. Nutzer werden zufällig an die DSLAM angeschlossen. Die Anzahl der Router, BRAS, DSLAM und Nutzer ist während aller Simulationsdurchläufe gleich. Die Bandbreite zwischen den Routern im Backbone-Netzwerk und zwischen Routern und BRAS ist auf 20 GBit/s eingestellt. Zwischen DSLAM und BRAS wurde der Wert 1 GBit/s gewählt. Diese Bandbreitenwerte sind angemessene Werte in der Praxis [PKAC08]. Jedem BitTorrent-Nutzer wird eine Download-Kapazität von 6 MBit/s und eine Upload-Kapazität von 1,5 MBit/s zugewiesen, wobei diese Werte gängige Werte für einen asymmetrischen Internet-Zugang darstellen (Stand zu Beginn der Simulation für diese Forschungsarbeit 2010 [SBS10]).

Zu Beginn eines Simulationsdurchlaufs gibt es genau einen Seeder. Der Seeder bleibt im Netzwerk, bis jeder BitTorrent-Nutzer des Schwarms seinen Download einer 100 MByte großen Datei beendet hat. Die maximale Anzahl von Nutzern, die gleichzeitig von einem

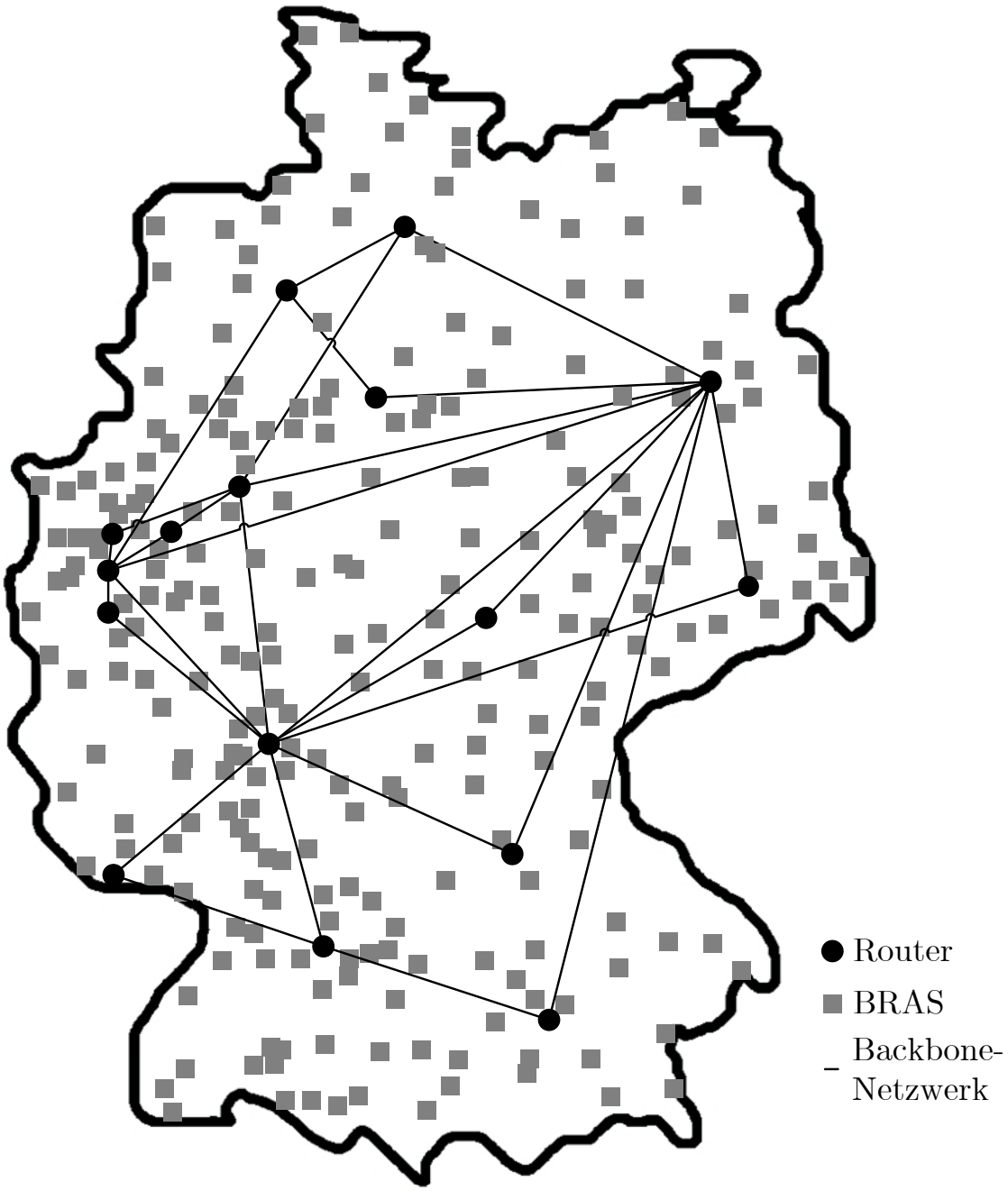


Abbildung 3.6.: Telefonicas Netzwerkinfrastruktur in Deutschland [Sko08, Tel12]

anderen Nutzer herunterladen dürfen, ist auf den Wert 4 eingestellt. Ein Nutzer verlässt während des Download-Vorgangs maximal vier Mal den Schwarm, um nach einer Stillstandzeit wieder zurückzukehren (siehe auch Abschnitt 3.5.2).

3.5.4. Simulationsergebnisse und Auswertung

Sowohl Standard- als auch modifizierter BitTorrent-Algorithmus (BTA) wurden auf der entwickelten Topologie simuliert. In den Simulationen wurden folgende Werte für unterschiedliche WF-Werte ermittelt, um beide Algorithmen zu vergleichen:

- Anzahl der physikalischen Hops: Aufsummierte Anzahl der physikalischen Hops, die die Datenpakete während der Simulation durch das Netzwerk zurückgelegt haben.
- Datenmenge im Kernnetzwerk: Aufsummierte Datenmenge, die während der Simulation die Router des Kernnetzwerks passiert hat.
- Anzahl der Paketverwürfe: Absolute Anzahl der Paketverwürfe durch Verkehrsstauungen.
- Maximaler Durchsatz im gesamten Netzwerk: Verkehrsspitze im gesamten Netzwerk und deren Zeitpunkt.
- Zeitpunkt der Fertigstellung des Datei-Downloads: Zeit, die der letzte BitTorrent-Nutzer braucht, um seinen Datei-Download zu vollenden.

Da Nutzer während eines Simulationsdurchlaufs zufällig mit den DSLAM verbunden werden, wurden für jeden WF-Wert auf der X-Achse 30 Messungen vorgenommen. In den Diagrammen ist für jeden WF-Wert der Mittelwert dieser Messungen auf der Y-Achse eingetragen. Zudem wird das 95 %-Vertrauensintervall (VI) angegeben, um zu verdeutlichen, dass die Genauigkeit und der Umfang der Messungen ausreichen, um Schlussfolgerungen abzuleiten.

Anzahl der physikalischen Hops: Wie in Abbildung 3.7 ersichtlich ist, sinkt die Anzahl der Hops, wenn der modifizierte BTA zur Anwendung kommt. Für WF = 0 zeigen die Simulationsergebnisse bereits eine geringfügige Reduktion der Hops um 0,4 % verglichen mit dem Standard-BTA. Dies ist dadurch bedingt, dass der Algorithmus in Abbildung 3.3 den Hop Count berücksichtigt, aber die Service Rate eines Nutzers als Peer-Auswahlkriterium dominiert. An dieser Stelle sei noch einmal angemerkt, dass der Hop Count *immer* durch den modifizierten BTA berücksichtigt wird und ein ansteigender

WF-Wert nur den Einfluss des Hop Counts *verstärkt*. Für alle anderen WF-Werte sinkt die Anzahl der Hops um 1,8 bis 2,3 %.

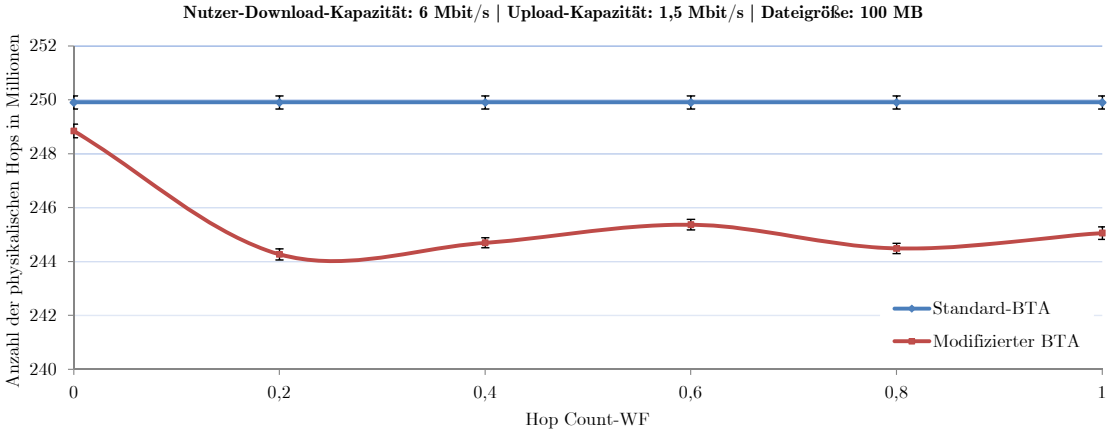


Abbildung 3.7.: Anzahl der Hops für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant.

WF	Standard-BTA		Modifizierter BTA		
	Datenmenge [GBit]	VI [MBit]	Datenmenge [GBit]	Datenmengen-reduzierung [%]	VI [MBit]
0	14,4	121	13,9	3,5	115
0,2			12,4	13,9	113
0,4			12,2	15,3	94
0,6			12,0	16,7	112
0,8			12,6	12,5	134
1,0			12,0	16,7	73

Tabelle 3.5.: Datenmenge im Kernnetz für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant.

Datenmenge im Kernnetzwerk: Diese relativ geringe Verminderung der Anzahl der Hops führt für die modifizierte BitTorrent-Variante zu einer erheblich niedrigeren Datenmenge im Kernnetzwerk. Tabelle 3.5 verdeutlicht diesen Umstand und zeigt eine Redu-

zierung der Last im Kernnetz um bis zu 17 % für WF = 0,6 und WF = 1.

Anzahl der Paketverwürfe: In Abbildung 3.8 ist die Anzahl der verworfenen Pakete ersichtlich. Paketwiederholungen sind dabei berücksichtigt, da TCP in ns-2 vollständig implementiert ist. Diese Anzahl erhöht sich im Fall des modifizierten BTA um 1 bis 20 %. Bei WF = 0,8 tritt dabei ein Topologie-abhängiges Minimum auf. Es kann also zu Paketverwürfen auf den BRAS kommen, da benachbarte Nutzer hohe Datenmengen austauschen. Allerdings ist die Last im Kernnetz niedriger, da sich der Datenverkehr aus dem Kernnetz an den Rand des Internets verlagert. Die erhöhte Anzahl der Paketverwürfe ist akzeptabel, da das Hauptaugenmerk auf einer verringerten Last im Kernnetz liegt, um Bandbreite für Datenverkehr durch andere Applikationen freizumachen und zudem belegen die Simulation, dass die QoE der Nutzer nicht leidet (siehe Abbildung 3.11).

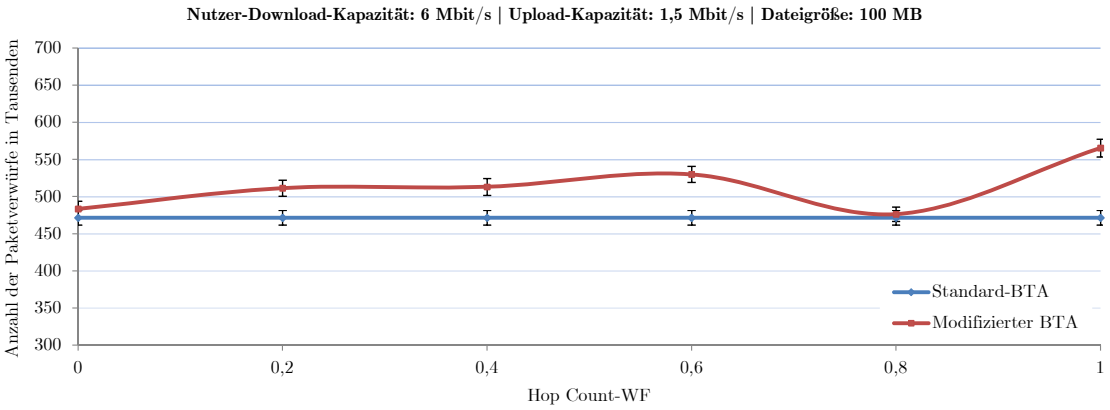


Abbildung 3.8.: Anzahl der Paketverwürfe für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant.

Maximaler Durchsatz im gesamten Netzwerk: Abbildung 3.9 zeigt den maximalen Durchsatz im gesamten Netzwerk. Er ist bis auf eine Ausnahme bei WF = 0 (Anstieg um 0,5 %) etwas niedriger, wenn der modifizierte BTA zur Anwendung kommt. Die Reduktion bewegt sich in einem Bereich von 2 % für WF = 1 bis zu 8 % für WF = 0,8.

Durch eine erhöhte Wichtung des Hop Counts als Peer-Auswahlkriterium sinkt also der maximale Durchsatz. Durch die damit einhergehende stärkere Verlagerung des Datenverkehrs in die Randbereiche kommt es zu einer geringeren Durchflutung des Kernnetzes. Somit treten keine so hohen Spitzenlasten mehr auf und Verkehrsstauungen im Kernnetz

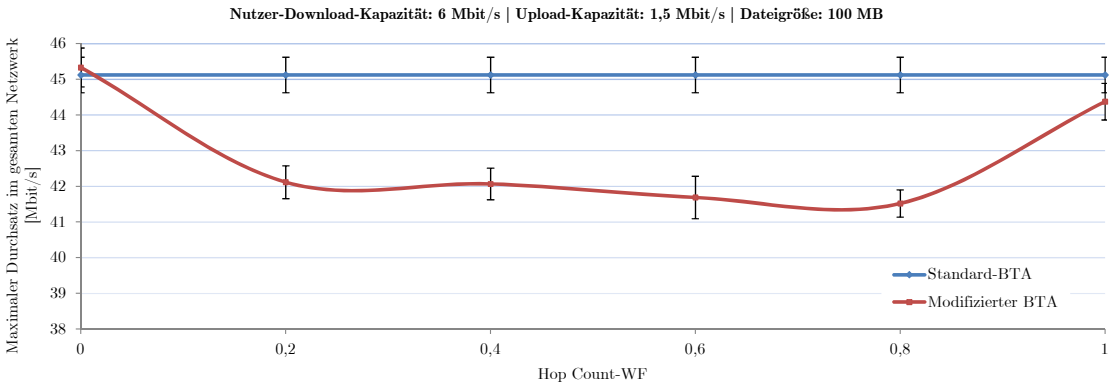


Abbildung 3.9.: Maximaler Durchsatz für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant.

werden reduziert. Bei WF = 1 tritt ein verhältnismäßig hoher maximaler Durchsatz für den modifizierten BTA auf, da dort Topologie-abhängig relativ viele Paketverwürfe auftreten (siehe Abbildung 3.8). Diese führen zu vielen neuen Paketen, die versandt werden müssen und somit zu einem höheren maximalen Durchsatz trotz der Verlagerung des Datenverkehrs in den Randbereich.

Der Zeitpunkt der Verkehrsspitze liegt zwischen Minute 15 und 17 nach Simulationsstart und weist nur minimale Unterschiede (0,5 bis 7 %) zwischen beiden Algorithmen auf (siehe Abbildung 3.10). Zu diesem Zeitpunkt hat die Mehrzahl der BitTorrent-Nutzer (ungefähr 60 %) das Netzwerk betreten, da der Eintrittszeitpunkt einer Weibull(t, 1200, 0.62)-Verteilungsfunktion folgt (siehe Abbildung 3.4 und Tabelle 3.4). Die Simulationsergebnisse in Abbildung 3.10 dienen der Veranschaulichung dieser Tatsache und bestätigen das erwartete Verhalten.

Zeitpunkt der Fertigstellung des Datei-Downloads: Darüber hinaus zeigen die Simulationsergebnisse, dass die Zeit, die notwendig ist, bis der letzte BitTorrent-Nutzer den Dateidownload vervollständigt hat, wesentlich niedriger ist, wenn der modifizierte BTA angewendet wird (siehe Abbildung 3.11). Tatsächlich wird die Zeit um 7 % (für WF = 0,6) bis zu 14 % (für WF = 0) reduziert. Wird WF auf 0,8 eingestellt, ist eine Zeitreduzierung um 12 % festzustellen; bei WF = 1 zeigt sich eine Zeitverringerung um 9 %. Generell steigt die Zeit bis zur Fertigstellung des Datei-Downloads mit steigendem

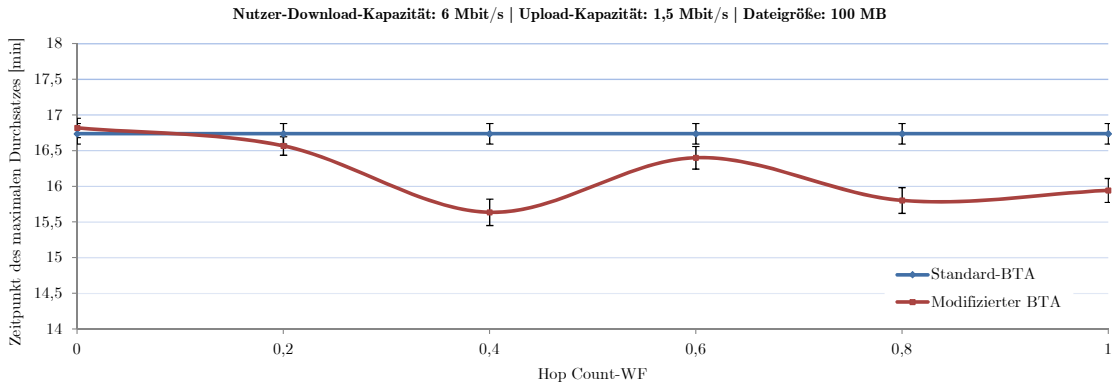


Abbildung 3.10.: Zeitpunkt des maximalen Durchsatzes für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant.

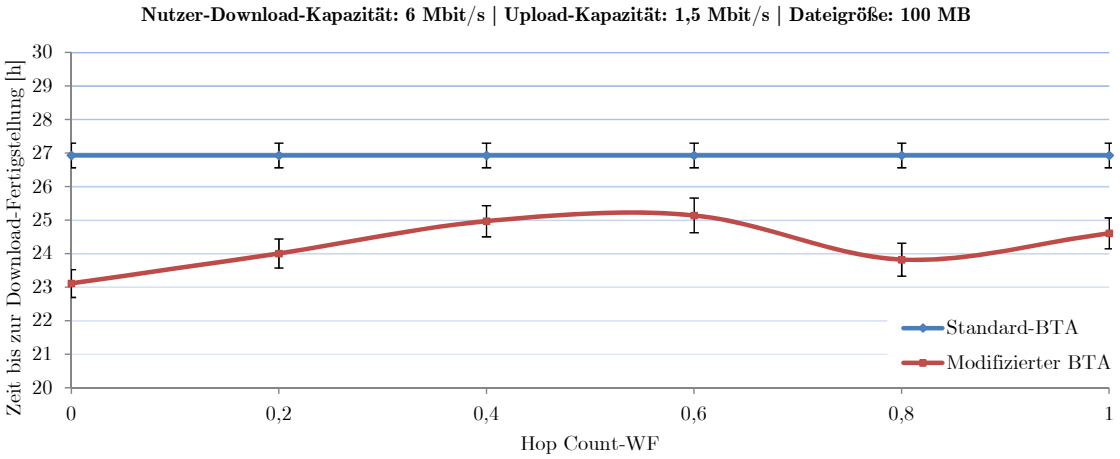


Abbildung 3.11.: Zeitpunkt der Fertigstellung des Datei-Downloads für unterschiedliche WF-Werte. Das Ergebnis für den Standard-BTA ist unabhängig von WF und daher konstant.

WF bis zu einem Wert von 0,6 an. Ein Kompromiss bezüglich der Wichtung von Hop Count und Service Rate reduziert somit zunächst nur die Last im Kernnetz. Erst bei WF = 1,0—also der maximalen Wichtung des Hop Counts—ist eine Zeitreduzierung

um 9 % bei gleichzeitiger Lastverringerung im Kernnetz um 17 % zu verzeichnen. Diese Verminderung umfasst eine Abnahme der Anzahl der Hops um 2 %. Bezüglich der Last im Kernnetz sowie der QoE der BitTorrent-Nutzer ist folglich offensichtlich WF = 1 eine günstige Wahl.

3.6. Zusammenfassung und Ausblick

In diesem Kapitel wird ein neuer Choking-Algorithmus für das BitTorrent-Protokoll vorgestellt, bei dem vorzugsweise physikalisch nahe BitTorrent-Nutzer für den Datenaustausch ausgewählt werden, um das Kernnetz von Netzbetreibern zu entlasten. Das Auswahlkriterium stellt hierbei der Hop Count dar. Er wird über die Differenz zwischen initialem TTL-Wert des Headers eines IP-Pakets und dessen TTL-Wert am Ziel des Pakets ermittelt. Da der initiale TTL-Wert nicht direkt verfügbar ist, wird er vom modifizierten BTA in BitTorrent-Handshake-Nachrichten eingefügt. Dabei erfordert der modifizierte BTA weder eine Modifikation des Konstruktionsalgorithmus von BitTorrent-Netzwerken noch ist das Senden zusätzlicher Pakete notwendig, um den Hop Count zu bestimmen.

Die durchgeführten Simulationen zeigen deutlich, dass Netzbetreiber von der Anwendung des modifizierten BTA profitieren, da die Last im Kernnetz um bis zu 17 % reduziert wird. Der Datenverkehr wird dabei örtlich eingegrenzt (die Anzahl der Hops reduziert sich um bis zu 2,3 %). Darüber hinaus belegen die Simulationen, dass sich die QoE von BitTorrent-Nutzern erhöht, da die Zeit, bis der letzte Nutzer seinen Download beendet hat, um bis zu 14 % abnimmt. Indem der Datenverkehr aus dem Kernnetz in den Randbereich des Internets verlagert wird, kann es hier zu Paketverwürfen kommen. Durch die hohen Datenmengen, die von nahen Nutzern ausgetauscht werden, führt dies zu einem Anstieg der Paketverwürfe um 1 bis 20 %. Allerdings ist dieser Effekt tolerierbar, denn zum einen leidet die QoE der Nutzer nicht, was die Simulationen belegen. Zum anderen liegt das Hauptaugenmerk auf einer verringerten Last im Kernnetz, um Bandbreite für Datenverkehr freizumachen, der durch andere Applikationen verursacht wird. Durch die stärkere Verlagerung des Datenverkehrs in die Randbereiche und die damit einhergehende geringere Durchflutung des Kernnetzes treten keine so hohen Spitzenlasten mehr auf und Verkehrsstauungen im Kernnetz werden reduziert.

Die Nutzung des Hop Counts zur Berücksichtigung netzwerktechnischer Nähe stellt einen

generischen Ansatz dar [DKW+08a]. Er ist somit auf andere P2P-Netzwerke wie z. B.das eDonkey2000-Netzwerk übertragbar, das Bestandteil von eMule ist.

Der derzeitige Mechanismus wurde für IPv4 implementiert, aber IPv6 wird im zukünftigen Internet das dominierende Protokoll sein. Deshalb werden sich weitere Arbeiten mit der Anpassung des neuen Mechanismus an IPv6-Umgebungen befassen.

Kapitel 4.

Auswahl und Erweiterung des P2P-Protokolls für dezentralisierte Kommunikationsinfrastruktur

Für die Realisierung einer P2P-basierten Netzwerkinfrastruktur, die einen Hauptpunkt dieser Forschungsarbeit darstellt, ist die Auswahl eines geeigneten P2P-Protokolls essentiell. In Abschnitt 2.4.7 wurden bereits die verschiedenen existenten P2P-Systeme vorgestellt und hinsichtlich Speicher- und Suchkomplexität verglichen. DHT-basierte P2P-Systeme bieten den besten Kompromiss bezüglich Such- und Speicherkomplexität. Sie sind zudem wenig fehleranfällig und weisen unter der Voraussetzung einer entsprechend eingestellten Suchtoleranz eine deterministische Suche auf, das heißt es kommt zu keinen False Negatives. Sie werden deshalb als P2P-Protokollform für den Entwurf P2P-basierter Netzwerkinfrastruktur ausgewählt.

4.1. Auswahl des DHT-basierten P2P-Protokolls

Es existieren diverse DHT-basierte P2P-Protokolle [SW05]. Es wurden die gängigen Protokolle Chord, Pastry, Kademlia, Symphony und Viceroy hinsichtlich verschiedener Kriterien auf ihre Eignung für den Einsatz für P2P-basierte Netzwerkinfrastruktur untersucht. Die Komplexität bezüglich der Größe der Routing-Tabelle für eine Knotenanzahl von N beträgt $O(logN)$ für Chord, Pastry und Kademlia. Für Symphony und Viceroy beträgt sie $O(1)$. Weiterhin wurde die Flexibilität der Routing-Tabelle untersucht. Flexibilität

bedeutet, dass für eine gegebene Anzahl von Knoten mehr als eine Routing-Tabelle existiert, das heißt, es gibt keine starre und eindeutige Routing-Tabelle. Dies bietet den Vorteil niedriger Beitritts- und Austrittskosten für neue Knoten und durch die Flexibilität der Routing-Tabelle ist der Wartungsaufwand minimal. Temporäre (also nicht unbedingt benötigte) Kontakte beschleunigen zudem den Lookup-Prozess. Chord und Viceroy besitzen eine starre Routing-Tabelle. Pastry, Kademlia und Symphony besitzen hingegen eine flexible Routing-Tabelle. Kademlia besitzt zudem gegenüber anderen DHT-basierten Lösungen vorteilhafte Analyseeigenschaften, die eine leichte formale Analyse seines Worst-Case-Verhaltens ermöglichen. Die Lookup-Performance beträgt für Chord, Pastry und Kademlia $O(logN)$; Symphony und Viceroy weisen hingegen eine quadratisch logarithmische Komplexität $O(logN^2)$ auf. Zudem ist die Suchkomplexität bei Chord, Pastry und Kademlia deterministisch bestimmbar; bei Symphony und Viceroy hingegen unterliegt sie der Wahrscheinlichkeitstheorie, da Zufallsprozesse eine Rolle spielen. Besonders die Flexibilität der Routing-Tabelle, hohe deterministische Lookup-Performance sowie gute Analyseeigenschaften sind wichtige Auswahlkriterien für ein DHT-basiertes P2P-Protokoll. Da der Lookup-Vorgang mit hoher Wahrscheinlichkeit die am meisten ausgeführte Operation und für alle DHT-basierten Netze grundlegend ist, liegt der Hauptaugenmerk auf der Lookup-Performance bzw. Suchkomplexität.

In der Tabelle 4.1 werden die Suchkomplexitäten der besprochenen DHT-basierten Netze Chord, Pastry und Kademlia mit deterministischer Lookup-Performance gegenübergestellt. Für Speicher- sowie Beitritts- und Austrittskomplexitäten gibt es in der Literatur unterschiedliche Angaben, so dass diesbezüglich keine konkreten Aussagen möglich sind [SW05, SR06a, MM02]. Die Suchkomplexität gibt die durchschnittliche Anzahl von Routing-Hops während eines Lookups an.Während bei Chord die Komplexität fest von der Anzahl der Knoten im Netz N abhängt, ist diese bei Pastry und Kademlia bzw. Kademlias Implementierungsvariante Kad von einem Parameter b abhängig. Dieser Parameter b gibt die Anzahl der Bits der Knoten-ID bzw. des -Hashwertes an, die pro Lookup-Schritt übersprungen werden können. D.h., je größer dieser Parameter ist, desto näher kommt man dem Lookup-Ziel pro Schritt [SW05, SR06a, MM02]. Werte der Originalimplementierung für b sind 4 für Pastry, 5 für Kademlia und laut den Ausführungen in [SR06a] im Mittel 6,98 für Kad.

Bei der Entwicklung von Kademlia wurde zudem eine Diskrepanz des Designs von Pastry aufgegriffen. Pastrys Routing-Metrik entspricht nicht unbedingt der tatsächlichen nume-

Netzwerk	Suche
Chord	$O(\frac{1}{2}log_2(N))$
Pastry	$O(log_{2^b}(N))$
Kademlia/Kad	$O(log_{2^b}(N))$

Tabelle 4.1.: Suchkomplexitäten der DHT-basierten P2P-Netze mit deterministischer Suchkomplexität in Abhängigkeit von der Anzahl der Knoten N und Parameter b [SW05, SR06a, MM02].

rischen Nähe der Knoten-IDs. Als Folge davon sind bei Pastry zwei Routing-Phasen erforderlich, die die Lookup-Performance beeinträchtigen und eine komplizierte formale Analyse des Worst-Case-Verhaltens bedingen. Kademlia nutzt daher die XOR-Routing-Metrik, die diese Probleme verringert und ermöglicht darüber hinaus parallele Lookup-Operationen [SW05]. Die Kademlia-Implementierungsvariante Kad nutzt außerdem mehr Buckets, als ursprünglich für Kademlia vorgeschlagen wurden und verringert zusätzlich noch einmal die durchschnittliche Anzahl von notwendigen Routing-Hops pro Lookup [Bru06, SR06a].

Da das Kademlia-Netzwerk eine geringe Suchkomplexität aufweist, sich als vorteilhaft gegenüber Pastry auszeichnet und durch den Parameter b hinsichtlich Such- und Speicherkomplexität sowie der Teilnahme am Netz optimierbar ist, bietet es die besten Voraussetzungen für die Realisierung P2P-basierter Infrastrukturen und wird deshalb dafür ausgewählt. Im Speziellen fällt die Wahl auf die Kademlia-Implementierungsvariante Kad, die die komplette Kademlia-Funktionalität umsetzt. Im Abschnitt 2.4.6 wurde das Kad-Protokoll bereits beschrieben, welches für den eMule-Client implementiert wurde und dort eingesetzt wird.

Trotz aller Vorteile des Kad-Protokolls bedarf dieses insbesondere hinsichtlich zweier Aspekte der Erweiterung, um die reibungslose Funktionalität des Netzwerks gewährleisten zu können:

1. Behandlung von Firewalls und NAT: Auch Peers im Kad-Netzwerk, die auf Grund von Firewalls und NAT nicht kontaktiert werden können, soll die Kommunikation ermöglicht werden.
2. Einführung einer dynamischen Suchtoleranz: Um in einem von vornherein unbe-

kannten Kad-Netzwerk eine deterministische Suche gewährleisten zu können, ist die Einführung einer dynamischen Suchtoleranz notwendig, die sich zur Laufzeit auf jede beliebige Netzwerkkonfiguration selbständig einstellt. Eine statische Suchtoleranz, die zur Kompilierzeit feststeht und zur Laufzeit unveränderlich bleibt, kann ohne vorherige Kenntnis des Kad-Netzwerks nicht sicherstellen, dass es für jeden auftretenden Hashwert einen Zielknoten gibt. Eine Einstellung der Suchtoleranz auf einen sehr hohen festen Wert, um dennoch eine Deterministik zu erreichen, verursacht eine hohe Redundanz in der Zuständigkeit, da praktisch jeder Knoten Ziel für jeden Hashwert ist.

Beide Erweiterungen wurden vollständig implementiert und kommen für das Kad-Netzwerk zum Einsatz.

4.2. Behandlung von Firewalls und NAT

4.2.1. Motivation

Da bei Provider-übergreifender Kommunikation von Zugangsknoten ggf. netzbetreiberseitige Firewalls und NAT Kommunikationshemmnisse darstellen, müssen geeignete Mechanismen zur Firewall/NAT-Erkennung und -Umgehung gewählt werden, um die Kommunikation dennoch zu ermöglichen [FSK05]. Das Problem bei NAT bzw. Firewalls ist, dass unbekannte eingehende Nachrichten den eigenen Ziel-Port nicht erreichen können [RWHM03]. Bei der üblichen Form von NAT, der Network Address Port Translation, erfolgt die Zuordnung mehrerer privater IP-Adressen zu einer öffentlichen IP-Adresse und zusätzlich ein Austausch der Ports. Um das Vorhandensein von Firewalls bzw. NAT festzustellen, wird in dieser Forschungsarbeit ein erweiterter Firewall/NAT-Überprüfungsmechanismus für das Kad-Netzwerk vorgeschlagen, der nach dem eigenen Bootstrapping zur Anwendung kommt und ggf. nach einer bestimmten Zeit wiederholt werden sollte (in der Kad-Originalimplementierung der Version 0.49a, die als Implementierungsgrundlage dieser Forschungsarbeit dient, jede Stunde).

Wird das Vorhandensein einer Firewall festgestellt bzw. befindet sich der Peer hinter einer NAT, wird ein Mechanismus für die Findung eines sogenannten Buddys erklärt, der die Regelung der Kommunikation übernimmt und somit die Umgehung der Firewall bzw. der

NAT ermöglicht. Der Buddy-Mechanismus wahrt den dezentralen Charakter des Kad-Netzwerks und hat sich somit als sinnvoll für den Einsatz in diesem Netz herausgestellt [Pas09]. Für weitere Mechanismen zur Umgehung von Firewalls bzw. NAT sei an dieser Stelle auf [SFK8q] verwiesen.

4.2.2. Stand der Technik

Die Implementierungsvariante Kad des Kademlia-Protokolls beinhaltet bereits einen einfachen Mechanismus zur Erkennung von NAT und Firewalls sowie zur Bestimmung eines Buddys zur Ermöglichung der Kommunikation trotz Firewall/NAT [Bru06].

NAT-Erkennung [Bru06]: Ein Peer A kontaktiert per UDP-Datagramm bei seinem Eintritt ins Kad-Netzwerk einen Peer B, der als nicht hinter einer Firewall/NAT angenommen wird und somit erreichbar ist. In dieses UDP-Datagramm wird der TCP-Port von Peer A zur späteren Kontaktaufnahme per TCP mit eingebettet. Das UDP-Datagramm kommt mit der IP-Adresse von Peer A bei Peer B an oder mit der IP-Adresse des ggf. vorhandenen NAT-Gerätes zwischen A und B. B schreibt nun die empfangene IP-Adresse in die Nutzlast des Antwortpakets und schickt es zurück an A. Peer A vergleicht die IP-Adresse in der Nutzlast der empfangenen Antwort mit seiner eigenen und stellt bei Nicht-Übereinstimmung fest, dass A und B über NAT verbunden sind [RWHM03]. Vorausgesetzt, das NAT-Gerät tauscht die in der Nutzlast enthaltene IP-Adresse nicht auch aus, stellt dies einen vergleichsweise einfachen Ausschluss von NAT zwischen zwei Netzwerkknoten dar.

Firewall-Erkennung für das TCP-Protokoll [Bru06]: Ein Peer kann bei Bedarf die Überprüfung der Erreichbarkeit seines im Kad-Netzwerk genutzten TCP-Ports vornehmen. Hierfür erfolgt die Bindung eines Stream-Sockets an seine IP-Adresse und seinen TCP-Port im Server-Modus und die Aufforderung an einen anderen Peer, sich als Client mit diesem Socket zu verbinden. Wenn die Verbindung gelingt, ist der TCP-Port erreichbar und die Verbindung wird abgebaut. Scheitert der Verbindungsaufbau nach einer bestimmten Zeit oder durch Signalisierung des anderen Peers per UDP-Datagramm, wird der TCP-Port nach Ausschluss einer NAT-Blockierung als durch die Firewall geschlossen angenommen.

Allerdings gibt es bei dieser Methode keine Sicherheit, dass die Firewall den für das TCP-Protokoll offenen Port nicht für die Kontaktaufnahme mittels UDP sperrt. Diesem

Aspekt widmet sich der folgende Abschnitt.

4.2.3. Mechanismen zur Erkennung von Firewalls und NAT für das UDP-Protokoll

Oftmals verfügt ein Router sowohl über eine Firewall-Funktionalität als auch über NAT. Darüber hinaus existieren auch transparente Firewalls, die keine eigene Netzwerk-Adresse beziehen, um sie schwieriger angreifbar zu machen. P2P-Knoten können dadurch in ihrer Funktionalität beeinträchtigt werden oder die Kommunikation wird sogar vollständig unterbunden. Deshalb sind geeignete Mechanismen notwendig, um vorhandene Firewalls bzw. NAT zunächst zu erkennen und dann die Kommunikationsfähigkeit betroffener Knoten vollständig wiederherzustellen.

Für die Gewährleistung der Funktionalität des Kad-Netzwerks ist die UDP-basierte Kommunikation der Knoten von entscheidender Bedeutung. UDP-Pakete an einen bestimmten Port werden von Circuit-Level- und Application-Layer-Firewalls sowie einfachen Paket-Filtern mit starren Regelsätzen behandelt, infolgedessen Pakete verworfen werden oder passieren dürfen. Sollte der UDP-Port, den Kad-Knoten zum Paketaustausch verwenden, gesperrt sein, wird diesen bereits der Beitritt in das Kad-Netzwerk verwehrt. Allerdings verhält sich die Mehrzahl der heutzutage eingesetzten Firewalls bezüglich UDP-Datagrammen wie ein dynamischer Paket-Filter. Ausgehende UDP-Pakete öffnen den Port für eine definierte Zeit, um die Antwort abzuwarten. Sollte also der Empfang eines UDP-Pakets auf eine Anfrage hin möglich sein, impliziert dies keineswegs, dass der Port grundsätzlich erreichbar ist. Diese Schlussfolgerung kann erst getroffen werden, wenn der Paketempfang nach dem Ablauf der definierten Zeit möglich ist.

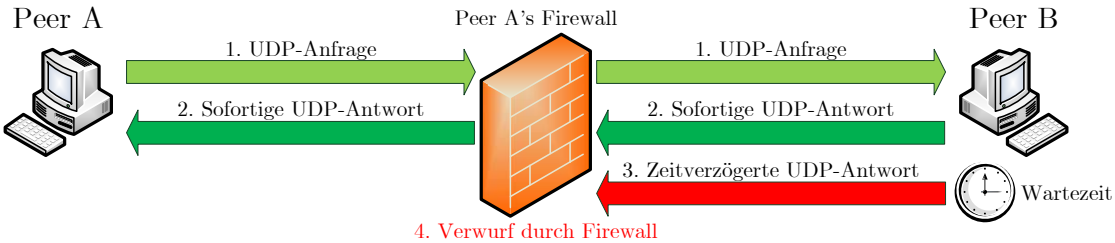


Abbildung 4.1.: Einfache Firewall-Erkennung mit Wartezeit [Bru06].

Einfache Firewall-Erkennung mit Wartezeit (UDP-Protokoll): Eine bereits in Brunner et al. [Bru06] beschriebene einfache Methode zur Erkennung einer Firewall ergibt sich darin, dass Peer A ein UDP-Paket an einen anderen Knoten B sendet (1.) und ihn zu einer sofortigen Bestätigung auffordert (2.) (siehe Abbildung 4.1). Nach dem Ablauf einer definierten Wartezeit soll B eine weitere Antwort (3.) an A senden, welche durch die Firewall verworfen wird (4.).

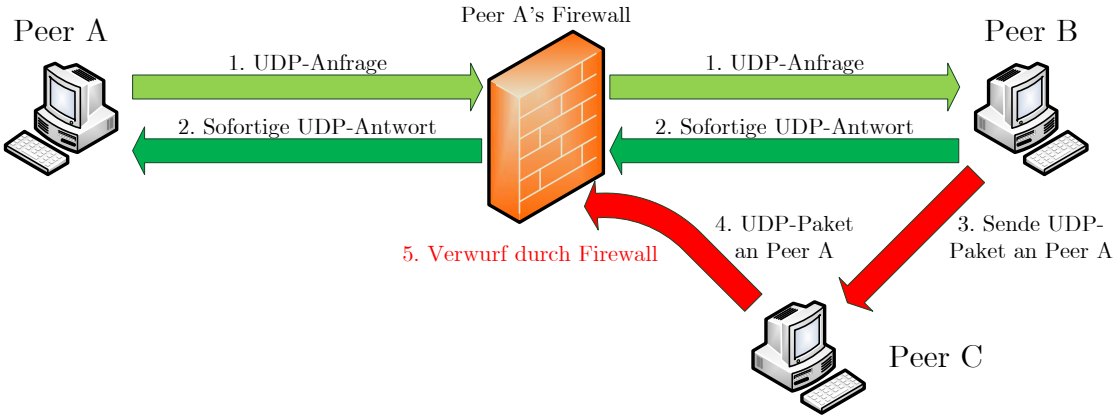


Abbildung 4.2.: Verteilte Firewall-Erkennung ohne Wartezeit.

Allerdings gibt es keinen standardisierten Wert für die Wartezeit, so dass keine Aussage getroffen werden kann, nach welcher Zeit der durch die Anfrage geöffnete Port wieder geschlossen wird und das weitere Antwortpaket gesendet werden muss. Daher kann es vorkommen, dass das Antwortpaket gesendet wird, während der Port noch offen ist. Zudem erhöht sich mit steigender Wartezeit auch die Wahrscheinlichkeit, dass Knoten B das Kad-Netzwerk bereits verlässt, bevor er das zweite Antwortpaket sendet. Somit weist dieser einfache Firewall-Erkennungsmechanismus niemals eine 100 %-ige Zuverlässigkeit auf und beinhaltet eine durch die Wartezeit bedingte Verzögerung. Daher wurde eine verteilte Firewall-Erkennung ohne Wartezeit entwickelt, die im Folgenden vorgestellt wird.

Verteilte Firewall-Erkennung ohne Wartezeit (UDP-Protokoll): Dieser Mechanismus nutzt den Umstand aus, dass dynamische Paket-Filter-Firewalls nur UDP-Pakete von Knoten passieren lassen, an die zuvor eine Anfrage gestellt wurde. Es kann für die angestrebten Anwendungsfälle davon ausgegangen werden, dass das Kad-Netzwerk in

seiner Größe mindestens einen dritten Peer C umfasst, den Knoten B instruiert (3.), ein UDP-Paket an A zu schicken (4.) (siehe Abbildung 4.2). Erreicht dieses Paket A, ist der überprüfte Port grundsätzlich erreichbar. Ist dies nicht der Fall, kann davon ausgegangen werden, dass der Port gesperrt ist (5.) und nur Antworten von vorher kontaktierten Knoten zugelassen werden (2.). Unter der Voraussetzung, dass Peer A zuvor C nicht kontaktiert hat, stellt diese Methode einen zuverlässigen Mechanismus zur Firewall-Erkennung dar, der durch den Wegfall der Wartezeit zudem schnell ist.

4.2.4. Der Buddy-Mechanismus zur Umgehung von Firewalls und NAT [Bru06]

Der Buddy-Mechanismus stellt eine für das Kad-Netzwerk geeignete Methode dar, um einem Peer A hinter einem NAT oder einer Firewall die Kommunikation zu ermöglichen und somit die UDP- und/oder TCP-Port-Sperre zu umgehen.

Der betroffene Peer A muss dafür eine Suche nach seinem eigenen Hashwert durchführen, um einen geeigneten Buddy (Peer C) zu lokalisieren (siehe Abbildung 4.3). Bei einer erfolgreichen Suche werden k Knoten gefunden, deren Hashwerte innerhalb der Suchtoleranz des suchenden Knotens A liegen. Gibt es solche Knoten nicht, kann kein Buddy ausgewählt werden und der betroffene Peer kann nicht an der Kommunikation teilnehmen.

Wenn mehrere Peers gefunden werden, kann als Buddy z. B. derjenige Peer C ausgewählt werden, der die geringste logische Distanz zu Peer A aufweist. Allerdings eignen sich alle gefundenen Knoten innerhalb der Suchtoleranz gleich als Buddy, da ihr Zuständigkeitsbereich deckungsgleich mit dem von Peer A ist (beispielsweise Peer C in Abbildung 4.3).

Dieser Peer C erhält die Anweisung, einen Stream-Socket im Server-Modus aufzubauen und auf einen Verbindungsaufbau zu warten. Ist Peer C dazu bereit, teilt er dies Peer A mit und sendet ihm seinen TCP-Port. Peer A baut daraufhin als Client einen Stream-Socket auf und verbindet sich mit dem Buddy-Peer C. Diese Verbindung ist erlaubt, da sie aus Sicht von Peer A eine ausgehende Verbindung darstellt. Die etablierte Verbindung bleibt bestehen, solange keiner der beiden Knoten das Netzwerk verlässt.

Der Buddy erhält nach dem Verbindungsaufbau alle Nachrichten (1.) aus dem Kad-Netzwerk (beispielsweise von Peer B in Abbildung 4.3), die eigentlich an Peer A gerichtet

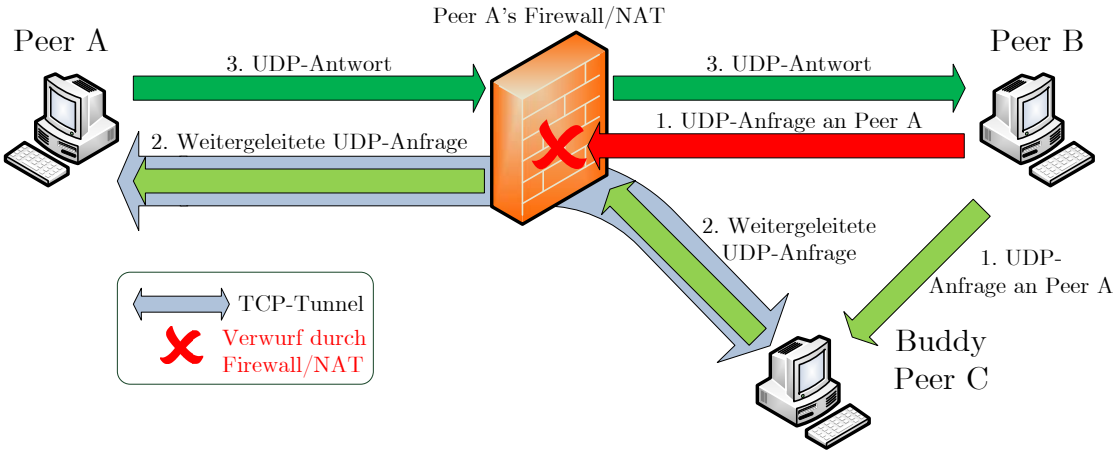


Abbildung 4.3.: Kontaktaufnahme von Peer B mit Peer A mit Hilfe von Buddy-Peer C [Bru06].

sind und die Peer A auf normalem Wege nicht erreichen. Diese Nachrichten werden durch den aufgebauten TCP-Tunnel an Peer A weitergeleitet (2. siehe Abbildung 4.3), woraufhin Peer A Peer B direkt antworten kann (3.).

Um auf Veränderungen im Netzwerk reagieren zu können, wird der Buddy-Mechanismus nach einer bestimmten Zeit wiederholt (in der Originalimplementierung der Version 0.49a alle 20 Minuten).

4.2.5. Zusammenfassung und Ausblick

Einzelne Knoten eines P2P-Netzwerkes wie dem Kad-Netz können durch die Zusammenarbeit mit anderen Knoten eine Erkennung durchführen, ob sie durch Firewalls oder NAT in ihrer Kommunikation eingeschränkt sind oder diese gar ganz verhindert wird. Bereits zwei Knoten reichen für eine einfache Firewall/NAT-Erkennung im Kad-Netzwerk aus. Drei Knoten ermöglichen ein selbst entwickeltes erweitertes Erkennungsverfahren (verteilte Erkennung ohne Wartezeit), bei dem als Gegenleistung für erhöhten Kommunikations- und Koordinationsaufwand schneller zuverlässigere Informationen über eventuell gegebene Kommunikationseinschränkungen gewonnen werden können.

Zahlreiche existente Lösungen setzen das Vorhandensein eines zentralen Servers voraus,

mit dem alle Knoten ständig kommunizieren müssen. Anstelle dieses zentralen Ansatzes, der deutlich anfälliger für den Ausfall einzelner Knoten ist, wird die redundante Datenspeicherung auf logisch benachbarten Knoten im Kad-Netzwerk ausgenutzt. Knoten, die sich nicht hinter einer NAT oder einer Firewall befinden, können eingehende Nachrichten über eine permanente TCP-Verbindung an logisch benachbarte Knoten hinter einem NAT oder einer Firewall weiterleiten und fungieren damit als sogenannter Buddy. Dadurch kann ein Knoten mit geeignetem Buddy vollständig an der Kommunikation im Kad-Netzwerk teilnehmen. Der dezentrale Charakter dieser Lösung bleibt dabei gewahrt und nur betroffene Knoten und Buddys müssen permanente Verbindungen aufrecht erhalten.

4.3. Dynamische Suchtoleranz

Das Kad-Protokoll muss die Anforderung eindeutiger Suchanfragen gewährleisten können, um einen wirklichen Ersatz eines Client-Server-Systems darzustellen. Die Deterministik einer Suche hängt bei Kad von der Suchtoleranz ab, innerhalb welcher ein P2P-Knoten für einen angefragten Hashwert noch als Ziel gilt. Die Suchtoleranz muss stets für die aktuell vorhandenen Knoten im Kad-Netzwerk alle möglichen Hashwerte abdecken.

4.3.1. Motivation

Eine statische Suchtoleranz, wie sie in der ursprünglichen Kad-Implementierung vorgesehen ist, reicht deshalb nicht aus, da nicht auf den Bei- und Austritt von Knoten reagiert werden kann und das Netz unbekannt ist. Es ist eine Dynamische Suchtoleranz (DST) notwendig, die angepasst wird, wenn neue Knoten in das Kad-Netz kommen oder dieses verlassen. Normalerweise kennt kein Knoten alle anderen Knoten, was aber notwendig ist, um die Suchtoleranz für alle Knoten festzulegen. Allerdings ist mit Hilfe eines am Institut entwickelten Algorithmus (genannt Kademlia Discovery (KaDis)-Algorithmus [Sko10, SDA+11a]), der nachfolgend erläutert wird, die globale Ermittlung aller Peers im Kad-Netzwerk mit einer hohen Wahrscheinlichkeit möglich, ohne dass ein Knoten von vornherein alle Knoten kennen muss.

Ein Master-Knoten kann somit die Suchtoleranz in Abhängigkeit der Knoten-IDs und der Größe des Adressraums mit Hilfe eines nachfolgend erläuterten Algorithmus berechnen,

nachdem mit dem KaDis-Algorithmus alle Knoten ermittelt wurden. Der Master-Knoten führt den KaDis-Algorithmus regelmäßig aus, um neue und ausgefallene Knoten festzustellen und dann ggf. die DST neu zu berechnen. Die initiale Suchtoleranz wird immer auf die Größe des Adressraums eingestellt. Eine Neuberechnung der Suchtoleranz ist nur erforderlich, solange die Suchtoleranz größer als ein voreingestellter Schwellwert ist. Nach der Neuberechnung der Suchtoleranz wird diese an alle Knoten gesendet.

Da die Berechnung der DST nur durch einen Knoten (Master-Knoten) vorgenommen werden soll, wird abschließend der Beweis erbracht, dass die ermittelte Suchtoleranz tatsächlich für alle anderen Knoten Gültigkeit besitzt. Die Beweisführung ist notwendig, da die Berechnung auf der Grundlage der Routing-Tabelle des Master-Knotens vorgenommen wird, aber jeder andere Knoten die Routing-Tabelle aus seiner lokalen Sicht ermittelt.

4.3.2. Stand der Technik

Der Lookup-Bereich bei Kademlia und Kad wird durch eine Toleranzzone bestimmt (siehe Abschnitt 2.4.6). Diese Zone ist durch die Anzahl der führenden Bits festgelegt, in denen zwei Hashwerte übereinstimmen müssen, um als ähnlich zu gelten. Die Anzahl der führenden Bits, die für die Zuordnung zu einem Bereich gewählt werden, wird durch den Parameter h angegeben. In der Kad-Implementierung der Version 0.49a wird dieser Parameter statisch auf die führenden 8 Bits des 128 Bit breiten Hashwerts eines Peers festgelegt. Wenn die ersten 8 Bits des Hashwerts von Peers also mit den ersten 8 Bits des Hashwerts eines Datums übereinstimmen, dann sind diese Peers für die Informationen zuständig. Kad routet dabei nicht exakt zu einem bestimmten Hashwert, sondern verteilt die Daten an alle Peers, die sich in der vordefinierten Toleranzzone befinden [Kru11, CN11, YLL09, SENB07, CCF10, Bru06].

Soll die Suchtoleranz dynamisch geregelt werden, so muss der Parameter h, der für die Anzahl der führenden Bits steht, im laufenden Prozess angepasst werden. Dabei muss sichergestellt werden, dass zu jeder Zeit mindestens ein Peer für ein beliebiges Datum verantwortlich ist. Momentan ist nach bestem Wissen des Autors kein Mechanismus existent, um diese Suchtoleranz automatisch an sich ändernde Bedingungen anzupassen.

4.3.3. Globale Peer-Ermittlung mittels KaDis-Algorithmus

Da es im Kad-Netzwerk keine zentrale Instanz gibt, ist die Anzahl der Peers unbekannt. Allerdings müssen für die Ermittlung der dynamischen Suchtoleranz alle Peers bekannt sein. Um trotzdem alle Knoten kontaktieren zu können, erhält jeder Knoten im Netzwerk eine eindeutige Zugangsknoten-ID, um den KaDis-Algorithmus anwenden zu können. Im Gegensatz zum in [Sko10, SDA+11a] vorgestellten Algorithmus wird hierbei direkt nach den eindeutigen Knoten-IDs und nicht nach deren Daten gesucht. Dabei sei angemerkt, dass den Zugangsknoten trotz ihrer eindeutigen Knoten-IDs dennoch Hashwerte zugewiesen werden müssen, um eine Verbindung zwischen Knoten und deren Daten zu etablieren.

Bei der Antwort eines ermittelten Knotens muss der Master-Knoten folgende Werte speichern:

- Knoten-Hashwert
- Knoten-IP-Adresse
- Knoten-UDP/TCP Port

Jeder Knoten wird in eine Liste nach seiner XOR-Distanz zum Master-Knoten einsortiert. Zudem ist mit jeder Ermittlung eines Knotens die Anzahl aller Knoten zu aktualisieren.

Der KaDis-Algorithmus benutzt einen Parameter k, der stets mit einem voreingestellten Schwellwert T verglichen wird und als Abbruchkriterium für den Algorithmus dient. Der Ablauf des KaDis-Algorithmus ist in Abbildung 4.4 dargestellt.

Es wird angenommen, dass die Zugangsknoten-IDs mit dem festen String *Knoten* anfangen und anschließend einen ansteigenden ganzzahligen Wert i enthalten. Zunächst wird versucht, Knoten i mit i=0 zu kontaktieren. Die Kontaktaufnahme folgt dabei der in Abschnitt 2.4.6 beschriebenen Lookup-Prozedur, bei der nach dem Hashwert der Knoten-ID gesucht wird. Ist die Kontaktaufnahme erfolgreich, so wird der Parameter k auf Null gesetzt, i um Eins erhöht und die Kontaktaufnahme mit dem nächsten Knoten versucht (ausgenommen bei der Kontaktaufnahme ist die eigene Knoten-ID). Ist die Kontaktaufnahme nicht erfolgreich, wird k um Eins erhöht. Sobald k den Wert T erreicht hat, bricht der Algorithmus ab. Die Wahrscheinlichkeit, alle Knoten zu finden, hängt somit von T ab. Beispielsweise kann T auf einen Wert eingestellt werden, der der Knotenausfallwahrscheinlichkeit P_A multipliziert mit der (angenommenen) Anzahl der Knoten im

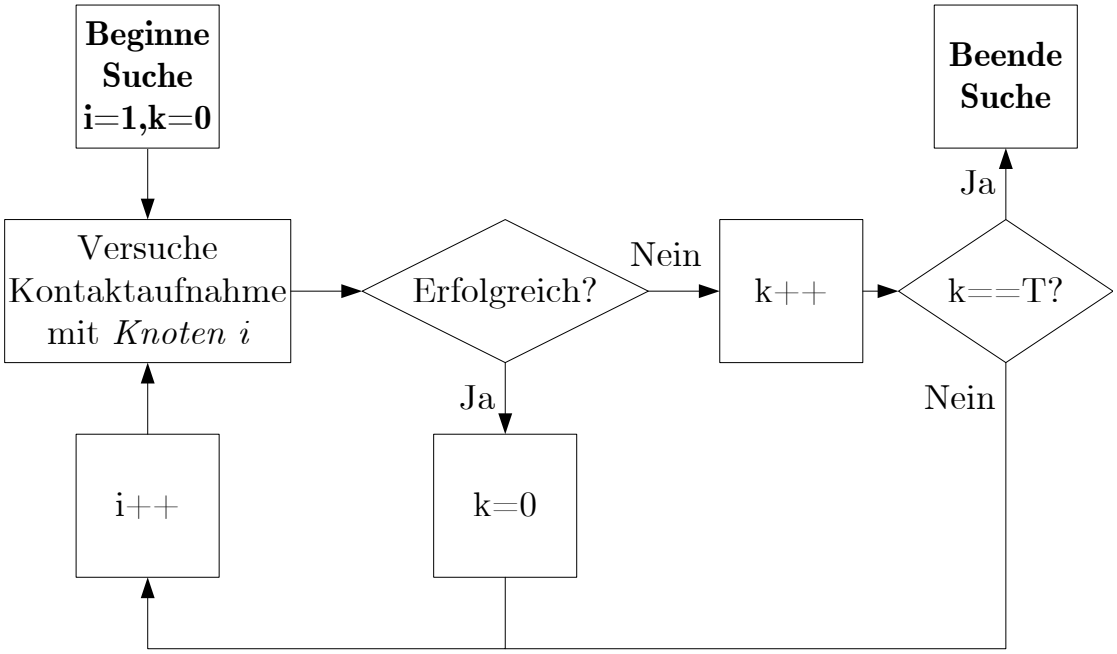


Abbildung 4.4.: Ablauf des KaDis-Algorithmus für die globale Peer-Ermittlung.

gesamten Netzwerk entspricht. Beispielsweise könnten so für P_A = 0,1 * 10.000 Knoten 1000 ausgefallene Knoten mit aufeinander folgenden IDs übersprungen werden.

In Tabelle 4.2 ist zur Veranschaulichung ein Beispielszenario dargestellt. Der Schwellwert T ist auf Drei eingestellt. Im Netzwerk befinden sich drei Knoten mit den Zugangsknoten-IDs *Knoten1*, *Knoten2* und *Knoten5*. Knoten1 und Knoten2 werden im Kad-Netz gefunden. Nach zwei nicht erfolgreichen Versuchen der Kontaktaufnahme mit den nicht vorhandenen Knoten3 und Knoten4 besitzt k den Wert Zwei, der immer noch kleiner ist als T. Anschließend wird Knoten5 erfolgreich kontaktiert und k wird auf Null zurückgesetzt. Der Algorithmus sucht weiter nach Knoten und bricht nach der erfolglosen Kontaktaufnahme mit Knoten8 ab. Nun wird angenommen, dass keine weiteren Knoten als *Knoten1*, *Knoten2* und *Knoten5* im Netzwerk sind.

i	Erfolgreich?	k	k==T?	Zugangsknoten-ID
1	Ja	0	Nein	Knoten1
2	Ja	0	Nein	Knoten2
3	Nein	1	Nein	Knoten3
4	Nein	2	Nein	Knoten4
5	Ja	0	Nein	Knoten5
6	Nein	1	Nein	Knoten6
7	Nein	2	Nein	Knoten7
8	Nein	3	Ja	Knoten8
Abbruch des Algorithmus				

Tabelle 4.2.: Beispielszenario für die Funktionsweise des KaDis-Algorithmus.

4.3.4. Algorithmus zur Berechnung der dynamischen Suchtoleranz

Gegeben ist ein Master-Knoten, der alle anderen Knoten mittels KaDis-Algorithmus ermittelt hat. Er besitzt somit Kenntnis über die Hashwerte aller anderen Knoten und deren Kontaktdaten. Die Knoten sind in eine Liste nach ihrer XOR-Distanz zum Master-Knoten einsortiert.

Gesucht ist die Suchtoleranz, für die es für jeden Hashwert des Adressraums mindestens ein/mehrere Ziel(e) gibt. Der Algorithmus für N Knoten, i = 0 und eine Suchtoleranz (ST) = 2^n lautet wie folgt: Der gesamte Adressraum, in dem sich an entsprechenden Stellen Knoten gemäß ihrer XOR-Distanzen zum Master-Knoten befinden, wird solange halbiert, wie sich in jeder Hälfte noch die vorgegebene Mindestanzahl von Knoten befindet. Pro Halbierung des Adressraums wird i um Eins erhöht. Nachdem der Adressraum so halbiert wurde, dass sich in den Hälften nicht mehr die Mindestanzahl von Knoten befindet, wird die Berechnung abgebrochen. Die Suchtoleranz ergibt sich mit ST = 2^{n-i}. Der Algorithmus hat eine Komplexität von $O(logN)$ [Kru12].

In den Abbildungen 4.5 - 4.8 wird der Algorithmus anhand von Iterationsschritten für N = 8 und n = 4 (also einem 4 Bit-Adressraum) verdeutlicht. Der gesamte Adressraum wird durch den Kad-Ring abgebildet. An einigen Hashwerten sind beispielhaft Knoten angeordnet. Bei jedem Schritt erfolgt eine Halbierung des Rings und die ST ändert sich

auf einen neuen Wert. Der Algorithmus wird abgebrochen, sobald sich in diesem Beispiel weniger als ein Knoten in einem beliebigen Ringsegment befindet.

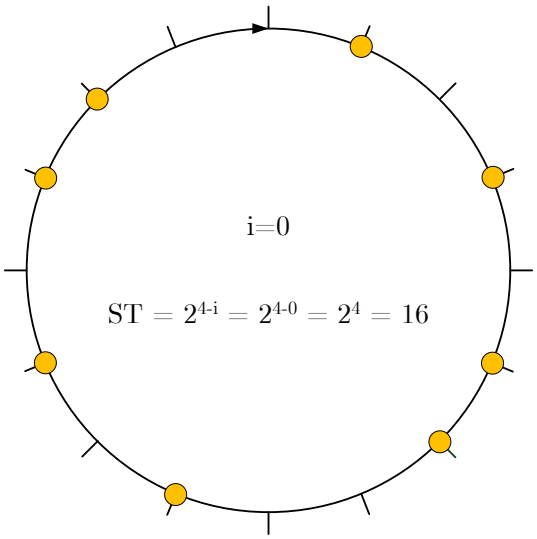


Abbildung 4.5.: Kad-Ring mit N=8 Knoten in einem 4 Bit-Adressraum (n=4) für i=0.

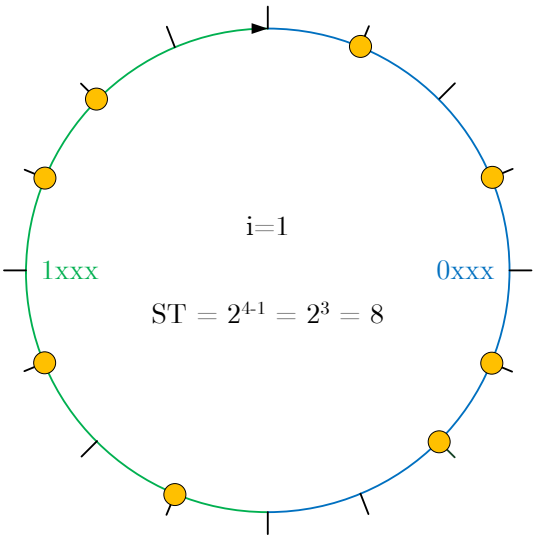


Abbildung 4.6.: Kad-Ring für i=1 (erste Iteration).

4.3.5. Beweis der Gültigkeit des DST-Algorithmus

Zu beweisen ist, dass die berechnete DST aus Sicht *eines* Knotens auch für *alle anderen* Knoten gilt. Die Berechnung der Suchtoleranz basiert auf einer Aufteilung des Rings in gleich große Segmente. Die Beweisführung erfolgt über den Nachweis, dass die Anzahl der Ringsegmente für jeden beliebigen zur Berechnung gewählten Knoten gleich bleibt und sie nur ihre Positionen ändern. In Abbildung 4.9 ist noch einmal der Kad-Ring mit 8 nummerierten Knoten dargestellt. Zur Veranschaulichung des Beweisführung ist in Abbildung 4.10 und 4.11 dargestellt, wie sich aus verschiedenen Knotensichten nach Durchführung des Algorithmus die Anzahl der Ringsegmente nicht ändert, sondern nur die Positionen verschieben.

In Abbildung 4.12 ist erneut der beispielhafte Kad-Ring dargestellt. In diesem sind zur

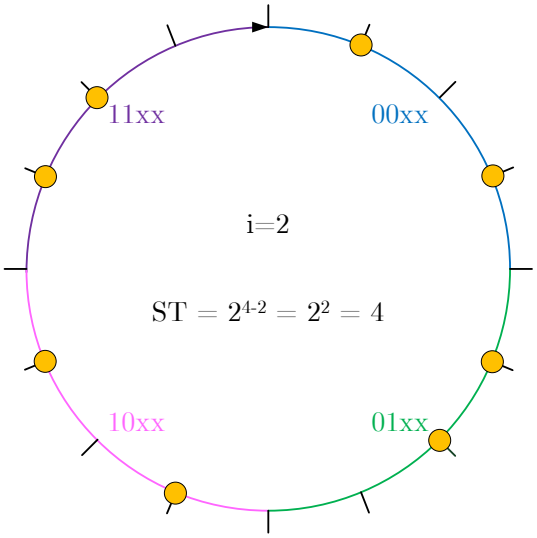


Abbildung 4.7.: Kad-Ring für i=2 (zweite Iteration).

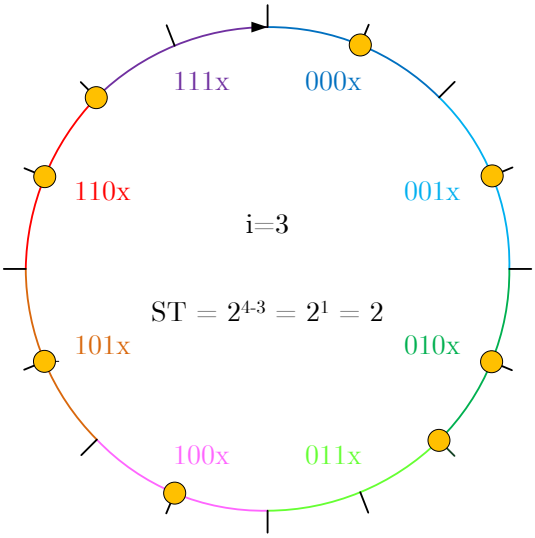


Abbildung 4.8.: Kad-Ring für i=3 (letzte Iteration vor dem Abbruch).

Vorbereitung der Beweisführung einige Formalisierungen ersichtlich. Der Adressraum umfasst n Bit, d.h. auch jeder Hashwert besteht aus n Bit. Die Teile des Rings (Ringsegmente) sind mit der Menge $M_{a_1,...,a_d}$ bezeichnet und umfassen jeweils alle entsprechenden Hashwerte $M_{a_1,...,a_d} := \{(a_1, ..., a_d, b_1, ..., b_{n-d}) \in \mathbb{Z}_2^n\}$ mit Hashwerten aus dem Vektorraum $\mathbb{Z}_2^n = (\{0,1\}^n, \oplus)$, in dem für alle Werte die Operation $\oplus$ (XOR, exklusives Oder) definiert ist. Der Präfix eines Hashwertes wird mit $a_1, ..., a_d$ bezeichnet; der Suffix entsprechend mit $b_1, ..., b_{n-d}$.

Weiterhin existiert eine Menge M, die alle Mengen $M_{a_1,...,a_d}$ umfasst:

$$M := \{M_{a_1,...,a_d} : a_1, ..., a_d \in \{0,1\}\}$$

Ferner sei eine Funktion $\phi_k : M \to M$ definiert, die eine beliebige Menge $M_{a_1,...,a_d}$ um einen Faktor k verschiebt, indem auf alle enthaltenen Hashwerte die Operation $\oplus$ angewendet wird:

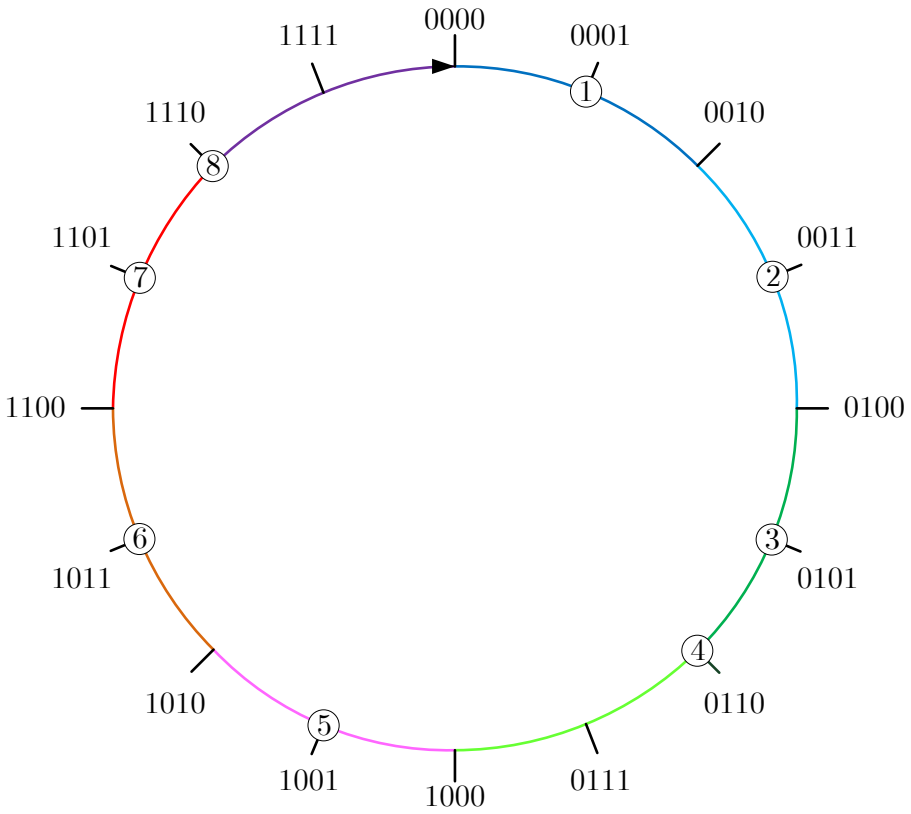


Abbildung 4.9.: Kad-Ring mit N=8 Knoten (nummeriert) in einem 4 Bit-Adressraum (n=4).

$$\phi_k(M_{a_1,...,a_d}) \;:=\; M_{a_1 \oplus k_1,...,a_d \oplus k_d}$$

Dabei sei $k \;:=\; (k_1, ..., k_n) \;\in\; \{0,1\}^n$ und für $a_i \;\neq\; a_i^{'}$ gilt $a_i \oplus k_i \;\neq\; a_i^{'} \oplus k_i$.

Zu zeigen ist, dass ϕ_k eine Bijektion ist. Eine Funktion ist bijektiv, wenn sie sowohl injektiv, d. h. kein Wert der Zielmenge mehrfach angenommen wird, als auch surjektiv ist, d. h. jeder Wert der Zielmenge angenommen wird. Zusammenfassend bedeutet das, dass eine vollständige Paarbildung zwischen den Elementen von Definitionsmenge und Zielmenge stattfindet. Nur Bijektionen behandeln ihren Definitionsbereich und ihren Wertebereich symmetrisch, sodass eine bijektive Funktion immer eine Umkehrfunktion hat bzw. invertierbar ist. Da die Menge M endlich ist, genügt es zu zeigen, dass ϕ_k injektiv ist. Damit

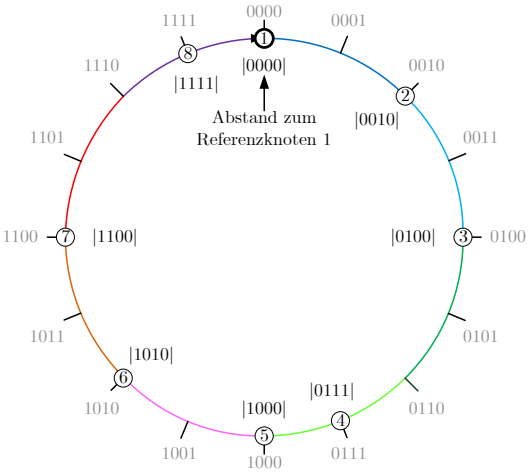


Abbildung 4.10.: Kad-Routing-Tabelle aus Sicht von Knoten 1.

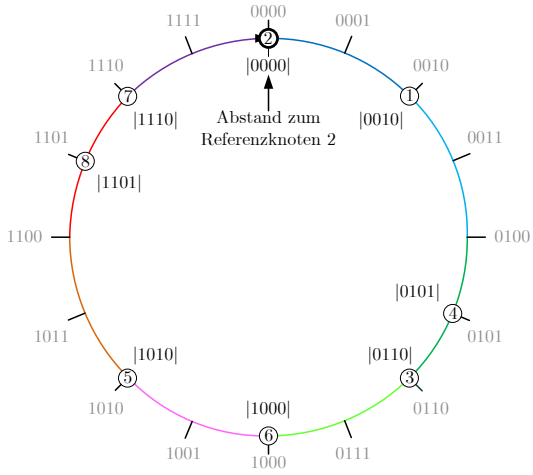


Abbildung 4.11.: Kad-Routing-Tabelle aus Sicht von Knoten 2.

ist ϕ_k dann automatisch auch bijektiv.

Formal gesprochen ist zu zeigen, dass zwei ungleiche Ringsegmente $M_{a_1,...,a_d} \neq M_{a_1',...,a_d'}$ durch Rotation mit dem gleichen Faktor k niemals auf das gleiche Ringsegment abgebildet werden, so dass $\phi_k(M_{a_1,...,a_d}) \neq \phi_k(M_{a_1',...,a_d'})$ gilt.

Gegeben seien nun zwei ungleiche Ringsegmente $M_{a_1,...,a_d} \neq M_{a_1',...,a_d'}$. Daraus folgt, dass auch die Präfixe a und a' unterschiedlich sind: $(a_1, ..., a_d) \neq (a_1', ..., a_d')$. Sie unterscheiden sich also zumindest in einem Bit i: $\exists i \ : \ a_i \neq a_i'$. Somit gilt sowohl für eine exklusive Veroderung dieses einzelnen Bits $a_i \oplus k_i \neq a_i' \oplus k_i$ als auch für alle Bits der Präfixe $(a_1 \oplus k_1, ..., a_d \oplus k_d) \neq (a_1' \oplus k_1, ..., a_d' \oplus k_d)$. Ebenfalls gilt Ungleichheit für die exklusiv veroderten zwei ungleichen Ringsegmente $M_{a_1 \oplus k_1,...,a_d \oplus k_d} \neq M_{a_1' \oplus k_1,...,a_d' \oplus k_d}$. Schlussendlich folgt daraus, dass $\phi_k(M_{a_1,...,a_d}) \neq \phi_k(M_{a_1',...,a_d'})$, was zu beweisen war.

Hiermit ist der Beweis erbracht, dass jeder Knoten mit Kenntnis aller Knoten die Suchtoleranz berechnen kann und diese aus Sicht jedes anderen Knoten auch Gültigkeit besitzt.

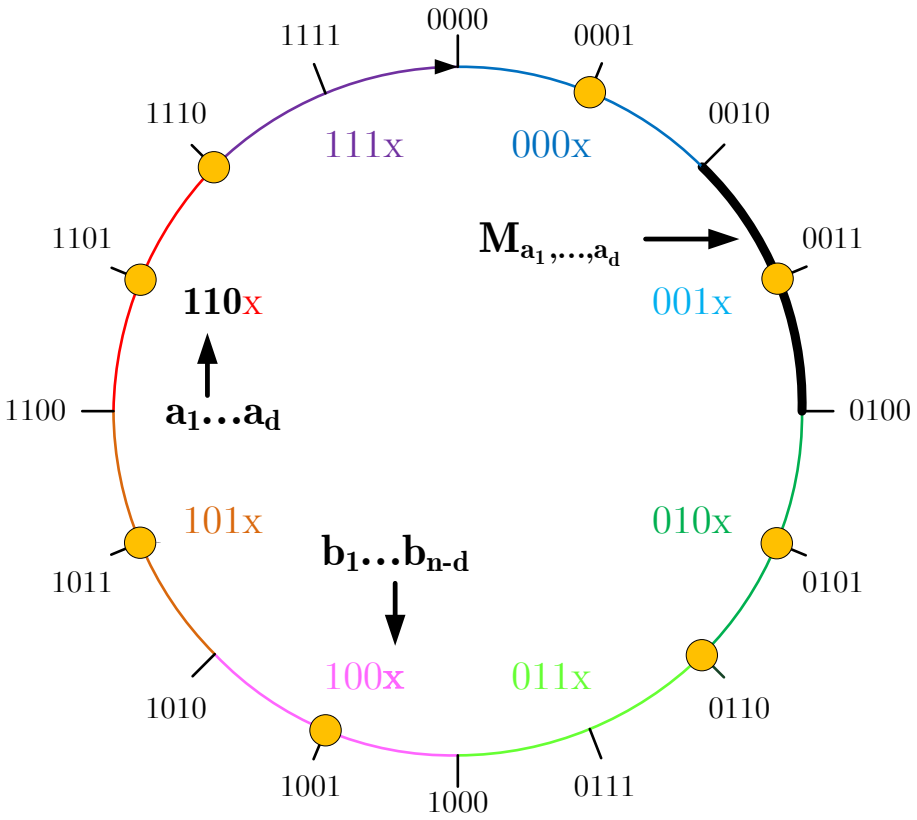


Abbildung 4.12.: Kad-Ring nach Aufteilung in Ringsegmente $M_{a_1,...,a_d}$

4.3.6. Einschränkungen des vorgestellten Ansatzes

Um den Master-Knoten als SPoF zu vermeiden, sollten zusätzliche Master-Knoten als Backup eingerichtet werden. Diese müssen regelmäßig überprüfen, ob sich der aktive Master-Knoten noch im Netzwerk befindet. Ist dies nicht der Fall, wird einer der Backup-Knoten zum aktiven Master-Knoten und bestimmt einen zusätzlichen Backup-Knoten zur Kompensation des ausgefallenen Knotens.

Der Beitritt und Ausfall eines Knotens wird durch den Master-Knoten festgestellt, da er periodisch den KaDis-Algorithmus ausführt. Hat sich die Netzwerkkonstellation verändert, muss eine Neuberechnung der Suchtoleranz sowie deren Verbreitung erfolgen. Je größer der Zeitraum zwischen zwei Ausführungen des KaDis-Algorithmus ist, desto höher ist die Wahrscheinlichkeit, dass sich die Netzwerkkonstellation verändert hat und die

Suchtoleranz nicht mehr aktuell ist. Je kleiner der Zeitraum gewählt wird, desto öfter ist Rechen- und Kommunikationsaufwand erforderlich, um die Knotenermittlung, Suchtoleranzberechnung sowie -verbreitung durchzuführen. In einem stabilen Kad-Netzwerk aus Zugangsknoten kann der Zeitraum jedoch auf einige Stunden eingestellt werden, ohne dass eine nicht aktuelle Suchtoleranz riskiert wird.

Da sich mit dem Hinzukommen neuer Knoten bzw. durch Knotenausfall durch die neue Suchtoleranz die Knoten-Zuständigkeit für Daten ändert, besitzt ein Knoten unter Umständen noch Daten, für die er nicht mehr verantwortlich ist. Daher werden sie bereits bei der Daten-Verteilung mit einer TTL versehen, so dass die Gültigkeit über den Ablauf ihrer TTL automatisch erlischt. Der TTL-Wert ist dabei Szenario-abhängig und wird entsprechend der Gültigkeitsdauer der Daten gewählt.

4.3.7. Zusammenfassung

Obwohl die Relevanz von DHT-basierten P2P-Netzwerken ständig zunimmt, findet die dynamische Einstellung der Suchtoleranz nur wenig Beachtung. Dieses rührt daher, dass P2P-Netze meistens im Bereich des File-Sharings eingesetzt werden (wie z. B. das Kad-Netzwerk, das Bestandteil von eMule ist) und daher eine hohe Anzahl von Nutzern vorausgesetzt wird. Somit kann in der Regel ein fester Wert eingestellt werden, ohne dass die Deterministik bei Suchanfragen leidet.

Dieser Abschnitt beschreibt ein Verfahren für die dynamischen Einstellung der Suchtoleranz im Kad-Netzwerk und beweist seine Gültigkeit. Ein einzelner Knoten, der Kenntnis über alle Knoten im Kad-Netzwerk besitzt, berechnet und aktualisiert regelmäßig die Suchtoleranz und verteilt die neu berechnete Suchtoleranz an alle Knoten. Somit gewährleistet das Kad-Protokoll die Anforderung eindeutiger Suchanfragen für die zu realisierende P2P-basierte Netzwerkinfrastruktur und stellt einen wirklichen Ersatz zum Client-Server-Paradigma dar.

Kapitel 5.

Entwurf und Realisierung P2P-basierter Netzwerkinfrastruktur

Auf Grund der steigenden Komplexität des Internets und seiner ständig wachsenden Nutzeranzahl weisen zahlreiche Internet-Basisdienste wie DNS und ISP-basierte Dienstleistungen wie die IP-Adressvergabe und deren Datenhaltung sowie Administrierungs- und Wartungsdienste schwerwiegende Probleme bezüglich Skalierbarkeit und Fehlertoleranz auf. Einerseits sind diese Basisdienste abhängig von alter (zentralisierter) Service-Infrastruktur, die nicht für die Dimension und Komplexität des heutigen und zukünftigen Internets ausgelegt ist. Andererseits muss kostenintensive neue Infrastruktur zur Verfügung gestellt werden, um die unzureichende Skalierbarkeit und Fehlertoleranz zu kompensieren.

Diese Forschungsarbeit widmet sich deshalb der Entwicklung P2P-basierter Kommunikationsinfrastruktur, da P2P-Technologie eine exzellente technologische Basis für die Realisierung effizienter dezentralisierter Lösungen darstellt, um existierende Strukturen zu ersetzen oder zu ergänzen. P2P-Netzwerke wie das Kad-Netzwerk weisen intrinsische Eigenschaften wie eine sehr hohe Skalierbarkeit sowie Widerstandsfähigkeit gegen Fehler und Angriffe auf, die somit per se beim Einsatz von P2P-Technologie nutzbar sind.

Ein durchschnittlicher Zugangsknoten eines Zugangsnetzwerks wie z. B. ein PowerQUICC II Pro (MPC8378) der Firma Freescale Semiconductor hat eine bestimmte verfügbare Speicher- und Rechenkapazität, die ohne zusätzliche Kosten nutzbar ist [Fre11, Nin11]. Übliche Werte aus der Praxis sind dabei eine Rechenleistung, die insgesamt 1234 Dhrystone-Millionen Instruktionen pro Sekunde (MIPS) beträgt und von der 40 % zur Verfügung gestellt werden können; selbst wenn aus Gründen der Reserve nie mehr als

50 % Grundlast auftreten sollen (siehe Abbildung 5.1) [Nin11]. Von der gebräuchlichen Random Access Memory (RAM)-Speicherkapazität in Höhe von 1 GByte RAM stehen ca. 400 MByte zur Verfügung [Nin11]. Darüber hinaus wird ein Zugangsknoten mit einigen MByte Flash-Speicher ausgestattet: als Beispiel aus der Praxis sei hier der Wert 64 MByte genannt, der aber hauptsächlich für das Bootimage und Konfigurationsparameter des Zugangsknotens genutzt werden soll [Nin11].

Auf Grund der verfügbaren Ressourcen sollen Zugangsknoten als Mitglieder des P2P-Netzwerks dienen und werden durch das selbstorganisierende DHT-basierte P2P-Netzwerk Kad verbunden. Jeder Zugangsknoten trägt somit einen Teil seiner RAM-Speicher- und Rechenkapazität zum verteilten DHT-Netzwerk bei.

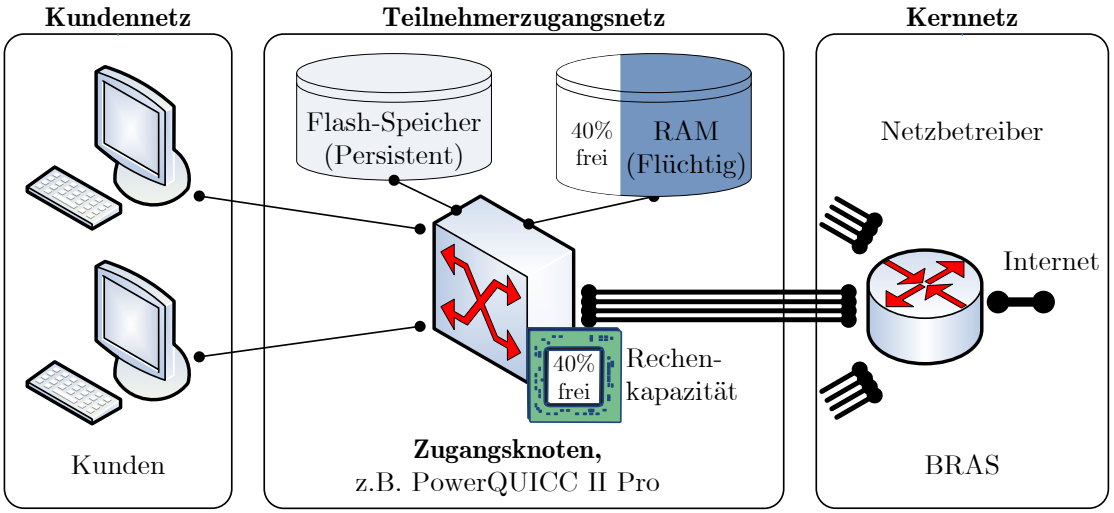


Abbildung 5.1.: TZN mit Zugangsknoten wie z. B. PowerQUICC II Pro (MPC8378). Ersichtlich sind der persistente Flash-Speicher, der flüchtige RAM-Speicher in Höhe von 1 GByte RAM sowie die 1234 Dhrystone-MIPS-Rechenkapazität. 40 % der RAM-Speicherressourcen sowie der Rechenkapazität stehen zur freien Verfügung [Fre11, Nin11].

In dieser Forschungsarbeit wurden

1. eine P2P-basierte Speicherplattform für die Speicherung von Sitzungsdaten in Internet-Zugangsnetzwerken (PSP) [DGT+10],

2. ein verteiltes Rechensystem auf der Grundlage einer verteilten P2P-Umgebung (DuDE) [Sko10, SDA+11a, SDA+11b]
3. sowie ein P2P-basiertes DNS (P-DONAS) entwickelt [DSA+bm, DSL+10, Weg08, Pfe09].

Für alle drei Systeme wurden zum Nachweis der Funktionalität voll funktionsfähige Prototypen entwickelt [Kar12b, SDA+11b, DSL+10]. Abschnitt 5.1 und 5.2 beschreiben ausgewählte Aspekte der P2P-basierten Speicherplattform PSP sowie des P2P-basierten verteilten Rechensystems DuDE. Als ein ausgewähltes Ergebnis wird die Realisierung des P2P-basierten DNS P-DONAS detailliert in Abschnitt 5.3 vorgestellt.

Zugangsknoten verbrauchen ferner einen bedeutenden Teil der in Telekommunikationsnetzwerken benötigten Energie [KD10, GLL08]. Insbesondere fehlendes Power Management und die ständige Verfügbarkeit („Always On“) dieser Netzwerkkomponenten, die von Nutzern gefordert wird, führen dazu, dass die Leistungsaufnahme in keiner starken Abhängigkeit vom auftretenden Datenverkehr steht. Folglich wird bei der üblichen Nutzung Energie verschwendet, was im Zuge der „Green IT“-Bestrebung eine Belastung für Umwelt und Ressourcen darstellt. Indem aber ungenutzte Ressourcen der Zugangsknoten für neue Aufgaben eingesetzt werden, wird die Energieeffizienz erhöht. An anderer Stelle können zudem Netzwerkkomponenten eingespart werden, die zuvor diese Aufgaben erfüllten, so dass weitere Energie eingespart werden kann.

Die Nutzung von P2P-Overlay-Netzwerken für die Realisierung administrativer Dienste in digitalen Netzwerken stellt einen Forschungsschwerpunkt innerhalb des übergeordneten Projekts Encompassing Technological Reuse of P2P in Access Networks (ENTROPIA) dar, bei dem die Broadband Access Division von Nokia Siemens Networks GmbH & Co. KG in Greifswald mit Inspiration und fortwährender Unterstützung mitwirkte. Für diesen Forschungsschwerpunkt läuft derzeit ein Europäisches Patentanmeldungsverfahren [BD10].

5.1. Die P2P-basierte Speicherplattform *PSP*

ISP müssen die Sitzungsdaten ihrer Kunden speichern, da diese zu Betriebs-, Verwaltungs- und Kontrollzwecken benötigt werden. Dabei zeichnet jeder Zugangsknoten des Zugangsnetzwerks eines ISP die Sitzungsdaten aller Kunden auf, die über physikalische Ports mit

ihm verbunden sind. Sitzungsdaten umfassen unter anderem Adressinformationen wie IP-Adressen, physikalische Ports, MAC-Adressen und „Lease Times“ von IP-Adressen. Ein Kunde, der sich mit dem Internet verbindet, fragt automatisch über DHCP eine IP-Adresse an (siehe Abschnitt A.2).

5.1.1. Motivation

Sitzungsdaten sind hochgradig flüchtig, da sich die Datenbasis fortwährend ändert, was zu häufigen Speicherzugriffen für das Schreiben der Daten führt. Darüber hinaus müssen sie dauerhaft gespeichert und regelmäßig abgerufen werden, da sie für die Datenweiterleitung sowie die Konfiguration der Session-Filter eines Zugangsknotens benötigt werden. Session-Filter blockieren Datenverkehr von nicht konfigurierten Teilnehmern, d.h., von Kunden, die keine IP-Adresse abgefragt haben. Im Falle des Neustarts oder Absturzes eines Zugangsknotens müssen die Daten neu geladen werden, d.h. die Wiederherstellung der Sitzungsdaten (Session Data Recovery) ist notwendig. Erst nach dieser Wiederherstellung werden auch die Session-Filter neu konfiguriert und der Kunde hat wieder Internet-Zugang. Ein Neustart eines Zugangsknotens findet im schlechtesten Fall 2 - 3 Mal die Woche statt, im Durchschnitt alle 4 Wochen und im besten Fall 0 - 1 Mal im Jahr [Nin11]. Um auch im schlechtesten Fall die Persistenz der Sitzungsdaten sicherzustellen, wird heutzutage Flash-Speicher eingesetzt. Allerdings ist dieser in seiner Verfügbarkeit und Wiederbeschreibbarkeit limitiert und außerdem für andere Zwecke wie die Speicherung von Konfigurationsparametern eines Zugangsknotens vorgesehen. Flash-Speicher hat eine Lebensdauer von 10.000 Schreibzugriffen im schlechtesten Fall [LFM11]. Für die Aktualisierung der Sitzungsdaten wird alle 15 min bis 1 h ein Schreibzugriff notwendig; somit kann der Flash-Speicher schon nach 104 Tagen ausfallen [Nin11].

Neben Flash-Speicher besitzen Zugangsknoten eine bestimmte *verfügbare* flüchtige RAM-Speicher- sowie Rechenkapazität, die ungenutzt ist und somit ohne zusätzliche Kosten verwendet werden kann (siehe Abbildung 5.1). Anders als Flash-Speicher kann RAM-Speicher fast unbegrenzt oft beschrieben werden. Deshalb sollen die RAM-Speicherressourcen aller Zugangsknoten gemeinsam genutzt werden, indem die Zugangsknoten mit dem DHT-basierten Kad-Netzwerk verbunden werden. Dabei dient das Kad-Netzwerk als semipermanenter verteilter Speicher für eine strukturierte Speicherung der Sitzungsdaten. Jeder Zugangsknoten trägt einen Teil seiner Speicher- und Rechenkapazität zum

Kad-Netzwerk bei. Nach dem Neustart oder Absturz eines Zugangsknotens wird die Wiederherstellung der Sitzungsdaten vorgenommen, indem die notwendigen Daten gezielt aus dem Kad-Netzwerk bezogen werden. Hierdurch wird die Persistenz der Daten gewährleistet.

Sitzungsdaten müssen mit einer sehr hohen Zuverlässigkeit verfügbar sein. Typischerweise muss die Verfügbarkeitswahrscheinlichkeit 99,999 % betragen. Da die Daten verteilt auf allen Zugangsknoten gespeichert sind, muss die Verfügbarkeit durch entsprechende Mechanismen gewährleistet werden, da Zugangsknoten ausfallen können. Um die Redundanz bei der Datenspeicherung minimal zu halten und gleichzeitig eine hohe Verfügbarkeit sicherzustellen, werden Reed-Solomon-Codes aus der Gruppe der Erasure Resilient Codes (ERCs) eingesetzt [W.K04].

Nachfolgend werden folgende Hauptbeiträge kurz dargestellt [DGT+10]:

- P2P-Technologie wird hinsichtlich ihrer Eignung für die Umsetzung einer dezentralisierten Speicherplattform analysiert. Anschließend erfolgt mit Hilfe der gewonnenen Erkenntnisse die Realisierung einer semipermanenten Speicherung von Sitzungsdaten in flüchtigem RAM-Speicher auf der Grundlage von P2P-Technologie.
- ERCs und einfache Datenreplikation (DR) werden hinsichtlich der Datenverfügbarkeit und dem Speicher-Overhead verglichen. Als geeigneter Vertreter der ERCs wird der Reed-Solomon-Code ausgewählt und umgesetzt, um nahezu 100 %-ige Datenverfügbarkeit sicherzustellen.
- Die Kad-basierte Architektur und Funktionalität der P2P-basierten Speicherplattform (PSP) werden beschrieben.

5.1.2. Stand der Technik

In den Arbeiten [J. 00, RA10, M. 07, C. 08, DR01, YYXT08] wird die Nutzung von DHT-basierten Lösungen für die verteilte Datenspeicherung vorgeschlagen. Im Speziellen sollen in [J. 00] global verteilte, nicht vertrauenswürdige Server genutzt werden. Zur Sicherstellung der Vertrauenswürdigkeit müssen spezielle Maßnahmen ergriffen werden, auf die PSP verzichten kann, da Zugangsknoten vertrauenswürdige Instanzen darstellen. Ribeiro et al. [RA10] stellen eine DHT-basierte Plattform mit einer niedrigen Peer-Verfügbarkeit bzw. hohen Ein- und Austrittsdynamik für persistente Datenspeicherung

vor. Die Lookup-Komplexität beträgt bei diesem Ansatz $O(N/log_2(N))$ notwendige Hops pro Lookup. PSP weist demgegenüber durch die Nutzung des Kad-Netzwerks Vorteile auf, da die Lookup-Komplexität $O(log_{2^b}(N))$ geringer ist (siehe Abschnitt 4.1). In [M. 07] stellen die Autoren ein öffentliches Datenmanagementsystem für Web-Applikationen vor. Das Hauptaugenmerk liegt auf der Daten- und Diensteintegration sowie der Bereitstellung von Funktionalität zur Verarbeitung von Datenbanken. Insbesondere die effiziente Datenbankabfrage steht im Vordergrund. Dieser Rechenaufwand ist für PSP nicht notwendig, da PSP den Fokus auf eine automatisierte Datenverteilung und -sammlung ohne Notwendigkeit für komplexe Anfragen legt. Dies gilt ebenso für den Ansatz in [C. 08], der sich auf eine Pastry-basierte skalierbare Speicherplattform für die Speicherung von aufgezeichneten IP-Datenströmen konzentriert. Hauptsächlich widmet sich dieser Ansatz der Performance-Verbesserung komplexer Suchoperationen für die Abfrage dieser Datenströme. Druschel et al. [DR01] beschreibt ein globales Internet-basiertes Speichersystem bestehend aus nicht vertrauenswürdigen Knoten auf der Grundlage von Pastry. Für die Gewährleistung hoher Datenverfügbarkeit nutzen die Autoren reine Datenreplikation. Für PSP kommen Reed-Solomon-Codes zur Anwendung, die hohe Datenverfügbarkeit mit wesentlich geringerer Datenredundanz sicherzustellen. PSP basiert zudem im Gegensatz zu [C. 08] und [DR01] auf dem Kad-Netzwerk anstatt auf Pastry und hebt sich diesbezüglich durch die in Abschnitt 4.1 beschriebenen Vorteile ab.

Toka et al. [TDM11] befasst sich mit der Speicherung von großen Datenmengen in der Größenordnung von einigen GByte auf Peers mit einer hohen Ein- und Austrittsdynamik. Da zudem eine sehr heterogene Peer-Performance angenommen wird, wurden spezielle Scheduling-Strategien für den Datentransfer integriert. Diese zusätzlichen Scheduling-Strategien sind für PSP nicht notwendig, da Zugangsknoten mit relativ homogener Performance angenommen werden können und somit solche Strategien nicht notwendig sind. In der Arbeit von Qin Xin et al. [Qin04] werden Datensätze in Redundanzgruppen eingeteilt. Dabei liegt der Fokus auf einer hohen Verfügbarkeit der Daten für P2P-basierte Applikationen mit einer hohen Peer-Wanderungsrate, d.h. einem häufigen Bei- und Austritt von Peers. Es erfolgt eine history-basierte Auswahl von Knoten, die Daten speichern sollen. Dieses Modell eignet sich besser für P2P-basierte File-Sharing-Applikationen mit einer hohen Peer-Wanderungsrate als für PSP.

Im Gegensatz zu dem Ansatz dieser Forschungsarbeit werden in keiner der oben vorgestellten Arbeiten verfügbare Ressourcen einer vertrauenswürdigen Infrastruktur genutzt,

um eine Speicherplattform für allgemeine Daten zu realisieren. Ye et al. [YYXT08] stellt eine verteilte Speicherplattform aus relativ vertrauenswürdigen Peers mit dem Fokus auf der Gewährleistung von Datenaktualität und -sicherheit vor. Allerdings sind zusätzliche Server notwendig, die vollständige Routing-Tabellen vorhalten und so kann jedes Ziel mit einem Hop erreicht werden. PSP stellt demgegenüber ein vollständig dezentralisiertes System dar und verzichtet auf den Einsatz von Servern, die einen Flaschenhals und SPoF bilden. So erreicht PSP hohe Skalierbarkeit und niedrige Fehleranfälligkeit.

5.1.3. P2P-Technologie für die Realisierung von PSP

Um die Nutzung des Flash-Speichers für eine zentralisierte Sitzungsdatenspeicherung auf jedem einzelnen Zugangsknoten zu vermeiden, ist eine dezentralisierte Speicherlösung wünschenswert, die *verfügbare* Ressourcen nutzt. Jeder Zugangsknoten ist über eine einzigartige ID identifizierbar, die aus einer alphanumerischen Zeichenkette bestehen kann. Diese Zugangsknoten, die sich im TZN befinden (siehe Abbildung 5.1), sollen als Plattform für die vorgeschlagene Lösung dienen.

Um solch ein dezentralisiertes System effizient zu realisieren, weist P2P-Technologie zahlreiche vielversprechende Eigenschaften auf. Über seine Anwendung für das File-Sharing hinaus stellt P2P ein neues Netzwerkparadigma dar (siehe Abschnitt 2.4), bei dem keine Clients und Server existieren. Alle Netzwerkknoten sind Peers, d.h. Zugangsknoten sind sowohl Client als auch Server und bilden oberhalb der existierenden Topologie ein logisches Overlay. Dabei sind hohe Skalierbarkeit und Ausfallsicherheit intrinsische Eigenschaften von P2P-Netzwerken, die ohne zusätzliche Kosten genutzt werden können.

Jeder Zugangsknoten trägt mit seinen verfügbaren Ressourcen zur dezentralisierten Speicherplattform bei. Das P2P-Netzwerk selbst wird somit zur *verteilten Speicherressource*, wobei die Zugangsknoten ihre verfügbaren RAM-Ressourcen teilen. Jeder Zugangsknoten hält dabei nur einen Teil des gesamten Sitzungsdatenbestandes. Da Zugangsknoten auch ihre Rechenleistung teilen, stellt das Netzwerk zudem eine *verteilte Rechenressource* dar.

In dem eingesetzten DHT-basierten Netzwerk Kad gibt es keinen SPoF, so dass hohe Widerstandsfähigkeit gegenüber DoS-Angriffen und Netzwerkausfällen als im Prinzip gegeben vorausgesetzt werden kann [SW05]. Kad ist als DHT-basiertes System *selbstorganisierend*, d.h., es kommt ohne zusätzliche Wartung von außen aus. Mit Hilfe der in

Abschnitt 4.3 beschriebenen DST ermöglicht Kad zudem einen deterministischen Lookup. Redundante Datenhaltung wird von Kad per se unterstützt, wodurch eine nochmals gesteigerte Ausfallsicherheit der Sitzungsdaten erreicht wird, da diese auch bei einem oder mehreren ausgefallenen Zugangsknoten noch mit einer hohen Wahrscheinlichkeit verfügbar sind. Um allerdings nahezu 100 %-ige Sitzungsdatenverfügbarkeit in flüchtigen RAM-Speichern zu erreichen, sind nachfolgend vorgestellte Maßnahmen notwendig.

5.1.4. Sicherstellung 99,999 %-iger Datenverfügbarkeit

Der wichtigste und entscheidende Aspekt bei der Datenspeicherung im Allgemeinen und Sitzungsdaten im Speziellen ist die Gewährleistung hoher Datenverfügbarkeit [W.K04]. Insbesondere im Fall verteilter Datenspeicherung müssen angemessene Maßnahmen getroffen werden, um eine Datenverfügbarkeit P_d von typischerweise mindestens 99,999 % zu garantieren, falls Teile des verteilten Speichersystems ausfallen.

Eine Methode, um einen hohen Wert für P_d sicherzustellen, ist pure DR. Alternativ lassen sich Daten auch mit Hilfe von ERCs in Datenstücke einteilen und werden dabei um einige verschränkte Kodierungsstücke ergänzt. Daten und ihre Kopien bzw. Datenstücke und Kodierungsstücke werden auf den Komponenten des verteilten Speichersystems abgelegt, die im vorliegenden Anwendungsfall Zugangsknoten sind. Die Zugangsknotenverfügbarkeit P_n ist dabei konstant und kann nur durch die Verwendung zuverlässigerer Hardware verbessert werden. Daher ist eine Erhöhung der Datenverfügbarkeit P_d nur durch das Hinzufügen von Redundanz möglich. Folglich steigt auch das zu speichernde Datenvolumen. Die Zunahme des Datenvolumens stellt somit die Kosten der höheren Datenverfügbarkeit dar und wird als Speicher-Overhead-Faktor S bezeichnet. S bezeichnet dabei das Verhältnis aus ursprünglicher Datenmenge + hinzugefügte Redundanz geteilt durch die ursprüngliche Datenmenge. In Tabelle 5.1 sind alle grundlegenden Parameter für die Klassifizierung der verteilten Speicherung zusammenfassend aufgeführt [Got09].

Da die Speicherkapazität der Zugangsknoten begrenzt ist, sollte S so klein wie möglich sein bei gleichzeitig möglichst hoher Datenverfügbarkeit P_d. Um den besten Kandidaten für die Erfüllung dieser Voraussetzung zu bestimmen, wird nachfolgend einfache DR mit ERCs hinsichtlich S und P_d verglichen.

Parameter	Bedeutung
P_n	Zugangsknotenverfügbarkeit
P_d	Datenverfügbarkeit
S	Speicher-Overhead-Faktor

Tabelle 5.1.: Grundlegende Parameter der verteilten Speicherung

5.1.4.1. Einfache Datenreplikation vs. Erasure Resilient Codes

Bei der Anwendung von einfacher DR werden Daten von dem Knoten, der die Daten besitzt, auf S-1 andere Knoten kopiert. Dabei ist S ein ganzzahliger Wert, da vollständige Datenkopien erstellt werden. Um die ursprünglichen Daten nach einem Knotenneustart oder -ausfall wiederherzustellen, müssen sie von einem dieser S-1 Knoten geholt werden. Folglich hängt P_d von P_n und S ab [W.K04]:

$$P_{d,\ DR} \ = \ \sum_{i=1}^{S-1} \binom{S-1}{i} P_n^i (1-P_n)^{S-1-i} \tag{5.1}$$

Wenn ERCs genutzt werden, werden die Daten vor der Speicherung in m Datenteile aufgeteilt (siehe Abbildung 5.2). Anschließend erfolgt die Generierung von k Kodierungsstücken, die Informationen von allen m Datenstücken enthalten. Insgesamt werden also n = m + k Stücke generiert, die die Daten repräsentieren.

Aus beliebigen m dieser n Stücke ist ein effizienter ERC in der Lage, die ursprünglichen Daten wiederherzustellen; selbst, wenn einige Stücke fehlen (Löschungen, siehe Abbildung 5.3). Daher hängt P_d von P_n, m und n ab, wobei das Verhältnis n/m dem Speicher-Overhead-Faktor S entspricht [W.K04]:

$$P_{d,\ ERC} \ = \ \sum_{i=m}^{n} \binom{n}{i} P_n^i (1-P_n)^{n-i} \tag{5.2}$$

Grundsätzlich erhöhen sowohl DR als auch ERCs die Datenverfügbarkeit. Allerdings erreichen ERCs die gewünschte Datenverfügbarkeit P_d mit deutlich weniger Speicher-Overhead als DR. Sei P_d beispielsweise 99,999 % und Knoten seien mit P_n = 90 % verfügbar. Bei der Nutzung von DR ergibt sich laut Gleichung 5.1 der Wert 6 für S,

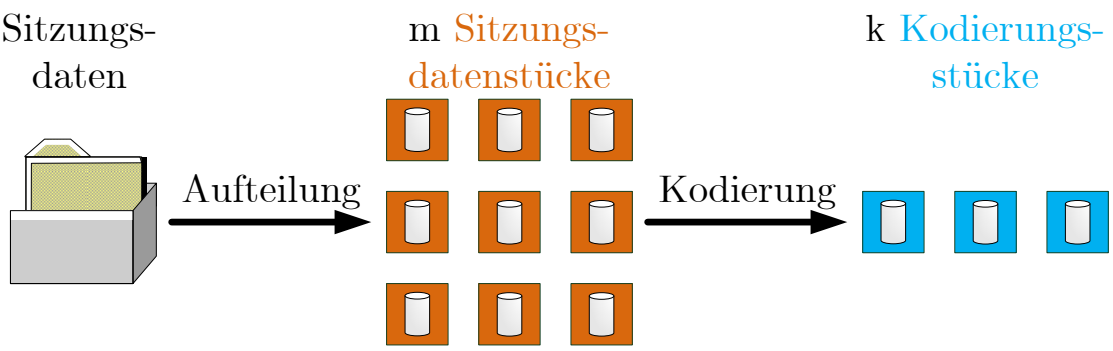


Abbildung 5.2.: Aufteilung der Sitzungsdaten in m=9 Datenstücke und Generierung von k=3 Kodierungstücken mittels ERCs.

d.h. die Daten müssten fünf mal repliziert werden. Im Gegensatz dazu bieten ERC P_d = 99,999 % mit einem viel niedrigeren Wert $S = 2$ (mit z. B. n = 32 und m =16). Mit Hilfe von ERC kann S also um den Faktor drei erniedrigt werden. Infolgedessen werden ERCs für die Gewährleistung hoher Datenverfügbarkeit genutzt.

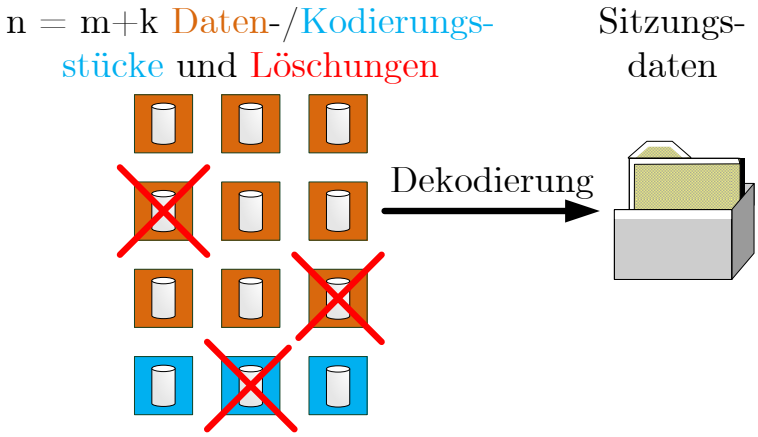


Abbildung 5.3.: Dekodierung der Sitzungsdaten aus beliebigen m=9 Stücken mittels ERCs.

ERCs weisen eine relativ hohe Komplexität bezüglich der Generierung von Kodierungsstücken sowie deren Dekodierung in eine vollständige Datei auf [Got09]. Zugangsknoten haben allerdings genügend Rechenkapazität zur Verfügung, um diese Rechenaufgabe zu

lösen.

5.1.4.2. Reed-Solomon-Codes

Reed-Solomon-Codes werden als geeigneter ERC ausgewählt, da sie am zweckmäßigsten für die Gewährleistung hoher Verfügbarkeit für die verteile Speicherung sind. Reed-Solomon-Codes wurden umfassend erforscht und sind von enormer praktischer Relevanz (z. B. für die Fehlerkorrektur von Digital Versatile Discs (DVDs), Satellitenkommunikation und Digital Video Broadcasting (DVB)) [Ang09, Got09].

Ein Vorteil ist ihre Fähigkeit der Datenwiederherstellung aus *exakt* m *beliebigen* Datenstücken [HP03]. Diese Eigenschaft macht sie zusammen mit Bose-Chaudhuri-Hocquenghem (BCH)-Codes einzigartig, da kein anderer Code diese Fähigkeit besitzt. Reed-Solomon-Codes nutzen somit die hinzugefügte Redundanz optimal aus. Ein weiterer Vorteil ergibt sich daraus, dass Reed-Solomon-Codes systematische Block-Codes sind. Das bedeutet, dass Daten- und Kodierungsstücke geordnet sind, wobei die Kodierungsstücke den Datenstücken folgen. Wenn m von n Stücken verfügbar sind und es sich bei diesen um Datenstücke handelt, kann die komplette Datei durch einfache Zusammensetzung dieser Datenstücke wiederhergestellt werden, d.h. *ohne* Dekodierung. Dadurch erweisen sich Reed-Solomon-Codes als äußerst effizient und sparen Rechenzeit und Energie für die Dekodierung.

5.1.5. Kad-basierte Realisierung von PSP

PSP wurde durch die Verknüpfung der Zugangsknoten eines Netzbetreibers als Kad-basierter DHT-Ring realisiert. Das Kad-Protokoll wurde dafür so modifiziert, dass die Übertragung von Datenstücken ermöglicht wird und wurde um eine Löschfunktionalität für das Entfernen von Datenstücken erweitert. Die Zugangsknoten führen eine strukturierte Speicherung der Sitzungsdaten- und Kodierungsstücke anderer PSP-Knoten durch (siehe Abbildung 5.4).

Da ein Zugangsknoten einen Peer repräsentiert, d.h. einen PSP-Knoten im Kad-Netzwerk, wird ihm ein Hashwert zugewiesen, der z. B. aus seiner IP-Adresse berechnet wird. Anhand dieses Hashwertes platziert sich der Zugangsknoten auf dem DHT-Ring, der den gesamten Hash-Adressraum umfasst.

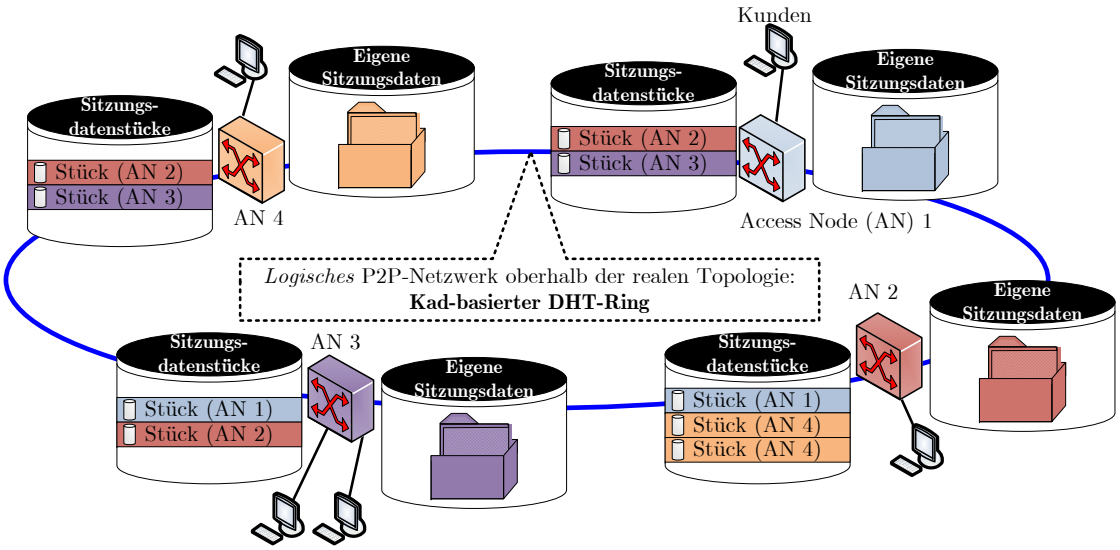


Abbildung 5.4.: Kad-basierter DHT-Ring für die strukturierte Speicherung von Sitzungsdaten- und Kodierungsstücken auf Zugangsknoten (ANs).

Dabei ist jeder Zugangsknoten dafür zuständig,

- seine eigenen Sitzungsdaten in seinem RAM zu speichern, um angeschlossene Kunden mittels DHCP mit IP-Adressen zu versorgen und
- Sitzungsdaten- und Kodierungsstücke anderer Knoten mit Hashwerten ähnlich dem eigenen Hashwert zu speichern. Hashwerte werden aus der Verknüpfung der Zugangsknoten-ID und dem „Dateinamen“ der Sitzungsdaten und Kodierungsstücke erstellt.

Als Machbarkeitsbeweis wurde ein Prototyp entwickelt, der einen Zugangsknoten mit PSP-Funktionalität emuliert. Die Implementierung erfolgte in C++ für Windows-Betriebssysteme.

5.1.5.1. Architektur eines PSP-Knotens

Die Architektur eines PSP-Knotens ist in Abbildung 5.5 ersichtlich. Die Hauptkomponenten sind ein Speicher mit den eigenen Sitzungsdaten (1), ein Speicher mit den Sitzungsdaten- und Kodierungsstücken anderer PSP-Knoten (2), eine Routing-Tabelle

(3), eine Komponente mit erweiterter Kad-Funktionalität (4) und ein Modul mit Steuerungsfunktionalität (5).

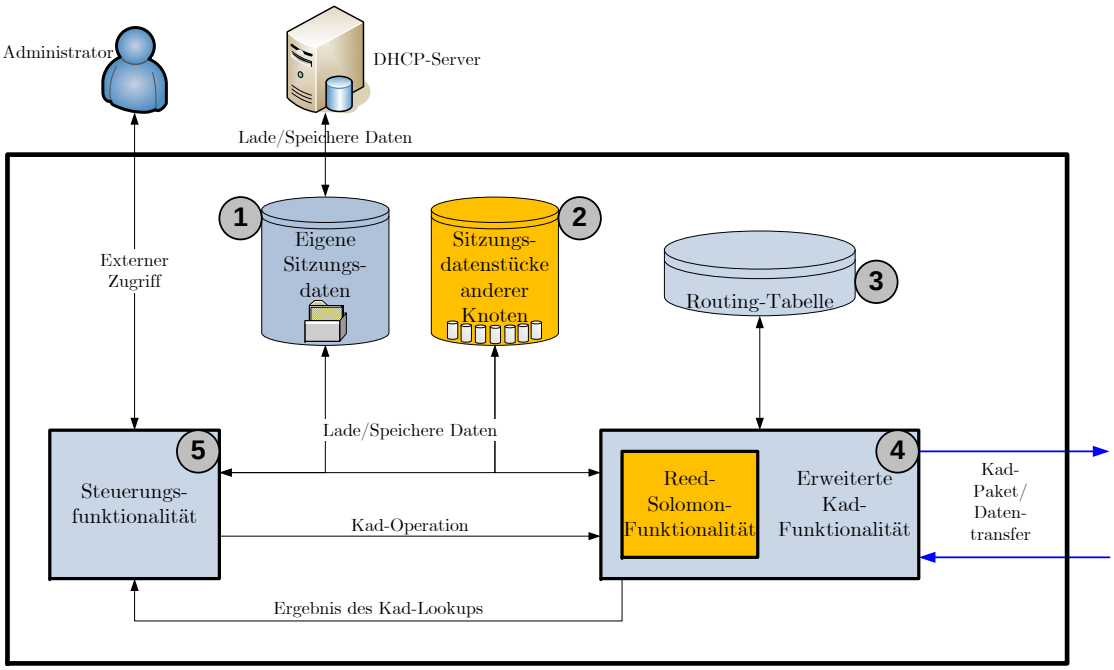


Abbildung 5.5.: Blockdiagramm eines PSP-Knotens. Der eigene Sitzungsdatenbestand, auf den ein DHCP-Server zur Bereitstellung von IP-Adressen zugreift, wird über das Kad-Protokoll auf anderen Knoten gesichert. Externe Steuerung durch einen Administrator ist möglich.

Der DHCP-Server hat Zugriff auf den Speicher mit eigenen Sitzungsdaten, um angeschlossene Kunden mit IP-Adressen und Lease-Zeiten versorgen zu können. Der zweite Speicher dient der Vorhaltung von Sitzungsdaten- und Kodierungsstücken anderer Knoten, für die der PSP-Knoten auf Grund seines Hashwertes zuständig ist. Die Routing-Tabelle enthält Kontaktinformationen über andere PSP-Knoten, um mit ihnen kommunizieren zu können. Der Kad-Block realisiert die erweiterte Funktionalität des Kad-Protokolls. Dies umfasst das Bootstrapping, die Instandhaltungsfunktionalität und die Ausführung von Lookups, Such- und Löschoperationen auf dem DHT-Ring. Es sei angemerkt, dass mindestens ein PSP-Knoten als ständig verfügbar angenommen wird, um für den Bootstrapping-Vorgang für eintretende Knoten zu dienen. Darüber hinaus um-

fasst der Kad-Block die Reed-Solomon-Funktionalität, um Sitzungsdaten aufzuteilen und Kodierungsstücke zu erzeugen und umgekehrt auch die Dekodierung von Datenstücken in Sitzungsdaten vorzunehmen. Datenstücke werden mittels TCP übertragen, wenn dies bei einer Lookup- oder Speicheroperation auf dem Kad-Ring notwendig wird. Wenn der PSP-Knoten eine Operation auf dem DHT-Ring ausführen soll, wird der Kad-Block angewiesen, Verbindung mit PSP-Knoten aus seiner Routing-Tabelle aufzunehmen. Eine vordefinierte Anzahl von PSP-Knoten, die dem Hashwert des jeweiligen Datenstücks am ähnlichsten ist, wird dabei gleichzeitig kontaktiert. Das Modul mit Steuerungsfunktionalität dient der Auslösung von Knotenaktivitäten, die sowohl interner Natur sein als auch extern durch einen Administrator vorgenommen werden kann. Nachfolgend erfolgt die Erläuterung der Gründe für die Auslösung von PSP-Knotenaktivitäten.

5.1.5.2. Auslösung von PSP-Knotenaktivitäten

PSP-Knotenaktivitäten können durch *interne* und *externe* Gründe ausgelöst werden. Der Knoten, der aus diesen Gründen aktiv wird, wird *auslösender Knoten* genannt.

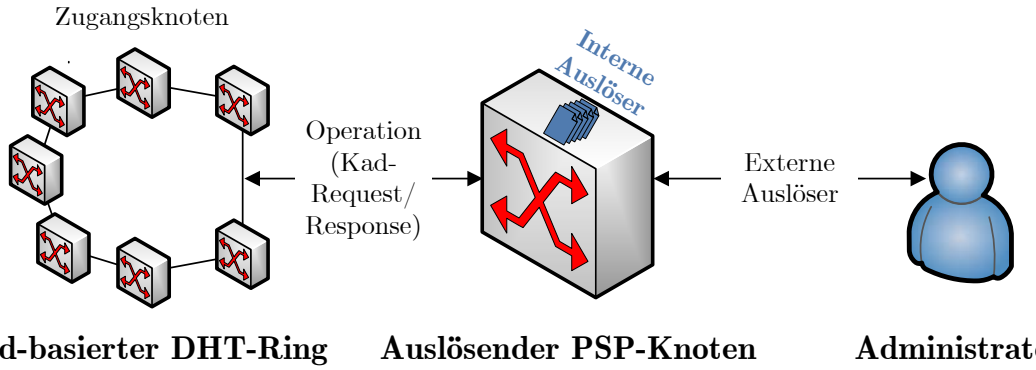


Abbildung 5.6.: Externe und interne Auslösung von Knotenaktivität und Interaktion mit anderen PSP-Knoten.

Interne Auslöser sind die Folge der Änderung oder des Verlustes von Sitzungsdaten auf einem Zugangsknoten. Externe Auslöser werden durch den Administrator eines Netzbetreibers gesteuert, der über eine Schnittstelle Zugriff auf den Zugangsknoten hat (siehe Abbildung 5.6). Über diese Schnittstelle können Lookups, Speicher- und Löschoperatio-

nen sowie Konfigurationsaufgaben angewiesen werden und die (nicht) erfolgreiche Umsetzung wird dem Administrator ebenso signalisiert. Die Anweisungen können entweder mittels Paketen über das Netzwerk gesendet oder durch ein spezielles Application Programming Interface (API) zwischen Administrator und PSP-Knoten gestellt werden. Die Schnittstelle erlaubt den flexiblen Zugriff auf die Sitzungsdaten eines Zugangsknotens und somit die Konfiguration durch eine administrative Instanz.

Über die Schnittstelle auf den Kad-basierten DHT-Ring werden Kad-Requests und Kad-Responses empfangen und gesendet, um Lookups, Speicher- und Löschoperationen auf dem DHT-Ring zu steuern. Pakete für das Bootstrapping und Instandhaltungsaufgaben werden ebenso hierüber ausgetauscht. Weiterhin erfolgt an dieser Stelle auch der Transfer von Datenstücken mittels TCP.

Nachstehend werden interne und externe Auslöser sowie die Interaktion mit anderen PSP-Knoten im Detail beschrieben.

Interne Auslöser auf einem PSP-Knoten: Interne Auslöser signalisieren, dass die Sitzungsdaten eines Zugangsknoten vom DHT-Ring gelesen oder gelöscht werden oder auf ihm gespeichert werden *müssen*.

Wenn ein Zugangsknoten nach einem Ausfall oder Reset neu gestartet wird, wird automatisch ein Lookup auf dem Kad-Ring ausgelöst. Dabei liest der neu gestartete Zugangsknoten seine eigenen Sitzungsdaten vom Ring, indem die entsprechenden Datenstücke von anderen Knoten bezogen werden. Dies ist nötig, da Sitzungsdaten im RAM eines Zugangsknotens gespeichert sind, nach einem Neustart verloren gehen und wiederhergestellt werden müssen.

Wenn ein Zugangsknoten Änderungen in seinem Sitzungsdatenbestand feststellt, löst er eine Löschoperation auf dem DHT-Ring aus. Dabei werden die entsprechenden Sitzungsdaten- und Kodierungsstücke der alten Sitzungsdaten gelöscht. Änderungen können sich durch abgelaufene Lease-Zeiten der IP-Adressen ergeben oder durch Kunden, die ihre IP-Adressen zurückgeben (DHCP Release, siehe auch Abschnitt A.2) sowie durch administrative Änderungen bei der Konfiguration von Sitzungsdaten.

Nachdem ein Zugangsknoten Änderungen in seinem Sitzungsdatenbestand bemerkt und die entsprechenden Datenstücke vom DHT-Ring gelöscht hat, werden neue Datenstücke aus den aktuellen Sitzungsdaten erstellt und auf dem DHT-Ring gespeichert.

Externe Auslöser durch einen Administrator: Externe Auslöser kommen nur von

einer administrativen Instanz des Netzbetreibers. Sie sind dafür gedacht, explizit Datenstücke anzufragen, zu speichern oder zu löschen, wenn die Sitzungsdaten eines Zugangsknotens modifiziert wurden oder wiederhergestellt werden *sollen*. Außerdem können die Parameter des Reed-Solomon-Codes konfiguriert werden, um die gewünschte Datenverfügbarkeit P_d sicherzustellen.

Wenn ein PSP-Knoten eine Suchanfrage von einem Administrator erhält, initiiert er einen Lookup auf dem DHT-Ring. Nach einem erfolgreichen Lookup wird eine Antwort zum Administrator geschickt, die die (nicht) erfolgreiche Erledigung der Suchanfragen anzeigt.

Wenn ein Knoten eine Aufforderung zum Speichern bekommt, teilt er seine Sitzungsdaten in Datenstücke auf und generiert Kodierungsstücke mit Hilfe des Reed-Solomon-Codes. Danach wird eine Speicheroperation auf dem DHT-Ring ausgelöst. Schlussendlich teilt der Knoten dem Administrator das Ergebnis der Operation mit.

Bei dem Erhalt einer Löschaufforderung wird eine Löschoperation auf dem DHT-Ring in Gang gesetzt und schließlich das Ergebnis der Operation mitgeteilt. Darüber hinaus kann ein Administrator die Löschung des Speichers mit den eigenen Sitzungsdaten anweisen. In diesem Fall werden die Datenstücke vom DHT-Ring *nicht* gelöscht. Die Löschung des Speichers mit den eigenen Sitzungsdaten kann notwendig werden, um Änderungen am Datensatz rückgängig zu machen und danach die ursprünglichen Sitzungsdaten vom DHT-Ring wiederherzustellen.

Interaktionen mit anderen PSP-Knoten:

Lookups auf dem DHT-Ring: Lookups werden durch Suchanfragen vom Administrator oder durch interne Auslöser initiiert. Während eines Lookups werden Knoten kontaktiert, deren Hashwerte denen der angefragten Datenstücke am ähnlichsten sind. Zuerst werden dazu Kontakte aus der Routing-Tabelle des auslösenden Knotens genommen. Im Laufe des Lookups wird durch die Antworten der kontaktierten Knoten Kenntnis über nähere Knoten erlangt. Schließlich werden Knoten mit Hashwerten gefunden und kontaktiert, die den Hashwerten der angefragten Datenstücke so ähnlich sind, dass sie innerhalb der Suchtoleranz liegen. Solch ein Knoten versucht, die Datenstücke in seinem Speicher zu finden. Hat er Erfolg, antwortet er dem auslösenden PSP-Knoten, der daraufhin sofort den Lookup abbricht. Wenn das angefragte Datenstück auf keinem der Knoten gefunden werden kann, beendet ein Time-out den Lookup. Da Reed-Solomon-Codes für die Gewährleistung hoher Datenverfügbarkeit P_d genutzt werden, liegt die Wahrscheinlichkeit

dafür aber nur bei 1-P_d. Somit sollten Time-outs, die einen Lookup terminieren, selten auftreten.

Speicheroperationen auf dem DHT-Ring: Speicheroperationen werden durch administrative Anfragen oder interne Auslöser hervorgerufen. Ebenso wie ein Lookup nimmt der auslösende Knoten dabei zunächst Verbindung mit Knoten aus seiner Routing-Tabelle auf, die dem Ziel-Hashwert am nächsten sind. Sobald ausreichend nahe Knoten kennengelernt wurden, werden diese angewiesen, die Datenstücke zu speichern. Die Speicheroperation wird beendet, sobald alle Datenstücke gespeichert wurden oder auf Grund eines Time-outs. Ein Time-out kann nur auftreten, wenn die Speicheroperation für eines der Datenstücke misslingt. In diesem Fall wird die Operation wiederholt, bis sie erfolgreich ist.

Löschoperationen auf dem DHT-Ring: Administrative Anfragen oder interne Auslöser können auch Löschoperationen nach sich ziehen, die prinzipiell wie Lookups oder Speicheroperationen ablaufen. Ausreichend nahe Knoten werden instruiert, ihre Datenstücke zu löschen. Die Löschoperation terminiert, nachdem alle Konten ihre Datenstücke entfernt haben oder durch einen Time-out. Ein Time-out tritt nur auf, wenn einer der Knoten mit zu löschenden Datenstücke aus dem Netzwerk ausgetreten ist.

Jede der Operationen trägt dazu bei, die Anzahl der Kontakte zu erhöhen, die der auslösende Knoten kennt. Dadurch wird das Kad-Netzwerk redundanter und robuster. Außerdem erhöht sich die Lookup-Performance potentiell, da einige Iterationsschritte auf dem Weg zu Ziel-Hashwert ausgelassen werden können, nachdem nähere Kontakte gelernt wurden.

5.1.5.3. Weitere Aspekte

Sitzungsdaten weisen einen Speicherbedarf in Höhe von einigen MByte pro Zugangsknoten auf. Daher ist die quantitative Speichermenge, die durch die Nutzung von Reed-Solomon-Codes verglichen mit DR gespart werden kann, für *eine gewünschte Datenverfügbarkeit* moderat. Allerdings gewährleisten Reed-Solomon-Codes umgekehrt eine wesentlich höhere Datenverfügbarkeit *für eine bestimmte verfügbare Speicherkapazität.*

Da sich durch den Bei- bzw. Austritt von Knoten bzw. durch Knotenausfall die Knoten-Zuständigkeit für Sitzungsdaten-Stücke ändern kann, besitzt ein Knoten unter Umständen noch Stücke, für die er nicht mehr verantwortlich ist. Daher werden diese bereits bei

der Daten-Verteilung mit einer TTL versehen, so dass die Gültigkeit über den Ablauf ihrer TTL automatisch erlischt. Der TTL-Wert ist dabei an die Aktualität der Sitzungsdaten anpassbar. Diese hängt von der Häufigkeit der Änderungen des Datenbestandes des DHCP-Servers ab. Jede Änderung führt zu einer Aktualisierung der Sitzungsdaten und damit einhergehend zu einer Neuverteilung von Sitzungsdaten-Stücken.

Durch den Einsatz des Kad-Netzwerks werden PSP einige Einschränkungen auferlegt. Da die Implementierungsvariante Kad auf dem Open-Source-Quellcode des eMule-Clients basiert, wird die Ausbeutung von Schwachstellen durch Hacker theoretisch vereinfacht. Allerdings sind Open-Source-basierte Apache-Web-Server einer der Marktführer und Linux vergrößert fortwährend seinen Marktanteil, was für die hohe Qualität von Open-Source-Quellcode spricht [NET09].

Die Kommunikation im Kad-Netzwerk führt zu zusätzlichem Datenverkehr im Netzwerk. Da Kad allerdings wie in Abschnitt 4.1 beschrieben über eine flexible Routing-Tabelle verfügt, ist der Traffic-Overhead durch die Instandhaltung sowie für Lookup-Vorgänge und die Teilnahme am Netz minimal.

5.1.6. Zusammenfassung

In diesem Abschnitt wurde die P2P-basierte Speicherplattform namens PSP vorgestellt, die die Nutzung von in seiner Verfügbar- und Wiederbeschreibbarkeit limitiertem Flash-Speicher für die Speicherung von Sitzungsdaten überflüssig macht. Stattdessen erfolgt der Gebrauch von verfügbarer RAM-Speicher- und Rechenkapazität von Zugangsknoten, da ein gewisser Anteil an Leerlaufzeit und freier Rechenleistung eines durchschnittlichen Zugangsknotens angenommen werden kann. Dadurch wird die Speicherung von Sitzungsdaten ohne zusätzliche Kosten für Netzwerke erreicht. Mit Hilfe des DHT-basierten P2P-Netzwerks Kad werden Zugangsknoten als logisches P2P-Overlay auf der existierenden Topologie verbunden und teilen dadurch ihre verfügbaren Speicher- und Rechenressourcen. Da hohe Skalierbarkeit und Ausfallsicherheit intrinsische Eigenschaften von Kad sind, weist PSP diese Merkmale ohne zusätzlichen Kapitalaufwand auf. Die Sitzungsdaten werden mit Hilfe von Reed-Solomon-Codes verschränkt gespeichert, so dass alle Daten auch bei einer hohen Anzahl von nicht verfügbaren Knoten wiederhergestellt werden können. Dabei ist wesentlich weniger Speicher-Overhead erforderlich, als die Nutzung einfacher Datenreplikation verursachen würde. Für die Gewährleistung einer Datenver-

fügbarkeit von 99,999 % mit einer Knotenverfügbarkeit von 90 % (dies umfasst auch einen funktionierenden Netzwerkzugriff der Knoten) beträgt der Speicher-Overhead so z. B nur 100 % im Gegensatz zu 500 % bei einfacher Datenreplikation.

Als Machbarkeitsbeweis wurde ein Software-Prototyp entwickelt, der einen Zugangsknoten mit PSP-Funktionalität abbildet. Zukünftig ist die Portierung dieses Prototypen für den Einsatz auf einem Xilinx-Evaluation Board geplant [Kar12b].

5.2. Das P2P-basierte verteilte Rechensystem *DuDE*

5.2.1. Motivation

Statistiken sind für ISP von grundlegender Bedeutung, um die Dienstgüte zu verbessern, Flaschenhälse in ihren Netzwerken und Attacken auf Netzwerkkomponenten zu erkennen. Die Verarbeitung von Log-Daten für die Berechnung dieser Statistiken stellt dabei für Netzbetreiber eine erhebliche Herausforderung dar. Zum einen sind existierende zentralisierte Rechensysteme nicht geeignet oder nicht dazu in der Lage, die ständig anwachsenden Log-Datenmengen einer stetig zunehmenden Anzahl von Internet-Nutzern zu verarbeiten (siehe Abbildung 5.7) [Exc11, Int11]. Zum anderen kann nur ein Standardsatz an *Kurzzeitstatistiken (KZS)* wie z. B. die CPU-Auslastung, die RAM-Auslastung oder die Anzahl verworfener und empfangener Pakete für einen *einzelnen* Zugangsknoten (Lokale Statistiken) erstellt werden. Überhaupt nicht unterstützt werden momentan:

- die Berechnung von *LZS*,
- Statistiken für *alle* Zugangsknoten (Globale Statistiken),
- die gleichzeitige Berechnung verschiedener Statistiken und
- die einfache Unterstützung neuer Statistiktypen.

Um diese Mängel zu beseitigen, wurde Distributed Computing System using a Decentralized P2P Environment (DuDE) entwickelt. Dazu werden Zugangsknoten mit Hilfe des selbst organisierenden DHT-basierten Kad-Netzwerks verbunden. Somit werden zusätzliche zentralisierte Systeme für die Statistikberechnung überflüssig. Zugangsknoten haben wie in Abbildung 5.1 ersichtlich eine bestimmte freie Speicher- und Rechenkapazität, die ohne zusätzliche Kosten genutzt werden kann. Zugangsknoten mit ausreichend Rechenleistung bringen sich bei der Statistikerstellung ein, wobei der Beitrag von den momentan

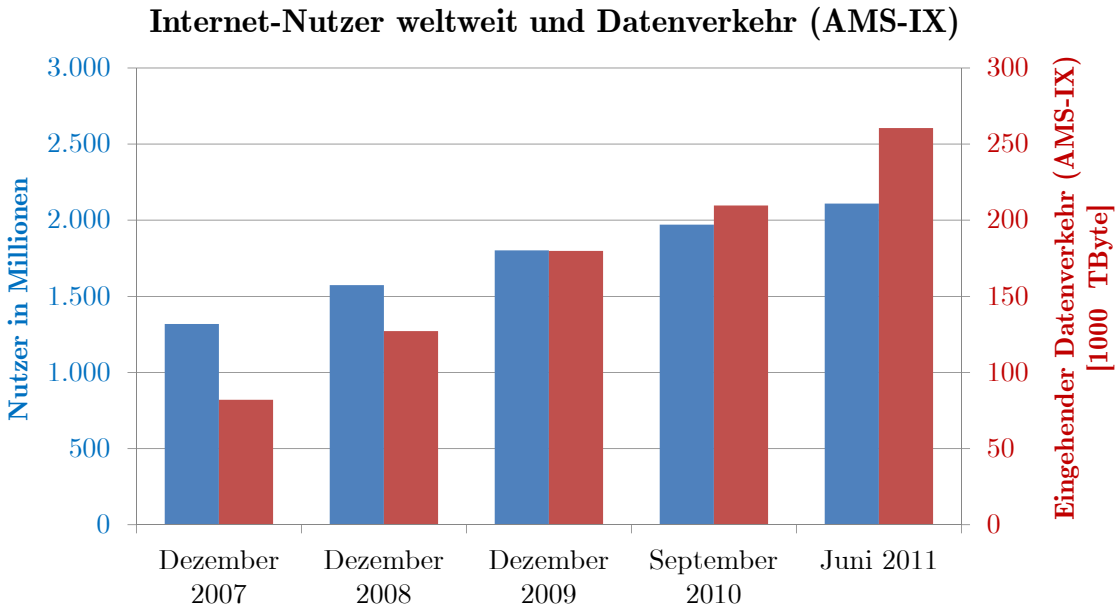


Abbildung 5.7.: Entwicklung der Internet-Nutzerzahlen weltweit und des anfallenden Datenverkehrs pro Monat am AMS-IX [Exc11, Int11].

verfügbaren Ressourcen abhängt. Diese Maßnahme verhindert die Überlastung eines einzelnen Knotens. Auf diese Weise skaliert das System im Gegensatz zu zentralisierten Systemen mit ansteigenden Log-Datenmengen. Hohe Ausfallsicherheit, die eine intrinsische Eigenschaft von P2P-Netzwerken ist [SW05], kombiniert mit ausgeklügelter redundanter Datenspeicherung durch Reed-Solomon-Codes (siehe auch Abschnitt 5.1.4.2) garantiert hohe Datenverfügbarkeit. DuDE unterstützt die Erstellung von KZS (einige Stunden) sowie LZS (mehrere Tage)–sowohl für einen *einzelnen* Knoten als auch für *alle* Zugangsknotens eines ISP. Auf Grund von DuDEs modularem Aufbau können zudem auch neue Statistikformate leicht integriert werden.

Nachfolgend werden folgende Hauptbeiträge kurz dargestellt [Sko10, SDA+11a]:

- Nach Vorüberlegungen und der Darlegung der Datenbasis wird die Anpassung des in Abschnitt 4.3.3 vorgestellten KaDis-Algorithmus zur globalen Daten-Ermittlung für das DuDE-System beschrieben.
- Der DuDE-Algorithmus für die Implementierung des verteilten Rechensystems wird

vorgestellt.

- Ein Backup-System und das Vorgehen bei der Gruppierung von Ressourcen, um die Auslastung von Knoten zu steuern, werden erläutert.
- Ergebnisse hinsichtlich Performance und Ressourcenverbrauch eines realen Testsystems werden präsentiert und mit einer zentralisierten Lösung verglichen.

5.2.2. Stand der Technik

In der Arbeit von Wu et. al. wird ein Grid-basierter Statistikdienst für den Einsatz in einer ausgedehnten dynamischen, heterogenen Umgebung vorgeschlagen [WD05] . Es handelt sich dabei nicht um einen DHT-basierten Dienst, sondern es kommt eine Semi-P2P-Struktur zum Einsatz, die eine gute Lastbalancierung erreicht und Flaschenhälse bezüglich der Performance vermeidet. Allerdings fehlt eine detaillierte Evaluierung der Performance und die Berechnung von Langzeitstatistiken wird nicht unterstützt. Darüber hinaus wird die Verfügbarkeit von Ressourcen nicht bei der Aufgabenverteilung berücksichtigt und verteilte Datenhaltung zur Erhöhung der Datenverfügbarkeit wird nicht beschrieben. DuDE widmet sich diesen ungelösten Problemstellungen bezüglich der Implementierung und stellt deutlich Performance-Vorteile gegenüber zentralisierten Rechensystemen heraus.

Der von Niu et. al. vorgeschlagene Ansatz konzentriert sich auf die Kompensation aus dem Netzwerk abgewanderter Peers, indem deren Daten verteilt im Netzwerk gespeichert werden [NL08]. Somit können die Daten abgewanderter Peers auch nach deren Ausscheiden von anderen Peers bezogen werden. Obwohl zufällige Netzwerkkodierung angewandt wird, um erhöhte Datenverfügbarkeit zu erreichen, muss allerdings immer noch ein zentraler Server die Daten von allen Peers einsammeln. Für den Flaschenhals, der durch den Server besteht, wird keine Abhilfe geschaffen, sondern das Problem wird nur gelindert. DuDE hingegen nutzt Reed-Solomon-Codes, um hohe Datenverfügbarkeit zu gewährleisten und vermeidet Flaschenhälse bezüglich der Berechnung durch die verteilte Statistikberechnung.

Computing Distributed architecture using the P2P paradigm (CoDiP2P) ist eine P2P-basierte Architektur für die verteilte Berechnung, die die gemeinsame Nutzung der Rechenressourcen von Nutzern ermöglicht [CBR+08]. Die Autoren beschreiben und bewer-

ten in ihrer Arbeit ein baumbasiertes P2P-Protokoll in seiner Funktionalität und hinsichtlich seiner Eigenschaften. Der Autor dieser Forschungsarbeit legt den Fokus hingegen auf die Evaluierung von DuDEs Performance für die Statistikberechnung, anstatt das eingesetzte P2P-Protokoll Kad zu rezensieren. Kad hat sich bereits in der Praxis bewährt und weist maßgebliche Vorteile gegenüber baumbasierten Protokollen auf; hauptsächlich hinsichtlich der Komplexität für die Restrukturierung und Instandhaltung des Netzwerks, da Kad auf einer flexiblen Routing-Tabelle basiert.

CompuP2P ist ein marktwirtschaftlichen Framework mit dem Ziel, basierend auf dem DHT-basierten Chord-Protokoll eine verteilte Nutzung der Rechenressourcen von Nutzern zu ermöglichen [GSS06]. Nutzer werden durch Anreize aus der Spieltheorie und Mikroökonomie dazu motiviert, ihre Ressourcen zu teilen. Auf das Chord-Protokoll werden weitere Overlays (Over-Overlays) wie z. B. für den Ressourcenhandel und die Diensteverwaltung aufgesetzt. Im Gegensatz zu diesem Ansatz nutzt DuDE eine erweiterte Version des Kad-Protokolls und verzichtet damit auf die Verwendung weiterer Over-Overlays und den damit verbundenen Overhead.

Das „JNGI“-Projekt ist ein Framework für großangelegte Berechnungen basierend auf dem hybriden P2P-Netzwerk Juxtapose (JXTA) [VNRS02]. JXTAs Konzept der Peer-Gruppen finden dabei Anwendung, wobei es drei Peer-Gruppen gibt (Monitor-, Task Dispatcher- und Worker-Gruppe). Es wird allerdings nicht klar, ob Peers auf Grund ihrer verfügbaren Ressourcen gruppiert werden, wie dies bei DuDE erfolgt. Anders als bei DuDE behält der Task Dispatcher weder den Überblick darüber, welche Worker welche Aufgabe bearbeiten noch über die Job Submitter, sondern Worker und der Job Submitter fragen ihrerseits den Task Dispatcher direkt. DuDEs Job Scheduler hingegen erzwingt eine schnelle Fertigstellung des Jobs, indem Time-outs für die Erledigung von Aufgaben durch die Worker definiert werden und fertiggestellte Statistiken sofort gesendet werden. Da JNGI auf JXTA basiert, kann das System darüber hinaus beträchtlich unter entstehenden Inkonsistenzen leiden, die seine Performance herabsetzen. Kad dagegen garantiert logarithmische Performance [SW05].

Die Arbeiten von [GKXS08] und [ZSZ03] befassen sich in erster Linie mit der Sammlung von Statistiken über Peers, indem ein Over-Overlay auf ein DHT-basiertes P2P-Netzwerk aufgesetzt wird. Dieses Over-Overlay ist dafür zuständig, Information wie Lasten und Kapazitäten von Peers zu sammeln, die in dem darunter liegenden DHT-Netzwerk organisiert sind. DuDEs Prozedur der Ressourcensammlung, um geeignete Peers für Re-

chenaufgaben zu finden, wird direkt im Kad-Netzwerk implementiert. Dadurch wird kein zusätzliches Overlay auf dem Kad-Netzwerk benötigt.

5.2.3. DuDE im Allgemeinen

P2P-Technologie in TZN: In Abbildung 5.1 wurde bereits ein typisches Zugangsnetzwerk mit einem durchschnittlichen Zugangsknoten dargestellt. Auf diesen Zugangsknoten wird das DuDE-System implementiert. Um hohe Rechenleistung zu erreichen, müssen Zugangsknoten kooperieren und deshalb verbunden werden. Damit geht ein Verzicht auf das Client-Server-Modell einher, da dieses Paradigma den Server als Flaschenhals aufweist. Das Ergebnis ist ein dezentralisiertes System, das P2P-Technologie mit seiner inhärenten Skalierbarkeit ausnutzt. Einem SPoF wie bei einem Client-Server-Modell wird dabei vorgebeugt. Eine erweiterte Version des Kad-Protokolls kommt auf Grund der in Abschnitt 4.1 beschriebenen Vorteile für die Verbindung der Zugangsknoten zum Einsatz.

Erweitertes Kad-Protokoll: Jeder Knoten ist im Kad-basierten DHT-Netzwerk neben seinen eigenen Log-Daten für einen Teil der Log-Daten anderer Knoten zuständig. DuDE unterstützt zudem zuverlässige Datenspeicherung durch Reed-Solomon-Codes im Netzwerk (siehe Abschnitt 5.1.4.2). Das Kad-Protokoll wurde insbesondere durch zusätzliche Suchobjekte erweitert und so modifiziert, dass eine verteilte Datenspeicherung und -sammlung möglich wird. Weiterhin wurde neue Pakete eingeführt, die die Aufgabenerfüllung von DuDE-Knoten ermöglichen und optimieren. Abbildung 5.8 zeigt das Netzwerk aus DuDE-Knoten, das als logischer Kad-basierter DHT-Ring oberhalb der realen Netzwerktopologie realisiert wurde. Die Datenbasis umfasst neben Knoten-eigenen Log-Daten und verteilten Log-Datenstücken ebenfalls Knoten-eigene LZS sowie LZS-Stücke anderer Knoten; eine detaillierte Erläuterung schließt sich unmittelbar im nachfolgenden Abschnitt an.

Verteiltes Rechnen: Ausgewählte Aspekte des verteilten Rechnens finden Anwendung, um überlastete Komponenten wie in zentralisierten Rechensystemen zu vermeiden. Zugangsknoten, die sich an der Statistikerstellung beteiligen, können abhängig von ihren verfügbaren Ressourcen *Job Scheduler* und/oder *Task Watcher* werden. Ein Job Scheduler ist ein sehr leistungsfähiger Knoten, der für die Verteilung von Teilen der Statistikerstellung (Tasks) an Knoten mit ausreichenden Ressourcen zuständig ist. Die Gesamtheit aller Tasks wird als *Job* bezeichnet. Jeder Task Watcher berechnet seinen Statistikanteil

und sendet ihn an den Job Scheduler nach der Fertigstellung zurück. Administratoren der Netzbetreiber können die Statistikerstellung an jedem Knoten in Auftrag geben und Statistiken abrufen und brauchen nicht direkt mit dem Job Scheduler verbunden zu sein.

Zusammengefasst stellen die Kombination von P2P-Technologie, ein erweitertes Kad-Protokoll mit zuverlässiger verteilter Datenspeicherung und Aspekten des verteilten Rechnens das neuartige System namens DuDE dar.

5.2.3.1. DuDEs Datenbasis: Log-Datensätze und Langzeitstatistiken

Log-Datensätze: Jeder Zugangsknoten hält Log-Daten in der Größenordnung von mehreren Hundert KByte in seinem RAM-Speicher, die z. B. die CPU-Auslastung des Knotens und die Anzahl der den Knoten passierenden Datenpakete umfassen. Der Speicher für die Log-Daten ist in Form eines Ringpuffers organisiert, der Daten über den Zeitraum einiger Stunden speichern kann. Um hohe Datenverfügbarkeit zu gewährleisten, wird wie bereits in Abschnitt 5.1.4.2 beschrieben der Reed-Solomon-Code eingesetzt. Ein Element namens Data Spreader (DS) teilt die Daten auf die erläuterte Art und Weise in z. B. 100 KByte große Stücke auf und erzeugt die Kodierungsstücke. Anschließend erfolgt entsprechend der Hashwerte die Verteilung der Datenstücke an Knoten im Kad-Netzwerk. Der DS dient ebenfalls der Gewährleistung der Aktualität der Log-Daten und verteilt in festen Intervallen die Log-Daten neu.

LZS: In dem Ringpuffer können wie erwähnt nur Log-Daten über den Zeitraum einiger Stunden gespeichert werden, die der Berechnung von KZS dienen. Um auch die Erfassung von LZS zu ermöglichen, müssen Knoten periodisch LZS erzeugen und diese in einem extra Speicher ablegen. LZS sind Statistiken über einen sehr umfangreichen Zeitraum für bestimmte Größen wie z. B. die CPU-Auslastung und die RAM-Auslastung. Dadurch können Netzwerkparameter über mehrere Tage beobachtet werden, um so eine aussagekräftigere Analyse des Netzwerkverhaltens vorzunehmen, als dies mit KZS von nur einigen Stunden möglich wäre. Für welche Messgröße jeweils LZS während der Laufzeit eines Zugangsknotens erzeugt werden, kann durch einen Administrator festgelegt werden.

Für die LZS-Erzeugung ist ein Element namens Periodic Accumulator (PA) vorgesehen. Der PA nutzt die aktuellen Log-Daten aus dem Ringpuffer, um fortwährend auf iterative Weise die LZS zu erzeugen. Neue Daten werden dabei an die bereits bestehenden LZS

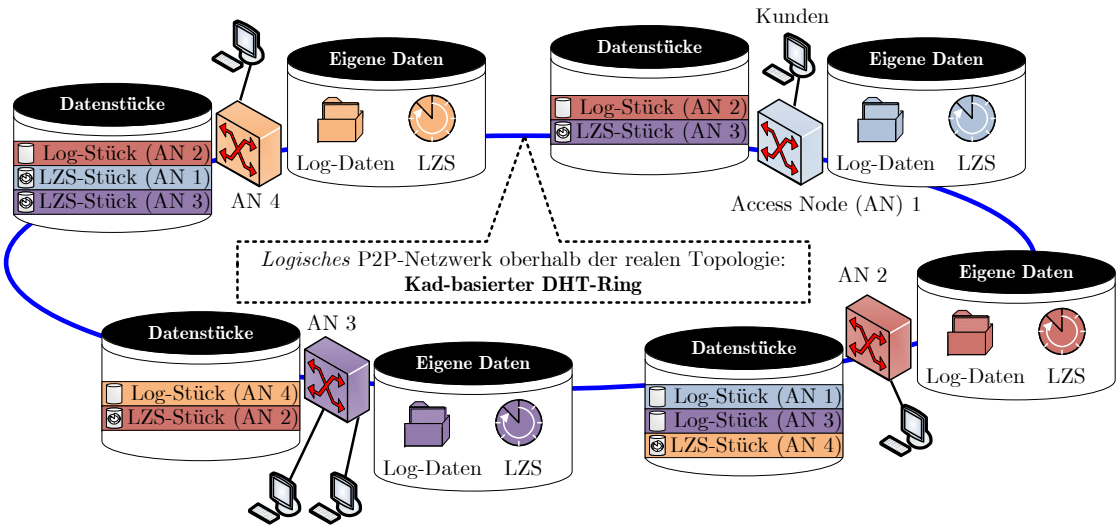


Abbildung 5.8.: Kad-basierter DHT-Ring für die verteilte Statistikberechnung und verteilte Datenhaltung. Jeder Knoten hält seine eigenen Log-Daten, erzeugt fortwährend daraus LZS und verteilt beide Datensätze.

angehängt. LZS werden ebenfalls mit dem Reed-Solomon-Code kodiert, in Stücke geteilt und schließlich im Kad-Netzwerk verteilt, um hohe Datenverfügbarkeit zu gewährleisten.

Aktualität der verteilten Datensätze: Der DS und PA richten sich bei der Koordinierung der Datenverteilung nach eingestellten Zeitparametern. Wann die Daten spätestens erneut verteilt werden müssen, hängt von der Größe des Ringpuffers ab. Angenommen, die Daten brauchen N Schritte, um durch den Ringpuffer zu laufen, so müssen sie vor Schritt N verteilt werden. Auch die Generierung der LZS muss vor Schritt N erfolgen, da ansonsten alte, aber noch nicht für die LZS verwendete Log-Daten im Ringpuffer überschrieben werden.

Eine permanente Aktualisierung führt zu einem hohen Ressourcenverbrauch für die Kodierung mittels Reed-Solomon-Codes und außerdem zu einer hohen Auslastung des Kommunikationskanals für die Datenverteilung. Zu große Zeitschlitze bewirken die Erzeugung von Statistiken aus zu alten Daten. Folglich stellt die Wahl der Länge des Zeitschlitzes einen Kompromiss zwischen akzeptablem Ressourcenverbrauch und Aktualität der Daten dar.

5.2.3.2. Globale Datenermittlung

Da DuDE keine zentrale Instanz besitzt, sind die Anzahl der Peers und damit auch die im Netzwerk vorhandenen Daten unbekannt. Allerdings werden für die globale Statistikberechnung die Daten aller Knoten benötigt. Um dies zu erreichen, enthalten die Bezeichnungen der Datensätze die eindeutige Knoten-ID (z. B. *Knoten i*, siehe auch Abschnitt 4.3.3). Somit gibt es pro Knoten i einen Log-Datensatz *Knoten i.log* sowie seine LZS über bestimmte Größen wie z. B. Prozessorauslastung (*Knoten i_ cpu.lzs*) oder RAM-Auslastung (*Knoten i_ ram.lzs*), die allesamt im Kad-Netzwerk verteilt sind. Der KaDis-Algorithmus (siehe Abbildung 5.9) findet alle im Netz verteilten Datensätze mit einer hohen Wahrscheinlichkeit, die wie in Abschnitt 4.3.3 vorgestellt vom Abbruchwert T abhängt.

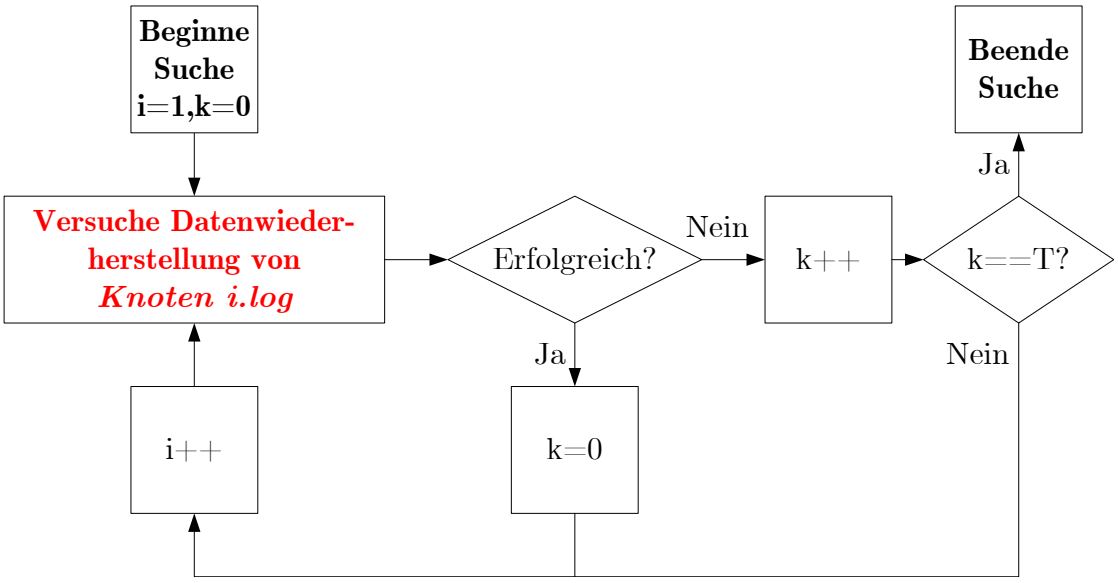


Abbildung 5.9.: KaDis-Algorithmus für die globale Log-Datenermittlung. Bei der Suche nach LZS wird entsprechend nach *Knoten i_ cpu.lzs*, *Knoten i_ ram.lzs* etc. gesucht.

Zunächst wird die Datenwiederherstellung des Log-Datensatzes *Knoten i.log* bzw. der LZS (*Knoten i_ cpu.lzs*, *Knoten i_ ram.lzs* etc.) mit i=0 versucht. Ist dies erfolgreich, bleibt k auf Null und i wird um Eins erhöht. Bei einem Misserfolg wird k um Eins erhöht.

Solange k den Schwellwert T nicht erreicht, wird i weiterhin um Eins inkrementiert. Sobald k gleich T ist, bricht der Algorithmus ab. Der Algorithmus ist dabei in der Lage, auch Daten von Knoten zu finden, die das Kad-Netzwerk bereits verlassen haben, wenn diese noch verteilt im Netzwerk vorliegen.

Tabelle 5.2 zeigt zur Veranschaulichung des Algorithmus ein Beispielszenario für die Wiederherstellung von Log-Daten. T hat hierbei den Wert Drei. Die Log-Datensätze von Knoten1, Knoten2 und Knoten5 seien im Netzwerk verteilt gespeichert. Nach zwei erfolglosen Versuchen für *Knoten3.log* und *Knoten4.log* erreicht k den Wert Zwei, ist aber immer noch kleiner als T=3. Danach wird *Knoten5.log* gefunden und K wird auf Null zurückgesetzt. Der KaDis-Algorithmus sucht anschließend weiter und bricht nach der erfolglosen Suche nach *Knoten8.log* ab. Nun wird angenommen, dass keine weiteren verteilten Datensätze im Netz vorhanden sind.

i	Erfolgreich?	k	k==T?	Datensatz
1	Ja	0	Nein	Knoten1.log
2	Ja	0	Nein	Knoten2.log
3	Nein	1	Nein	Knoten3.log
4	Nein	2	Nein	Knoten4.log
5	Ja	0	Nein	Knoten5.log
6	Nein	1	Nein	Knoten6.log
7	Nein	2	Nein	Knoten7.log
8	Nein	3	Ja	Knoten8.log
Abbruch des Algorithmus				

Tabelle 5.2.: Beispielszenario für die Funktionsweise des für DuDE angepassten KaDis-Algorithmus bei der Suche nach Log-Daten.

5.2.3.3. Der DuDE-Algorithmus

Der DuDE-Algorithmus basiert auf Ian Fosters Konzept für den Entwurf paralleler Algorithmen und wird ebenso in Phasen gegliedert [Fos95]. Dabei wurde der DuDE-Algorithmus von vier auf sieben Phasen erweitert, die in Abbildung 5.10 (Phase 1-4) und Abbildung 5.11 (Phase 5-7) dargestellt sind. Die Erweiterung ist notwendig, da DuDE zunächst

den Job Scheduler und Task Watcher bestimmen muss, wodurch mehr Verwaltungsaufwand anfällt. Allerdings funktioniert DuDE dadurch ohne zentrale Koordinierungsinstanz.

Als Veranschaulichung des Algorithmus wird ein Beispielszenario mit vier DuDE-Knoten mit freien Rechen- und Speicherressourcen angenommen. Tabelle 5.3 zeigt die freien Kapazitäten der vier Knoten im Verhältnis zu ihrer Maximalleistung.

Knoten	Freie Rechenkapazität [%]	Freie Speicherkapazität [%]
1	90	90
2	70	70
3	50	50
4	40	40

Tabelle 5.3.: Beispielszenario für die Veranschaulichung des DuDE-Algorithmus mit vier Knoten. Freie Rechen- und Speicherkapazitäten sind aufgeführt.

Phase 1: Während der ersten Phase sucht Knoten 1 auf Geheiß des an ihn angeschlossenen Administrators nach Knoten mit ausreichenden Ressourcen, um Job Scheduler zu werden. Jeder Knoten kann Job Scheduler werden und es muss nicht unbedingt derjenige Knoten sein, mit dem der Administrator direkt verbunden ist. Die kontaktierten Knoten müssen innerhalb einer definierten Zeit antworten. Ist diese Zeit verstrichen, werden ihre Antworten nicht mehr akzeptiert und sie kehren in den Ausgangszustand zurück.

Phase 2: Knoten 1 hat die meisten freien Ressourcen und wird somit Job Scheduler. Neben der reinen Addition der *relativ* verfügbaren Rechen- und Speicherressourcen sind auch viele andere Ansätze denkbar; Knoten könnten z. B. eine bestimmte *absolute* freie Rechen- und Speicherkapazität aufweisen müssen, um Job Scheduler zu werden. Alle anderen Knoten außer Knoten 1 werden freigegeben. Freigegebene Knoten sind nicht mehr blockiert und können weitere Anfragen entgegen nehmen.

Phase 3: Der Job Scheduler sucht nun ebenfalls nach Ressourcen. Wie in Phase 1 müssen Knoten innerhalb einer bestimmten Zeit antworten. Der Ansatz der erneuten Ressourcensuche wurde gewählt, da sich freie Kapazitäten der Knoten in der Zwischenzeit geändert haben können.

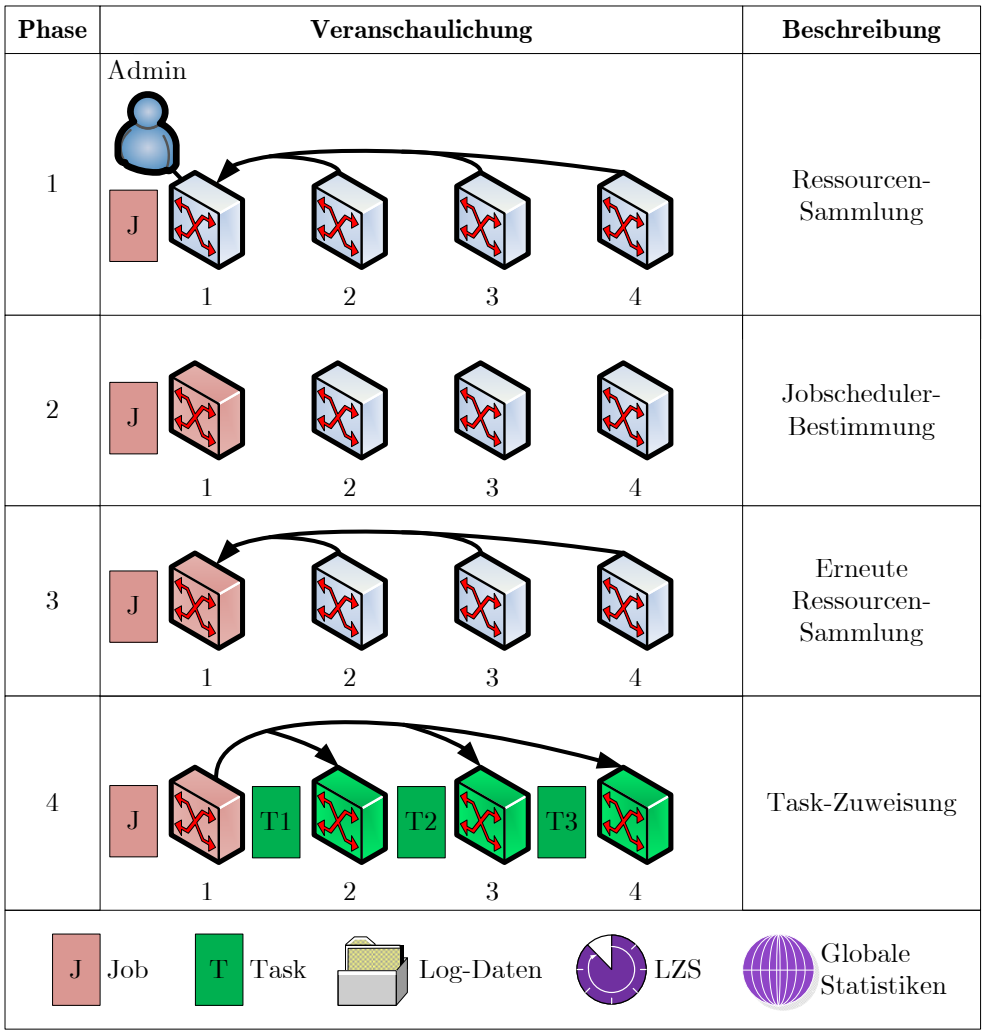


Abbildung 5.10.: DuDE-Algorithmus für die verteilte Statistikberechnung (Phase 1-4).

Phase 4: Phase 4 dient der Aufteilung eines Jobs in einzelne Tasks. Diese Tasks werden Knoten mit ausreichend freien Ressourcen zugewiesen. Im Beispiel besteht der Job aus drei einzelnen Tasks: Es sollen für *alle* Knoten folgende Statistiken (Globale Statistiken) berechnet werden:

- KZS über die Prozessorauslastung (T1, CPU-Auslastung) aus den Log-Daten aller vier Knoten,
- LZS über die Speicherauslastung (T2, MEM-Auslastung) aus den einzelnen LZS

aller vier Knoten und

- KZS über die Anzahl der Paketverwürfe (T3, #Paketverwürfe) aus den Log-Daten aller vier Knoten.

Ebenso könnte jede Statistik nur für jeweils einen einzelnen Knoten erzeugt werden, der frei bestimmbar ist (Lokale Statistiken). Knoten 2 übernimmt im gegebenen Fall T1, Knoten 3 wird mit der Erledigung von T2 beauftragt und Knoten 4 erhält T3 zur Verarbeitung. Bei der Task-Zuweisung ist eine Priorisierung möglich, so dass rechenintensive Tasks nur an Knoten mit ausreichenden Ressourcen gegeben werden. Jeder Knoten kann dabei selbst entscheiden, ob er seine Ressourcen zur Verfügung stellen will. Ebenso ist es möglich, Ressourcenuntergrenzen für Tasks zu definieren, die ein Knoten mindestens erfüllen muss, um eine Task verarbeiten zu können. Neben der allgemeinen Aufgabe ist auch der spezielle Quellcode zur Erledigung der Aufgabe an den jeweiligen Knoten übertragbar. Knoten, deren Ressourcen nicht benötigt werden, werden freigegeben, so dass sie anschließend für andere Aufgaben zur Verfügung stehen.

Phase 5: Während der Phase 5 sammelt jeder Task Watcher mit Hilfe des in Abschnitt 5.2.3.2 erläuterten KaDis-Algorithmus Datenstücke aus dem Netz, um daraus den jeweiligen Datensatz für die Statistikberechnung zu erhalten.

Phase 6: Anschließend werden in Phase 6 die Statistiken aus den gesammelten Datensätzen erzeugt.

Phase 7: Die letzte Phase dient der Rücksendung der generierten Statistiken an den Knoten (Knoten 1), der den Job entgegengenommen hat. Zunächst werden dazu alle Statistiken an den Job Scheduler gesendet, der in diesem Fall auch der Knoten ist, der auf Anweisung des Administrator den Job entgegengenommen hat. Anschließend können die erstellten Statistiken angezeigt und ausgewertet werden.

Der vorgestellte DuDE-Algorithmus ermöglicht die strukturierte Berechnung von Statistiken und kann als Grundlage für die Implementierung verteilter Rechensysteme auf der Grundlage von P2P-Technologie verwendet werden. Darüber hinaus reduziert der Algorithmus die Kommunikation zwischen Knoten auf das Notwendigste, so dass eine reibungslose Abarbeitung des Jobs garantiert wird.

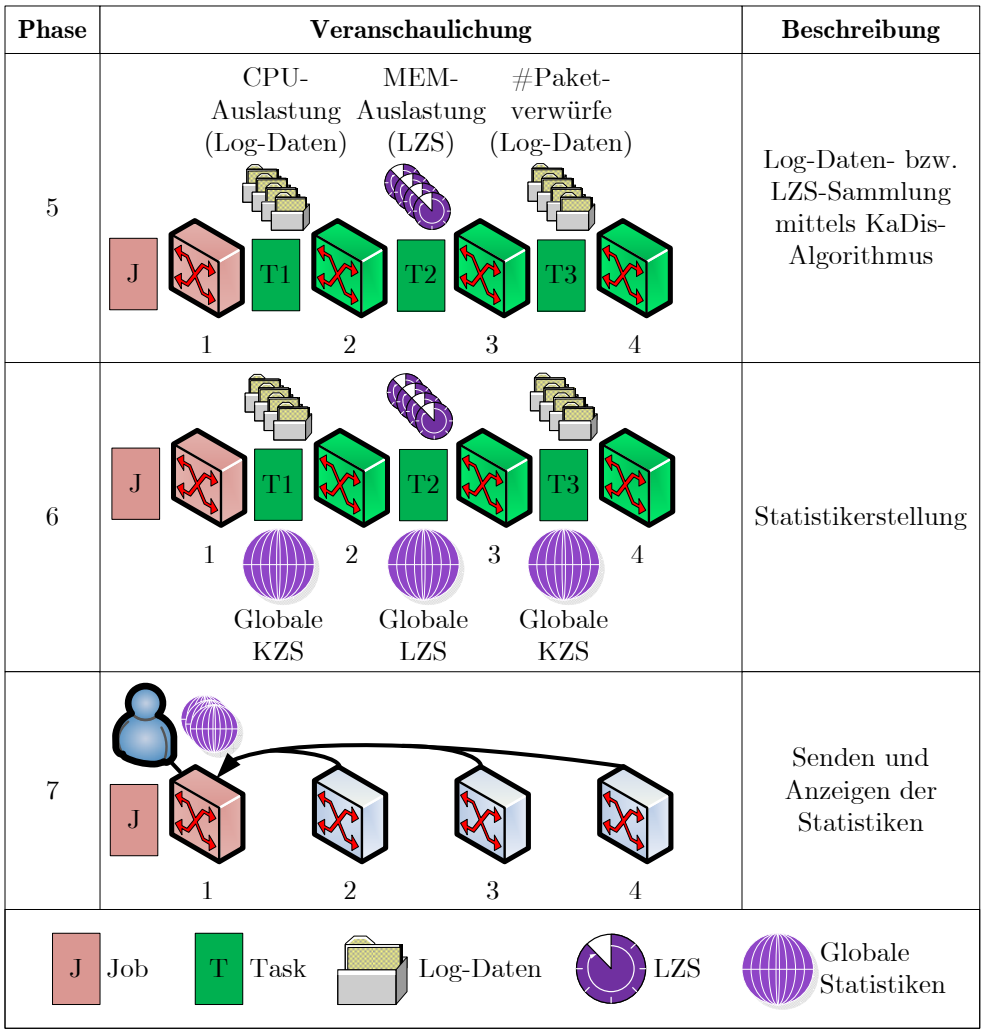


Abbildung 5.11.: DuDE-Algorithmus für die verteilte Statistikberechnung (Phase 5-7).

5.2.3.4. Backup-System

Um eine höhere Zuverlässigkeit von DuDE zu erreichen, wurde ein Backup-System integriert. Dieses Backup-System ist optional und dient der Kompensation von ausgefallenen Task Watchern und dem Job Scheduler.

Gleichzeitig mit der Zuweisung eines Tasks an einen Task Watcher wird ein zweiter Knoten angewiesen, diesen Task ebenfalls zu bearbeiten (Alternativ kann auch mehr als ein Backup-Knoten bestimmt werden). Beide Knoten nutzen dabei die gleiche Datenbasis

für die Erstellung der Statistiken. Wenn der Task Watcher ausfällt, kann der Backup-Knoten anstelle dessen die Ergebnisse sofort zurücksenden, da er parallel mitgearbeitet hat.

Der Job Scheduler erwartet die fertig gestellte Statistik nach einer bestimmten Zeit. Ist sie dann nicht verfügbar, werden sie von dem Backup-Knoten angefragt und der ausgefallene Knoten bekommt den Befehl, in den Ausgangszustand zu gehen, wenn er noch erreichbar ist.

Um einen ausgefallenen Job Scheduler auszugleichen, können ein oder mehrere weitere Knoten vom Administrator zeitgleich mit dem Job Scheduler zum Ersatz-Job Scheduler bestimmt werden. Diese erhalten nach der Fertigstellung ebenfalls sämtliche Statistiken. Kann der Administrator von einem ausgefallenen Job Scheduler keine Statistiken abrufen, fragt er den/die Ersatz-Job Scheduler. Ist kein Job Scheduler-Ausfall zu beklagen und können somit die Statistiken vom ursprünglichen Job Scheduler bezogen werden, so werden nach dem Abruf der Statistiken die Ersatz-Job Scheduler wieder freigegeben.

5.2.3.5. Ressourcen-Gruppierung

Die Nutzung eindeutiger Zugangsknoten-IDs (siehe Abschnitt 5.2.3.2) erlaubt die Gruppierung der Zugangsknoten. Es ist vorstellbar, dass Zugangsknoten-IDs geographische, Besitz- oder Kapazitätsinformationen in Form von Zeichenketten enthalten. Somit wird die Zuweisung bestimmter Tasks zu einer speziellen Gruppe von Knoten möglich, indem Zugangsknoten mit bestimmten vordefinierten Zeichenketten ausgewählt werden. Zusätzlich erlaubt diese Gruppierung einem Administrator, die Tasks flexibel aufzuteilen, ohne dabei Kenntnis über das Netzwerk zu besitzen.

5.2.4. Software-Realisierung von DuDE

Die Systemarchitektur eines DuDE-Knotens ist in Abbildung 5.12 ersichtlich. Die Hauptkomponenten sind ein Speicher mit eigenen Log-Daten und LZS eines Knotens (1), ein Speicher mit Log-Daten- und LZS-Stücken (2), die Routing-Tabelle mit Kontaktinformationen über andere Knoten (3), das erweiterte Kad-Protokoll (4) mit integrierter Reed-Solomon-Funktionalität für die verteilte Datenspeicherung sowie ein Block mit Steuerungsfunktionalität, der externe Anweisungen von einem Administrator entgegennimmt

und die Statistikberechnung startet (5). Außerdem gibt es einen Speicher, um die gewünschten Statistiken vorzuhalten (6): Dies können sowohl KZS als auch LZS für einzelne Knoten (Lokale Statistiken) oder alle Knoten sein (Globale Statistiken). Die Funktionalität zur Statistikberechnung (7) umfasst die bereits erläuterten Elemente PA und DS.

Jeder DuDE-Knoten speichert bestimmte Datenstücke, für die er auf Grund seines Hashwertes zuständig ist. Dabei wird jedem Knoten ein Hashwert zugewiesen, der aus seiner Zugangsknoten-ID mittels MD5-Hash-Funktion erzeugt wird [Riv92]. Die MD5-Hash-Funktion kann dabei Kollisionen erzeugen, so dass manche Knoten gleiche Hashwerte bekommen können. Dies gilt insbesondere, wenn nur einige Bytes des Hashwertes für die Hash-ID des Knotens genutzt werden. Das Kad-Netzwerk kann allerdings damit umgehen, so dass Knoten mit gleichen Hashwerten zwar einander nicht sehen, aber von anderen Knoten gefunden werden können.

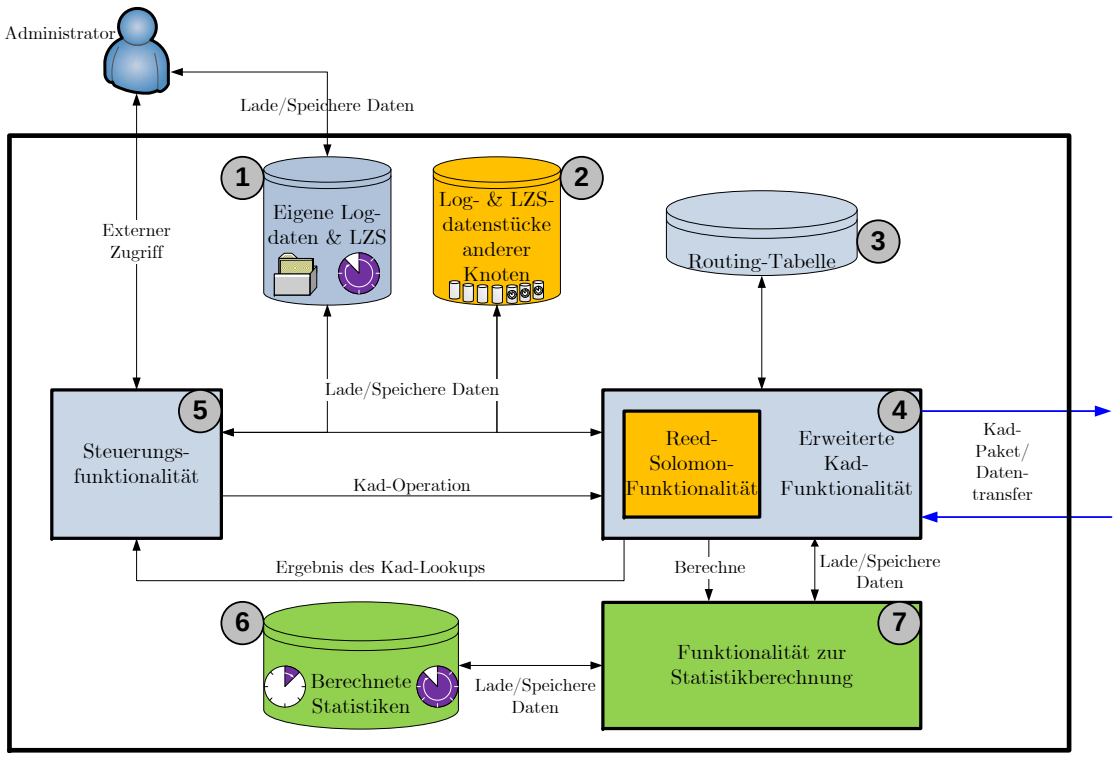


Abbildung 5.12.: Systemarchitektur eines DuDE-Knotens.

Um hohe Datenverfügbarkeit zu erreichen, werden sowohl Log-Daten als auch LZS durch die Reed-Solomon-Funktionalität in n Datenstücke mit redundantem Anteil geteilt (siehe Abschnitt 5.1.4.2) und in Speicher (2) abgelegt. Wenn Statistiken berechnet werden sollen, werden dazu die entsprechenden verteilten Datenstücke eingesammelt, um aus diesen die Datenbasis für die Statistikgenerierung zu gewinnen.

5.2.5. Testszenario

Um DuDEs Performance und die benötigen Systemressourcen eines Knotens zu ermitteln, wurde ein Testsystem aus sieben identischen PCs aufgebaut. Jeder PC besitzt einen 1,5 GHz Pentium 4-Prozessor und 512 MByte Arbeitsspeicher. Als Betriebssystem wird Windows XP SP3 genutzt. Diese Ausstattung ist mit der Konfiguration eines durchschnittlichen Zugangsknotens (siehe Abbildung 5.1) vergleichbar.

Mit Hilfe dieses Testszenarios soll die Beschleunigung bei der Statistikberechnung durch DuDE gegenüber einem heutzutage eingesetzten zentralisierten System (aus einem einzelnen Zugangsknoten) gezeigt werden. Das Testszenario legt den Fokus auf eine realitätsnahe Demonstration und Validierung von DuDEs Funktionalität. Umfangreiche Untersuchungen der Skalierbarkeit und des auftretenden Datenverkehrs durch die Kommunikation im Kad-Netzwerk wurden nicht vorgenommen. Es existieren einige Parameter, die einen Einfluss auf die Messergebnisse haben wie z. B. die Größe der Datensätze. Das Testszenario beschränkt sich bei der Datenbasis für die Statistikerstellung auf Log-Datensätze, um die Funktionalität zu veranschaulichen und sieht von der Auswertung von LZS ab.

Tabelle 5.4 gibt einen Überblick über die einstellbaren Parameter des Testsystems. Die Größe der Log-Datensätze ist auf 100 KByte eingestellt und der Abbruchwert des KaDis-Algorithmus beträgt Eins. Der Wert Eins wurde gewählt, da die Knoten und deren Log-Datensätze im Netzwerk bekannt sind und keine unbekannten Knoten gefunden werden müssen. Pro Testdurchlauf wurden die Anzahl der Log-Datensätze, die Rechenlast oder die Anzahl der Tasks variiert.

Anzahl der Log-Datensätze: Je nach Anzahl der Log-Datensätze verteilen entsprechend viele Knoten ihre Log-Datensätze im Kad-Netzwerk. Dadurch kann getestet werden, welchen Einfluss die Anzahl auf die Rechenzeit hat. Mit einer steigenden Anzahl im Kad-Netzwerk müssen die Knoten mehr Daten für die Statistikberechnung verarbeiten,

Parameter	Wert	Beschreibung
Dateigröße	100 KByte	Größe eines Log-Datensatzes
T	1	Abbruchwert des KaDis-Algorithmus
Log-Datensätze	Variabel	Anzahl der verfügbaren Log-Datensätze im Kad-Netz
Rechenlast	Variabel	Anzahl der Berechnungen von Π in Tausend
Tasks	Variabel	Anzahl der Tasks pro Job

Tabelle 5.4.: Einstellbare Parameter des Testsystems

wodurch eine höhere Rechenzeit zu erwarten ist. Zur Statistikberechnung sammelt jeder Knoten die Stücke aller vorhandenen Log-Datensätze ein, stellt die Log-Datensätze daraus wieder her und durchsucht diese nach den gewünschten Werten. Die Log-Datensätze enthalten dabei lediglich Beispieldaten und dienen nur dazu, die Funktionalität der Statistikberechnung zu testen.

Rechenlast: Die Statistikerzeugung entspricht im Testszenario dem Parsen eines Log-Datensatzes nach den gewünschten Daten. Für die Statistik relevante Werte werden in eine neue Datei geschrieben, die die Statistik repräsentiert. Um dabei eine beliebig hohe Rechenlast zu erreichen, wurde die Bailey-Borwein-Plouffe (BBP)-Formel beispielhaft implementiert [BBP97]. Sobald ein relevanter Wert gefunden wurde, wird mit Hilfe der BBP-Formel die Konstante Π bis auf neun Stellen genau berechnet (siehe Formel 5.3). Die Rechenlast gibt die Anzahl der Berechnung von Π in Tausend an. Eine Rechenlast von 10 bedeutet also 10.000 Berechnungen von Π bis zur neunten Stelle hinter dem Komma.

$$\pi = \sum_{k=0}^{9} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right) \tag{5.3}$$

Anzahl der Tasks pro Job: Da für das Testszenario sieben PCs gewählt wurden, wird die Anzahl der Tasks pro Job auf sechs beschränkt. Jeder Knoten kann somit eine Task übernehmen und ein Knoten wird Job Scheduler. Der Job Scheduler selbst übernimmt keine Task, wobei aber auch das möglich wäre. Folgende KZS werden exemplarisch durch die Tasks mittels Parsen aus den Log-Datensätzen aller Knoten gewonnen:

- Task 1: CPU-Auslastung

- Task 2: RAM-Auslastung
- Task 3: Anzahl aller Pakete, die den Zugangsknoten passiert haben
- Task 4: Anzahl verworfener Pakete
- Task 5: Anzahl aller UDP-Pakete
- Task 6: Anzahl aller TCP-Pakete

5.2.6. Auswertungsergebnisse

Folgende Messwerte wurden während jedes Testdurchlaufs aufgenommen, um die Vorteile von DuDE gegenüber einem heutzutage eingesetzten zentralisierten System (Einzelner Zugangsknoten (AN)) zu zeigen:

- Prozessor- und Speicherauslastung der Knoten: Ressourcen-Auslastung, die durch die Belastung mit den zu erledigenden Aufgaben auftritt (Job oder Tasks).
- Anzahl der Pakete in der Warteschlange eines Knotens: Alle Knoten kommunizieren über UDP miteinander (mit Ausnahme des reinen Datenaustausches, der über TCP erfolgt). Eingehende UDP-Pakete werden der Reihe nach abgearbeitet, so dass ggf. einige Pakete auf ihre Verarbeitung müssen. Diese werden in einer Warteschlange gespeichert.
- Zeit für die Fertigstellung des Jobs: Gemessene Zeit, bis alle Tasks des in Auftrag gegebenen Jobs verarbeitet wurden und die Statistiken bereit zur Anzeige und Auswertung sind.

Sowohl DuDE als auch das zentralisierte System bestehend aus einem Zugangsknoten berechnen dazu die gleichen Statistiken aus den gleichen Log-Datensätzen. DuDEs Performance-Gewinn spiegelt sich in der Beschleunigung bei der Statistikberechnung wider. Die Auswertung der gemessenen Werte erlaubt es, Schlussfolgerungen über das Verhalten eines Systems mit realen Nutzlasten zu ziehen.

Prozessorauslastung: Knoten, die als Task Watcher Statistiken berechnen, weisen eine Prozessorauslastung von 100 % während der Berechnung auf, da jeder PC nur mit einem Prozessorkern ausgestattet ist. Dies gilt sowohl für alle Knoten des DuDE-Systems als auch für den einzelnen Knoten. Der DuDE-Prozess wurde allerdings auf allen Knoten mit niedriger Priorität gestartet, so dass trotz 100 %-iger Prozessorauslastung wichtige

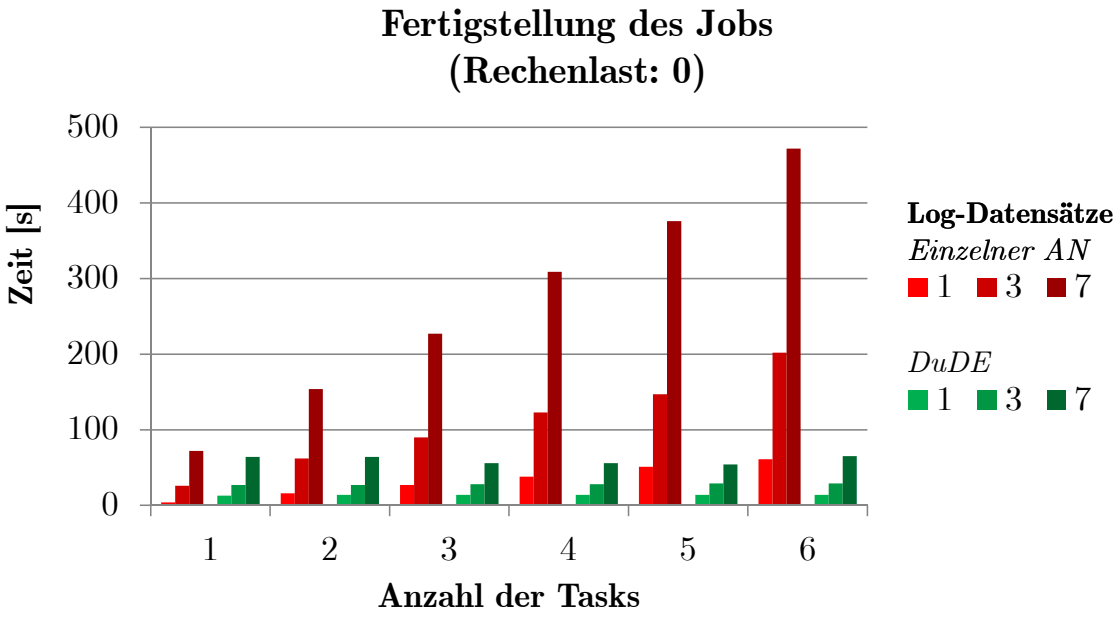


Abbildung 5.13.: Zeit für die Erledigung des Jobs für eine ansteigende Anzahl von Tasks und einer unterschiedlichen Anzahl von Log-Datensätzen (Rechenlast = 0).

Systemprozesse Vorrang haben. Der Job Scheduler im DuDE-System weist eine maximale Prozessorauslastung von 22 % auf. Diese niedrige Auslastung ergibt sich aus der Tatsache, dass der Job Scheduler nur Verwaltungsaufgaben übernimmt und selbst keine Statistik berechnet.

Anzahl der Pakete in der Warteschlange eines Knotens: DuDE-Knoten speichern zur Kommunikation verwendete UDP-Pakete in einer Warteschlange, bevor sie verarbeitet werden. Hoher Datenverkehr kann demzufolge zu einer erhöhten Anzahl von Paketen in der Schlange führen. Allerdings überschreitet die Anzahl der Pakete auf Grund des niedrigen Kommunikationsaufwands und der sofortigen Verarbeitung der Pakete im Testszenario niemals den Wert 1.

Zeit für die Fertigstellung des Jobs: Abbildung 5.13 zeigt, dass ein einzelner Knoten zur Erledigung des Jobs eine linear zunehmende Zeit für eine ansteigende Anzahl von Tasks benötigt. Die Anzahl der Log-Datensätze ist dabei gleich der Anzahl der Kno-

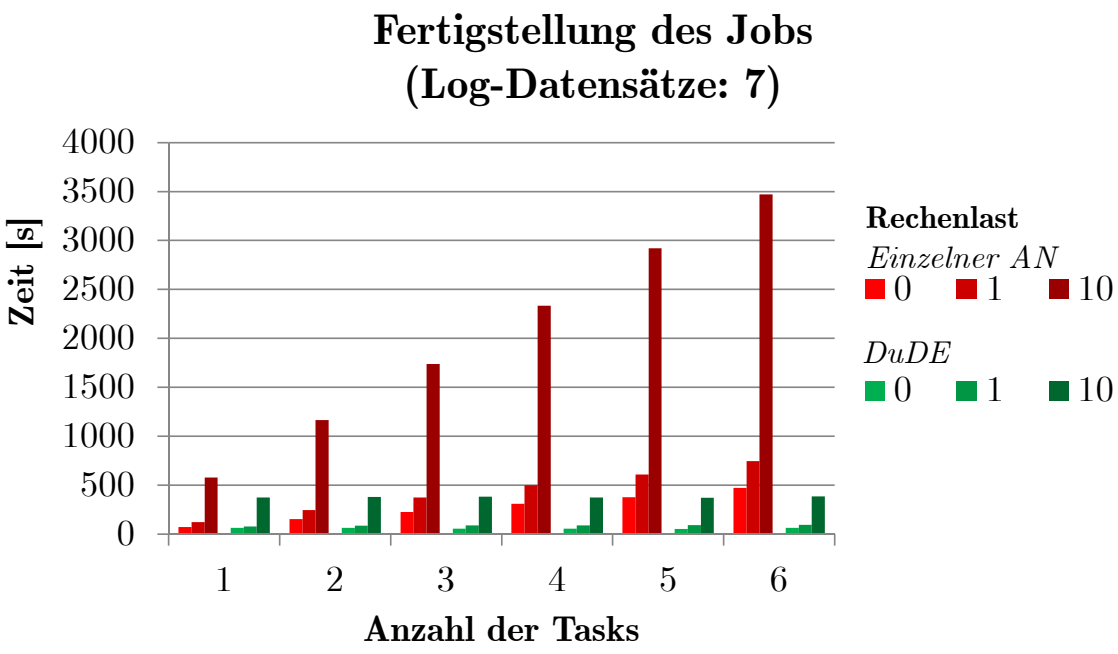


Abbildung 5.14.: Zeit für die Erledigung des Jobs für eine ansteigende Anzahl von Tasks und unterschiedlichen Rechenlasten (Log-Datensätze = 7).

ten im Testszenario. DuDEs Berechnungszeit bleibt hingegen erwartungsgemäß nahezu konstant. Das zentralisierte System aus einem Zugangsknoten ist einzig in einem Ausnahmefall schneller (4 statt 13 s), wenn nur ein Log-Datensatz im Netz existiert. Dies ist dadurch bedingt, dass in dem Fall DuDEs Zeit zur Bestimmung des Job Schedulers und der Task Watcher besonders stark ins Gewicht fällt.

Weiterhin hängt die Berechnungszeit auch von der Anzahl der Log-Datensätze ab. Mit einer steigenden Anzahl von Log-Datensätzen steigt auch die notwendige Zeit zur Statistikgenerierung an, da die Auswertung größerer Datenmengen mehr Berechnungen erfordert. Durch die Verteilung der Statistikberechnung ist DuDE auch diesbezüglich dem zentralisierten System klar überlegen und weist für 7 Log-Datensätze und 6 Tasks eine Verbesserung um 86 % auf.

Das gleiche Verhalten für die benötigte Rechenzeit zeigt sich mit steigender Rechenlast durch Nutzung der BBP-Formel für eine zunehmende Anzahl von Tasks. Wie in Abbildung 5.14 ersichtlich ist, braucht DuDE bei höherer Rechenlast mehr Zeit für die

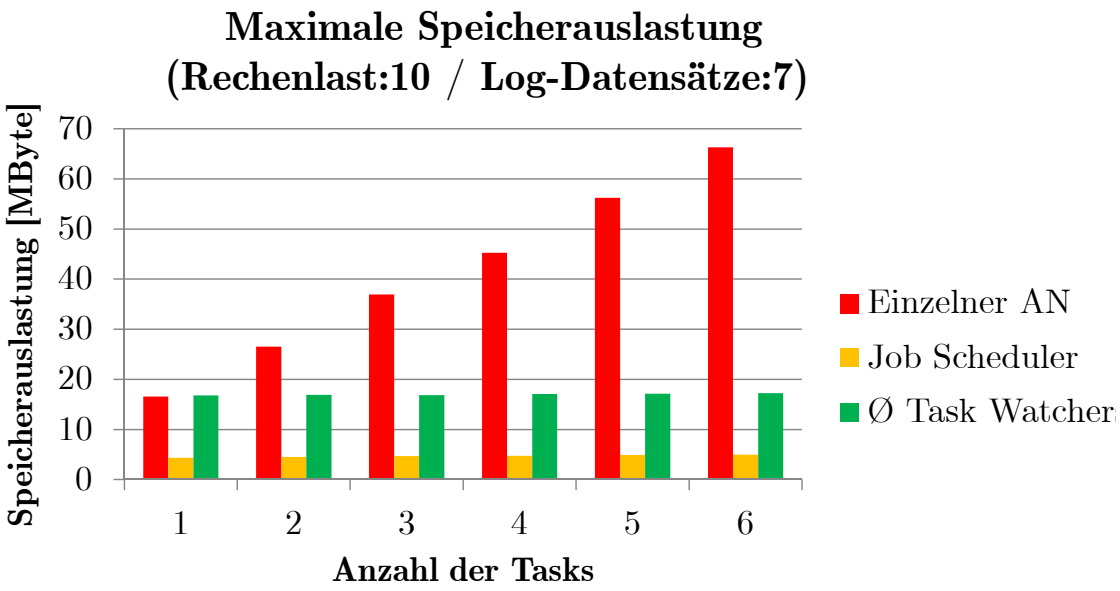


Abbildung 5.15.: Maximale Speicherauslastung der Knoten für eine ansteigende Anzahl von Tasks (Rechenlast = 10, Log-Datensätze = 7).

Statistikberechnung, da jeder Task Watcher Π mehrere tausendmal berechnen muss. Die Zeit für eine höhere Task-Anzahl ist konstant und zeigt für 7 Log-Datensätze und einer Rechenlast von 10 eine Verbesserung um 89 % gegenüber dem zentralisierten System. Solange es ausreichend viele Task Watcher gibt, ist die Berechnungszeit folglich unabhängig von der Anzahl der Tasks eines Jobs.

Maximale Speicherauslastung: Die Anzahl der Tasks eines Jobs ist entscheidend für die maximale Speicherauslastung. Abbildung 5.15 verdeutlicht, dass die maximale Speicherauslastung des Job Schedulers konstant ist, da er nur eine Verwaltungsfunktion erfüllt. Die konstante durchschnittliche maximale Speicherauslastung der Task Watcher ist dadurch bedingt, dass jedem Task Watcher genau ein Task des Jobs zugewiesen wird. Der einzelne Knoten des zentralisierten Systems muss sämtliche Daten der Log-Datensätze allein verarbeiten, wodurch die maximale Speicherauslastung steigt und bei 6 Tasks bereits 284 % über der von DuDE liegt. Allerdings zeigt Abbildung 5.16, dass die Speicherauslastung nicht von der Rechenlast abhängt. Eine höhere Rechenlast führt nicht zu höheren Datenmengen während der Statistikberechnung, da die BBP-Formel einer datenunabhän-

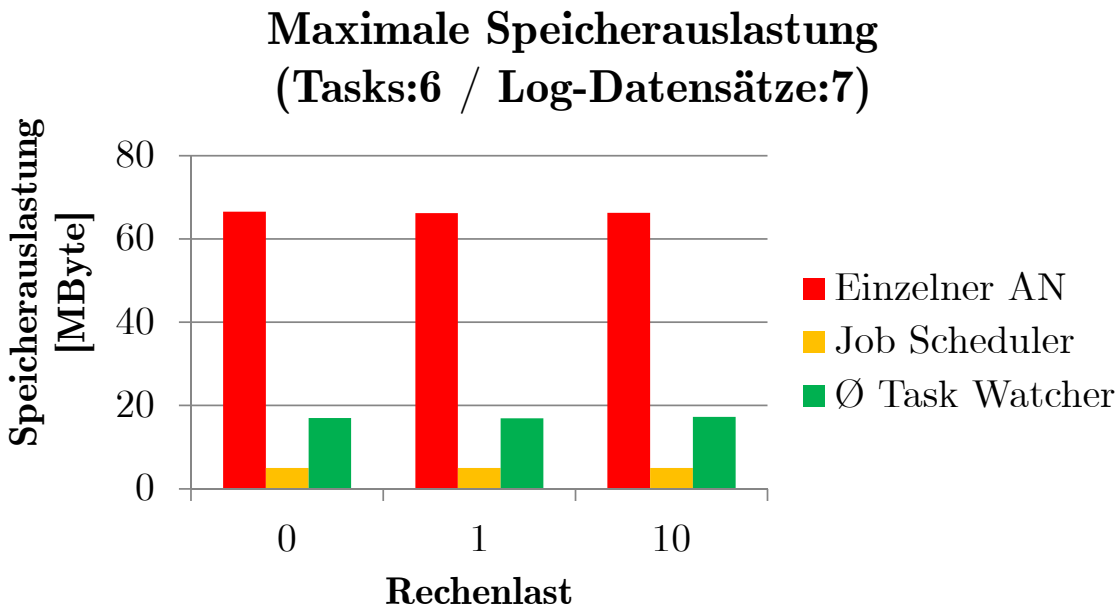


Abbildung 5.16.: Maximale Speicherauslastung der Knoten für eine ansteigende Rechenlast (Tasks = 6, Log-Datensätze = 7).

gigen mathematischen Berechnung dient. Höhere zu verarbeitende Datenmengen durch eine zunehmende Anzahl von Log-Datensätze verursachen stattdessen eine steigende maximale Speicherauslastung. Abbildung 5.17 belegt, dass die maximale Speicherauslastung der Task Watcher und des zentralisierten Systems mit einer zunehmenden Anzahl von Log-Datensätzen ansteigt. Die durchschnittliche maximale Speicherauslastung der Task Watcher ist allerdings sehr viel geringer als die des einzelnen Knotens im zentralisierten System und ergibt für 7 Log-Datensätze eine Verbesserung von 74 %. Die Speicherauslastung der Task Watcher steigt zudem sehr viel langsamer an. Der Job Scheduler weist einen konstanten Speicherverbrauch auf.

Die erzielten Ergebnisse stellen zunächst eine erfolgreiche Validierung von DuDE dar und heben darüber hinaus deutlich die Vorteile des Systems gegenüber einem zentralisierten Rechensystems hervor, das heutzutage zum Einsatz kommt. Erwartungsgemäß skaliert DuDE mit einer zunehmenden Anzahl von DuDE-Knoten mit nahezu perfekter linearer Beschleunigung bei der Statistikberechnung, wenn die Task-Anzahl erhöht wird. Der Kommunikations-Overhead durch die Kommunikation der Knoten wird auf ein Mi-

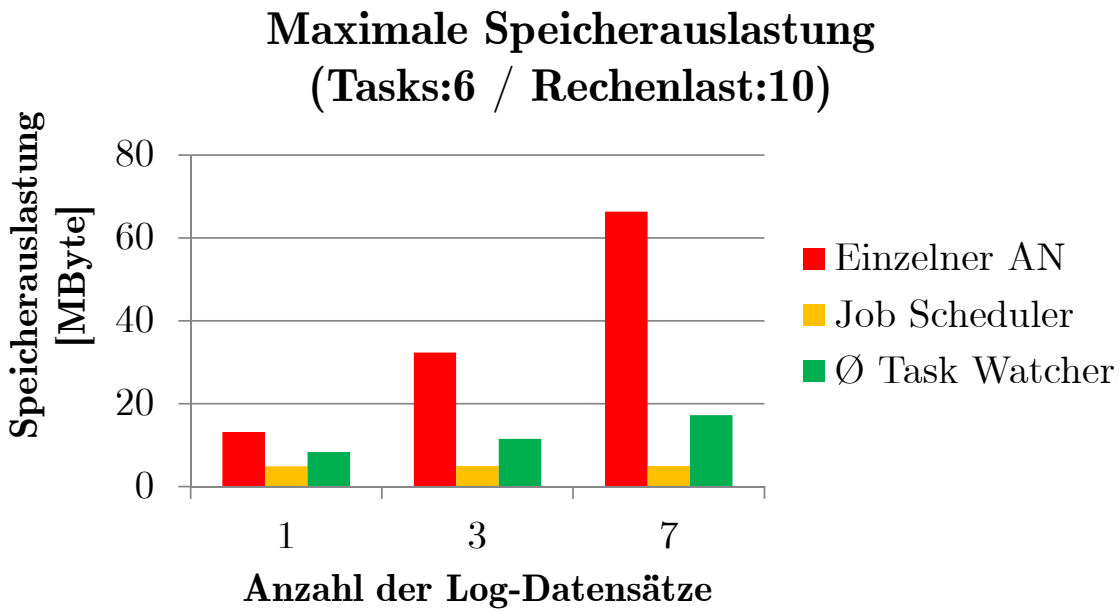


Abbildung 5.17.: Maximale Speicherauslastung der Knoten für eine ansteigende Anzahl von Log-Datensätzen (Rechenlast = 10).

nimum beschränkt, um hohe Performance zu gewährleisten und hatte keinen negativen Auswirkungen auf die Ergebnisse.

5.2.7. Zusammenfassung

Das P2P-basierte System für die verteilte Statistikberechnung namens DuDE wurde in diesem Abschnitt vorgestellt. DuDE nutzt den KaDis-Algorithmus zur Ermittlung der Datensätze aller Peers im Netzwerk ohne zentrale Instanz. Reed-Solomon-Codes sorgen dabei für hohe Datenverfügbarkeit bei der verteilten Datenspeicherung. Selbst bei einem Ausfall eines Knotens sind dessen Daten noch im Kad-Netz verfügbar, wenn der ausgefallene Knoten sie zumindest einmal vor dem Ausfall verteilt hat. Jeder Knoten hält zudem seine eigenen Daten im RAM vor, so dass er auch bei dem Ausfall seiner Netzwerkschnittstelle Zugriff darauf hat. Der Arbeitsablauf bei der Statistikerstellung, bestehend aus sieben Schritten, wurde vorgestellt und eignet sich als möglicher Ansatz für P2P-basierte verteilte Rechensysteme. Die Priorisierung von bestimmten Statistik-

berechnungen ist ebenso möglich wie die gezielte Nutzung von Rechenressourcen z. B. entsprechend der geographischen Lage oder Leistungskapazität.

Für eine realitätsnahe Demonstration und Validierung von DuDEs Funktionalität wurde ein Testsystem vorgestellt und mit einer heutzutage eingesetzten zentralisierten Plattform verglichen. Die Ergebnisse zeigen, dass DuDEs Rechenleistung mit einer steigenden Anzahl von Knoten skaliert und die Statistikberechnung nahezu perfekt linear beschleunigt.

Durch die Nutzung bereits vorhandener Ressourcen von Zugangsknoten für die Realisierung von DuDE stellt dieses System eine kostengünstige Möglichkeit für Netzbetreiber dar, ihre heutigen Engpässe bei der Statistikerstellung zu beheben. Durch umfassendere globale Langzeitstatistiken, die DuDE überhaupt erst ermöglicht, können so Schwachstellen im Netzwerk besser identifiziert und behoben werden. Infolge dessen kann eine höhere Zuverlässigkeit und Dienstgüte der Netze gewährleistet werden.

Zukünftige Arbeiten befassen sich mit der Erforschung weiterer Anwendungsfälle für das verteilte Rechensystem DuDE sowie weiteren umfassenden Simulationen und Analysen des Netzwerks.

5.3. Das P2P-basierte DNS *P-DONAS*

Während der Anfänge des Internet wurden Domainnamen und ihre IP-Adressen in einer zentral verwalteten Textdatei gespeichert. In den 80er Jahren war die Anzahl von Domainnamen bereits zu groß, um sie noch effizient in dieser Form verwalten zu können. Dadurch entwickelte sich das traditionelle DNS. Kurz zusammengefasst nimmt das DNS eine bidirektionale Zuweisung von IP-Adressen zu Domainnamen (oft auch Bestandteil von so-genannten Uniform Ressource Locators (URLs) [Moc87]) vor. Dadurch kann eine IP-Adresse für einen Domainnamen angefragt werden und umgekehrt.

5.3.1. Motivation

Ähnlich der Entwicklung in den 80er Jahren reicht das traditionelle DNS nicht mehr für die ständig wachsende Anzahl von Domainnamen und Nutzern im Internet aus. Dabei ist die limitierte Skalierbarkeit des DNS die größte Herausforderung [RS04].

ISP müssen heutzutage regionale DNS-Server-Farmen bereitstellen, um eine hohe Ausfallsicherheit und angemessene ökonomische Lastverteilung zu erreichen. Was ISP an Skalierbarkeit einsparen, fehlt bei der Leistungsfähigkeit und Ausfallsicherheit des DNS. Für eine hohe Skalierbarkeit müssen ISP in gesonderte Ausrüstung investieren, d.h. zusätzliche DNS-Server-Farmen. Dabei entstehen neben hohen Anschaffungskosten hohe Kosten für das Betreiben, die Verwaltung sowie den Energieverbrauch dieser zusätzlichen Infrastruktur, die 24 Stunden am Tag und 7 Tage in der Woche laufen muss.

Demzufolge ist die Hauptmotivation für das entwickelte P2P-basierte DNS, genannt P-DONAS, die zusätzlichen Kosten des ISP zu sparen, die durch zusätzliche DNS-Server-Farmen entstehen [DSA+bm]. Für die Realisierung von P-DONAS werden Zugangsknoten des Zugangsnetzes eines ISP (siehe Abbildung 5.1) mit Hilfe des DHT-basierten P2P-Netzwerks Kad organisiert. Ein durchschnittlicher Zugangsknoten verfügt, wie bereits dargelegt, über eine bestimmte verfügbare Speicher- und Rechenkapazität, die ohne zusätzliche Kosten genutzt werden kann.

Dabei bilden die Zugangsknoten einen logischen DHT-Ring auf der existierenden Netzwerktopologie. Da hohe Skalierbarkeit und Ausfallsicherheit intrinsische Eigenschaften von DHT-basierten P2P-Netzwerken sind, verfügt auch P-DONAS ohne zusätzliche Kosten darüber [SW05]. Jeder Zugangsknoten agiert als traditioneller DNS-Nameserver, aber speichert nur einen Teil des gesamten DNS-Datenbestandes. DNS-Anfragen an einen Zugangsknoten mit P-DONAS werden mittels P2P-Lookups beantwortet, wobei vollständige Kompatibilität und Verbindung zum traditionellen DNS gewährleistet ist.

5.3.2. Stand der Technik

Es gibt bereits einige Bestrebungen, ein P2P-basiertes DNS zu entwerfen. In [CMM02] wird das Distributed DNS (DDNS) vorgeschlagen. Die Motivation hierbei lag in der Lösung administrativer Probleme sowie unzureichender Lastverteilung und niedriger Ausfallsicherheit des traditionellen DNS. Nutzer dürfen mittels DNSSEC verifizierte DNS-Einträge in das DDNS einbringen, so dass eine Unabhängigkeit von durch ISP konfigurierten DNS-Nameservern erreicht werden soll. DDNS nutzt DHash, ein DHT-basiertes P2P-Netzwerk, das auf Chord aufsetzt. DNS-Einträge werden pseudo-zufällig repliziert und Caching von Einträgen wird entlang des Lookup-Pfads vorgenommen. Es gibt eine Reihe von Problemen mit DDNS wie z. B. die im Vergleich zum traditionellen DNS sehr

viel höhere Antwortlatenz. Außerdem müssen Endknoten, also Nutzer, DNS-Daten für Domainnamen bereitstellen, ohne dass sie dafür einen Anreiz erhalten. Der Verlust administrativer Kontrolle und die Abhängigkeit von der Mitarbeit von Endknoten erschweren den Einsatz eines solchen Systems. Im Gegensatz zu dieser Idee ist P-DONAS ein vertrauenswürdiger und zuverlässiger DNS-Dienst, der durch den ISP zur Verfügung gestellt wird. Außerdem lässt DDNS eine Menge Fragen bezüglich der Implementierung offen, die P-DONAS beantwortet; so z. B. die Möglichkeit zum Löschen von RRs in DDNS.

In [RS04] wird das sogenannte Cooperative Domain Name System (CoDoNS) vorgestellt. Die Motivation der Autoren war die Lösung der Probleme des traditionellen DNS bezüglich Skalierbarkeit, Konsistenz sowie Performance. CoDoNS ist ebenfalls DHT-basiert und nutzt Pastry als P2P-Netzwerk. Als Knoten bei CoDoNS sollen zusätzliche Server dienen. Replikation wird auf der Grundlage der Popularität von DNS-Einträgen vorgenommen. Um niedrige Latenz zu erreichen, wurde ein proaktives Caching-System entwickelt. CoDoNS ist kompatibel mit dem traditionellen DNS. Ergebnisse dieser Forschungsarbeit sind der Einsatz bisher auf weltweit 700 Knoten (PlanetLab) und eine sehr niedrige durchschnittliche Antwortlatenz (niedriger als traditionelles DNS). Allerdings ist CoDoNS abhängig von global verteilten Servern, die in Form eines globalen gemeinsam genutzten DNS-Caches zur Verfügung gestellt werden müssen. Im Gegensatz dazu ist P-DONAS so konzipiert und realisiert, dass DNS-Funktionalität mit Hilfe verfügbarer Ressourcen von Zugangsknoten anstatt durch zusätzliche Server bereit gestellt werden kann. P-DONAS basiert zudem im Unterschied zu CoDoNS auf dem Kad-Netzwerk anstatt auf Pastry und hebt sich diesbezüglich durch die in Abschnitt 4.1 beschriebenen Vorteile ab.

In [PMTZ06] wird eine Vergleichsstudie über DHT-basierte DNS-Alternativen vorgestellt, wobei als DHT-Protokoll Chord gewählt wurde. Die Arbeit stützt sich auf Analysen und Simulationen, um die Performance und Verfügbarkeit von traditionellem DNS und Chord-basiertem DNS zu vergleichen. Das Chord-basierte DNS wird als anfälliger gegenüber zufälligen Knotenausfällen dargestellt und zeigt nur höhere Widerstandsfähigkeit gegen gezielte Attacken als traditionelles DNS. Die gleiche Performance wie traditionelles DNS lässt sich nur durch zusätzliche Maßnahmen wie proaktives Caching erreichen. Als finale Schlussfolgerung gelangt die Arbeit zu dem Ergebnis, dass es dennoch keinen eindeutigen Gewinner gibt und beide Ansätze Vor- und Nachteile besitzen. Als Haupthindernis für den Einsatz eines DHT-basierten DNS wird angeführt, dass vielmehr das traditionelle und bewährte System verbessert werden sollte, bevor ein komplett neues noch

zu optimierendes Design dieses ersetzt. Der Autor der vorliegenden Forschungsarbeit vertritt demgegenüber die Auffassung, dass die *praktische* Umsetzung eines Kad-basierten DNS durchaus mit dem traditionellen DNS konkurrieren kann und vor allem Kosten für zentralisierte Lösungen durch Nutzung *vorhandener* Ressourcen spart. Die Evaluierung in einem realistischen Testszenario gewährt zudem detaillierte Einblicke in die Funktionsweise und Leistung eines DHT-basierten DNS und stellt einen weiteren Schritt zur Ersetzung bzw. Ergänzung des traditionellen DNS dar. Der Autor ist davon überzeugt, dass auch die Wahl von Kad als DHT-Protokoll auf Grund seiner in Abschnitt 4.1 beschriebenen Vorteile eine wichtige Voraussetzung für eine effiziente Realisierung eines P2P-basierten DNS darstellt. Neueste Entwicklungen zeigen zudem die hohe Relevanz und Notwendigkeit des in dieser Forschungsarbeit untersuchten Ansatzes. Vor dem Hintergrund der Vorbeugung einer eventuellen Zensur des DNS sowie hoher Widerstandsfähigkeit gegen DoS-Angriffe hat der Mitbegründer von Pirate Bay, Peter Sunde, ein Projekt zur Entwicklung eines P2P-basierten DNS gestartet [wir10].

In der Arbeit von Song et al. [SK11] wird ein hybrides P2P-basiertes DNS (HDNS) vorgestellt, das zu einem Teil aus der traditionellen hierarchischen DNS-Baumstruktur (die sogenannte Internal Zone) und zum anderen Teil aus einer flachen DHT-Struktur (die sogenannte Public Zone) besteht. Simulationen weisen für das System die hohe Robustheit eines DHT-Netzwerks gegen gezielte Attacken sowie die hohe Lookup-Effizienz des traditionelles DNS aus, wobei die gleiche Effizienz jedoch nicht vollständig erreicht wird. Der Fokus liegt auf der Darstellung des Kompromisses bezüglich der Aufteilung in Internal und Public Zone. Mit steigendem Anteil der Public Zone steigt die Robustheit des Systems, während die Performance abnimmt. P-DONAS legt das Hauptaugenmerk auf die Entwicklung und *vollständige Umsetzung* eines P2P-basiertes DNS als potentiell kompletter Ersatz für das traditionelle DNS durch die Verbindung verteilter verfügbarer Ressourcen. Es wird gezeigt, dass P-DONAS hinsichtlich seiner Performance sowie seines Ressourcenverbrauchs einen geeigneten Ersatz des traditionellen DNS für den praktischen Einsatz darstellt.

5.3.3. Herausforderungen des traditionellen DNS

Wie bereits erwähnt, ist die begrenzte Skalierbarkeit die größte Herausforderung des traditionellen DNS, der sich ISP stellen müssen. Andere Herausforderungen wie die Inkonsi-

stenz von DNS-Einträgen, die durch das TTL-getriebene Caching-System hervorgerufen wird, sind nicht Bestandteil dieser Arbeit. Der Vollständigkeit halber wird nachfolgend dennoch kurz auf Inkonsistenz, Ursachen für niedrige DNS-Performance sowie Lame Delegations (siehe Abschnitt 5.3.3.4) eingegangen.

5.3.3.1. Begrenzte Skalierbarkeit

Ein Aspekt der Skalierbarkeit ist die Ausfallsicherheit im Falle von Fehlern und Attacken. Wie in [RS04] gezeigt wird, hat die niedrige Ausfallsicherheit ihre Wurzeln in Delegation und physikalischen Bottlenecks sowie Implementierungsfehlern. Tatsächlich ist das traditionelle DNS extrem empfindlich gegenüber DoS-Attacken und Netzwerkfehlern.

Niedrige Ausfallsicherheit: Bezüglich der Ausfallsicherheit weist das ursprüngliche DNS einige Flaschenhälse (Bottlenecks) auf. Diese Bottlenecks lassen sich für Distributed Denial of Service (DDoS)-Attacken missbrauchen. In einer Stichprobe über 593160 Domainnamen und 164089 Nameserver wurden Delegation und physikalische Bottlenecks ermittelt [RS04].

Ein Delegation Bottleneck ist die minimale Anzahl der Nameserver in der Delegationskette jeder Domain, die kompromittiert werden muss, um die Domain zu kontrollieren. Wie in Abbildung 5.18 ersichtlich ist, werden fast 80 % der Domains von nur 2 Nameservern bedient. Abbildung 5.19 zeigt zudem, dass rund 63 % der Top 500-Webseiten von nur 2 Nameservern bedient werden.

Physikalische Bottlenecks sind die minimale Anzahl von Netzwerk-Gateways oder Routern zwischen Nutzern und Nameservern, die kompromittiert werden müssen, um die Domain zu kontrollieren. Abbildung 5.20 zeigt, dass über 30 % der Domains den Flaschenhals bei einem einzelnen Gateway bzw. Router haben.

DDoS-Attacken weit oben in der Hierarchie z. B. auf die Root-Server können das ganze DNS negativ beeinflussen. Im Oktober 2002 wurden Attacken auf Root-Server geführt, wobei die Root-Server bereits überproportional belastet sind [CAI04]. In der Folge waren neun Root-Server durch die Angriffe schlecht bis gar nicht erreichbar. Dies ergibt sich aus der verstreuten Anordnung der Root-Server weltweit (siehe Abbildung 2.8). Root Anycast hilft, das Problem zu vermindern, aber löst das grundlegende Problem nicht.

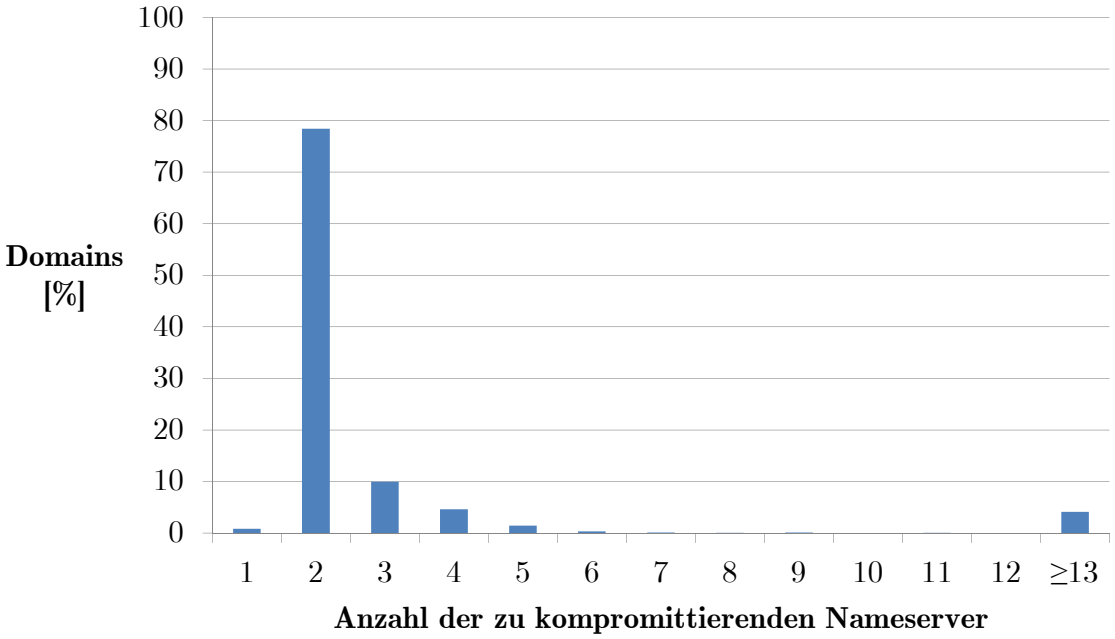


Abbildung 5.18.: Minimale Anzahl der zu kompromittierenden Nameserver für alle Domains (Delegation Bottlenecks) [RS04].

Ungleichmäßige Lastverteilung: Ein weiterer Aspekt der Skalierbarkeit ist die Lastverteilung, die direkt die DNS-Lookup-Performance beeinflusst. Da nur einige wenige Nameserver für die Hauptlast des DNS-Verkehrs verantwortlich sind, weist traditionelles DNS keine angemessene Lastverteilung auf. Besonders DoS-Attacken auf höhere Ebenen der typischen DNS-Hierarchie, d.h. auf DNS-Root-Server führen zu niedriger Lookup-Performance, wenn diese Root-Server zusammenbrechen.

5.3.3.2. Niedrige Performance

DNS-Anfragen beeinflussen die Web-Latenz. Rund 20-40 % der Zeit für den Bezug von Web-Inhalten wird durch DNS-Anfragen verbraucht. Darüber hinaus dauern rund 20-30 % der DNS-Anfragen mehr als 1 s. Manuelle Administration von DNS-Datenbanken führt zu Inkonsistenz und sogenannten Lame Delegations (siehe Abschnitt 5.3.3.4). 15 % der Domains weisen Lame Delegations auf, die zu Latenzen von bis zu 30 s führen können.

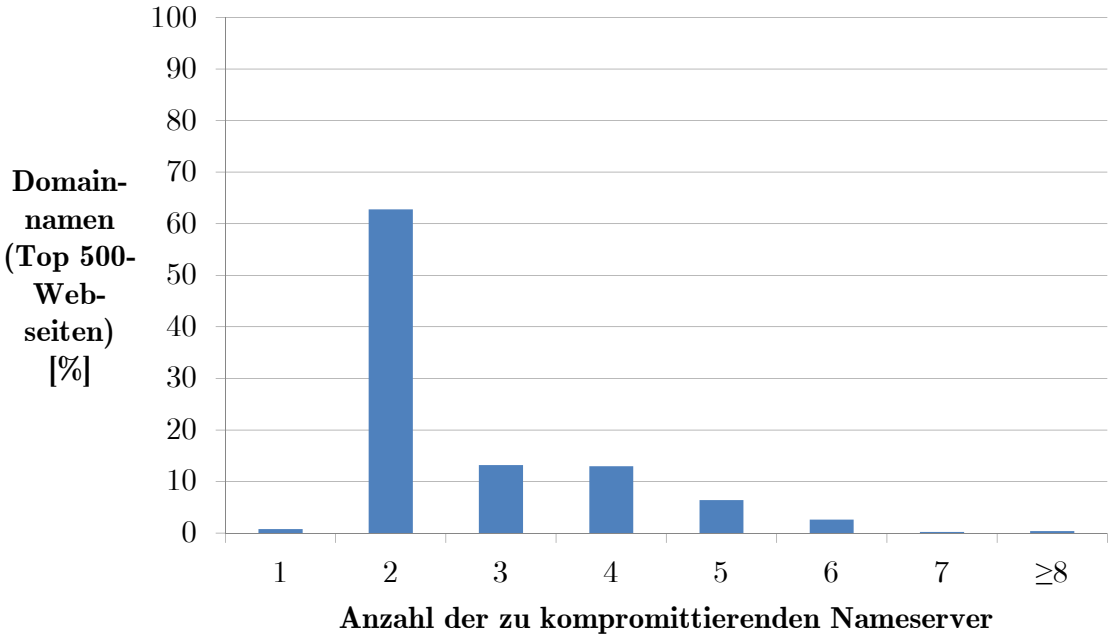


Abbildung 5.19.: Minimale Anzahl der zu kompromittierenden Nameserver für die Top 500-Webseiten (Delegation Bottlenecks) [RS04].

5.3.3.3. Inkonsistenz

Das für DNS eingesetzte Caching ist TTL-getrieben, das heißt, nach einer gewissen Weile erlischt die Gültigkeit der gecachten Einträge. Daraus ergibt sich ein Konflikt bei der Wahl der Dauer der TTL, denn dabei muss ein grundlegender Kompromiss zwischen Lookup- und Update-Performance gefunden werden. Einerseits führt eine lange TTL dazu, dass Änderungen nicht überall sofort im Netz sichtbar sind (86 % der RRs besitzen eine TTL $>$ 0,5 h). Eine kurze TTL ($<$ 10 min) auf der anderen Seite führt zu einer erhöhten Antwortlatenz.

5.3.3.4. Lame Delegations

Eine Lame Delegation bezeichnet eine fehlerhafte DNS-Einstellung. Dabei ist ein Nameserver als verantwortlich für einen Domainnamen gekennzeichnet, kann aber kein entsprechendes RR mit der zum Domainnamen gehörigen IP-Adresse vorweisen. Als Beispiel sei

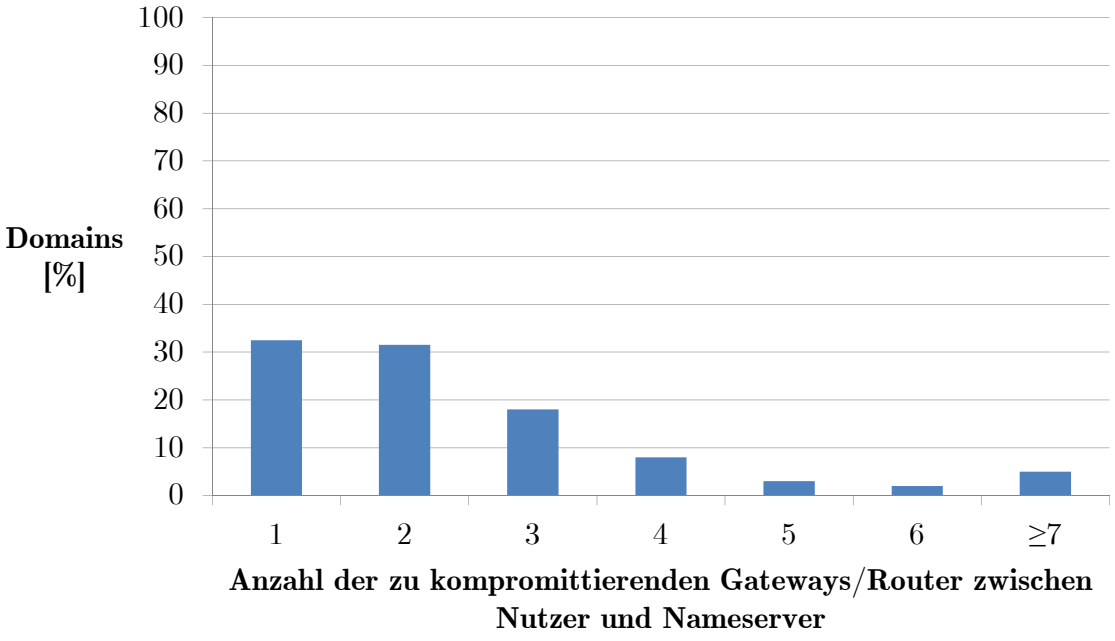


Abbildung 5.20.: Minimale Anzahl der zu kompromittierenden Gateways/Router zwischen Nutzer und Nameserver für alle Domains (Physikalische Bottleneck) [RS04].

hier eine Anfrage nach example.com durch den Host cache.example.net angeführt (siehe Abbildung 5.21).

Zuerst wird der voreingestellte Root-Server nach Verweisen für .com gefragt und liefert einen entsprechenden gTLD-Server, der nach Verweisen für example.com gefragt wird. Daraufhin wird der Nameserver 2 befragt, der die IP-Adresse des angefragten Domainnamens kennen müsste, aber einen Verweis zurück zum Root-Server liefert. Somit liegt eine Lame Delegation für Nameserver 2 vor, die geloggt wird. Anschließend wird Nameserver 1 befragt, der die Anfrage mit der entsprechenden IP-Adresse beantworten kann.

5.3.4. Konzipierung von P-DONAS

Um die Skalierbarkeit von traditionellem DNS zu verbessern, stellen ISP üblicherweise eine zusätzliche DNS-Infrastruktur in Form von DNS-Server-Farmen zur Verfügung. Die-

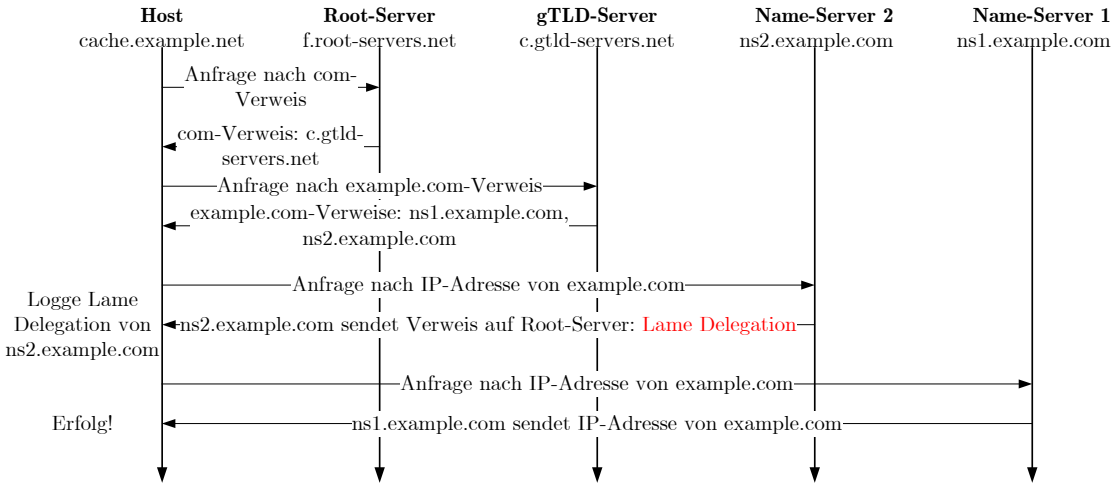


Abbildung 5.21.: Beispiel für das Auftreten einer Lame Delegation.

se befinden sich im Kernnetz eines ISP und versorgen einige Access Sites (ASs) eines ISP, die sich wiederum im TZN befinden (siehe Abbildung 5.22). ASs umfassen mehrere Zugangsknoten. DNS-Server-Farmen sollen die existierende DNS-Infrastruktur entlasten.

Um in erster Linie die Kosten für diese zusätzliche Infrastruktur einzusparen, wird P-DONAS als Ergänzung und Ersatz zum traditionellen DNS entworfen. Dabei werden ASs als ein P2P-basierter DHT (Kad-Netzwerk) organisiert. Auf den Zugangsknoten dieser als DHT organisierten AS werden DNS-Einträge strukturiert gespeichert (siehe Abbildung 5.23), wobei verfügbare Ressourcen der Zugangsknoten genutzt werden.

Jede AS bekommt dabei einen Hashwert und ordnet sich entsprechend diesem Hashwert auf einem DHT-Ring an, der den gesamten Hash-Adressraum repräsentiert. Die DNS-Daten werden entsprechend ihres Domainnamens auch gehasht. ASs mit einem Hashwert ähnlich dem des DNS-Datums speichern dieses Datum. Dabei hat der DHT-Ring Verbindung zum traditionellen DNS, so dass dieser mit DNS-Daten von dort ausgestattet werden kann. Der DHT-Ring bietet eine geringe Fehleranfälligkeit, da kein SPoF existiert. Durch den erweiterten Kad-Anfragemechanismus (siehe Abschnitt 4.3) kann eine deterministische Anfrage erreicht werden, die eine hohe Anfrage-Performance ermöglicht.

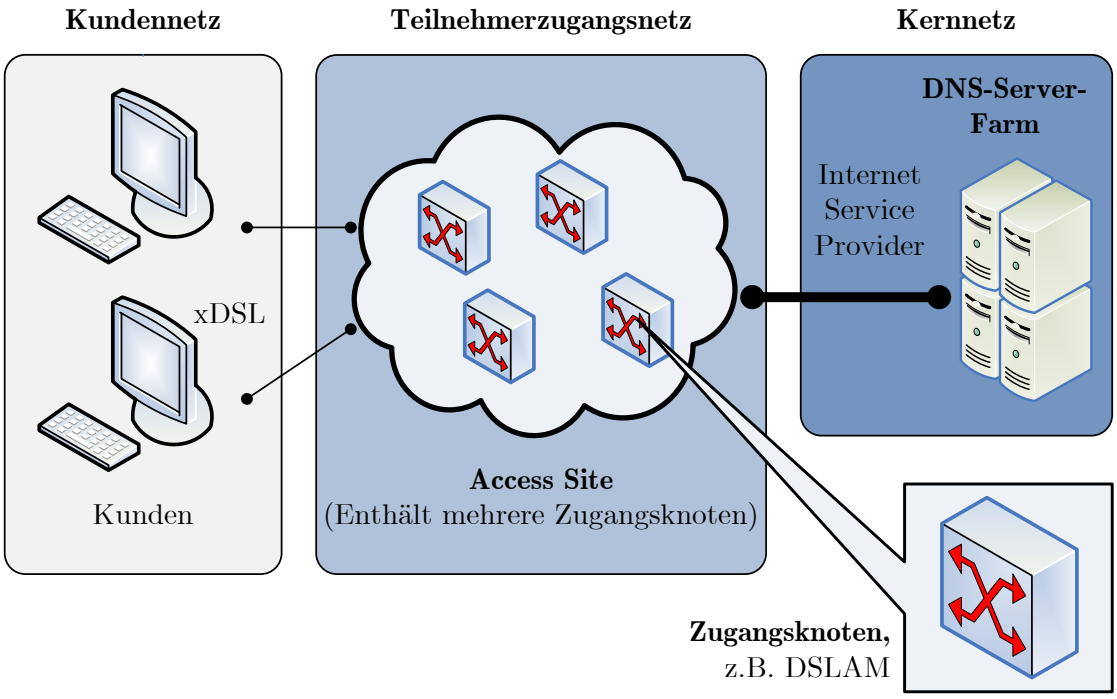


Abbildung 5.22.: AS (enthält mehrere Zugangsknoten), die mit einer DNS-Server-Farm verbunden ist.

5.3.4.1. Anforderungen an ein P2P-basiertes DNS

Ein P2P-basiertes DNS muss eine gute Performance aufweisen, d. h. eine niedrige Antwortlatenz besitzen, so dass für Nutzer keine spürbaren Wartezeiten auf die Beantwortung ihrer DNS-Requests auftreten. Um eine geringe Antwortlatenz zu erreichen, ist über eine intelligente Caching-Strategie nachzudenken, um durch geschicktes Datencaching die Antwortlatenz zu reduzieren. Ein P2P-DNS muss ausfallsicher und robust sein, um Schutz vor DoS- und DDoS-Angriffen zu bieten. Dafür sollte es keinen SPoF geben. Durch Redundanz bei der Datenspeicherung kann die Ausfallsicherheit weiter erhöht werden. Ein weiterer Aspekt ist die Selbstorganisation des P2P-DNS. Bei der Datenhaltung darf keine Inkonsistenz auftreten. In diesem Zusammenhang ist wiederum die Caching-Strategie wichtig, da durch hohe TTLs von gecachten DNS-Einträgen Inkonsistenzen im Netz auftreten.

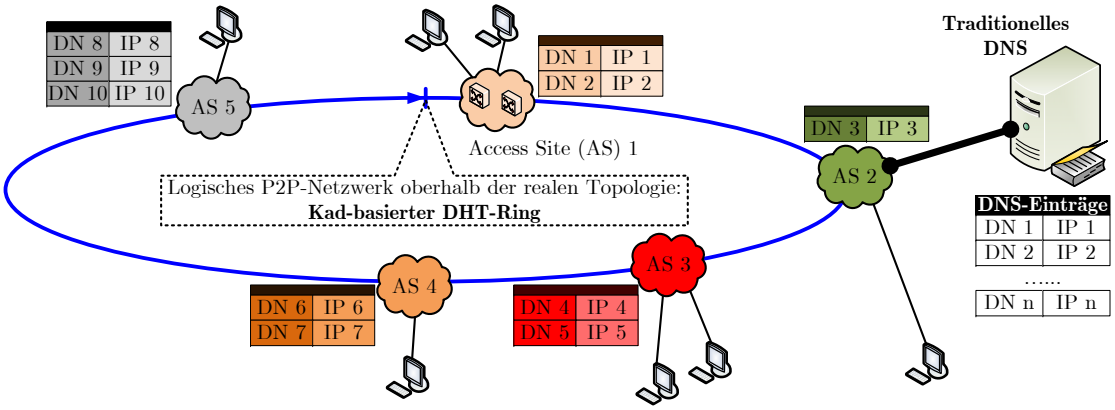


Abbildung 5.23.: Mittels Kad verbundene ASs mit DNS-Einträgen (Domainname (DN)-IP-Adresse (IP)-Paar).

Hohes Vertrauen in das P2P-DNS ist Grundvoraussetzung für die Annahme des neuen Dienstes. Durch Wahl des Zugangsknotens als Ort für die Implementierung von P-DONAS ist diese Voraussetzung gegeben. Damit einher geht der sicherlich wichtigste Anreiz aus Sicht eines Netzbetreibers, der durch Nutzung der Zugangsknoten die Kosten für zusätzliche DNS-Server-Farmen sparen kann.

5.3.4.2. Kad-basierter Entwurf von *P-DONAS*

In diesem Abschnitt wird das Konzept vorgestellt werden, welches das DNS- mit dem Kad-Konzept verbindet und die zu realisierende Schnittstelle implementiert. Der Nutzer stellt wie gewohnt DNS-Requests und erhält dementsprechende DNS-Responses. Zudem ist es weiterhin möglich, über das traditionelle DNS Anfragen zu verschicken. Die angesprochene Schnittstelle wird auf einem Zugangsknoten einer AS realisiert und im Folgenden als P-DONAS-Knoten bezeichnet. Da ein Zugangsknoten noch eine Vielzahl weiterer Aufgaben als die hier beschriebenen hat, wird im weiteren Verlauf nur noch die Bezeichnung P-DONAS-Knoten genutzt. Einem solchen P-DONAS-Knoten wird, wie einem DNS-Nameserver, eine IP-Adresse zugewiesen. Um P-DONAS nutzen zu können, muss ein Nutzer diese Adresse kontaktieren. Andernfalls werden alle DNS-Requests über das herkömmliche DNS beantwortet. Ein P-DONAS-Knoten hat die Aufgabe, RRs zu speichern, Anfragen von Nutzern sowie anderer P-DONAS-Knoten zu bearbeiten, ggf.

mit dem korrekten Domainnamen zu beantworten und analog zum DNS die Aktualität der RRs zu gewährleisten. Jeder P-DONAS-Knoten ist mit weiteren P-DONAS-Knoten sowie mit einem oder mehreren Nutzern verbunden und besitzt entsprechend der Kad-Spezifikation einen Hashwert. Die P-DONAS-Knoten bilden auf der physikalischen Topologie der zugehörigen Zugangsknoten eine logische Ring-Topologie (siehe Abbildung 5.23). Die Nutzer können jetzt ihre Requests anstatt an den Nameserver an ihren P-DONAS-Knoten stellen. Dieser verarbeitet dann die Anfrage weiter. In jedem Ring existiert ein Master-P-DONAS-Knoten (enthalten in AS 1.3 und AS 2.1 in Abbildung 5.24).

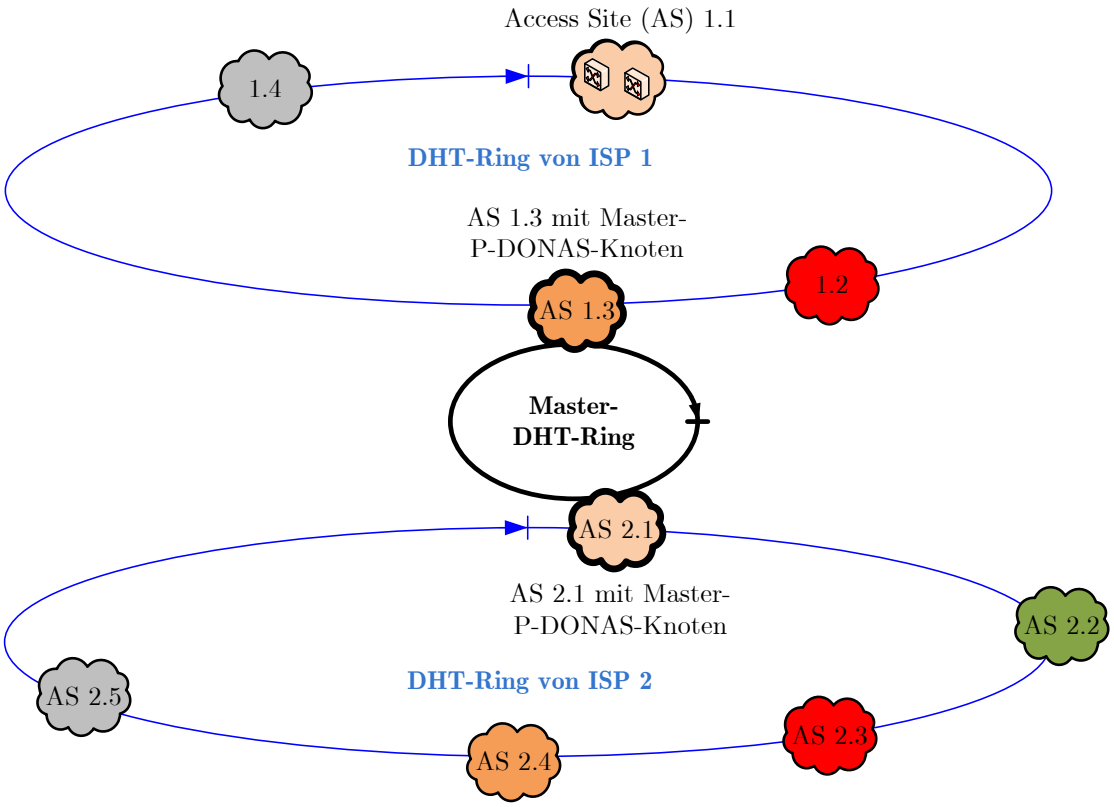


Abbildung 5.24.: Beispiel für den Aufbau einer DHT-Ring-Struktur mit Master-P2P-Knoten zur Verbindung zweier ISP. Aus Übersichtlichkeitsgründen sind die DNS-Einträge nicht dargestellt.

Dieser steht, zusätzlich zu seinen Verbindungen im Ring, in Kontakt mit einem Master-

P-DONAS-Knoten eines anderen Rings (siehe Abbildung 5.24). Ein Ring verbindet die ASs *eines* ISP. Über die Master-P-DONAS-Knoten, die mittels Master-DHT-Ring verbunden sind, ist somit eine ISP-übergreifende Kommunikation möglich. Die P-DONAS-Knoten jedes Rings besitzen außerdem eine Liste von Nameservern, die sie kontaktieren können. Ein DNS-Request an einen P-DONAS-Knoten löst eine Suche auf den Kad-Knoten entsprechend dem Kad-Suchalgorithmus aus. Falls eine Suche auf den Knoten der ASs des eigenen Rings erfolglos bleibt, kann der Master-P-DONAS-Knoten des Rings die Anfrage an den Master-P-DONAS-Knoten eines anderen Rings weiterleiten. Für den Fall, dass auch diese Suche kein Ergebnis liefert, besitzt jeder Knoten eine Liste von Nameservern, denen er das DNS-Request zusenden kann. Dies entspricht dem normalen Vorgehen im traditionellen DNS.

5.3.4.3. Architektur eines *P-DONAS*-Knotens

Die Architektur eines P-DONAS-Knotens ist in Abbildung 5.25 in Form eines Blockschaltbilds dargestellt. Die wichtigsten Bestandteile sind der Cache-Speicher (1), ein Speicher mit Knoten-Einträgen (2), die Routing-Tabelle (3), der Kad-Block sowie ein Block mit der Funktionalität zur Verarbeitung von DNS-Paketen (5). Der Cache dient als temporärer Speicher für RRs. Hier werden vor allem populäre RRs gespeichert, für die der P-DONAS-Knoten aufgrund der Distanz zwischen seinem Hashwert und dem des RR gemäß der Kad-Spezifikation nicht verantwortlich ist.

Analog dazu ist der Speicher für die Knoten-Einträge aufgebaut. In diesem werden allerdings die RRs abgelegt, für die der P-DONAS-Knoten aufgrund der Distanz zwischen seinem Hashwert und dem Hashwert der RRs verantwortlich ist. Die Routing-Tabelle enthält alle dem P-DONAS-Knoten bekannten Kontakte zu anderen Knoten und die nötigen Daten zur Kommunikation mit diesen. Der Kontakt zum Master-P-DONAS-Knoten muss in jedem Fall in der Tabelle enthalten sein. Soll der P-DONAS-Knoten eine Aktion auf dem Ring ausführen, beispielsweise die Suche nach einem bestimmten Domainnamen, so wird die Aktion zuerst mit einem oder mehreren Kontakten aus dieser Liste initialisiert. Der Kad-Block realisiert die Funktionsweise der Kad-Mechanismen wie sie in Abschnitt 2.4.6 beschrieben wurden. Die Kad-Funktionalität wurde um die Möglichkeit, DNS-Requests und die entsprechenden RRs für die Erzeugung des DNS-Reponses in Kad-Paketen zu übertragen und um eine Löschfunktionalität erweitert. Die Kad-Funktionalität umfasst

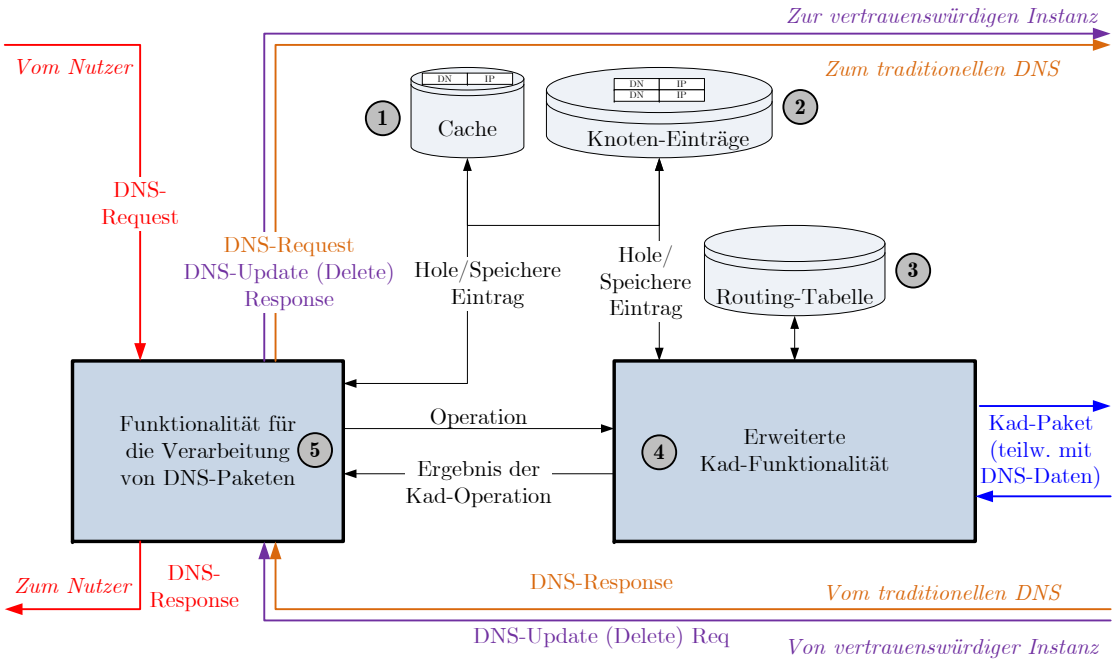


Abbildung 5.25.: Blockschaltbild eines P-DONAS-Knotens.

auch das Bootstrapping sowie die Wartung der Routing-Tabelle, wodurch eine stets konsistente und aktuelle Liste von Kontakten garantiert wird. Es sei angemerkt, dass mindestens ein P-DONAS-Knoten als immer im Kad-Netzwerk verfügbar angenommen wird, der für das Bootstrapping genutzt wird. Tritt der P-DONAS-Knoten neu in das Netzwerk ein, wird zunächst sein Hashwert berechnet. Die Algorithmen zur Hashwertberechnung werden nachfolgend erläutert.

5.3.4.4. Algorithmen zur Hashwertberechnung

Es werden zwei Hash-Algorithmen zur Schlüsselberechnung verwendet. Der weit verbreitete MD5-Algorithmus wird genutzt [Riv92], um den P-DONAS-Knoten-Hashwert sowie den Hashwert eines Domainnamens zu berechnen. Die Byte-Anzahl des Hashwerts ist zur Kompilierzeit frei auf einen ganzzahligen Wert größer Null einstellbar und kann somit an die konkrete Anzahl der Knoten im Kad-Netzwerk angepasst werden. Um den Domainnamen-Hashwert zu berechnen, wird der Domainname aus einem RR oder ggf. aus

eines DNS-Response, einem -Update oder einem -Request extrahiert. Aus ihr wird dann der Hashwert berechnet, welcher dazu genutzt wird, um Cache- oder Knoten-Einträge eindeutig identifizieren zu können oder die Distanz zwischen einem solchen Hashwert und einem P-DONAS-Knoten-Hashwert zu ermitteln.

Der zweite Algorithmus ist der CRC32-Algorithmus [Dig82]. Dieser wird benutzt, um aus dem Domainnamen die logische Anfangsadresse zu berechnen, ab der der Speichereintrag im Cache beziehungsweise in den Knoten-Einträgen auf einem P-DONAS-Knoten abgelegt werden soll. Da der CRC32-Algorithmus für die Realisierung in Hardware geeignet ist und ein schneller Hardware-basierter Zugriff auf Cache-Einträge unterstützt wird, wurde dieser Algorithmus gewählt [Jai92].

5.3.4.5. Formate der Speichereinträge

In Abbildung 5.26 ist ein Speichereintrag für das Beispiel „www.google.de“ dargestellt, wie er in den Knoten-Einträgen eines P-DONAS-Knotens gespeichert sein kann. Ein Datenblock des im Rahmen dieses Projekts verwendeten Speichers ist 32 Bit breit, kann aber leicht modifiziert werden. Das ergibt 4 Byte pro Datenblock. Diese Breite wurde gewählt, da die Wortbreite des Static RAM (SRAM)-Speichers eines Xilinx ML405-Boards unterstützt wird [XIL10], in dem perspektivisch Speichereinträge für den schnellen Hardware-Zugriff abgelegt werden sollen.

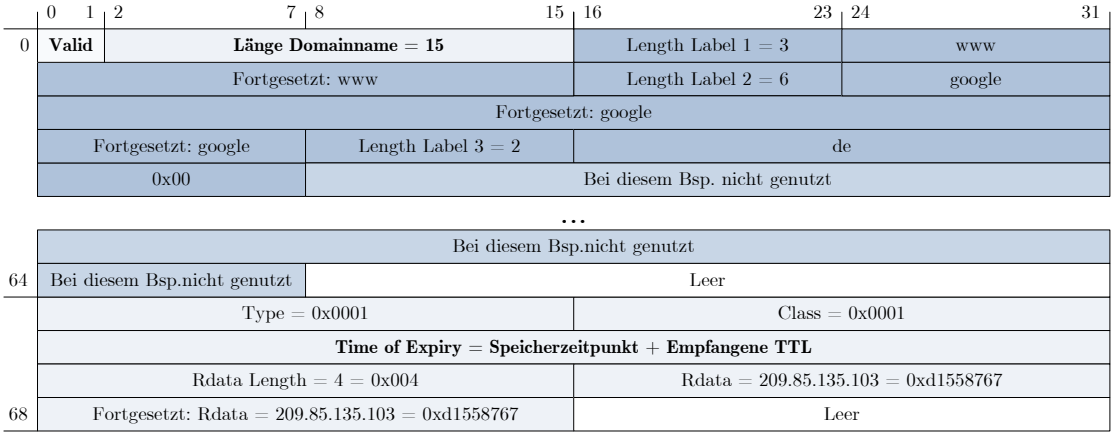


Abbildung 5.26.: Aufbau eines Knoten-Eintrags.

Valid	Bedeutung
00	Leer
01	Reserviert
10	Gelöschter Eintrag
11	Gültiger Eintrag

Tabelle 5.5.: Bedeutung des Valid-Felds eines Speichereintrags

Ein Speichereintrag entspricht im Wesentlichen einem aus dem DNS bekannten RR (siehe Abbildung B.4). Die wesentlichen Unterschiede bestehen im *Valid*-Feld, dem Feld *Länge Domainname* und dem Feld *Time of Expiry (TOE)*. Das Valid-Feld (siehe Tabelle 5.5) ist ein 2 Bit langer Wert, welcher signalisiert, ob der Speichereintrag gültig oder gelöscht ist oder ob es sich um einen leeren Speichereintrag handelt.

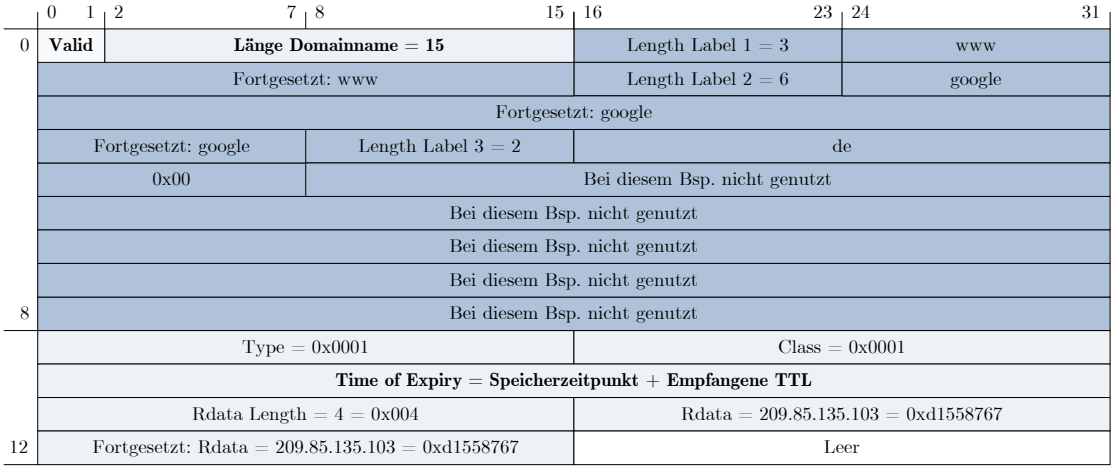


Abbildung 5.27.: Aufbau eines Cache-Eintrags.

Die Bitkombination 01 ist reserviert und momentan nicht belegt. *Länge Domainname* gibt die Länge des Domainnamens in Byte einschließlich der Length Labels - sowie des 0x00 -Feldes an (siehe auch Abbildung B.3). Mit Hilfe dieses Feldes müssen beim Auslesen des Domainnamens nicht zwingend alle physikalischen Adressen des Speichereintrages angesprochen werden, sondern nur die tatsächlich für den jeweiligen Domainnamen ge-

nutzten. Der gesamte Namensteil ist 260 Byte lang und wird ggf. mit Nullen aufgefüllt. Im Anschluss daran folgen die bereits bekannten Datenfelder für ein RR wie z. B. RData Length. Dieser Teil ist 14 Byte lang und wird auf 16 Byte aufgefüllt. Eine Besonderheit stellt das TOE-Feld dar. Dieses gibt im Gegensatz zur TTL nicht die Lebensdauer eines Eintrags an, sondern den Zeitpunkt, ab dem der Eintrag seine Gültigkeit verliert. Dieser Zeitpunkt berechnet sich aus dem Zeitpunkt der Speicherung des Eintrags sowie der TTL, die aus dem RR gewonnen wird. Der Zeitpunkt der Speicherung muss somit nicht gespeichert werden. Da ein P-DONAS-Knoten eine interne Systemzeit besitzt, kann leicht überprüft werden, ob die Systemzeit größer als die TOE ist. Ist dies der Fall, ist der Eintrag ungültig. Ein Speichereintrag in den Knoten-Einträgen ist insgesamt 276 Byte lang, was seine Ursache im verwendeten Adressierungsschema hat. Einträge im Cache werden in ihrer Größe reduziert und können nur Domainnamen mit einer zur Kompilierzeit festlegbaren maximalen Länge aufnehmen.

Dem liegt zugrunde, dass die meisten Domainnamen eine Länge von maximal 31 Byte haben [For06] und somit kann der Cache trotz dieser Einschränkung einen Großteil der auftretenden Domainnamenlängen aufnehmen. In Abbildung 5.27 ist ein Beispiel für einen Cache-Speichereintrag dargestellt, der Domains mit einer maximalen Domainnamenlänge von 34 Byte aufnehmen kann. Ergänzt man die 16 Byte für Type, Class, TOE, Rdata Length, Rdata und 2 Füll-Bytes, ergibt sich so für dieses Beispiel eine Größe von 52 Byte für Speichereinträge im Cache.

5.3.4.6. Adressierungsschema

Soll ein RR gespeichert oder gesucht werden, so wird zunächst aus dem Domainnamen mittels CRC32 ein 32 Bit langer Hashwert gebildet. Von diesen 32 Bit werden genau so viele Bits genutzt, dass jede logische Adresse im zur Verfügung stehenden Adressraum (Cache- oder Knoten-Einträge) angewählt werden kann. Es sei angemerkt, dass sich für unterschiedliche Domainnamen der gleiche Hashwert ergeben kann. Dadurch wird zunächst die gleiche Adresse für unterschiedliche RRs errechnet. Wenn allerdings eine solche sogenannte Kollision auftritt, wird das entsprechende RR an der berechneten Adresse + 1 (Kollisionsauflösung) gespeichert. Das gleiche Vorgehen wird bei der Suche nach einem Domainnamen angewandt.

Bei einer Länge eines Knoten-Eintrags von 276 Byte werden auf Grund der Wortbreite

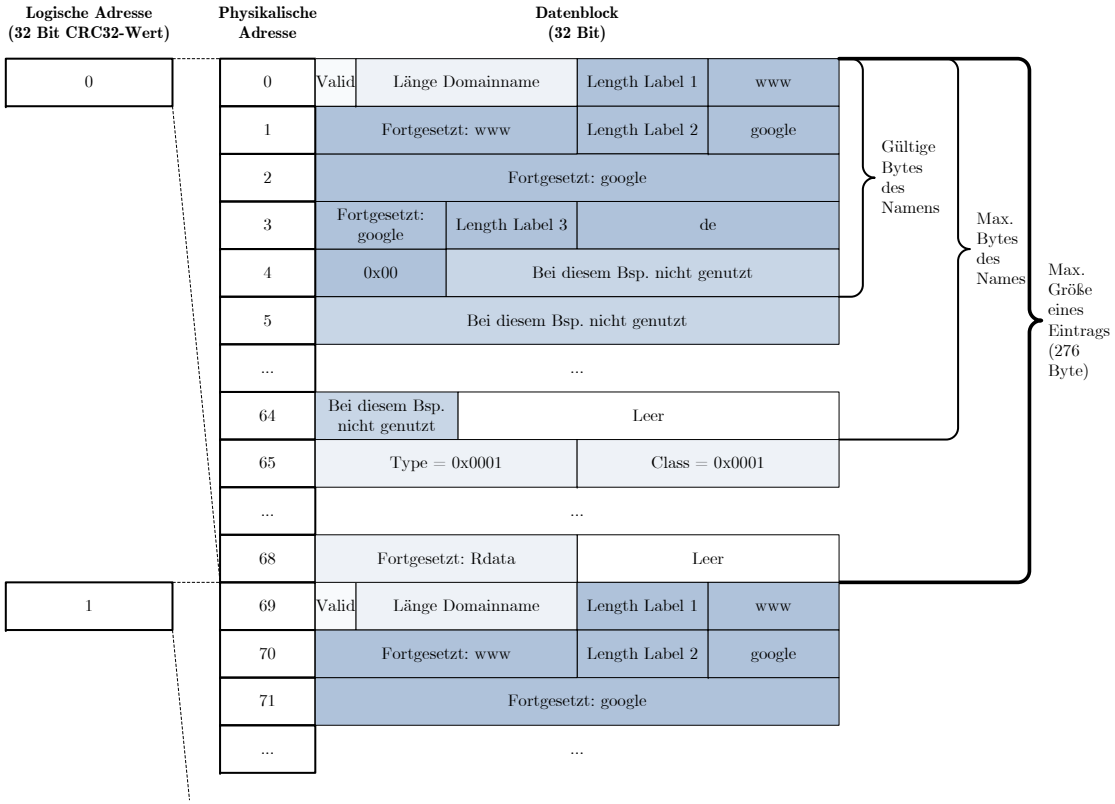


Abbildung 5.28.: Adressierungsschema für Knoten-Einträge.

von 4 Byte 69 Datenblöcke benötigt; für Cache-Einträge demzufolge z. B. 13 für eine maximale Domainnamenlänge von 34 Byte. Dies ist auch der Grund für das Auffüllen bestimmter Abschnitte eines Speichereintrags mit Nullen, da sich durch diese Methode Einträge ergeben, die ein Vielfaches eines solchen Datenblocks sind. Somit vereinfacht sich die Adressierung. Eine logische Adresse belegt also 69 physikalische Adressen (Knoten-Einträge) bzw. z. B. 13 physikalische Adressen (Cache-Einträge). Es wird nur die Hälfte des *DNS_RAM_SIZE* bzw. *CACHE_RAM_SIZE* Byte großen Speichers belegt, d.h. es ergibt sich eine maximale Anzahl von (DNS_RAM_SIZE/276)/2 bzw. (CACHE_RAM_SIZE/52)/2 Einträgen. Die Bedingung, dass nur die Hälfte des Speichers belegt werden soll, führt zu wenigen Kollisionen bei der Suche nach Einträgen im Speicher [Wid08]. Um für P-DONAS Antwortzeiten zu erreichen, die so niedrig wie möglich sind, ist eine geringe Anzahl von Kollisionen unabdingbar. Wird bei der Suche eine

physikalische Adresse erreicht, auf der keine Daten abgelegt sind, ist der Eintrag nicht im Speicher vorhanden. Das Adressierungsschema für Knoten-Einträge ist in Abbildung 5.28 dargestellt bzw. in Abbildung 5.29 für Cache-Einträge.

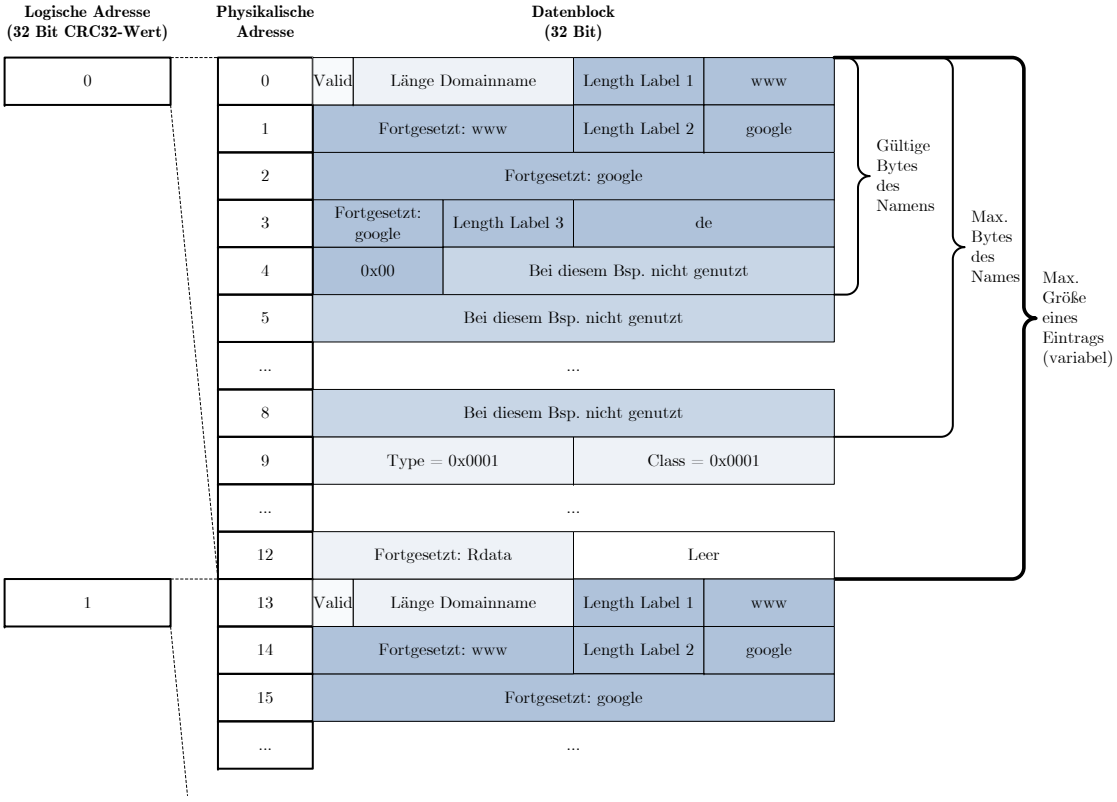


Abbildung 5.29.: Adressierungsschema für Cache-Einträge.

5.3.4.7. Auslösung von Knotenaktivitäten

Ein Knoten besitzt drei Schnittstellen. Zunächst gibt es eine Schnittstelle zum Nutzer.

Des weiteren gibt es die Schnittstelle zu anderen P-DONAS-Knoten im Kad-basierten DHT-Ring und die Schnittstelle zum traditionellen DNS. Abbildung 5.30 zeigt die drei Schnittstellen inklusive der über sie initiierten Aktivitäten.

Über die Schnittstelle zum Nutzer bekommt ein Knoten DNS-Requests vom Endnutzer und antwortet mit DNS-Responses. Über die Schnittstelle zum DHT-Ring werden Kad-

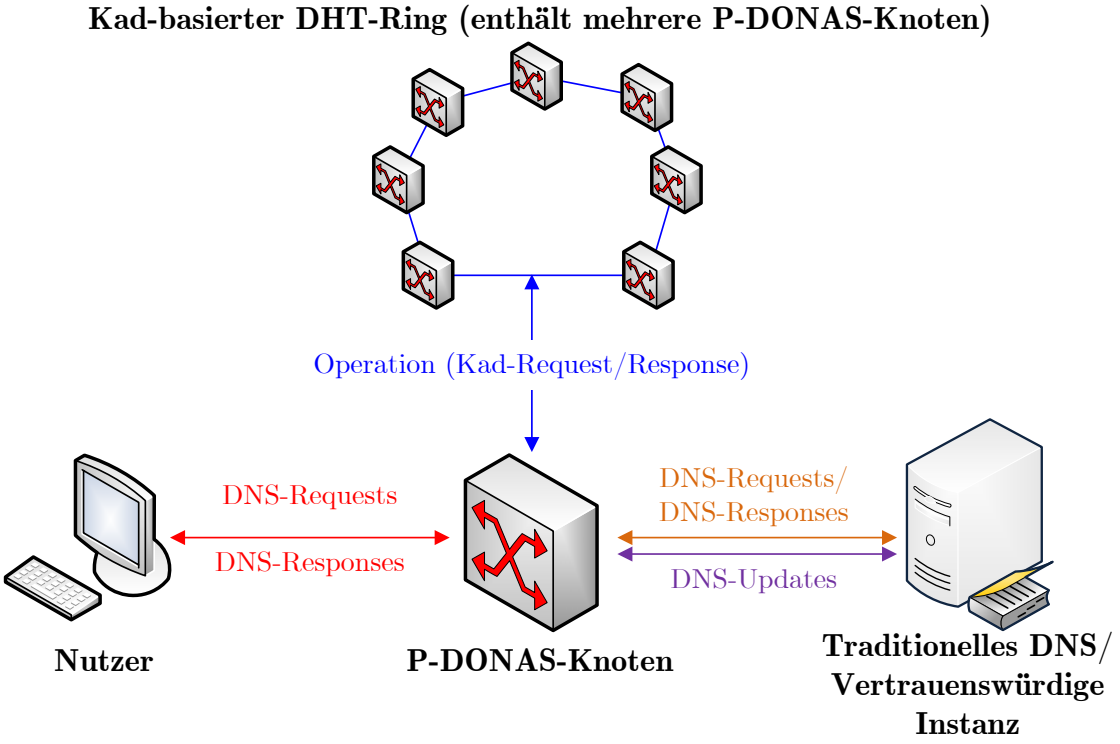


Abbildung 5.30.: Auslöser für Knotenaktivitäten: Nutzer, Kad-Netzwerk oder traditionelles DNS bzw. vertrauenswürdige Instanz.

Requests und -Responses empfangen und versendet, die Lookups sowie Speicher- und Löschoperationen bewirken. Über die Schnittstelle zum traditionellen DNS werden DNS-Update Requests sowie DNS-Update Delete Requests (Aufbau von DNS-Updates siehe Abbildung B.6) von einer vertrauenswürdigen Instanz sowie DNS-Responses von einem DNS-Nameserver empfangen.

Es sei angemerkt, dass ein spezielles Format von DNS-Updates für das Löschen von RRs (vom Autor DNS-Update Delete Request genannt) genutzt wird [Vix97]. Dazu müssen in der Update Section Class auf 0x00FF für Any gesetzt werden, die TTL auf Null sowie Rdata Length ebenfalls auf Null. Damit existiert kein Rdata-Teil. In Richtung des traditionellen DNS werden DNS-Update Responses, DNS-Update Delete Responses und weitergeleitete DNS-Requests vom Nutzer gesendet. Im Folgenden soll detailliert auf über diese Schnittstellen ausgelöste Aktionen eingegangen werden.

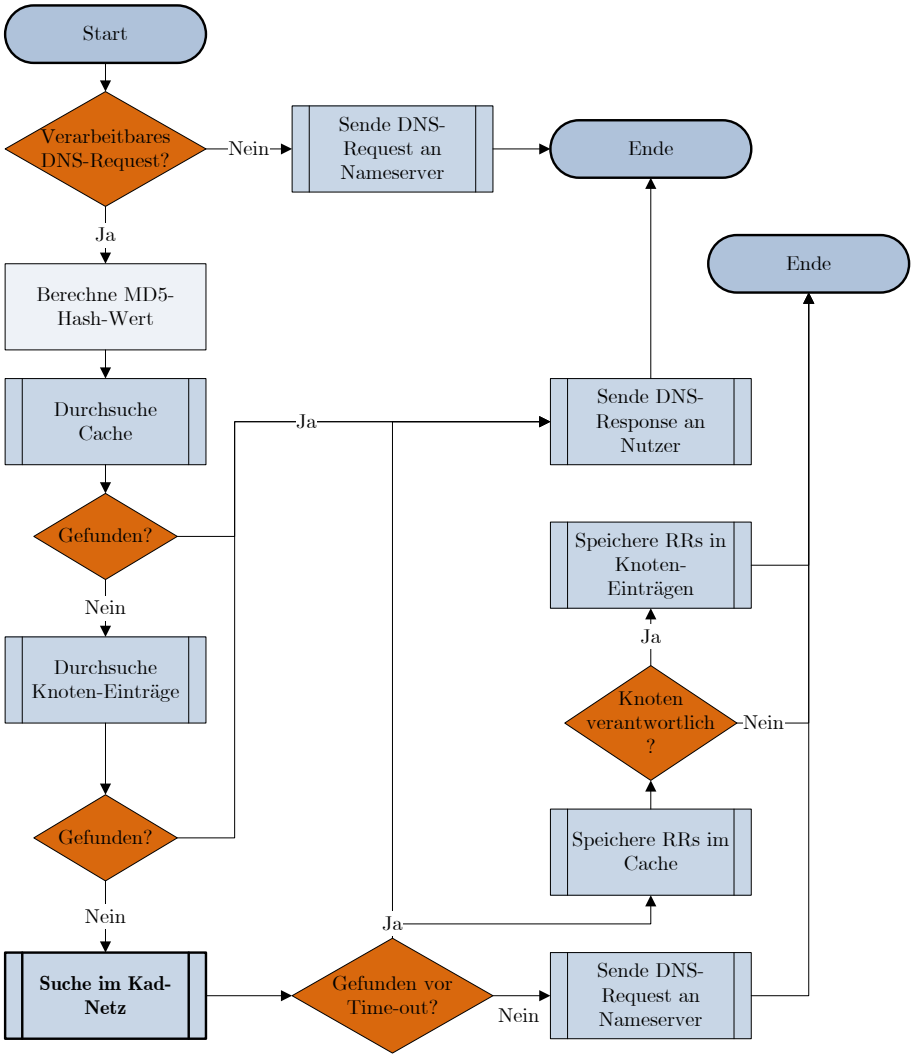


Abbildung 5.31.: Verarbeitung eines DNS-Requests vom Nutzer.

Interaktionen mit einem Nutzer

Der interne Ablauf in einem P-DONAS-Knoten während eines DNS-Requests ist in Abbildung 5.31 schematisiert. Bekommt der P-DONAS-Knoten ein Paket von einem Nutzer, wird zunächst geprüft, ob es sich um ein verarbeitbares DNS-Request handelt. Für eine Aufstellung verarbeitbarer DNS-Pakete sei auf Abschnitt B.3 verwiesen. Ist dies nicht der Fall, wird das Paket an einen vorkonfigurierten DNS-Nameserver weitergeleitet. Ist es aber ein verarbeitbares DNS-Request, wird der Hashwert des angefragten Domainnamens

gebildet. Anschließend wird mit Hilfe des Hashwertes überprüft, ob RRs des gesuchten Domainnamens im Cache des P-DONAS-Knotens gespeichert ist.

Ist dies der Fall (und sind die RRs laut ihren TTLs noch gültig), wird die Suche beendet und das DNS-Response wird dem Nutzer gesendet. Ist der RR aber nicht im Cache vorhanden, wird in den Knoten-Einträgen nach dem gleichen Muster gesucht. Ist die Suche erfolgreich, wird das DNS-Response zum Nutzer gesendet. Ist die Suche in den Speichern des vom Endnutzer angefragten P-DONAS-Knoten nicht erfolgreich, wird die Suche im Kad-Netzwerk ausgelöst: es kommt also zu einer Interaktion mit anderen P-DONAS-Knoten.

Interaktion mit anderen P-DONAS-Knoten

Suchoperationen auf dem DHT-Ring: Eine durch den Nutzer begonnene Suche (fett hervorgehoben in Abbildung 5.31) nach zu einem Domainnamen passenden RRs führt zunächst zu einer Suche nach Kontakten, die dem Ziel, also dem Hashwert des Domainnamens, am nächsten liegen. Dazu werden gleichzeitig bis zu *alpha* Kademlia FindValue Requests versendet (alpha besitzt in der Original-Kad-Implementierung den Wert Drei). Jeder Knoten, der ein solches Paket erhält, beantwortet die Anfrage mit einem Kademlia FindValue Response-Paket. Dieses enthält neben dem eigenen Knoten-Hashwert, den Hashwert des Domainnamens sowie neue und ggf. dem Ziel nähere Kontakte. Die Suche nach zielnahen Knoten wird so lange fortgesetzt, bis einer oder mehrere Knoten innerhalb der Suchtoleranz um das Ziel gefunden worden sind oder ein Time-out die Suche beendet. Diese Abbruchbedingungen für das Löschen des Suchobjekts werden durch einen separaten Thread überwacht. Knoten innerhalb der Suchtoleranz wird eine Anfrage vom Typ Kademlia FindValue Action Request übersandt, woraufhin sie ihre gespeicherten RRs durchsuchen. Die Antwort erfolgt im Falle einer Übereinstimmung nach dem bereits beschriebenen Schema. Jedes Ergebnis einer erfolgreichen Suche mittels Kademlia FindValue Action Request wird im Cache und ggf. im Speicher mit Knoten-Einträgen des ursprünglich vom Nutzer kontaktierten Knotens abgelegt, um so schneller auf zukünftige Anfragen für diesen Domainnamen reagieren zu können.

Es wäre ebenfalls denkbar, entsprechende Cache-Einträge entlang des beschrittenen Suchpfades vorzunehmen. Die für die Suche nötigen Kontakte bezieht der Kad-Algorithmus aus der Routing-Tabelle. Für den Fall, dass die Suche im Kad-Netzwerk erfolgreich war (dies schließt auch die Suche auf anderen DHT-Ringen über den Master-P2P-Knoten ein,

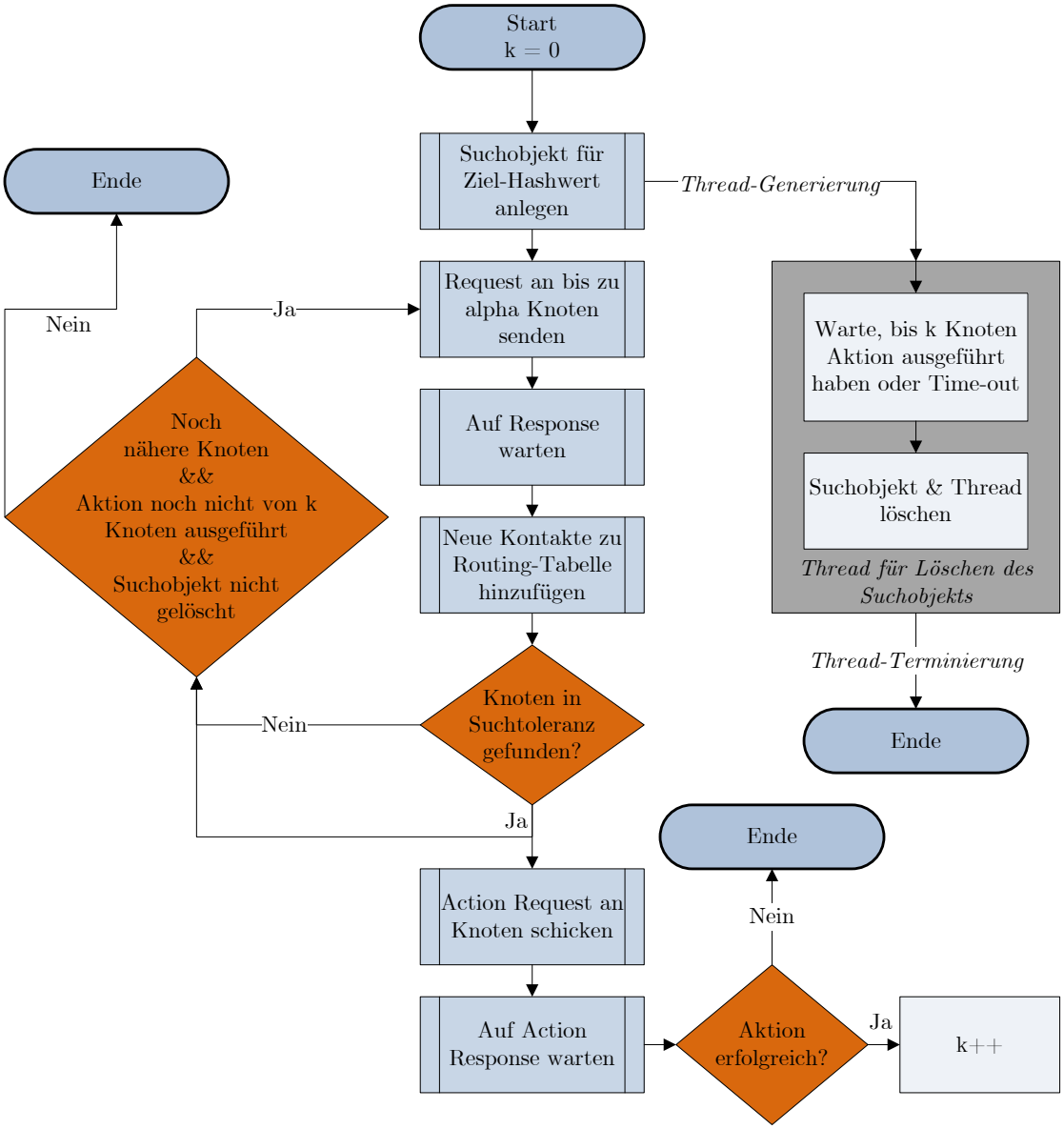


Abbildung 5.32.: Lookup-Vorgang auf dem Kad-Ring. Bei der Aktion handelt es sich um die Suche nach Knoten, dem Anfordern von RRs oder einer Speicher- oder Löschoperation.

siehe Abbildung 5.24) wird das DNS-Response für den Nutzer generiert.

Die beschriebene Suche im Kad-Netz ist in Abbildung 5.32 ersichtlich und erläutert die allgemeine Prozedur bei einem Lookup auf dem Kad-Ring. Entsprechend dem Suchtyp unterscheidet sich die Prozedur nur durch die finale Aktion. Bei der Aktion kann es sich um die Suche nach Knoten (ausgelöst durch Wartungsmechanismen wie Self- und Random-Lookup, siehe Abschnitt 2.4.6), das Anfordern von RRs oder eine Speicher- oder Löschoperation handeln.

Speicheroperationen auf dem DHT-Ring: Speicheroperationen auf dem DHT-Ring (fett hervorgehoben in 5.33 und 5.34) werden durch DNS-Update Requests oder DNS-Responses ausgelöst, die RRs zum Speichern enthalten. Bei einem Speichervorgang im Kad-Netz führt der entsprechende Knoten zunächst eine iterative Suche nach Knoten durch, die den geringsten Abstand zum Hashwert des in den RRs aufgeführten Domainnamens haben (allgemeine Lookup-Prozedur siehe Abbildung 5.32). Hierzu versendet er Kademlia Store Request-Pakete an ihm bekannte, dem Ziel nahe Knoten. Diese sollten mit einer Liste von näheren Knoten antworten. Die neu erlernten Knoten werden einerseits in den Bestand bekannter Knoten eingepflegt, andererseits aber auch der Liste zu befragender Knoten hinzugefügt. Von dieser Liste werden daraufhin die dem Ziel am nächsten liegenden, noch nicht kontaktierten Knoten ausgewählt. An diese wird nun ein Paket des Typs Store-Request verschickt. Erlangt der Ursprungsknoten im Laufe der Iterationsschritte Kenntnis über einen Kontakt innerhalb der Suchtoleranz um das Ziel herum, so wechselt der Typ der versendeten Nachricht in Kademlia Store Action Request, was einer direkten Aufforderung zum Abspeichern des mitgelieferten Domainnamen-Hashes sowie der dazugehörigen RRs entspricht. Wie bereits erläutert, werden auch bei allen speicherungsbezogenen Interaktionen die Absenderkontakte im Routingbaum des Empfängers aktualisiert. Als Antwort auf eine Kademlia Store Action Request erhält dessen Absender eine Kademlia Store Action Response, in dem die Information über den (Miss-)Erfolg der Speicherung enthalten ist. Der Absender zählt die Anzahl der erhaltenen Kademlia Store Action Responses mit Erfolgsmeldungen und beendet die Suche bzw. den Speichervorgang beim Erreichen des Wertes k (in der ursprünglichen Kad-Implementierung 10) oder durch einen Time-out. Zusätzlich dazu werden die gefundenen RRs im Cache des empfangenen Knotens gespeichert, um weitere Anfragen schneller beantworten zu können.

Löschoperationen auf dem DHT-Ring: Löschoperationen auf dem DHT-Ring (fett hervor-

gehoben in Abbildung 5.34) werden durch DNS-Update Delete Requests ausgelöst und funktionieren wie Lookups oder Speicheroperationen. Dem Ziel-Hashwert ausreichend nahe Knoten werden angewiesen, entsprechende RRs zu löschen. Die Löschoperation bricht ab, wenn entweder eine vorgegebene Anzahl von Knoten die RRs gelöscht hat oder durch ein Time-out. Durch die Tatsache, dass Zugangsknoten in das Kad-Netzwerk eintreten und es verlassen, kann sich die Verantwortlichkeit für RRs verändern, da sie auf Hashwerten und der eingestellten Suchtoleranz basiert. Demzufolge können Löschoperationen einen Zugangsknoten nicht erreichen, der für RRs verantwortlich war, aber dessen Verantwortlichkeit von einem anderen Zugangsknoten übernommen wurde. Allerdings erlischt die Gültigkeit dieser RRs automatisch nach einer TTL, die Teil eines jeden RRs ist [Moc87].

Jede Operation auf dem DHT-Ring trägt dazu bei, die Anzahl von Kontakten zu erhöhen, die der vom Nutzer angefragte Knoten kennt. Dadurch wird das Kad-Netzwerk redundanter und robuster. Außerdem erhöht sich die Lookup-Performance potentiell, da einige Iterationsschritte auf dem Weg zu einem Ziel ausgelassen werden können, da nähere Kontakte gelernt wurden.

Interaktion mit einem Nameserver

Liefert auch die Suche im Kad-Netzwerk kein Ergebnis oder kommt es zu einem Time-out, wird das DNS-Request an einen Nameserver weitergeleitet. Zu diesem Zweck besitzt der P-DONAS-Knoten die nötigen Daten, um mit einem vorkonfigurierten Nameserver in Kontakt zu treten. Erhält der P-DONAS-Knoten dann das entsprechende DNS-Response, leitet er dieses zum Nutzer weiter und speichert das entsprechende RR auf dem DHT-Ring. Dies geschieht mithilfe der Speicheroperation. Da offenbar kein P-DONAS-Knoten im Netz das gesuchte RR besitzt, wird dessen Hashwert gebildet. Mit diesem werden die P-DONAS-Knoten ermittelt, die laut ihres Hashwertes für die Daten des DNS-Response zuständig sind. Ist der das DNS-Response empfangende P-DONAS-Knoten laut seinem Hashwert nicht unter den zuständigen Knoten, werden die Daten nur in den Cache geschrieben. Anderenfalls werden die Daten den Knoten-Einträgen hinzugefügt. Der prinzipielle Ablauf ist in Abbildung 5.33 dargestellt. Für eine Aufstellung verarbeitbarer DNS-Pakete sei auf Abschnitt B.3 verwiesen.

Empfängt der P-DONAS-Knoten von Seiten des Netzes ein DNS-Update (Delete) Request, wird der in Abbildung 5.34 gezeigte Mechanismus auf einem P-DONAS-Knoten

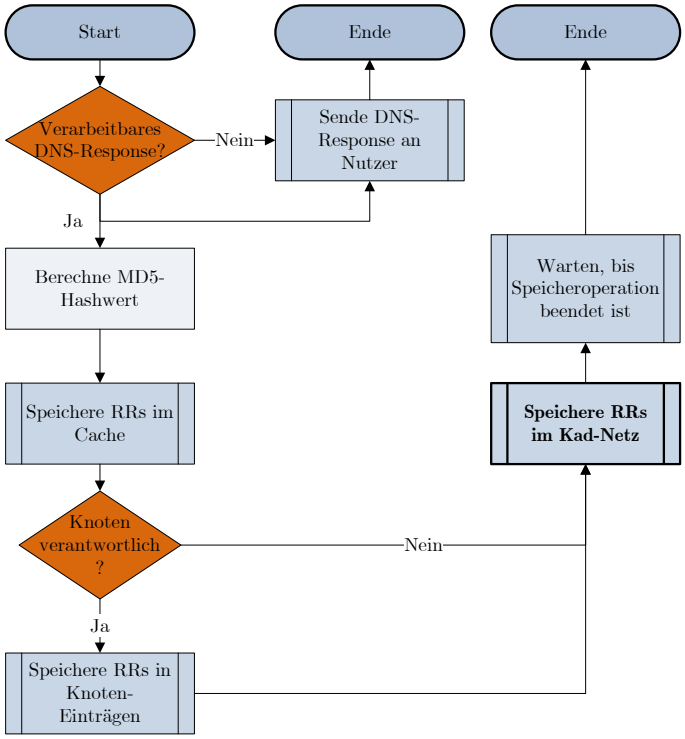


Abbildung 5.33.: Verarbeitung eines DNS-Response vom Nameserver. Operationen im Kad-Netzwerk sind fett hervorgehoben.

auslöst. Ein solches Update darf nur von einer vertrauenswürdigen Instanz, im Allgemeinen ein autorisierter Nameserver, durchgeführt werden. Es wird nun der RR-Hashwert des Updates gebildet und geprüft, ob der P-DONAS-Knoten für dieses RR zuständig ist. Ist dies der Fall, wird das RR in den Knoten-Einträgen gespeichert bzw. gelöscht. In jedem Fall muss die Verbreitung des RR im Kad-Netz per Speicher- bzw. Löschoperation stattfinden und es muss ein DNS-Update (Delete) Response generiert werden, das zur vertrauenswürdigen Instanz geschickt wird, um ihr ein (nicht) erfolgreiches Update zu bestätigen.

5.3.5. Implementierung von P-DONAS

Der Referenz-Code für das Kad-Netzwerk, auf dem diese Forschungsarbeit aufbaut, ist der eMule-Quellcode. Dieser ist frei verfügbar. Zum Zeitpunkt des Beginns der Arbeit war

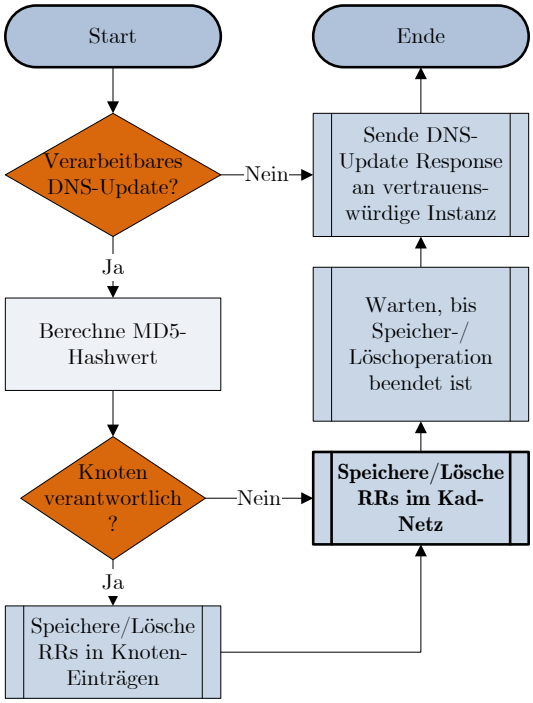


Abbildung 5.34.: Verarbeitung eines DNS-Update (Delete) Requests von einer vertrauenswürdigen Instanz. Die Speicher-/Löschoperation betrifft nur die Knoten-Einträge. Der Cache dient nur als temporärer Speicher; Cache-Einträge verschwinden nur durch den Ablauf ihrer TTL und werden nicht explizit gelöscht.

die Version 0.49a erhältlich, auf welcher die Kad-Implementierung basiert. Als Oberfläche und Compiler für die Implementierung wurde Microsoft Visual Studio 2005 ausgewählt. Das zuvor beschriebene Konzept wurde als voll funktionsfähiger Software-Prototyp in C++ für Windows realisiert. Darüber hinaus wurde P-DONAS ebenfalls vollständig für den Betrieb auf einem Xilinx ML405 Evaluation Board portiert [Sko09], so dass der Cache-Zugriff für den beschleunigten Zugriff in Hardware möglich ist. Dieses Board ist mit einem Virtex-4 FX20-Field-Programmable Gate Array (FPGA) ausgestattet, welcher mitsamt externem SRAM- und Double Data Rate (DDR)-Synchronous Dynamic RAM (SDRAM)-Speicher auch auf einem durchschnittlichen Zugangsknoten (siehe Abbildung 5.1) zur Verfügung steht [Nin11]. Der Prototyp liegt somit ebenfalls als Hardware-Software-Co-Design vor, war zum Zeitpunkt der Fertigstellung der Arbeit allerdings noch

in der Testphase und somit werden nur die Software-Prototypen beschrieben.

Die Hashwerte zum Finden und Speichern von RRs im Cache oder in den Knoten-Einträgen werden per CRC32 aus den entsprechenden Domainnamen erzeugt. Der Funktion wird zu diesem Zweck der Domainname in Form eines Strings übergeben. Zur Berechnung des Hashwertes eines Domainnamens oder eines P-DONAS-Knotens wurde der MD5-Algorithmus genutzt, welcher die gleichen Eingabewerte wie der CRC32-Algorithmus bekommt. Der Rückgabewert ist maximal 16 Byte lang; die Anzahl der genutzten Bytes ist einstellbar.

Grundlegend basiert die Implementierung zusammengefasst auf der Kommunikation zwischen mehreren Sockets, Funktionalität zur Verarbeitung des DNS-Protokolls sowie einer Erweiterung des Kad-Konzepts. Die einzelnen Funktionalitäten laufen parallel in Threads, welche miteinander über Shared Buffer und Warteschlangen Daten austauschen. Die Inter-Thread-Kommunikation erfolgt durch die Nutzung von Semaphoren; die Sicherung der Shared Buffer wird über das Mutex-Verfahren realisiert. Die Funktionsweise der implementierten Realisierung wird in den folgenden Abschnitten genauer beschrieben.

5.3.5.1. Socket-Kommunikation

Zur vollständigen Abdeckung der Kommunikation sind vier UDP-Sockets nötig: je ein Socket zur Kommunikation mit den Nutzern, mit einer vertrauenswürdigen Instanz, einem Nameserver sowie anderen P-DONAS-Knoten. Zur korrekten Definition eines Sockets werden die IP-Adressen der Netzwerkschnittstellen, die Klassifizierung des Netzwerks, in dem der Socket existiert und ein Port benötigt. Die IP-Adressen sind die des P-DONAS-Knotens, auf dem das Programm läuft. Die Klasse des Netzwerks ist die des Internets und wird mit AF INET übergeben. Da der Socket für die Verbindung zum Nutzer auf DNS-Requests warten soll, ist der Port für diesen Socket mit 53 vorgegeben. Der Socket für den Empfang von DNS-Updates von einer vertrauenswürdigen Instanz nimmt Pakete ebenfalls auf Port 53 entgegen. Der Port für den Socket zum Nameserver muss keinen bestimmten Wert annehmen, solange kein für andere Zwecke reservierter Port genutzt wird. Er wurde auf 59.636 gesetzt. Der Port, den eMule für den Kad-Socket vorsieht, ist mit 4672 vorgegeben.

Diese Daten werden der Socket-Methode übergeben. Hier wird auch festgelegt, dass es sich um einen UDP-Socket (SOCK DGRAM) handeln soll. Um Fehler abzufangen,

wird der Erfolg geprüft, indem der Rückgabewert ausgelesen wird. Liegt ein Fehler vor, wird der weitere Vorgang abgebrochen und die Cleanup-Methode aufgerufen, welche den Socket wieder abbaut. War der Vorgang erfolgreich, muss dem Socket über die Setsockopt-Methode mitgeteilt werden, dass die angegebene IP-Adresse mit genutzt werden soll. Wird dieser Schritt ausgelassen, kommt es zu Fehlern, da die Routine standardmäßig nicht vorsieht, eine IP-Adresse mehrfach zu vergeben. Kommt es zu einem Fehler, wird der Socket mithilfe der Closesocket-Methode geschlossen und Cleanup wird aufgerufen. War die Prozedur erfolgreich, muss der Socket noch an die übergebenen Adressdaten gebunden werden. Dies geschieht durch die Bind-Methode, welche nach gleichem Prinzip Erfolg und Fehlschlag behandelt. Wurde der Socket erfolgreich gebunden, lauscht er von jetzt an auf eingehende UDP-Pakete.

Jeder Socket hat beim Empfang und Senden Zugriff auf jeweils eine Warteschlange, in der bis zu 32.768 Pakete Platz finden. Hiermit wird sichergestellt, dass keine gerade empfangenen Pakete verloren gehen, während sich andere Pakete in der Verarbeitung befinden. Nach der Verarbeitung werden zu sendende Pakete in eine Warteschlange geschrieben. Der Socket sendet diese, sobald er bereit ist und neue Pakete werden sofort verarbeitet.

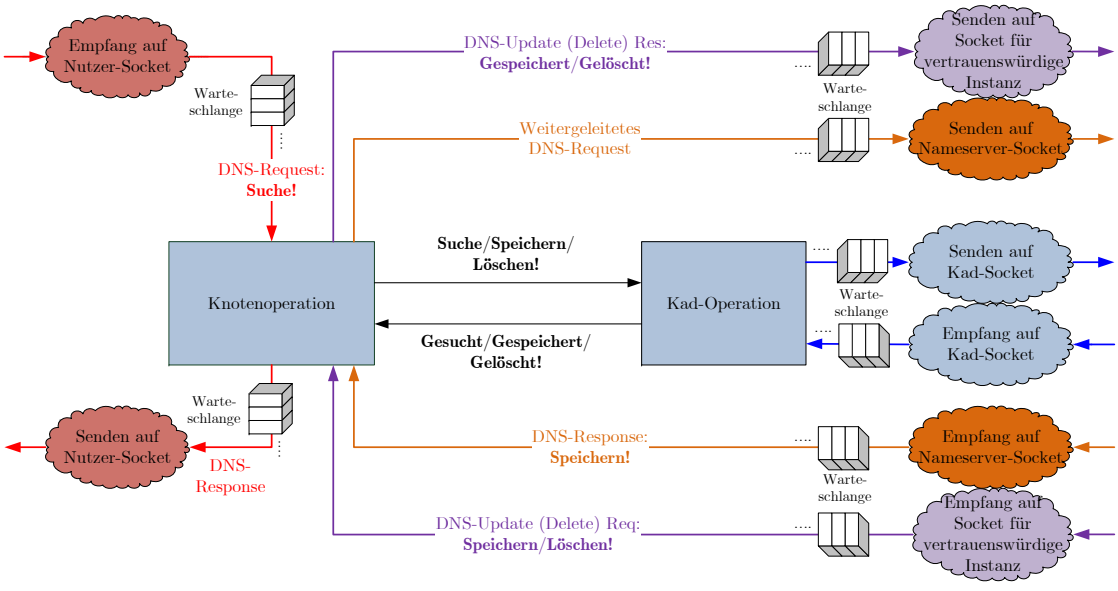


Abbildung 5.35.: Implementierung der Sockets.

Die vier Sockets kommunizieren sowohl untereinander als auch mit dem Nutzer, der vertrauenswürdigen Instanz, den P-DONAS-Knoten des DHT-Rings und einem vorkonfigurierten Nameserver. Abbildung 5.35 stellt den grundlegenden Ablauf der Kommunikation zwischen den Sockets dar. Der Socket für die Kommunikation mit dem Nutzer ist rot dargestellt, der für die vertrauenswürdige Instanz lila, der für den DHT-Ring blau und der für den vorkonfigurierten Nameserver braun.

5.3.5.2. Threads und Inter-Thread-Kommunikation

Um die programminterne Kommunikation zwischen den Sockets zu realisieren und Pakete zu verarbeiten, muss auf Threads zurückgegriffen werden. Pro Socket gibt es einen Thread, der für den Empfang der UDP-Pakete zuständig ist. Ein weiterer Thread pro Socket dient dem Senden von UDP-Paketen, die während der Verarbeitung der empfangenen Pakete erzeugt wurden. Für das Verarbeiten der empfangenen Pakete gibt es pro Socket 64 parallel arbeitende Threads, um den Ablauf zu beschleunigen. Eine Ausnahme bildet der Socket für Pakete von/zu der vertrauenswürdigen Instanz. In der derzeitigen Implementierung werden hierfür die 64 parallel arbeitenden Threads für Pakete vom/zum Nutzer genutzt, da momentan nur zu Testzwecken über diesen Sockets DNS Update (Delete)-Pakete empfangen und gesendet werden.

Die Threads kommunizieren untereinander durch sogenannte Semaphore. Dies sind im Grunde gesehen einfache Zähler, welche den Threads ein Ereignis signalisieren. Des Weiteren wurden für jeden Socket zwei Warteschlangen angelegt, welche durch das Mutex-Verfahren vor unberechtigtem oder gleichzeitigem Zugriff durch die Threads geschützt sind. Auf die erste Warteschlange hat der Thread für den Paket-Empfang schreibend Zugriff, während alle parallelen Threads für die Paket-Verarbeitung lesenden Zugriff besitzen. Entsprechend schreiben die Verarbeitungs-Threads ihre erzeugten Pakete in die zweite Warteschlange, aus der der Sende-Thread diese ausliest und an das gewünschte Ziel schickt. Der Ablauf vom Empfang eines UDP-Pakets bis zum Senden eines neuen Pakets ist in Abbildung 5.36 dargestellt.

Der Empfangs-Thread prüft zunächst, ob der Speicher, den sich dieser Thread mit dem Verarbeitungs-Thread teilt, verfügbar ist und kann in dem Fall über den Empfangssocket UDP-Pakete empfangen. Ist dies nicht der Fall, wartet der Thread bis der Speicher freigegeben wird. Darf der Thread auf den Speicher zugreifen und empfängt ein Paket, so wird

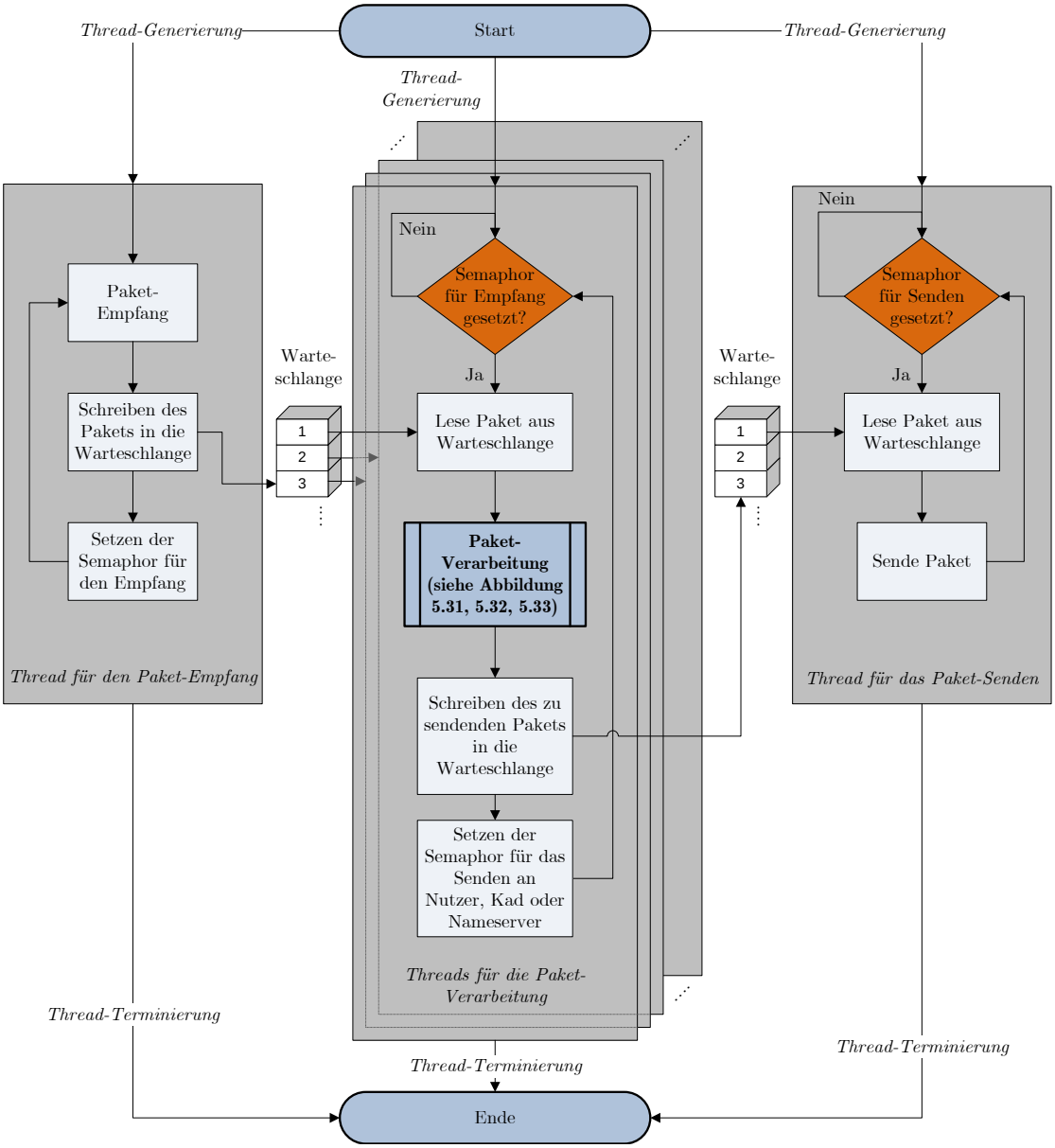


Abbildung 5.36.: Inter-Thread-Kommunikation.

der Speicher gesperrt, um einen Zugriff seitens des Verarbeitungs-Threads zu verhindern. Der Empfangs-Thread prüft zunächst, ob die Warteschlange noch nicht voll ist und

schreibt das empfangene Paket, seine Länge sowie die Adresse des Absenders in den Speicher. Ist dieser Vorgang abgeschlossen, signalisiert der Thread über ein Semaphor den Verarbeitungs-Threads, dass diese anfangen sollen, ihre Aufgaben auszuführen. Während der ganzen Zeit haben die Verarbeitungs-Threads bereits auf das Semaphor gewartet, welches ihnen erlaubt zu beginnen. Bekommen sie das Signal, lesen sie die benötigten Daten aus der Warteschlange aus. Jetzt können die Daten verarbeitet werden. Im Anschluss prüft der Verarbeitungs-Thread analog zur ersten Warteschlange, ob die zweite Warteschlange noch nicht voll ist. Ist dies der Fall, schreiben die Threads die Ergebnisse der Verarbeitung in den Speicher. Zum Schluss signalisieren die Verarbeitungs-Threads mit einer Semaphor, dass mit dem Senden begonnen werden kann. Der Sende-Thread hat bereits parallel zum beschriebenen Ablauf auf das Sende-Semaphor gewartet. Bekommt er nun das Signal, liest er die zum Senden benötigten Daten sowie die Adresse aus dem Speicher und veranlasst den Socket, das Paket zu senden.

Der Vorteil der Kommunikation über Threads besteht darin, dass die vier Sockets parallel den oben beschriebenen Kreislauf durchlaufen können und somit parallel arbeiten. Andernfalls könnte immer nur ein Socket empfangen, verarbeiten und senden. Die Verwendung zweier zugriffsbeschränkter Warteschlangen bietet den Vorteil, dass niemals zwei Threads gleichzeitig schreibend auf dieselben Daten zugreifen können. Dadurch wird beispielsweise vermieden, dass Daten überschrieben werden können, bevor sie verarbeitet wurden und somit verloren wären. Des Weiteren ist es mit diesem Vorgehen möglich, die Adressen für das Senden vom Empfangs-Thread bis zum Sende-Thread durchzureichen oder sie unter Umständen an beliebiger Stelle zu modifizieren und anzupassen.

Zusätzlich zu den 200 Threads zur Socket-Kommunikation werden folgende Threads benötigt:

- 1 Haupt-Thread, in dem alle anderen Threads als Worker-Threads laufen.
- 4 Threads: Je ein Thread zur Übergabe von DNS-Paketen vom Client, von einer vertrauenswürdigen Instanz, DNS-Paketen vom Nameserver und Kad-Paketen an einen freien Verarbeitungs-Thread.
- 1 Thread für das Erlernen neuer Peers in der Nähe des eigenen Hashwertes, der alle 4 Stunden aktiv wird (Self-Lookup, siehe Abschnitt 2.4.6 und [Bru06]).
- Variable Anzahl von Threads für das Erlernen neuer Peers in bestimmten Buckets, die einmal in der Stunde arbeiten (Random-Lookup, siehe Abschnitt 2.4.6 und

[Bru06]).

- Variable Anzahl von Threads für die Überprüfung, ob Peers mit abgelaufener Lebenszeit entfernt werden müssen (siehe Abschnitt 2.4.6 und [Bru06]). Die Überprüfung findet jede Minute statt.
- 1 Thread für die Stabilisierung und Aktualisierung der Routing-Tabelle [Bru06], der alle 45 Minuten aktiv wird.
- 1 Thread für die Aktualisierung der Cache- und Knoten-Einträge alle 5 Minuten. Hierdurch werden Einträge mit abgelaufener TTL entfernt.
- Variable Anzahl von Threads für das Löschen von Kad-Suchobjekten (siehe Abbildung 5.32). Für jeden Lookup-Vorgang auf dem Kad-Ring wird ein Thread angelegt und nach dem Löschen des Suchobjekt wieder terminiert.
- 1 Thread für das minütliche Aufnehmen von Messwerten.
- 1 Thread für das automatische Schließen der Log-Dateien nach Beendigung der Messwert-Aufnahme. Diese Maßnahme verhindert das ständige Öffnen und Schließen der Dateien und eine damit einhergehende Verlangsamung des Prototypen. Dieser Thread wurde nicht auf der portierten ML405-Variante implementiert, da die Messwerte dort direkt auf die Konsole ausgegeben werden.

Sämtliche Threads werden nur zu den jeweils definierten Zeitpunkten aktiv oder warten auf die Signalisierung per Semaphor bzw. auf einen Time-out und schlafen ansonsten. Somit wird der Prozessor nicht unnötig belastet. Die zeitlichen Angaben wurden der Kad-Originalimplementierung der Version 0.49a entnommen, die als Implementierungsgrundlage dieser Forschungsarbeit dient.

5.3.5.3. Software-Portierung auf ein Xilinx Evaluation Board

Die portierte Software-Implementierung auf dem Xilinx Evaluation Board umfasst im Wesentlichen die Funktionalität, die auch die Implementierung unter Windows bietet. Für die Umsetzung des Projekts wurden das Embedded Development Kit (EDK) 11.4 und das Software Development Kit (SDK) 11.4 genutzt. Das EDK wurde für die Hardware-Spezifikation verwendet. Das SDK wurde zur Bearbeitung des Software-Projektes eingesetzt.

Hardware-Spezifikation: Das Xilinx Virtex-4 ML405 Evaluation Board bildet die Testplattform und spiegelt einen Knoten wider [XIL10]. In der Tabelle 5.6 sind die wichtigsten genutzten Hardware-Ressourcen und Speicher des ML405-Boards wiedergegeben. Darüber hinaus wird der Virtex-4-FPGA (XC4VFX20-FF672) für die Realisierung der Hardware genutzt und stellt u. A. 153 KByte großen integrierten Block RAM (BRAM) (von dem 32 KByte für die Software reserviert werden) sowie 8544 logische Funktionseinheiten (Slices) bereit.

Komponente	Beschreibung
PowerPC 405	300 MHz Prozessor
BRAM	153 KByte Speicher
SRAM	1 MByte Speicher
DDR-SDRAM	64 MByte Speicher
TriMode_MAC_GMII	Ethernet-Schnittstelle, betrieben mit 100 MBit/s

Tabelle 5.6.: Hardware-Spezifikation des ML405-Boards.

Der PowerPC 405 dient zur Abarbeitung des Programmcodes und ist mit 300 MHz getaktet. Der SRAM ist für den schnellen Hardware-Zugriff auf Cache-Einträge vorgesehen und wird durch den Software-Prototypen momentan nicht genutzt. Der BRAM-Speicher wird durch das SDK vorgegeben, um den Verweis auf den Start des Programms zu setzen. Im DDR-SDRAM befindet sich der Programmcode. Der benötigte Heap und Stack des Programms befindet sich ebenfalls im DDR-SDRAM, der mit 64 MByte genügend Speicher liefert, um das Programm lauffähig zu machen. Ein detaillierter Schaltplan ist in Abbildung 5.37 ersichtlich.

Die einzelnen Komponenten sind über den Processor Local Bus (PLB) miteinander verbunden, welches die einzelnen Komponenten als Master und Slaves behandelt. Der PLB ist ein voll synchroner Bus [XIL12]. Der Joint Test Action Group (JTAG)-PowerPC-Controller zum Debugging des Programms im laufenden Betrieb und das System-Reset-Modul sind Master bezüglich des Prozessors. Über die JTAG-Schnittstelle wird ebenfalls die Programmierung des Boards vorgenommen. Der TriMode_MAC_GMII-Controller für die Ethernet-Kommunikation ist ein Master bezüglich der TriMode_MAC_GMII-Fifo. Alle anderen Komponenten sind als Slaves im System eingebunden. Der Clock-

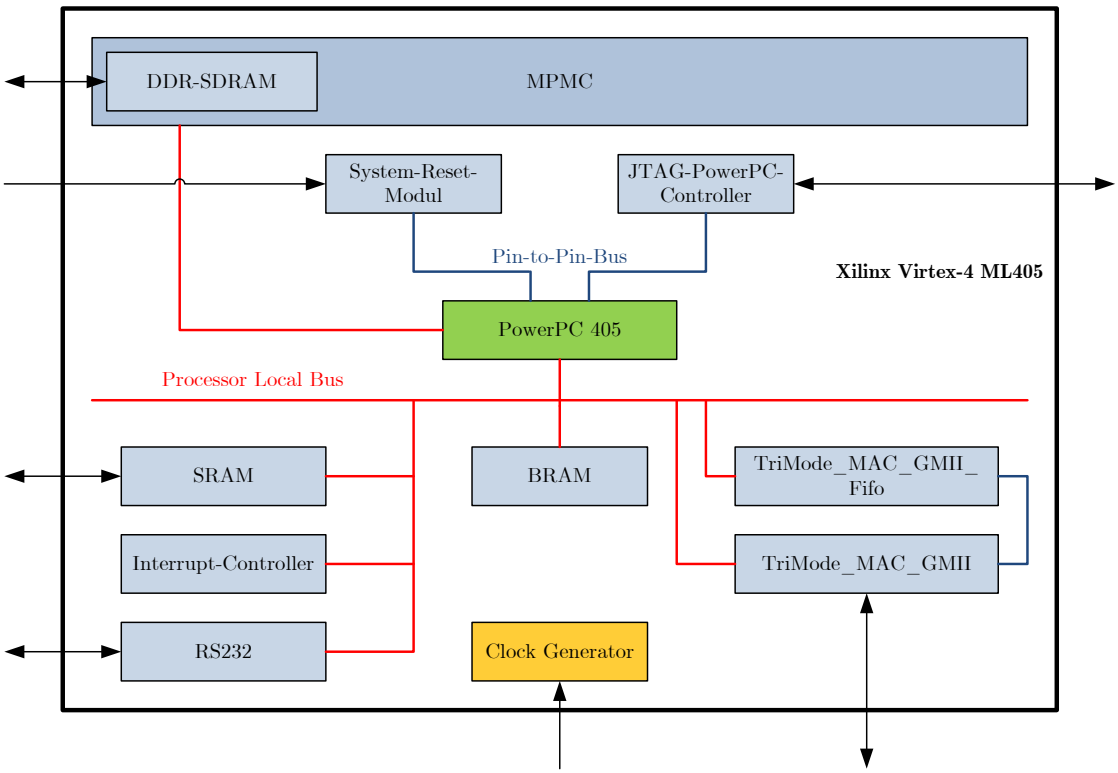


Abbildung 5.37.: Schaltplan der Hardware-Spezifikation. Aus Übersichtlichkeitsgründen sind die Taktleitungen des Clock-Generators zum Betreiben des Boards und der Busnetze nicht dargestellt.

Generator erzeugt die Frequenz zum Betreiben des Boards und der Busnetze. Der Interrupt-Controller verarbeitet sämtliche Interrupt-Eingänge der Komponenten zu einem einzelnen Interrupt-Ausgang für den PowerPC 405-Prozessor [XIL11a]. Der PowerPC 405-Prozessor ist mittels des Xilinx Pin-to-Pin-Busses direkt mit der JTAG-Schnittstelle und dem Reset verbunden. Der TriMode_MAC_GMII-Controller ist ebenfalls mittels Pin-to-Pin-Bus mit der TriMode_MAC_GMII-Fifo, die Ethernet-Pakete zwischenspeichert, zusammengeschlossen. Dadurch tritt ein direkter Kontakt ein, ohne den PLB nutzen zu müssen. Der in der Abbildung ersichtliche DDR-SDRAM wird über das Multi-Port Memory Controller (MPMC)-Module Interface angesteuert. Der MPMC ist ein vollwertiger Speicher-Controller, dessen Parameter individuell einstellbar sind [XIL11c]. Über die

RS232-Schnittstelle können Ausgaben während des laufenden Betriebs auf einem Terminal eines angeschlossenen PCs ausgegeben werden. Die benötigten Ressourcen auf dem Xilinx Virtex-4 ML405-Board, die für die Realisierung der Komponenten-Wrapper und Speicher-Controller notwendig sind (siehe Abbildung 5.37), belaufen sich insgesamt auf 5348 von 8544 verfügbaren Slices des Virtex 4-FPGAs (63 %) sowie 69 % des verfügbaren BRAM.

Software-Projekt: Diese Hardware-Spezifikation wurden dem SDK übergeben, in welchem das Softwareprojekt erstellt wurde. Das Software-Projekt besteht aus zwei Segmenten. Zum einen ist dies die „Software Platform“, in der alle zusätzlichen Funktionen bezüglich des Boards eingestellt werden können. Zum anderen werden hier der Xilkernel und der LightWeight IP (lwIP)-Stack aktiviert und es werden automatisch die Software-Bibliotheken zum Projekt hinzugefügt. Das zweite Segment ist die Software-Applikation selbst, welche mit der „Software-Platform“ verbunden wird.

Der lwIP-Stack ist eine Grundvoraussetzung, um die P-DONAS-Applikation nutzen zu können, denn er bietet viele netzwerktechnische Kommunikationsschnittstellen, welche auf dem verwendeten ML405 Evaluation Board nutzbar sind. Die Socketprogrammierung ist ähnlich der BSD-Programmierung und alle Funktionsaufrufe werden durch den Vorsatz „lwip“ ergänzt. Des Weiteren sind keine Unterschiede in der praktischen Socketprogrammierung gegenüber dem BSD-Standard vorhanden [XIL11b]. Verwendet werden nur UDP-Pakete bzw. UDP-Sockets. Es werden keine TCP-Verbindungen aufgebaut, da das Kad-Netzwerk und das DNS UDP-basiert arbeiten.

Zum Ausführen der P-DONAS-Applikation muss der Speicher definiert werden, in dem das Programm abläuft. Insbesondere müssen der Heap und der Stack auf eine ausreichende Größe eingestellt werden, um die Funktionalität von P-DONAS zu garantieren. Im Stack werden alle zur Kompilierzeit feststehenden Daten gespeichert. Im Heap hingegen werden dynamische Objekte abgelegt, die zur Laufzeit alloziert und gelöscht werden können. Die Größe und Platzierung der Speicher werden direkt im Linker-Skript eingestellt. Der Heap wurde auf eine maximale Größe von 16 MByte eingestellt; der Stack umfasst 1 MByte. Insgesamt umfasst die P-DONAS-Applikation als ausführbare Datei 31 MByte und wurde im DDR-SDRAM abgelegt.

5.3.5.4. Erweiterung um ein Sicherheitsprotokoll

Um die DNS-Kommunikation zwischen einem P-DONAS-Knoten und Endnutzer sowie DNS-Nameserver zu sichern, wird momentan TSIG (siehe Abschnitt 2.3.2) für den entwickelten Prototypen genutzt. DNSSEC wird allerdings perspektivisch unterstützt.

5.3.6. Auswertung

5.3.6.1. Testaufbau

Zur Demonstration der Funktionalität und Leistungsfähigkeit von P-DONAS auf Zielplattformen mit unterschiedlich vielen freien verfügbaren Ressourcen wurde ein Testszenario aus 8 heterogenen P-DONAS-Knoten aufgebaut, die alle über das Kad-Protokoll miteinander kommunizieren (siehe Abbildung 5.38). Das Testszenario legt den Fokus auf eine realitätsnahe Demonstration, Validierung und Evaluierung der P-DONAS-Funktionalität im laufenden Betrieb. Umfangreiche Untersuchungen der Skalierbarkeit mit Hilfe groß angelegter Simulationen wurden nicht vorgenommen.

Die Hashwerte der Knoten sind fest eingestellt und stammen aus einem Adressraum, der 256 mögliche Werte umfasst, d.h. der Hashwert-Adressraum umfasst 1 Byte. Die Suchtoleranz wurde auf den Wert 130 eingestellt, so dass für jeden möglichen Hashwert mindestens 3 Knoten als Ziel gefunden werden (Redundanzfaktor 3). Die Cache-Größe beträgt zunächst 46.400 Byte, so dass 200 Einträge in diesen passen, wurde allerdings für spätere Messungen reduziert, um den Einfluss des Kad-Netzes hervorzuheben. Ein Cache-Eintrag ist dabei 116 Byte groß und der Cache-Speicher wird nur zur Hälfte belegt, um die Anzahl auftretender Hashwert-Kollisionen beim Durchsuchen auf ein Minimum zu reduzieren (siehe Abschnitt 5.3.4.6). Die maximale Domainnamen-Länge, die ein Cache-Eintrag aufnehmen kann, beträgt 98 Byte. Die Speichergröße für die Aufnahme von Knoten-Einträgen ist auf 1 MByte festgelegt, so dass 1899 Einträge in den Speicher passen. Ein Knoten-Eintrag ist 276 Byte groß und dieser Speicher wird ebenfalls nur zur Hälfte belegt. Dabei wird die maximale Domainnamen-Länge von 255 Byte unterstützt.

Es wurde eine auf einem Nameserver des Instituts für Angewandte Mikroelektronik und Datentechnik (MD) aufgezeichnete DNS-Nutzlast zum Testen ausgewählt, die sich über 1 Tag erstreckt. Bei diesem Nameserver handelt es sich um einen Rechner mit dem

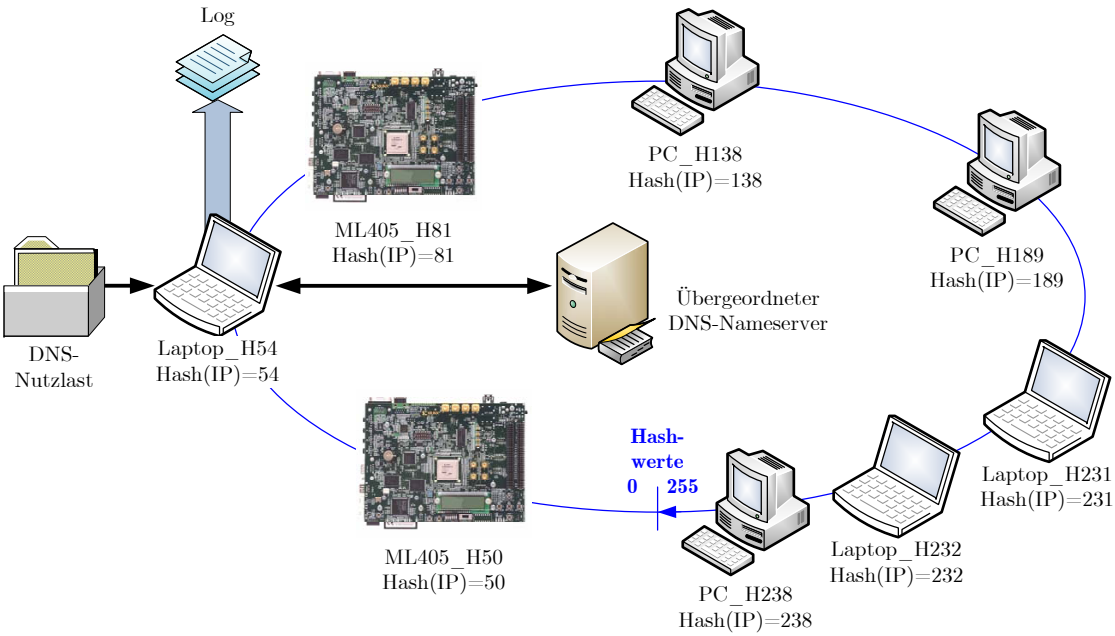


Abbildung 5.38.: P-DONAS-Testaufbau.

Betriebssystem Windows-Server 2008 R2, der über eine Intel Core 2 Quad-CPU mit 3 GHz sowie 8 GByte RAM verfügt.

Die aufgezeichnete DNS-Nutzlast wurde während der Tests auf den Log-Rechner Laptop_H54 abgespielt, der die Hauptmesswerte aufnimmt. Der Log-Rechner ist ein Laptop mit dem Betriebssystem Windows XP SP 3, einer Genuine Intel-CPU T2500 mit zwei 2 GHz-Prozessorkernen sowie 1024 MByte RAM. Der Log-Rechner delegiert DNS-Requests, die nicht beantwortet werden können, an einen übergeordneten DNS-Nameserver.

Eine Überblick über die Ausstattung aller eingesetzten Rechner bzw. ML405-Boards ist in Tabelle 5.7 ersichtlich. Sämtliche Rechner verfügen über 100 MBit/s schnelle Ethernet-Schnittstellen.

P-DONAS-Knoten	Betriebssystem	CPU	Arbeitsspeicher
ML405_H50	Xilkernel	PowerPC 405 300 MHz	64 MByte
Laptop_H54	Windows XP SP 3	Genuine Intel-CPU T2500 2 GHz	1024 MByte
ML405_H81	Xilkernel	PowerPC 405 300 MHz	64 MByte
PC_H138	Windows XP SP 3	Intel 4-CPU 1,5 GHz	512 MByte
PC_H189	Windows XP SP 3	Intel 4-CPU 1,5 GHz	512 MByte
Laptop_H231	Windows XP SP 3	Intel M-CPU 1,5 GHz/897 MHz	1528 MByte
Laptop_H232	Windows XP SP 3	Intel M-CPU 1,7 GHz/594 MHz	1024 MByte
PC_H238	Windows XP SP 3	Intel 4-CPU 1,5 GHz	512 MByte

Tabelle 5.7.: Ausstattung der für den P-DONAS-Testaufbau eingesetzten Rechner bzw. Xilinx ML405 Boards.

5.3.6.2. Aufgezeichnete DNS-Nutzlast

Die DNS-Nutzlast wurde vom 31.08 - 01.09.2011 aufgezeichnet und umfasst 191.700 DNS-Requests. Die Aufzeichnung begann um 12:36 Uhr und exakt 24 h wurden aufgezeichnet. Pro Minute wurden durchschnittlich 133 DNS-Requests gestellt. Der Verlauf der Anfragen über der Testzeit ist in Abbildung 5.39 ersichtlich.

Von den aufgezeichneten DNS-Requests konnte der Nameserver des Instituts MD 63.405 Pakete (33,1 %) nicht selbst beantworten und hat sie deshalb an übergeordnete Nameserver delegiert. Standardmäßig wurden davon 62.202 Pakete (32,4 %) an einen übergeordneten Nameserver der Universität Rostock delegiert. Wegen Überlastsituationen wurden 55 Pakete (0,1 %) nicht beantwortet. Weitere 1.203 Pakete (0,6 %) wurden an andere

Nameserver weitergeleitet, die nur kontaktiert werden, wenn der standardmäßige übergeordnete Nameserver nicht reagiert. Wegen Überlastsituationen wurden davon 355 Pakete (29,5 %) nicht beantwortet.

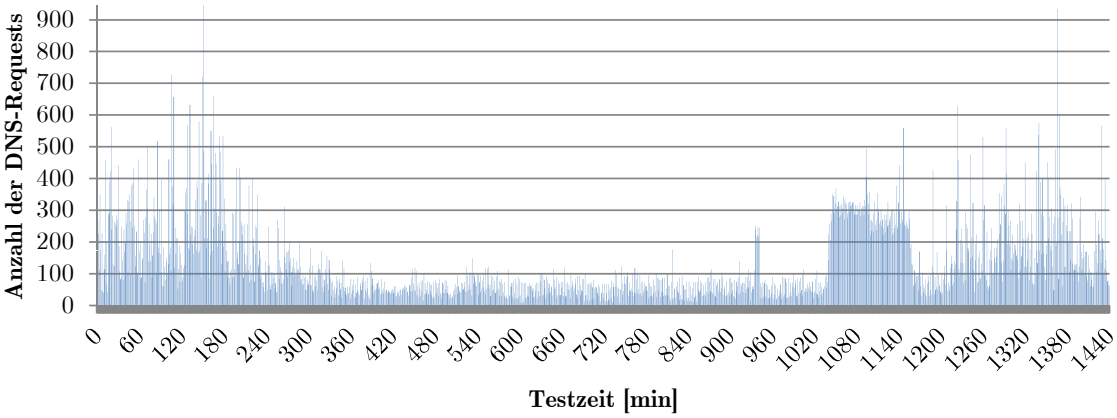


Abbildung 5.39.: Anzahl der DNS-Requests pro Minute über der Testzeit.

Antwortlatenz: Die durchschnittliche Latenz von der Anfrage per DNS-Request bis zur Beantwortung durch ein DNS-Response beträgt 45 ms. Die durchschnittliche Latenz pro Minute über der Testzeit ist in Abbildung 5.40 ersichtlich.

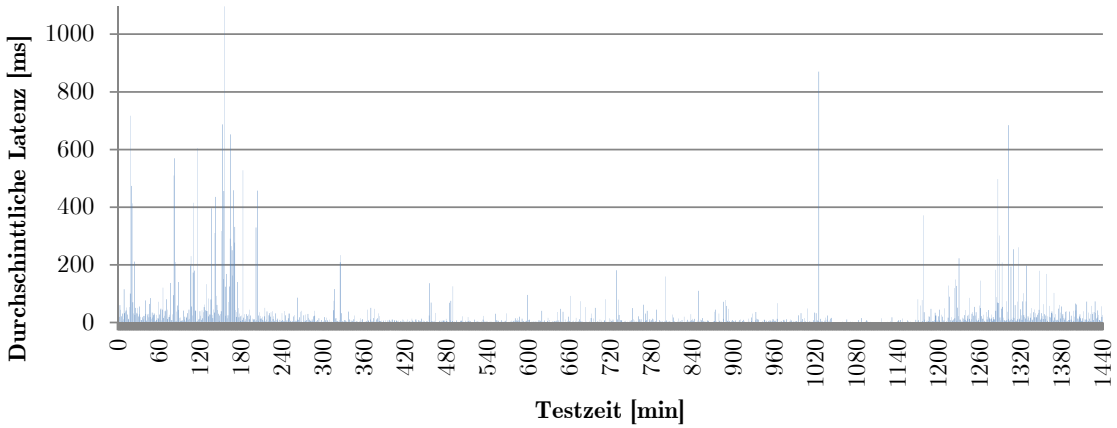


Abbildung 5.40.: Durchschnittliche Latenz pro Minute über der Testzeit.

Domainnamen-Länge: Die durchschnittliche Domainnamen-Länge beträgt 28 Byte.

Die Längen der Domains bewegen sich insgesamt im Bereich von 4 bis 250 Byte. Die Kumulative Verteilungsfunktion (KVF) der Domainnamen-Längen ist in Abbildung 5.41 dargestellt. Wie aus dieser Abbildung hervorgeht, sind 99,9 % der Domains im Cache speicherbar, da der Cache Domains mit einer Länge von bis zu 98 Byte aufnehmen kann.

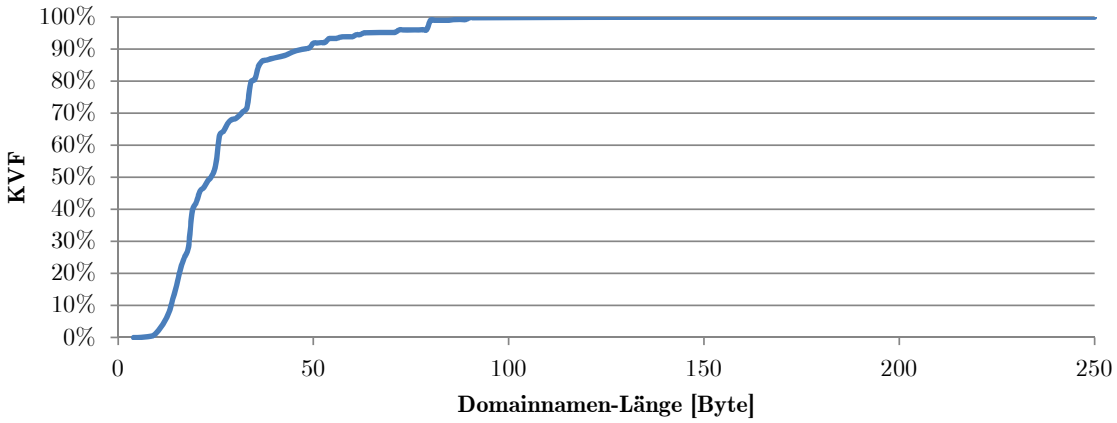


Abbildung 5.41.: KVF der Domainnamen-Längen.

Gesamtanzahl der nicht verarbeitbaren Requests: 109.782 (57,3 %) Pakete dieser Nutzlast werden durch den P-DONAS-Prototypen nicht unterstützt (siehe Abschnitt B.3). Eine Aufstellung der Zusammensetzung dieser nicht verarbeitbaren Pakete ist in Tabelle 5.8 ersichtlich. Die Anteile sind dabei bezogen auf 109.782 Pakete. Somit sind 81.918 DNS-Requests (42,7 %) durch P-DONAS verarbeitbar.

CPU-, Speicherauslastung und Anzahl der Threads: Die durchschnittliche minütlich gemessene CPU-Auslastung des Nameservers des Instituts MD ist mit 0,019 % festzustellen. Während der Messungen lag die durchschnittliche Speicherauslastung des Nameservers bei 167,4 MByte und 19 Threads waren aktiv.

5.3.6.3. Ergebnisse

Leerlauftests: Zunächst wurden die 8 Knoten des Testaufbaus gestartet, ohne die aufgezeichnete DNS-Nutzlast auf den Log-Rechner Laptop_H54 abzuspielen (2 Leerlauftests). Diese Maßnahme dient der Ermittlung der grundlegenden Datenlast im Kad-Netzwerk durch Instandhaltungsmaßnahmen. Die über alle Knoten akkumulierte Datenlast betrug

RR-Typ der Anfrage [Moc87]	Anzahl	Anteil [%]
IPv6 (0x001c)	66.144	60,25
SOA (0x0006)	17.480	15,92
PTR (0x000c)	12.966	11,81
SRV (0x0021)	9.988	9,10
TXT (0x0010)	2.153	1,96
A (0x0001) + 1 Additional RR	1.027	0,94
NS (0x0002)	17	0,01
MX (0x000f)	7	0,01

Tabelle 5.8.: Zusammensetzung der durch P-DONAS nicht verarbeitbaren Pakete.

12,1 MByte über 24 h und es wurden insgesamt 267.060 Kad-Pakete während dieses Zeitraums verarbeitet.

Weiterhin wurden die maximalen Füllstände der Kad-Warteschlangen (alle Knoten) mit 0 bis 3 Paketen ermittelt, so dass die verfügbare Kapazität von 32.768 Paketen pro Warteschlange ausreicht. Die durchschnittliche CPU-und Speicherauslastung pro Minute (alle Knoten) ist in Abbildung 5.42 ersichtlich. Die mittlere CPU-Auslastung liegt in jedem Fall unter 1 %. Auch die Speicherauslastung ist mit unter 10 MByte sehr niedrig. Die mittlere Anzahl der Threads pro Minute beträgt 220 pro Knoten; maximal arbeiten 221 Threads pro Knoten.

Tests mit variierendem Time-out-Wert für die Suche im Kad-Netz: Anschließend wurde die aufgezeichnete DNS-Nutzlast auf den Log-Rechner Laptop_H54 abgespielt. Auf der Grundlage der durch die Messungen gewonnenen Daten werden folgende Ergebnisse vorgestellt:

- Durchschnittliche Antwortlatenz (Log-Knoten Laptop_H54)
- Anteil der Antworten auf verarbeitbare DNS-Requests aus dem Cache (Log-Knoten Laptop_H54)
- Anteil der Antworten auf verarbeitbare DNS-Requests aus Knoten-Einträgen (Log-

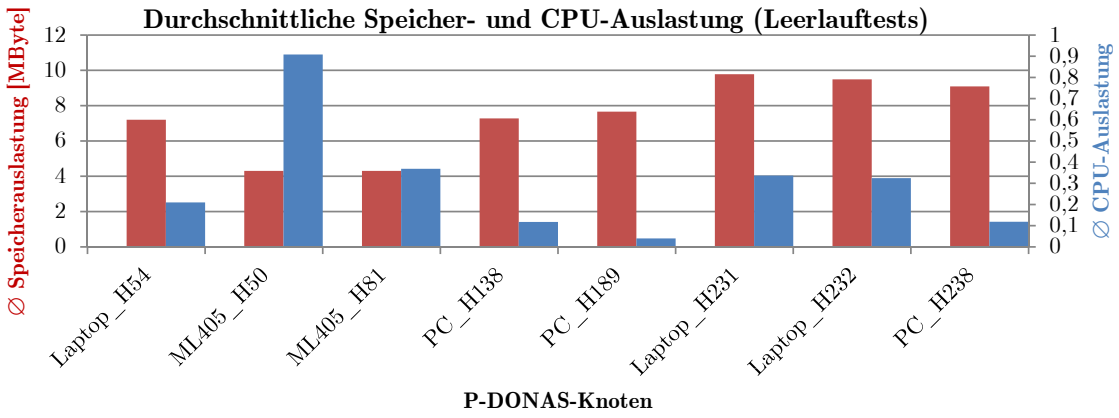


Abbildung 5.42.: Leerlauftests: Durchschnittliche Speicher- und CPU-Auslastung pro Minute (alle Knoten).

Knoten Laptop_H54)

- Anteil der Antworten auf verarbeitbare DNS-Requests aus dem Kad-Netz (Log-Knoten Laptop_H54)
- Anteil der Antworten auf verarbeitbare DNS-Requests von übergeordnetem Nameserver (Log-Knoten Laptop_H54)
- Durchschnittliche Anzahl der gespeicherten Knoten-Einträge (alle Knoten) und Cache-Einträge (Log-Knoten Laptop_H54)
- Anzahl und Größe der ausgetauschten Kad-Pakete (alle Knoten)
- Maximale Füllstände der Client- und Nameserver-Warteschlangen (Log-Knoten Laptop_H54) sowie der Kad-Warteschlangen (alle Knoten)
- Durchschnittliche CPU-Auslastung (alle Knoten)
- Durchschnittliche Speicherauslastung (alle Knoten)
- Maximale/Durchschnittliche Anzahl der Threads (alle Knoten)

Bei jeder Messung wurde der Wert für den Time-out der Suche im Kad-Netz variiert (siehe Abbildung 5.31 und 5.32). Dieser Time-out-Wert bestimmt maßgeblich, wie lange auf Antworten von Knoten im Kad-Netz gewartet wird, bis ein DNS-Request an den übergeordneten Nameserver weitergeleitet wird. Es wurden die Werte 0, 40, 80, 120, 160,

200 und 500 ms gewählt und jeweils 2 Messungen pro Wert vorgenommen. Jede Messung dauerte hierbei 24 Stunden.

Während der Messungen traten minimale Abweichungen bezüglich der Anzahl der gestellten DNS-Requests im Vergleich zur aufgezeichneten DNS-Nutzlast auf (22 bis 29 mehr gestellte DNS-Requests bzw. 0,01 % bis 0,02 % mehr pro Messung). Dieser Effekt konnte bereits während der Leerlauftests beobachtet werden. Die Diskrepanzen erklären sich dadurch, dass der Rechner zum Abspielen der DNS-Nutzlast einige zusätzliche DNS-Requests stellte. Zudem wurden zu Spitzenlastzeiten einige weitergeleitete DNS-Requests durch den übergeordneten Nameserver nicht verarbeitet und somit nicht beantwortet. Der insgesamt somit nicht beantwortete Anteil der DNS-Requests ist mit 0,12 % bis 0,41 % und im Mittel mit 0,18 % festzustellen.

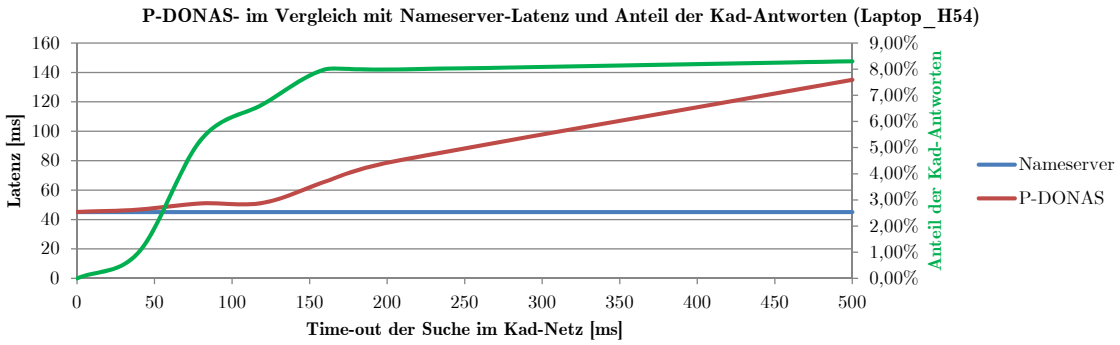


Abbildung 5.43.: Durchschnittliche Latenz über dem Time-out-Wert im Vergleich mit der Nameserver-Latenz und dem Anteil der Kad-Antworten auf verarbeitbare DNS-Requests.

Auf Grund dieser minimalen Diskrepanzen weicht der Wert für die Anzahl der durch das P-DONAS-System verarbeitbaren DNS-Requests ebenfalls geringfügig von der aufgezeichneten DNS-Nutzlast ab. Es traten Abweichungen von 0,01 % bis 0,04 % (im Mittel 0,03 %) zum Referenzwert von 81.918 verarbeitbaren DNS-Requests auf.

Antwortlatenz: In Abbildung 5.43 ist die durchschnittliche Antwortlatenz (Log-Knoten) über dem Time-out-Wert der Suche im Kad-Netz im Vergleich mit der Nameserver-Latenz und dem Anteil der Kad-Antworten auf verarbeitbare DNS-Requests ersichtlich.

Die Antwortlatenz nimmt mit zunehmendem Time-out-Wert etwa linear zu, da mehr Wartezeit auf Antworten aus dem Kad-Netz bleibt. Sie übersteigt aber bis zu einem Time-out-Wert von 200 ms nicht den Wert 80 ms, was noch keine spürbaren Verzögerungen beim Warten eines Nutzers auf ein DNS-Request bedeutet, da Menschen erst Verzögerungen ab einer Zehntelsekunde als spürbar empfinden [Fis09]. Zudem zeigt sich bezüglich dem Anteil der Kad-Antworten auf beantwortbare DNS-Requests, dass sich für die aufgezeichnete DNS-Nutzlast ab einem Time-out-Wert von 160 ms die Anzahl der Kad-Antworten nicht mehr maßgeblich ändert und bei ungefähr 8 % stagniert. Dieses Verhalten wurde durch die Wahl des hohen Time-out-Werts in Höhe von 500 ms bestätigt.

Antwortverhalten: In Abbildung 5.44 ist der Anteil der Cache-, Knoten-, Kad- und Nameserver-Antworten auf verarbeitbare DNS-Requests (Log-Knoten) dargestellt. An dieser Stelle sei noch einmal auf Abbildung 5.25 verwiesen, in der der Cache sowie der Speicher für die Aufnahme von Knoten-Einträgen, für die der Knoten aufgrund seines Hashwertes verantwortlich ist, ersichtlich ist.

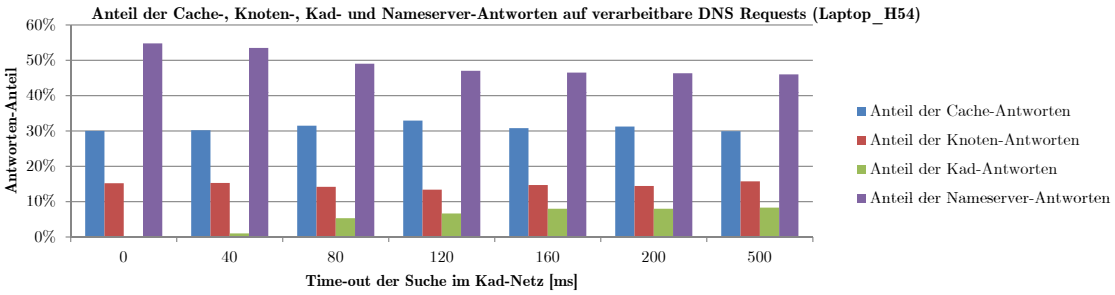


Abbildung 5.44.: Anteil der Cache-, Knoten-, Kad- und Nameserver-Antworten auf verarbeitbare DNS-Requests über dem Time-out-Wert.

Die Cache- und Knoten-Anteile sind für alle Time-out-Werte relativ konstant. Der Kad-Anteil nimmt mit zunehmendem Time-out-Wert gegenüber einem sinkenden Nameserver-Antworten-Anteil zu. Bei dem Time-out-Wert von 160 ms liegt der akkumulierte Anteil von Cache-, Knoten- und Kad-Antworten bei 53 %, so dass das P-DONAS-System die Mehrheit aller verarbeitbaren DNS-Requests selbst beantwortet. Unter Berücksichtigung der niedrigen Antwortlatenz von 66 ms an diesem Punkt ist dieser Time-out-Wert für

die aufgezeichnete DNS-Nutzlast optimal. Dabei wird deutlich, dass bereits aus einem 46.400 Byte kleinen Cache, in den maximal 200 Einträge passen, ein Großteil (bis zu 33 %) der verarbeitbaren DNS-Requests beantwortet werden kann.

Anzahl der Einträge: Wie Abbildung 5.45 zeigt, ist der Cache dabei im Mittel zu 86 % bis 91 % ausgelastet.

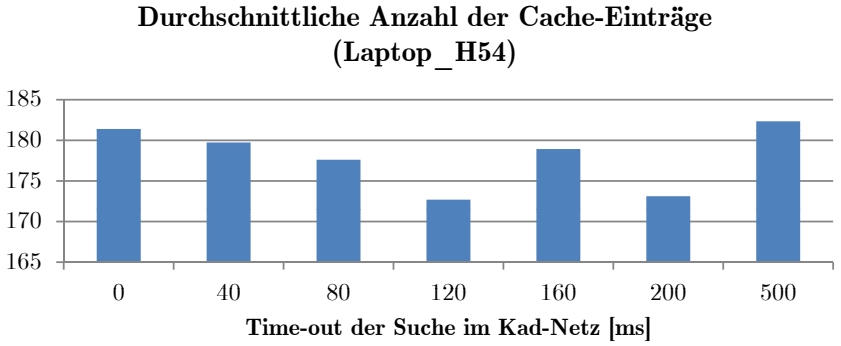


Abbildung 5.45.: Durchschnittliche Anzahl der Cache-Einträge pro Minute über dem Time-out-Wert.

Die durchschnittliche Anzahl der Knoten-Einträge pro Minute für jeden einzelnen Knoten des Testaufbaus ist in Abbildung 5.46 dargestellt.

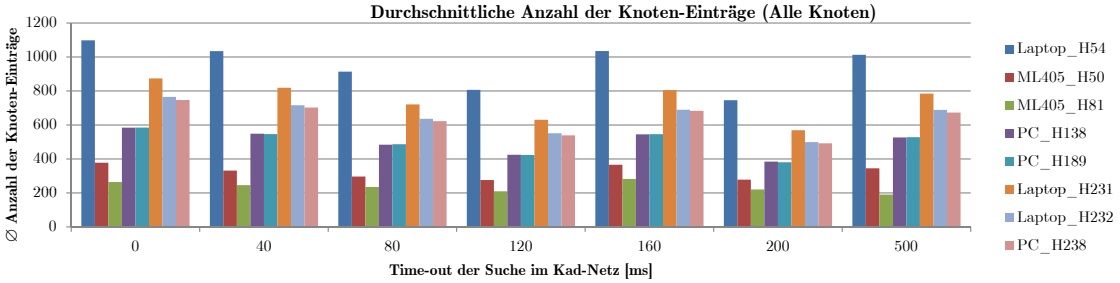


Abbildung 5.46.: Durchschnittliche Anzahl der Knoten-Einträge pro Minute über dem Time-out-Wert.

Durch die unterschiedlichen Zuständigkeiten eines Knotens für DNS-Einträge, die sich durch die Knoten-Hashwerte und die Hashwerte der Domainnamen sowie die eingestellte Suchtoleranz ergeben, gibt es hier unterschiedliche Anzahlen. Jeder Knoten kann bis zu

1899 Einträge speichern, so dass sich Auslastungen von 10 % (ML405_H81) bis 58 % (Laptop_H54) ergeben. Während jeder Messung reicht der 1 MByte große Speicher somit aus.

Kad-Datenverkehr: Die Anzahl der verarbeiteten Kad-Pakete über dem Time-out-Wert wird durch Abbildung 5.47 ersichtlich.

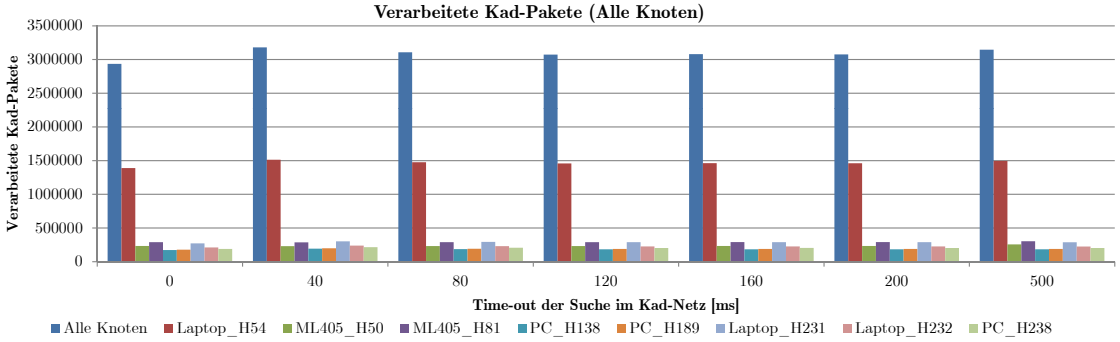


Abbildung 5.47.: Anzahl der verarbeiteten Kad-Pakete über dem Time-out-Wert.

Abbildung 5.48 zeigt das Volumen des verarbeiteten Kad-Datenverkehrs über dem Time-out-Wert.

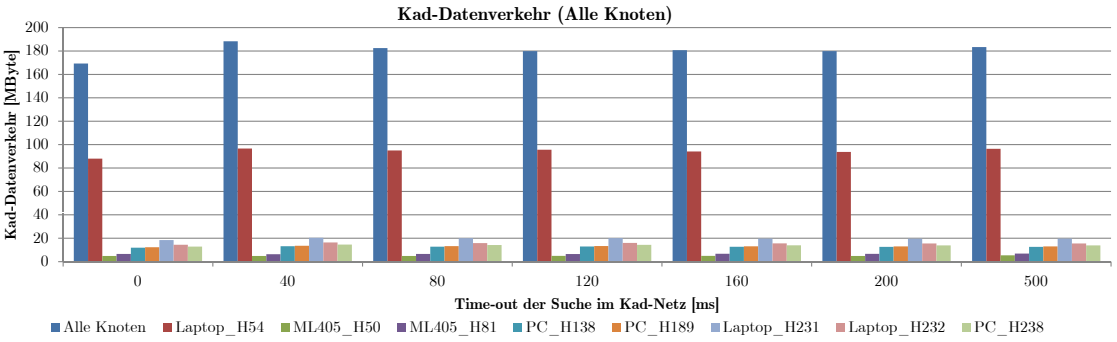


Abbildung 5.48.: Volumen des verarbeiteten Kad-Datenverkehrs über dem Time-out-Wert.

Da der Log-Knoten Laptop_H54 die aufgezeichnete DNS-Nutzlast verarbeiten muss, weist er hier erwartungsgemäß die höchsten Werte auf. Für alle anderen Knoten bewegen

sich die Werte in den selben Größenordnungen. Während der Messungen trat als höchstes Datenvolumen in der Summe 188 MByte auf, was anteilig durch die Knoten verarbeitet wurde. Schließt man von diesem Datenvolumen und der Testdauer von 24 h auf die durch P-DONAS erzeugte zusätzliche Verkehrslast, ist diese mit unter 7,8 MByte/h oder 2,2 KByte/s festzustellen. Die Verkehrslast ist tatsächlich noch geringer, da bei dieser Betrachtung von einem Knoten gesendete und von einem anderen Knoten empfangene Pakete doppelt gezählt wurden.

Wie die Leerlauftests gezeigt haben, ist eine Grundlast im Kad-Netz von 12,1 MByte dabei unvermeidbar, was im dargelegten Fall 6 % ausmacht.

Füllstände der Warteschlangen: Für eine Übersicht über alle implementierten Warteschlangen sei zunächst auf Abbildung 5.35 verwiesen. Der Füllstande der Wartschlangen für Pakete von/zu der vertrauenswürdigen Instanz wurden nicht untersucht, da während der Tests keine DNS Update (Delete)-Pakete versandt wurden. Die Füllstände der Warteschlangen für Pakete vom und zum Client bzw. übergeordneten Nameserver (Laptop_H54) sind in Abbildung 5.49 dargestellt.

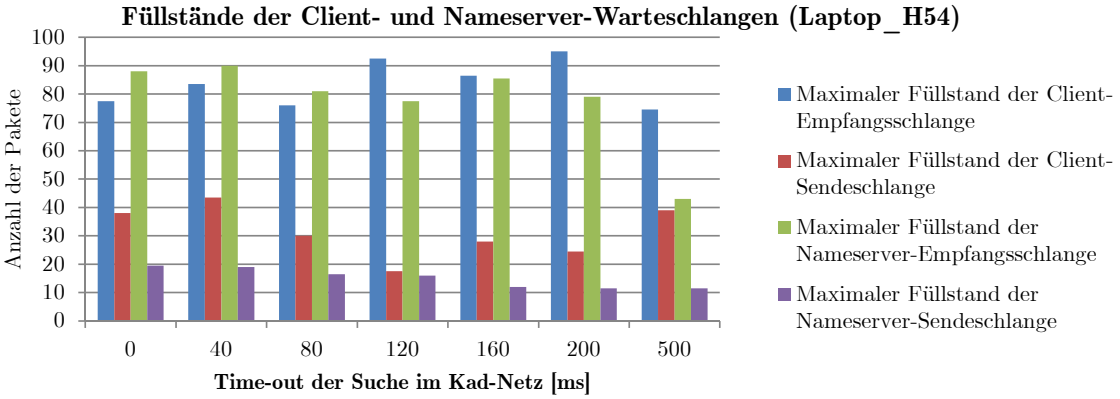


Abbildung 5.49.: Maximale Füllstände der Client- und Nameserver-Warteschlangen (Laptop_H54) über dem Time-out-Wert.

Die Füllstände der Warteschlangen für Pakete vom bzw. zum Kad-Netzwerk (alle Knoten) sind in Abbildung 5.50 bzw. in Abbildung 5.51 ersichtlich.

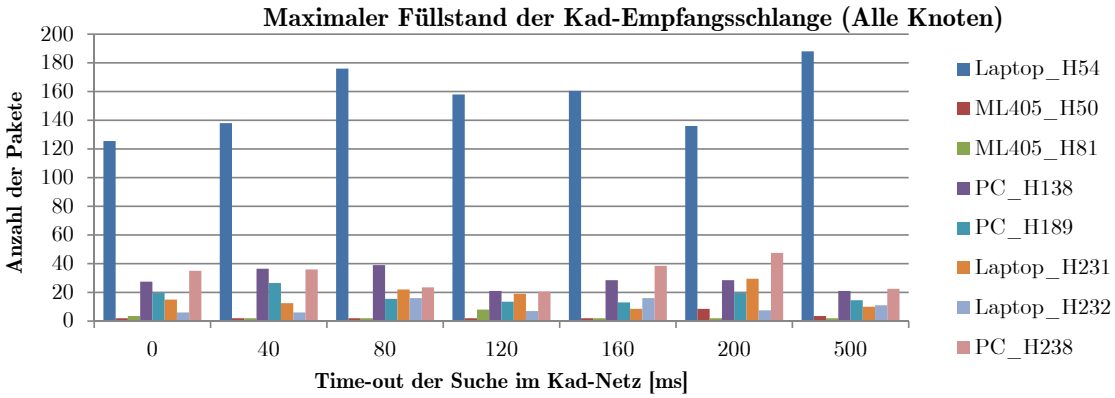


Abbildung 5.50.: Maximale Füllstände der Kad-Empfangs-Warteschlangen (alle Knoten) über dem Time-out-Wert.

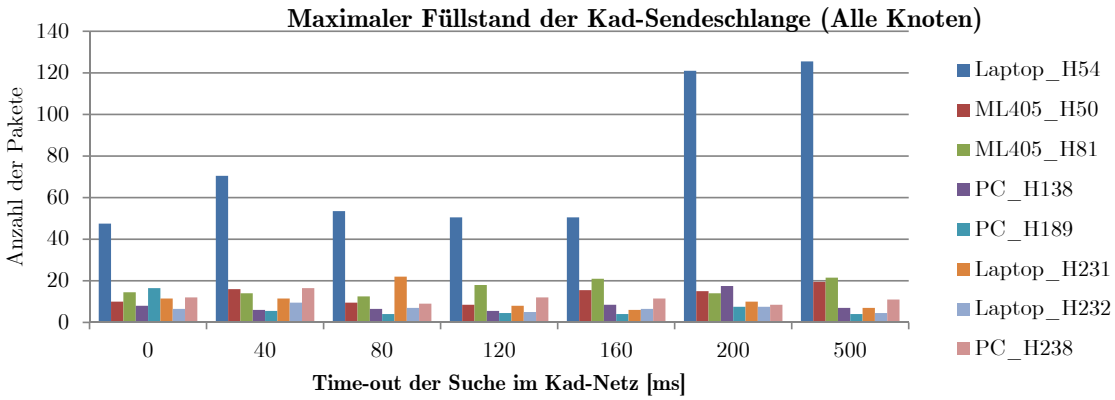


Abbildung 5.51.: Maximale Füllstände der Kad-Sende-Warteschlangen (alle Knoten) über dem Time-out-Wert.

Alle Empfangs- und Sendewarteschlangen bleiben mit einem maximalen Füllstand von 188 Paketen deutlich unter der verfügbaren Kapazität von 32.768 Paketen pro Warteschlange. Dies bedeutet, dass die Kapazitäten der Warteschlangen deutlich herabgesetzt werden können, ohne dass es zu Paketverlusten kommt.

CPU-, Speicherauslastung und Anzahl der Threads: Die durchschnittliche CPU-Auslastung pro Minute (alle Knoten) ist in Abbildung 5.52 über dem Time-out-Wert dargestellt.

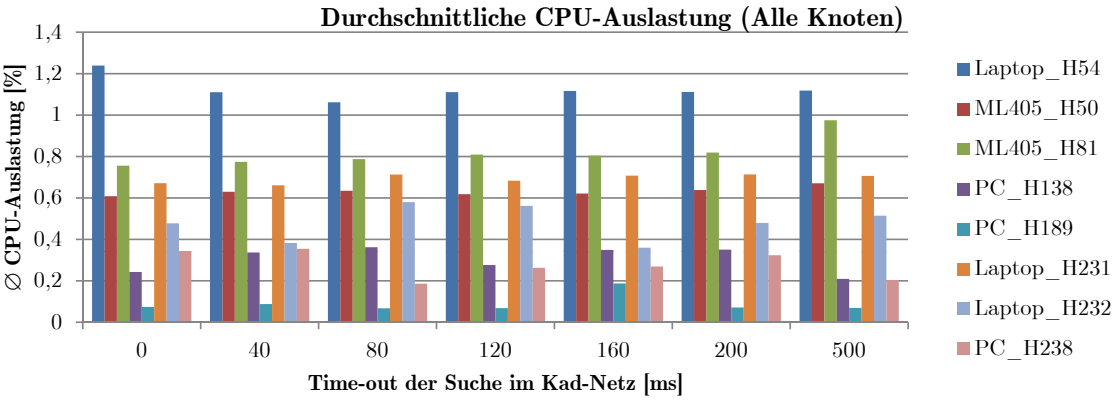


Abbildung 5.52.: Durchschnittliche CPU-Auslastung pro Minute (alle Knoten) über dem Time-out-Wert.

Abbildung 5.53 zeigt die durchschnittliche Speicherauslastung pro Minute (alle Knoten).

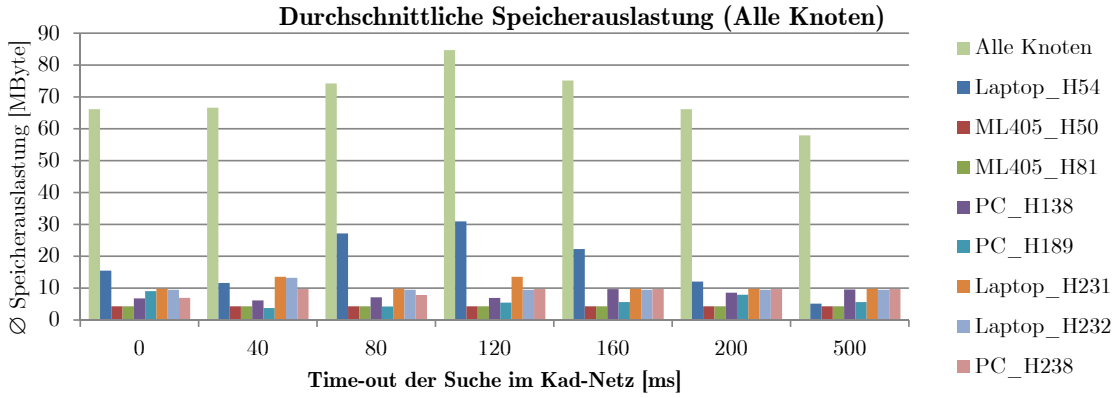


Abbildung 5.53.: Durchschnittliche Speicherauslastung pro Minute (alle Knoten) über dem Time-out-Wert.

Verglichen mit der gemessenen CPU-Auslastung (0,019 %) des DNS-Servers des Instituts MD weist P-DONAS ähnlich niedrige Werte auf. Die CPU-Auslastung ist auf dem Log-Knoten erwartungsgemäß am höchsten, steigt aber nie über 1,24 %. Auf allen anderen Knoten liegt sie immer unter 1 %. Über alle Knoten aufsummiert ist die Speicheraus-

lastung mit maximal 84,7 MByte zudem deutlich unter der des DNS-Nameservers des Instituts MD in Höhe von 167,4 MByte und reduziert sich auf 50,6 % demgegenüber.

Auf dem Log-Knoten (Laptop_H54) waren während der Messungen bis zu 303 Threads aktiv; durchschnittlich belief sich der Wert pro Minute auf 221. Die mittlere Anzahl der Threads pro Minute für alle anderen Knoten beträgt wie während des Leerlauftests 220; maximal arbeiten 221 Threads pro Knoten. Trotz der hohen Anzahl an Threads zeigen sich bezüglich CPU- und Speicherauslastung keine hohen Werte. Somit ist die mit Hilfe der Threads erreichte parallele Paketverarbeitung sinnvoll und trägt zu niedrigen Antwortlatenzen bei.

Tests mit verkleinertem Cache: Wie oben dargelegt, ist die Größe des Caches mit maximal 200 Einträgen für die aufgezeichnete DNS-Nutzlast angemessen dimensioniert. Um allerdings den Einfluss des Kad-Netzwerks hervorzuheben, würde der Cache auf ein Viertel seiner ursprünglichen Größe reduziert, so dass er Platz für maximal 50 Einträge bietet. Während der Messungen stellte sich dadurch ein Durchschnittswert von 47 Cache-Einträgen ein. Als Time-out-Wert für die Suche im Kad-Netz wurde fest 160 ms eingestellt, da aus Abbildung 5.43 hervorgeht, dass sich ab diesem Wert die Anzahl der Kad-Antworten nicht mehr maßgeblich erhöht. Die Antwortlatenz ist hier mit einem Wert von 66 ms nur geringfügig höher als die des zentralen Nameservers des Instituts MD mit 45 ms. Diese Erhöhung um 21 ms beeinflusst die Wahrnehmung eines Nutzers nicht, wenn er auf eine Antwort auf DNS-Requests wartet [Fis09].

Als Ergebnisse für die Antwortlatenz ergab sich mit verkleinertem Cache ein Wert von 60 ms, der lediglich 15 ms über dem Wert des Nameservers des Instituts MD liegt. Weitere interessante Ergebnisse sind der Anteil der Cache- (24 %), Knoten- (18 %), Kad- (11 %) und Nameserver-Antworten (47 %) auf verarbeitbare DNS-Requests. Insgesamt beläuft sich der Anteil an Antworten, die das P-DONAS-System auf verarbeitbare DNS-Requests selbst geben kann, somit auf 53 %. Alle weiteren untersuchten Daten verhalten sich entsprechend den Ergebnissen der Tests mit variierendem Time-out-Wert und viermal so großem Cache.

Der Anteil der Kad-Antworten ist erwartungsgemäß bei reduzierter Cache-Größe auf 11 % angestiegen. Abhängig von der DNS-Nutzlast und ihrer Zusammensetzung ist ein höherer Anteil möglich. So ist für häufig angefragte, lange Domainname, die nicht in den

Cache des angefragten Knotens passen, eine weitere deutliche Steigerung anzunehmen.

Zusammenfassung der Ergebnisse: Eine Zusammenfassung der wichtigsten Ergebnisse ist in Tabelle 5.9 ersichtlich. Für variierende Time-out-Werte für die Suche im Kad-Netz sind die Antwortlatenz von P-DONAS sowie der Anteil aller Antworten bzw. Kad-Antworten auf verarbeitbare DNS-Requests dargestellt. Für den Time-out-Wert 160 ms sind die Ergebnisse für einen Cache mit reduzierter Größe ebenfalls angegeben.

TIME-OUT-WERT $[ms]$	0	40	80	120	**160**	200	500
Antwortlatenz $[ms]$	45	47	51	52	**66/60***	79	135
Antworten-Anteil [%]	45	46	50	53	**53/53***	54	54
Davon Kad-Anteil [%]	0	2	10	11	**15/19***	15	15

* Kleiner Cache

Alle unmarkierten Werte wurden mit einem großen Cache aufgenommen.

Tabelle 5.9.: Zusammenfassung der wichtigsten Ergebnisse. Für die untersuchten Time-out-Werte für die Suche im Kad-Netz sind die Antwortlatenz und der Anteil aller Antworten bzw. der Kad-Antworten von P-DONAS auf verarbeitbare DNS-Requests ersichtlich.

Unter Berücksichtigung der besten Werte für Antwortlatenz, Antworten-Anteil insgesamt und Anteil der Kad-Antworten ist der Time-out-Wert 160 ms optimal für die aufgezeichnete DNS-Nutzlast. Die entsprechenden Werte sind in Tabelle 5.9 fett hervorgehoben.

5.3.7. Zusammenfassung und Ausblick

In diesem Abschnitt wurde ein P2P-basiertes DNS namens P-DONAS vorgestellt, das zusätzliche durch den Netzbetreiber bereitgestellte DNS-Nameserver-Farmen ergänzen und potentiell auch ersetzen kann. Dadurch werden Kosteneinsparungen für Netzbetreiber ermöglicht. Mit Hilfe des DHT-basierten P2P-Netzwerks Kad werden Zugangsknoten zu einem logischen P2P-Overlay auf der existierenden Topologie verbunden. Da Zugangsknoten vertrauenswürdige Elemente des TZN eines Netzbetreibers sind, kann hohes Vertrauen in das P2P-basiertes DNS als Grundvoraussetzung für die Annahme des neuen

Dienstes sichergestellt werden. Zugangsknoten tragen mit ihren verfügbaren Speicher- und Rechenressourcen zu dem verteilten DNS-Dienst bei. Da hohe Skalierbarkeit und Widerstandsfähigkeit intrinsische Eigenschaften von Kad sind, verfügt P-DONAS ohne zusätzliche Kosten über diese Vorteile. Darüber hinaus gewährleistet P-DONAS volle Kompatibilität mit dem traditionellen DNS, so dass P-DONAS ohne Schwierigkeiten mit der existierenden DNS-Infrastruktur funktioniert.

Als Machbarkeitsbeweis wurde ein Software-Prototyp sowohl unter Windows als auch auf einem Xilinx ML405-Board umgesetzt, der einen Zugangsknoten mit P-DONAS-Funktionalität emuliert. Der Prototyp ist hinsichtlich Rechen-, Speicher- und Hardware-Ressourcenverbrauch ohne Weiteres auf einem durchschnittlichen Zugangsknoten lauffähig. Dieser Prototyp wird zukünftig um eine Hardware-Komponente erweitert werden, um Antworten aus dem Cache in Höchstgeschwindigkeit geben zu können. So können die Antwortzeiten auch für eine größere Anzahl von Knoten als im Testaufbau in einem größeren Kad-Netz gering gehalten werden.

Mit dem entwickelten P-DONAS-Prototypen wurde bewiesen, dass mit Hilfe eines P2P-basierten DNS bestehende DNS-Nameserver-Farmen um bis zu 53 % entlastet werden können, wenn es als Ergänzung eingesetzt wird. Die Entlastung ist bezogen auf DNS-Datenverkehr, den der P-DONAS-Prototyp verarbeiten kann. Eine Ergänzung um bisher nicht verarbeitbare DNS-Pakete ist ohne Weiteres für den industriellen Einsatz möglich. Ein Großteil der Antworten kann zudem bereits mit Hilfe eines relativ kleinen Cache-Speichers auf dem angefragten Knoten gegeben werden.

Die Prozessorauslastung auf den Knoten des Testsystems bewegt sich in der gleichen Größenordnung wie auf einem klassischen DNS-Nameserver. Hinsichtlich der Antwortzeiten (Latenzen) ist festzustellen, dass das entwickelte Testsystem Antwortzeiten liefert, die mit denen des klassischen DNS-Nameservers vergleichbar sind und zu keinen spürbaren Wartezeiten für den Nutzer führen.

Zusammengefasst stellt P-DONAS einen Schritt in Richtung einer dezentralisierten DNS-Lösung dar, die einen zuverlässigen DNS-Dienst im TZN anbietet. P-DONAS kann das traditionelle DNS sowohl ergänzen als auch ersetzen.

5.4. Kapitelzusammenfassung und Ausblick

Dieses Kapitel widmet sich der P2P-Netzwerktechnologie und studiert Einsatzmöglichkeiten in TZN. Es wird untersucht, inwiefern mit geeigneten Mitteln Kosten der Netzbetreiber reduziert werden können, wenn P2P-Technologien im TZN eingesetzt werden und Zugangsknoten mit verfügbaren Speicher- und Rechenressourcen selber Teilnehmer eines P2P-Netzes werden. Auf diese Weise können vorhandene Internet-Dienste, die Schwachstellen besitzen, ersetzt bzw. komplementiert werden.

Wie die in diesem Kapitel dargelegten Resultate beweisen, konnten diese Dienste verbessert und um neue Funktionalitäten erweitert werden. Kostenintensive zusätzliche Infrastruktur kann somit eingespart werden. Zudem wird die Energieeffizienz der vorhandenen Zugangsknoten durch eine bessere Ausnutzung gesteigert. Als Ergebnisse wurden voll funktionsfähige Prototypen für eine P2P-basierte Speicherplattform, ein verteiltes Rechensystem sowie ein P2P-basiertes DNS entwickelt.

Der beschriebene Ansatz skalierbarer IT- und Web-Dienste wird oft auch als „Cloud Computing“ umschrieben [SG11, Kar12a]. Oftmals wird dieser Begriff einzig für Dienste auf Abrechnungsbasis verstanden: Dienste der *Public* Cloud wie Speicher- und Rechenkapazität werden dem Nutzer ohne menschliches Eingreifen zur Verfügung gestellt. Sogenannte *Private* Clouds bilden hier eine Ausnahme. Sie „bezeichnen unternehmensinterne Rechenzentren, die der Öffentlichkeit nicht über ein verbrauchsabhängiges Abrechnungsmodell zugänglich gemacht werden“ [BLRK09]. Generell weisen Public und Private Clouds technisch gesehen den gleichen Aufbau auf. Bei Private Clouds fehlen jedoch die Geschäftsmodelle.

Für die Speicherung von Daten, die in Unternehmen hohen Sicherheitsanforderungen unterliegen, sind unternehmensexterne Lösungen nicht geeignet [Kar12a]. Daher besteht nach wie vor ein hoher Bedarf an unternehmenseigenen Rechen- und Speicherkapazitäten, die die gewünschten Sicherheitsanforderungen erfüllen können. Auf dem gleichen Abstraktionsniveau wie Public Clouds könnten Private Clouds diesem Bedarf gerecht werden, im Gegensatz zu Public Clouds aber den schärferen Sicherheitsbedürfnissen Rechnung tragen. IT-Infrastruktur mitsamt ihren verfügbaren Ressourcen wird so unabhängig von Art, Örtlichkeit und Leistungsvermögen zur Verfügung gestellt. Mit der Dienste-Virtualisierung gehen Kosteneinsparungen einher, da für deren Ausführung nicht nur zusätzlich angeschaffte Hardware eingesetzt werden könnte, sondern auch verfügbare,

derzeit nicht nutzbare Ressourcen.

Netzbetreiber könnten so beispielsweise überschüssige Ressourcen im TZN in Form eines P2P-basierten Netzwerkes sammeln und diese als Dienst dem Unternehmen zur Verfügung stellen. Das Netzwerk aus Zugangsknoten würde somit eine Private Cloud darstellen, die Dienste zur Verfügung stellt und die notwendige Datenbasis mit garantierter Datenverfügbarkeit vorhält [Kar12a]. In diesem Zusammenhang sind die Ansätze dieser Forschungsarbeit nutzbar, um erhebliche Speicher- und Rechenkapazitäten zu mobilisieren, die über Schnittstellen virtualisiert angeboten werden könnten. Wenn in Zukunft Cloud-Ressourcen mittels etablierter Standards angefragt werden können, ist eine Kombination der beiden Prinzipien Cloud Computing und P2P vorstellbar [Pin12].

Kapitel 6.

Zusammenfassung

Die vorliegende Forschungsarbeit widmet sich der Berücksichtigung netzwerktechnischer Nähe in P2P-Netzen und studiert Einsatzmöglichkeiten von P2P-Netzwerktechnologie in TZN.

6.1. Berücksichtigung netzwerktechnischer Nähe in P2P-Netzen

In den letzten Jahren haben verschiedene P2P-Applikationen wie z. B. das File-Sharing im Bereich der privaten Internet-Nutzung eine weite Verbreitung erlangt. P2P-Datenverkehr nimmt daher einen großen Teil des gesamten Internetdatenverkehrs ein. Deshalb wurde zunächst untersucht, inwiefern gezieltes P2P-Routing hilft, diese P2P-Datenmassen besser zu bewältigen und die Netzwerkinfrastruktur zu entlasten. Denn obwohl P2P-Nutzer eine lukrative Einkommensquelle für Netzbetreiber sind, stellen die großen P2P-Datenmengen eine große Herausforderung bezüglich der Lenkung dieses Datenverkehrs (Traffic Engineering) dar. Da P2P-Routing üblicherweise die zu Grunde liegende physikalische Netzwerktopologie nicht berücksichtigt, werden herkömmliche Mechanismen von Netzbetreibern für das Traffic Engineering außer Kraft gesetzt. Das bedeutet, dass zwei auf dem P2P-Overlay benachbarte Peers, die Daten austauschen, netzwerktechnisch weit voneinander entfernt sein können und viele Netzwerkressourcen verbrauchen. Es wurde ein Algorithmus entwickelt, der die Peer-Auswahl in dem P2P-Netzwerk BitTorrent durch Hop Count-Nutzung verbessert.

Eine Auswertung des neuen BitTorrent-Algorithmus stellt dabei deutlich seine Vorteile

gegenüber dem Standard-Algorithmus sowohl für den Nutzer als auch für Netzbetreiber heraus. Die Last im Kernnetz der Netzbetreiber wird um bis zu 17 % reduziert. Der Datenverkehr wird dabei örtlicher eingegrenzt, was sich in einer Reduktion der Anzahl der Hops um bis zu 2,3 % widerspiegelt. Darüber hinaus erhöht sich die QoE von BitTorrent-Nutzern, da die Zeit, bis der letzte Nutzer seinen Download beendet hat, um bis zu 14 % abnimmt. Indem der Datenverkehr aus dem Kernnetz in den Randbereich des Internets verlagert wird, kann es hier zu Paketverwürfen kommen. Allerdings ist dieser Effekt tolerierbar, denn zum einen leidet die QoE der Nutzer nicht und zum anderen liegt das Hauptaugenmerk auf einer verringerten Last im Kernnetz, um Bandbreite für Datenverkehr freizumachen, der durch andere Applikationen verursacht wird.

6.2. Entwurf und Realisierung P2P-basierter Netzwerkinfrastruktur

Für die Realisierung effizienter dezentralisierter Netzwerkinfrastrukturen für die Verteilung und Verarbeitung von Daten bietet P2P-Technologie eine exzellente technische Basis. Es wurde deshalb untersucht, inwiefern mit geeigneten Mitteln Kosten der Netzbetreiber reduziert werden können, wenn P2P-Technologien im TZN eingesetzt werden und Zugangsknoten mit verfügbaren Speicher- und Rechenressourcen selber Teilnehmer eines P2P-Netzes werden. Auf diese Weise können vorhandene Internet-Dienste, die Schwachstellen besitzen, ersetzt bzw. komplementiert werden.

Zunächst war hierfür die Auswahl eines geeigneten P2P-Protokolls essentiell. Das DHT-basierte Kademlia-Netzwerk bzw. seine Implementierungsvariante Kad bietet diesbezüglich die besten Voraussetzungen und wurde deshalb ausgewählt. Da bei Netzbetreiberübergreifender Kommunikation von Zugangsknoten ggf. netzbetreiberseitige Firewalls und NAT Kommunikationshemmnisse darstellen, wurde ein erweitertes Firewall- bzw. NAT-Erkennungsverfahren für das Kad-Netzwerk entwickelt. Um die Anforderung eindeutiger Suchanfragen gewährleisten können, wurde zudem ein Verfahren für die dynamische Einstellung der Suchtoleranz im Kad-Netzwerk zur Laufzeit entwickelt und seine Gültigkeit bewiesen.

Als Ergebnis der Untersuchungen wurden voll funktionsfähige Prototypen für eine P2P-basierte verteilte Speicher- (PSP) und Rechenplattform (DuDE) für Netzbetreiber und

ein P2P-basiertes Domain Name System (P-DONAS) vorgestellt:

- Die P2P-basierte Speicherplattform namens PSP macht die Nutzung von in seiner Verfügbar- und Wiederbeschreibbarkeit limitiertem Flash-Speicher für die Speicherung von Sitzungsdaten überflüssig. Stattdessen erfolgt der Gebrauch von verfügbarer RAM-Speicher- und Rechenkapazität von Zugangsknoten.
- DuDE ist ein P2P-basiertes System für die verteilte Statistikberechnung für Netzbetreiber, das heutige Engpässe bei der Statistikerstellung behebt. DuDE ermöglicht umfassende globale Langzeitstatistiken, mit deren Hilfe Schwachstellen im Netzwerk besser identifiziert und behoben werden können. Infolge dessen kann eine höhere Zuverlässigkeit und Dienstgüte der Netze gewährleistet werden.
- Das P2P-basierte DNS namens P-DONAS macht zusätzliche durch den Netzbetreiber bereitgestellte DNS-Nameserver-Farmen überflüssig. Es stellt eine vertrauenswürdige, ressourcensparende, hoch skalierbare und widerstandsfähige Lösung dar, die kompatibel zum traditionellen DNS ist.

Wie die Resultate dieser Forschungsarbeit beweisen, konnten diese ausgewählten Internet-Dienste verbessert und um neue Funktionalitäten erweitert werden. Kostenintensive zusätzliche Infrastruktur kann durch die Nutzung vorhandener Ressourcen eingespart werden. Zudem wird die Energieeffizienz der vorhandenen Zugangsknoten durch eine bessere Ausnutzung gesteigert.

6.3. Ausblick

Eine Verbreitung des vorgestellten Ansatzes für die verbesserte Peer-Auswahl in BitTorrent-Netzwerken über den akademischen Bereich hinaus ist nicht ausgeschlossen. So ist eine Weiterentwicklung für den Einsatz in der Internetdienstleister-Branche vorstellbar.

Die in diesem Forschungsbereich gewonnenen wissenschaftlichen Kenntnisse im Bereich der P2P-Technologie können auf Grund ihres generischen Charakters auf weitere Forschungs- und Anwendungsbereiche übertragen werden. Der als P2P-Netz organisierte Verbund aus Zugangsknoten mitsamt ihren überschüssigen Ressourcen könnte von Netzbetreibern Unternehmen als Dienst angeboten werden. In solch einer Private Cloud könnten mit Hilfe der Ansätze dieser Forschungsarbeit große Speicher- und Rechenkapazitäten

gebündelt und über Schnittstellen zur Verfügung gestellt werden [Kar12a]. Standardisierungsbestrebungen könnten zu einer Vereinheitlichung der Schnittstellen führen, so dass perspektivisch eine genormte Abfrage von Cloud-Ressourcen möglich wird [Pin12]. Somit ist eine Kombination von Cloud Computing- und P2P-Prinzipien zukünftig denkbar.

P2P-Techniken kommen zudem für die Unterstützung bei der Übertragung von Multimediadiensten wie z. B. IPTV zum Einsatz. Vielfach ist hierbei die Dienstgüte allerdings noch nicht ausreichend [Sim12]. Deshalb sollte eine mögliche Qualitätsverbesserung für P2P-basiertes IPTV untersucht werden, indem eine Entlastung von Zugangsnetzen durch Bandbreitenersparnis im Upload vorgenommen wird [Zie12]. Dazu melden Peers den Zugangsknoten im Zugangsnetz, dass diese P2P-basierten IPTV-Verkehr abfangen sollen und an den nächsten Peer weiterleiten. Zu untersuchen sind Auswirkungen bezüglich Bandbreitenersparnis im Upload sowie einer Verringerung der Latenz.

Anhang A.

DHCP-Pakete und -Protokoll

A.1. Aufbau von DHCP-Paketen

In Abbildung A.1 ist der Aufbau eines DHCP-Pakets ersichtlich [JT07]. Opcode be-

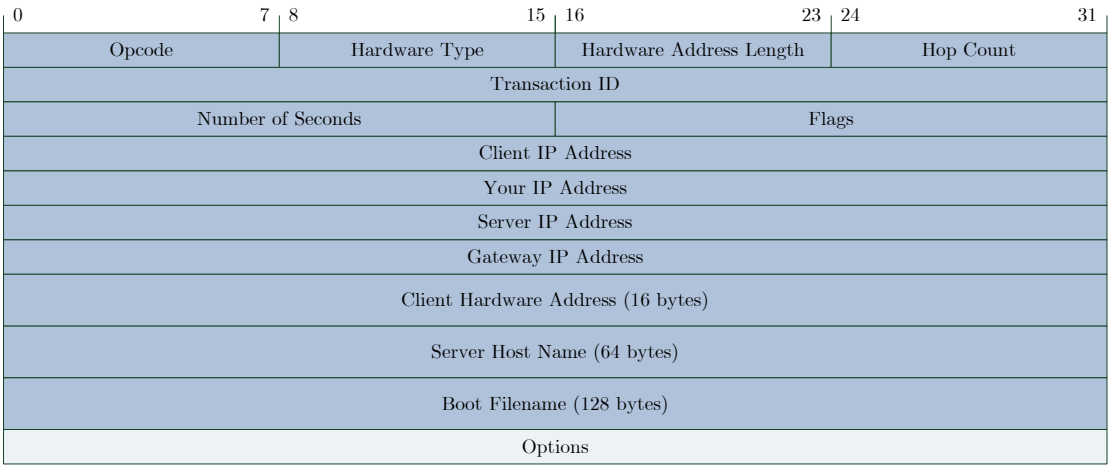


Abbildung A.1.: Aufbau eines DHCP-Pakets.

schreibt die Art des DHCP-Pakets: 1 steht für Boot Request und 2 für Boot Reply. Hardware Type beschreibt das verwendete Übertragungsprotokoll und nimmt für Ethernet z. B. den Wert 1 an. Hardware Address Length gibt die Länge der Client-Hardwareadresse an. Das Feld Hop Count wird durch Relay Agents genutzt. Die Transaction ID dient zum Matchen von Requests und Replies. Number of Seconds enthält die vergangene Zeit in

Sekunden seit Beginn des Adressbezugs bzw. der Releaseerneuerung durch den Client. Das Feld Flags enthält ein Broadcast-Bit, das den Wert 0 für Unicast und 1 für Broadcast annimmt. Client IP Address umfasst die IP-Adresse des anfragenden Clients. Your IP Address gibt Auskunft über die vom Server an den Client angebotene IP-Adresse. Server IP Address stellt die IP-Adresse des nächsten DHCP-Servers dar. Gateway IP Address informiert über die IP-Adresse des DHCP-Relay Agents. Client Hardware Address enthält die Client-Hardwareadresse wie z. B. die MAC-Adresse bei Ethernet. Server Host Name ist ein optionaler Server-Hostname und Boot Filename gibt den Namen der Datei an, die der Client zum Booten verwenden soll. Das Feld Options definiert über den DHCP Message Type die Art des DHCP-Pakets und enthält weiterhin z. B. die vergebene „Lease Time“ einer IP-Adresse.

A.2. Ablauf des DHCP-Protokolls

Das DHCP-Protokoll umfasst sieben mögliche Nachrichten (siehe Tabelle A.1), die über den DHCP Message Type im Options-Feld angegeben werden. DHCP Discover ist ein

DHCP Message Type	Name der Nachricht
1	DHCP Discover
2	DHCP Offer
3	DHCP Request
4	DHCP Decline
5	DHCP Ack
6	DHCP Nak
7	DHCP Release

Tabelle A.1.: Nachrichten des DHCP-Protokolls.

Client-Broadcast, der der Suche nach Servern dient. Ein DHCP Offer dient als Offerte des Servers an Client mit einer vorgeschlagenen Konfiguration. Das DHCP Request stellt die Annahme oder Ablehnung der Offerte durch den Client dar. Das DHCP Ack ist die positive Bestätigung durch den Server; DHCP Nak wiederum die negative Bestätigung. Ein DHCP Decline wird als Fehlermeldung durch den Client genutzt, dass die zuge-

wiesene Adresse bereits in Gebrauch ist. Mit Hilfe eines DHCP Release gibt der Client die zugewiesene Adresse frei. Über ein DHCP Inform verlangt der Client nur sekundäre Konfigurationsparameter, da seine IP-Adresse extern konfiguriert ist.

Der Ablauf des DHCP-Protokolls gliedert üblicherweise sich in sechs Phasen und ist in Abbildung A.2 ersichtlich [JT07]:

- In der *Anforderungsphase* schickt ein Client ein DHCP Discover per IP-Broadcast an alle Server im gleichen Subnetz. Das DHCP Discover enthält die MAC-Adresse des Clients. Jeder kontaktierte DHCP-Server entscheidet, ob er für die Anfrage zuständig ist. DHCP-Relay Agents leiten Anfragen an DHCP-Server weiter.
- Während der *Angebotsphase* bekommt ein Client von allen kontaktierten DHCP-Servern ein DHCP Offer. Zunächst reserviert ein Server dafür eine IP-Adresse und stellt Konfigurationsdaten wie „Lease Time“, Subnetzmaske usw. zusammen. Dann antwortet ein Server mittels IP-Broadcast und der anfragende Client erkennt die Antwort an seiner MAC-Adresse.
- In der sich anschließenden *Auswahlphase* wählt der Client ein Angebot z. B. anhand der „Lease Time“ aus und antwortet per DHCP Request als Broadcast an alle Server. In diesem DHCP Request ist die IP-Adresse des ausgewählten Servers als Option enthalten. Somit weiß der ausgewählter Server, dass er gewählt wurde. Für alle anderen bedeutet dies eine Absage.
- Die *Bestätigungsphase* dient dem ausgewählten Server dazu, dem Client ein DHCP Ack mit Quelladresse=Server-IP-Adresse und Zieladresse=IP-Broadcast zu schicken. Der Client übernimmt IP–Adresse und Konfigurationsdaten wie z. B. die IP „Lease Time“. Zuvor soll/kann der Client die Eindeutigkeit der IP-Adresse überprüfen.
- Die Phase der *Releaseerneuerung* beginnt nach halb abgelaufener „Lease Time“. Dabei schickt der Client ein DHCP Request an den Server und dieser schickt ein DHCP Ack, um die „Lease Time“ zu verlängern. Wenn die „Lease Time“ nicht verlängert wird, löst der Client ein DHCP Discover aus.
- In der *Beendigungsphase* gibt ein Client seine IP-Adresse frei, indem er ein DHCP Release an den Server schickt. Danach steht diese IP-Adresse wieder zur Verfügung.

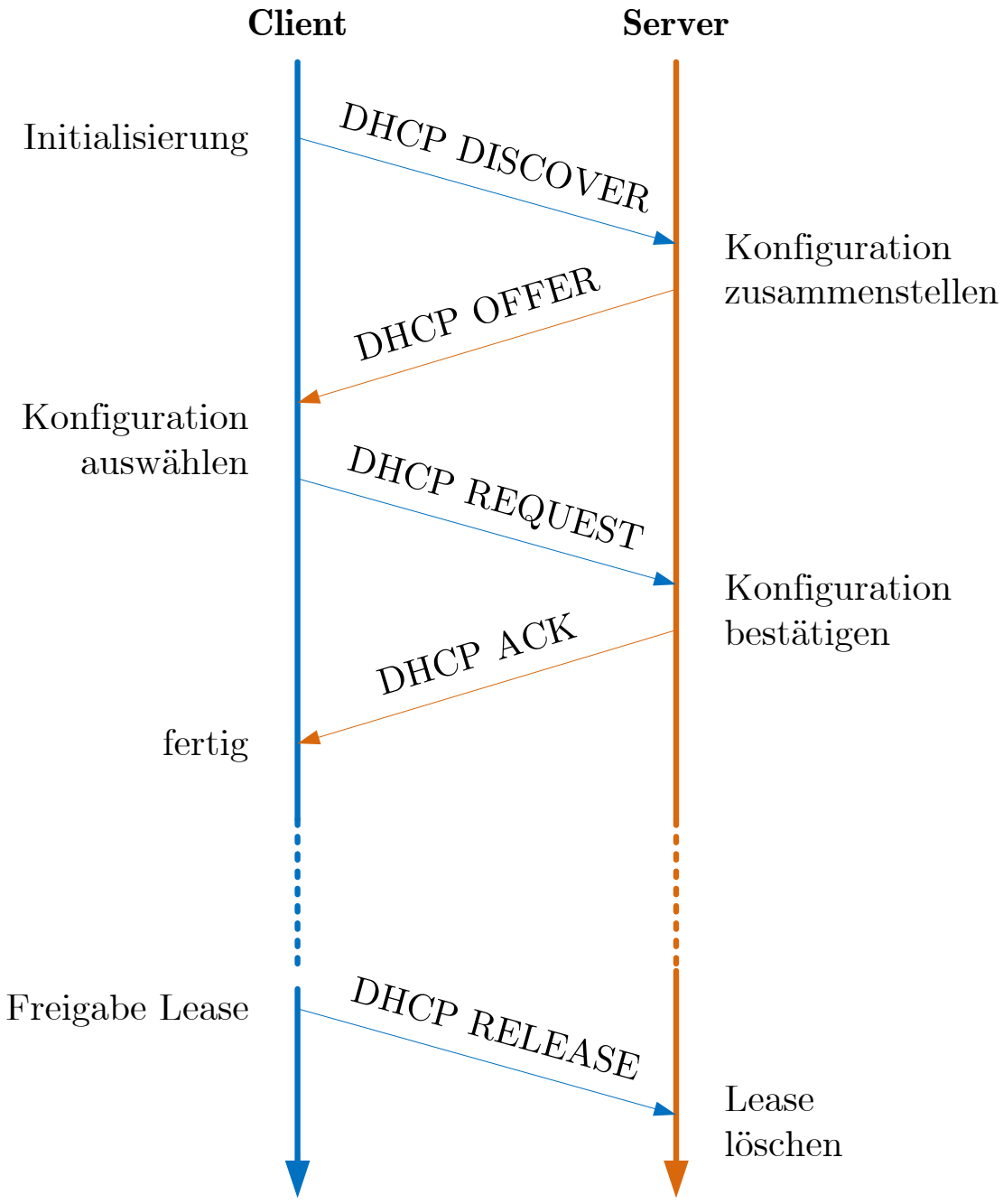


Abbildung A.2.: Ablauf des DHCP-Protokolls.

Anhang B.

DNS-Pakete und -Protokoll

B.1. Allgemeiner Aufbau

Der Aufbau von DNS-Paketen (außer DNS-Updates) ist in Abbildung B.1 dargestellt [Moc87]. Das Identification-Feld dient dazu, Request- und Reply-Pakete anhand einer

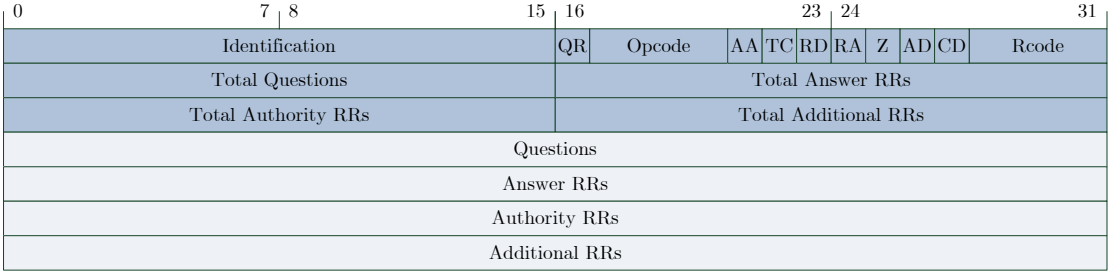


Abbildung B.1.: Aufbau eines DNS-Pakets (außer DNS-Updates) [Moc87].

16 Bit-ID zu matchen. Das QR-Feld gibt an, ob es sich um ein DNS-Query oder ein DNS-Response handelt (0 = Query, 1 = Response). Das Opcode-Feld legt die Art des DNS-Pakets fest (siehe Tabelle B.1).

Das Flag AA ist '1', wenn der antwortende Nameserver Autorität für den angefragten Domainnamen ist, sonst '0'. TC = '1' zeigt an, dass nur die ersten 512 Byte der Antwort geliefert wurden und der Rest abgeschnitten wurde. RD = '1' weist einen Nameserver an, Queries rekursiv zu betreiben. Rekursives Query ist optional. RA = '1' dient als Anzeige dafür, dass rekursives Query unterstützt wird. '0' zeigt demzufolge eine Nichtverfügbarkeit der Rekursion an. Das Flag Z ist reserviert. AD gibt an, ob die Daten

Opcode	Beschreibung	Referenzen
0	QUERY, Standard Query	[Gib07]
1	IQUERY, Inverse Query	[Gib07]
2	STATUS, Server Status Request	[Gib07, Law02]
3	Reserviert	
4	NOTIFY, Standard Query	[Vix96]
5	UPDATE	[Vix97]
6-15	Reserviert	

Tabelle B.1.: Opcode-Feld von DNS-Paketen.

verifiziert wurden. CD steht für Checking Disabled. Der Rcode dient als Benachrichtigung, ob ein DNS-Request erfolgreich verarbeitet werden konnte und im Falle eines Misserfolgs als Klassifikation des Fehlers. Total Questions gibt die Anzahl der Einträge in der Questions-Liste an. Total Answer RRs bezieht sich auf die Anzahl der Einträge in der Answer RRs-Liste. Total Authority RRs bezeichnet die Anzahl der Einträge in der Authority RR-Liste. Total Additional RRs steht für die Anzahl der Einträge in der Additional RR-Liste.

Eine Questions-Struktur (siehe Abbildung B.2) ist von variabler Länge und kann 0 oder mehr Query-Strukturen (siehe Abbildung B.3) enthalten. Eine Query-Struktur besteht

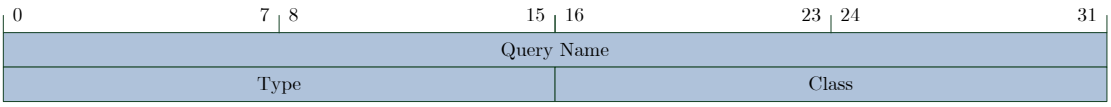


Abbildung B.2.: Aufbau einer Questions-Struktur.

aus mehreren Labels. Das erste Byte eines jeden Labels gibt die Länge dieses Labels an. Das letzte Byte einer Query-Struktur ist immer 0x00 und gibt damit das Ende einer Query-Struktur an. Ein Beispiel ist www.google.de. www ist Label 1 und davor steht ein Byte Length Label 1 = 3.

Ein RR hat den in Abbildung B.4 ersichtlichen Aufbau. Es hat eine variable Länge. Es können Answer RRs, Authority RRs oder Additional RRs im DNS-Paket enthalten sein.

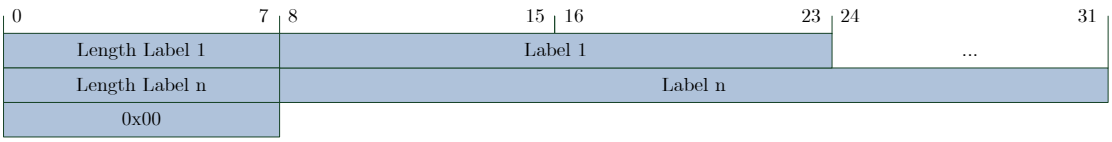


Abbildung B.3.: Aufbau einer Query-Struktur, die in einer Questions-Struktur enthalten sein kann.

Ein wichtiger Wert für das Type-Feld (2 Byte großer Bestandteil von Questions- und

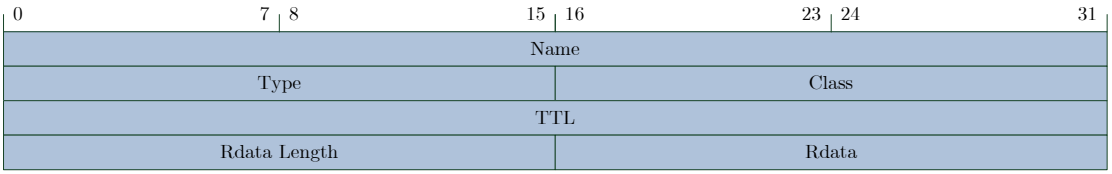


Abbildung B.4.: Aufbau eines RRs.

RR-Strukturen) ist z. B. Type = 1. Questions-Strukturen dieses Typs stellen Anfragen nach IPv4-Adressen dar; RR-Strukturen dieses Typs sind A RRs.

Der übliche Wert für das Class-Feld (2 Byte großer Bestandteil von Questions- und RR-Strukturen) ist Class = 1. Dieser Wert bedeutet Internet-Klasse.

Das DNS-Protokoll nutzt eine Form der Nachrichtenkomprimierung. Diese wird genutzt, um keinen Domainnamen wiederholen zu müssen. Statt der Wiederholung eines Domainnamens wird die in Abbildung B.5 dargestellte Struktur verwendet. Offset gibt den

Abbildung B.5.: Struktur zur DNS-Nachrichtenkomprimierung.

Byte-Offset von Beginn der DNS-Nachricht an (d.h. vom 1. Byte von DNS Identification an). An dieser Stelle steht der einzusetzende Domainname.

B.2. Aufbau von DNS-Updates

Mit Hilfe eines DNS-Updates können RRs oder RRsets (Menge von RRs) in einem angegebenen Bereich (Zone) hinzugefügt oder gelöscht werden [Vix97]. Es können separat Voraussetzungen angegeben werden, die Abhängigkeiten vom Vorhandensein oder Nicht-Vorhandensein von RRsets/einzelnen RRs für DNS-Update-Operationen definieren. Wenn diese Voraussetzungen nicht zutreffen, wird kein DNS-Update durchgeführt. Ein DNS-Update hat im Gegensatz zum in Abbildung B.1 dargestellten Aufbau einen modifizierten Header (siehe Abbildung B.6) und enthält andere Daten. Der Opcode eines

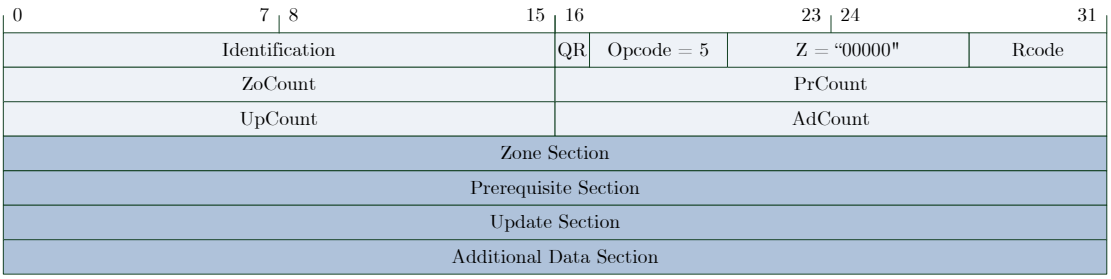


Abbildung B.6.: Aufbau eines DNS-Updates [Vix97].

DNS-Updates ist 5. Z ist für zukünftige Zwecke reserviert und sollte den Wert Null annehmen. ZoCount gibt die Anzahl der RRs in der Zone Section an. PrCount bezeichnet die Anzahl der RRs in der Update Section. In der Update Section befindet sich die mit UpCount angegebene Anzahl von RRs. In der Additional Data Section sind AdCount viele RRs vorhanden. Die Zone Section (siehe Abbildung B.7) hat den gleichen Aufbau wie eine Query-Struktur (siehe Abbildung B.3). Allerdings wurden die Felder in ZName,

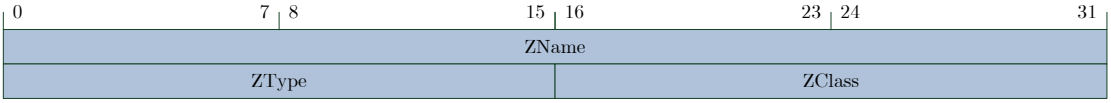


Abbildung B.7.: Aufbau einer Zone Section.

ZType und ZClass umbenannt. Die Zone Section wird von einem DNS-Update genutzt, um den Bereich der RRs festzulegen, die zu aktualisieren sind. Alle RRs, die zu aktua-

lisieren sind, müssen im gleichen Bereich sein. Deshalb darf die Zone Section genau ein RR vom ZType = SOA enthalten. ZClass bezeichnet die Klasse des Bereichs. Die Prerequisite Section enthält eine Menge von RRset-Voraussetzungen, die vom Primary Master Server (existiert nur einmal pro Zone) zum Zeitpunkt der Ankunft des DNS-Updates erfüllt werden müssen. Der Aufbau der Prerequisite Section ist derselbe wie bei einem RR (siehe Abbildung B.4).Die Prerequisite Section kann fünf verschiedene Bedeutungen haben:

1. RRset existiert (Wertunabhängig): Mindestens ein RR mit angegebenem Name und Type (in dem durch dem Zone Section festgelegten Bereich und Klasse) muss existieren.
2. RRset existiert (Wertabhängig): Eine Menge von RRs mit angegebenem Name und Type existiert und hat die gleichen Bestandteile mit den gleichen Daten wie das RRset, das in dieser Section angegeben ist.
3. RRset existiert nicht: Keine RRs mit angegebenem Name und Type (in dem durch dem Zone Section festgelegten Bereich und Klasse) existieren.
4. Name ist in Gebrauch: Mindestens ein RR mit angegebenem Name (in der durch dem Zone Section festgelegten Bereich und Klasse) muss existieren. Diese Voraussetzung wird nicht durch leere Nichtterminalsymbole erfüllt.
5. Name ist nicht in Gebrauch: Kein RR irgendeines Typs mit angegebenem Name existiert. Diese Voraussetzung wird durch leere Nichtterminalsymbole erfüllt.

Die Update Section enthält RRs, die zu einem Bereich hinzugefügt oder von diesem gelöscht werden sollen. Der Aufbau der Update Section ist derselbe wie der eines RRs (siehe Abbildung B.4). Die Update Section kann vier verschiedene Bedeutungen haben:

1. Füge RRs zu einem RRset hinzu: RRs mit angegebenem Name, Type, TTL, Rdata Length und Rdata und der gleichen Class wie die in der Zone Section definierte ZClass werden hinzugefügt.
2. Lösche ein RRset: RRs mit angegebenem Name und Type löschen (Ausnahme: Name = ZName -> Alle RRs außer mit Type = SOA oder NS löschen). Hierbei muss die TTL mit Null angegeben werden. Class muss auf Any gesetzt werden. Rdata Length ist Null und somit gibt es keine Rdata.
3. Lösche alle RRsets eines Namens: Alle RRsets mit angegebenem Name löschen

(Ausnahme: Name = ZName -> Alle RRs außer mit Type = SOA oder NS löschen). Type und Class müssen mit Any angegeben werden. Die TTL muss Null sein. Rdata Length ist Null und somit gibt es keine Rdata.

4. Lösche ein RR von einem RRset: RRs mit angegebenem Name, Type, Rdata Length und Rdata löschen (Ausnahme: Name = ZName -> RR mit Type = SOA oder NS nicht löschen). TTL muss Null sein. Class ist auf None zu setzen, um es vom Hinzufügen eines RRs zu unterscheiden.

Die Additional Data Section enthält RRs, die sich auf das Update selber oder neue durch das Update hinzuzufügende RRs beziehen. Es kann immer nur ein DNS-Update zurzeit verarbeitet werden. Nach Abschluss aller Update-Operationen, wird ein DNS-Update Response generiert und an den Absender des DNS Updates an den entsprechenden UDP- bzw. TCP-Port zurückgeschickt. Das DNS Identification- und Opcode-Feld des DNS-Updates ist zu übernehmen. Das QR-Bit ist auf 1 für Response zu setzen. Die ZoCount-, PrCount-, UpCount-, und AdCount-Felder sowie die zugehörigen Sections können aus dem DNS-Update kopiert werden oder auf Null gesetzt werden. Im letzten Fall sind die Sections auch leer.

B.3. Durch P-DONAS unterstützte Teile des DNS-Protokolls

Pakettyp	#Total Questions/ ZoCount	#Total Answer RRs/ PrCount	#Total Authority RRs/ UpCount	#Additional RRs/ AdCount
DNS-Request	1*	0	0	0
DNS-Response	0 oder 1	Beliebig**	Beliebig	Beliebig
DNS-Update	1***	0	1****	0

Tabelle B.2.: Unterstützte Teile des DNS-Protokolls.

* Es werden nur Anfragen nach A RRs (Type=1) unterstützt.

** Von allen enthaltenen RRs werden nur A RRs (Type=1) ausgewertet und gespeichert.

*** In der Zone Section muss genau ein RR vom Typ SOA (Type=6) stehen.

**** Das zu speichernde RR muss vom Typ A RR (Type=1) sein.

Literaturverzeichnis

[ABFW04] Aggarwal, V. ; Bender, S. ; Feldmann, A. ; Wichmann, A.: Methodology for Estimating Network Distances of Gnutella Neighbors, GI Jahrestagung (2), 2004, S. 219–223

[Adc07] Adconion Media Group: *Joost - Free Online TV*. `http://www.joost.com/`. Version: 2007

[AH99] Al-Harbash, Thamer: *Raw IP Networking FAQ*. `http://www.ntua.gr/rin/rawfaq.html`. Version: 1999

[BBP97] Bailey, David ; Borwein, Peter ; Plouffe, Simon: ON THE RAPID COMPUTATION OF VARIOUS POLYLOGARITHMIC CONSTANTS. In: *Mathematics of Computation* Bd. 66, 1997, S. 903–913

[BD10] Bahls, Thomas ; Duchow, Daniel: *P2P overlay network for administrative services in a digital network*. `http://www.freepatentsonline.com/EP2148493A1.html`. Version: Januar 2010

[BHK07] Baumgart, Ingmar ; Heep, Bernhard ; Krause, Stephan: OverSim: A Flexible Overlay Network Simulation Framework. In: *Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007*, 2007, S. 79–84

[BL07] Baker, Mark ; Lakhoo, Rahim: Peer-to-Peer Simulators / ACET, University of Reading. 2007. – Forschungsbericht

[BLRK09] Böhm, Markus ; Leimeistet, Stefanie ; Riedl, Christoph ; Krcmar, Helmut: Cloud Computing: Outsourcing 2.0 oder ein neues GeschäftsmodelI zur BereitstelIung von IT-Ressourcen? In: *IM - Die Fachzeitschrift für Information Management und Consulting* 2 (2009), Mai, S. 5–14

[Bru06] Brunner, René: *A Performance Evaluation of the Kad-Protocol*, University of Mannheim, Deutschland, Master-Arbeit, November 2006

[C. 08] C. Morariu et al.: DIPStorage: Distributed Storage of IP Flow Records, 16th IEEE LANMAN, 2008, S. 108–113

[CAI04] CAIDA: *Nameserver DoS Attack October 2002.* `http://www.caida.org/projects/dns/dns-root-gtld/oct02dos.xml`. Version: 2004

[CBR+08] Castella, D. ; Barri, I. ; Rius, J. ; Gine, F. ; Solsona, F. ; Guirado, F.: CoDiP2P: A Peer-to-Peer Architecture for Sharing Computing Resources. In: *Advances in Soft Computing* Bd. 50, 2008, S. 293–303

[CCF10] Cholez, Thibault ; Chrisment, Isabelle ; Festor, Olivier: Monitoring and Controlling Content Access in KAD. In: *IEEE ICC*, 2010, S. 1–6

[CDHR02] Castro, M. ; Druschel, P. ; Hu, Y. ; Rowstron, A.: Exploiting Network Proximity in Distributed Hash Tables. In: *International Workshop on Future Directions in Distributed Computing (FuDiCo)*, 2002, S. 52–55

[CMM02] Cox, Russ ; Muthitacharoen, Athicha ; Morris, Robert T.: Serving DNS using a Peer-to-Peer Lookup Service, IPTPS, 2002, S. 155–165

[CN11] Carle, Prof. Dr.-Ing. G. ; Niedermayer, Dr. H.: *Peer-to-Peer Systems and Security.* Lehrveranstaltung, TU München, 2011

[Coh03] Cohen, Bram: Incentives Build Robustness in BitTorrent, First Workshop on the Economics of Peer-to-Peer Systems, Juni 2003

[Dig82] Digital, Xerox Intel: *The Ethernet - A Local Area Network: Data Link Layer and Physical Layer Specifications.* Spezifikation, Version 2.0, 1982

[DR01] Druschel, P. ; Rowstron, A.: PAST: a large-scale, persistent peer-to-peer storage utility. In: *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems, 2001*, 2001, S. 75–80

[Eas97] Eastlake, D.: *Domain Name System Security Extensions.* RFC 2065, Januar 1997

[EHBK07] Eger, K. ; Hossfeld, T. ; Binzenhofer, A. ; Kunzmann, G.: Efficient Simulation of Large-Scale P2P Networks: Packet-level vs. Flow-level Simulations, UPGRADE-CN'07, 2007, S. 9–16

[Eve90] Everhart, C.: *New DNS RR Definitions.* RFC 1183, Oktober 1990

[Exc11] Exchange, AMS-IX Amsterdam I.: *Historical Traffic Data.* Oktober 2011

[FG00] Fujii, K. ; Goto, S.: Correlation between Hop Count and Packet Transfer Time, APAN/IWS, 2000

[Fis09] Fischer, Mario: *Website Boosting 2.0: Suchmaschinen-Optimierung, Usability, Webseiten-Marketing.* Mitp, Redline, 2009

[For06] Forbes, Dennis: *Interesting Facts About Domain Names.* `http://blog.yafla.com/Interesting_Facts_About_Domain_Names/`. Version: März 2006

[Fos95] Foster, Ian: *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering.* Addison Wesley Pub Co Inc, 1995. – 27–57 S.

[Fre11] Freescale Semiconductor: *PowerQUICC II Pro (83xx) Communications Processors.* Integrated Processor Hardware Specifications. `http://www.freescale.com/webapp/sps/site/taxonomy.jsp?code=PCPPCMPC83XX`. Version: Mai 2011

[FSK05] Ford, Bryan ; Srisuresh, Pyda ; Kegel, Dan: Peer-to-Peer Communication Across Network Address Translators. In: *Proceedings of the annual conference on USENIX Annual Technical Conference*, 2005, S. 179–192

[FV08] Fall, Kevin ; Varadhan, Kannan: *The ns Manual (The VINT Project).* `http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf`. Version: 2008

[Gib07] Gibbard, Steve: Geographic Implications of DNS Infrastructure Distribution. In: *The Internet Protocol Journal* Bd. 10, 2007

[GKXS08] Graffi, Kalman ; Kovacevic, Aleksandra ; Xiao, Song ; Steinmetz, Ralf: SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems, ICPADS, 2008, S. 279–286

[GLL08] Gladisch, A. ; Lange, C. ; ; Leppla, R.: Power Efficiency of Optical Versus Electronic Access Networks. In: *ECOC*, 2008, S. 1–4

[gol12] golem.de: *Bittorrent startet Dropbox-Konkurrenten.* `http://www.golem.de/1201/88846.html`. Version: Januar 2012

[GPM+04] García, Pedro ; Pairot, Carles ; Mondéjar, Rubén ; Pujol, Jordi ; Tejedor, Helio ; Rallo, Robert: PlanetSim: A New Overlay Network Simulation Framework. In: *Proceedings of the 19th IEEE International Conference on Automated Software Engineering (ASE). Workshop on Software Engineering and Middleware (SEM)*, 2004, S. 123–136

[GSS06] Gupta, Rohit ; Sekhri, Varun ; Somani, Arun K.: CompuP2P: An Architecture for Internet Computing Using Peer-to-Peer Networks. In: *IEEE Transactions on Parallel and Distributed Systems* 17 (2006), S. 1306–1320

[Hal06] Halsall, Fred: *Computer Networking and the Internet*. 5. Addison-Wesley, 2006

[HJS+03] Harvey, N. J. A. ; Jones, M. B. ; Saroiu, S. ; Theimer, M. ; Wolman, A.: SkipNet: A Scalable Overlay Network with Practical Locality Properties. In: *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems (USITS)*, 2003

[HP03] Huffman, W. C. ; Pless, V.: *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2003

[IET09] IETF: *Application-Layer Traffic Optimization (ALTO)*. `http://www.ietf.org/html.charters/alto-charter.html`. Version: 2009

[Int11] Internet World Stats: *INTERNET GROWTH STATISTICS* . `http://www.internetworldstats.com/emarketing.htm`. Version: Juni 2011

[ipo09] ipoque GmbH: *Internet Study 2008/2009*. 2009

[ISI81] Information Sciences Institute, University of Southern C.: *Internet Protocol Specification*. RFC 791, September 1981

[J. 00] J. Kubiatowicz et al.: OceanStore: An Architecture for Global-Scale Persistent Storage, ASPLOS, 2000, S. 190–201

[Jai92] Jain, R.: A Comparison of Hashing Schemes for Address Lookup in Computer Networks. In: *IEEE Transactions on Communications* 40 (1992), Nr. 10, S. 1570–1573

[JT07] Javvin Technologies, Inc.: *Network Protocols Handbook*. 4. Javvin Technologies, Inc., 2007

[KD10] Koutitas, George ; Demestichas, Panagiotis: A Review of Energy Efficiency in Telecommunication Networks. In: *Inter Telecommunication Forum TELFOR* 2 (2010), Nr. 1, S. 2–7

[Law02] Lawrence, D.: *Obsoleting IQUERY*. RFC 3425, November 2002

[LCC+10] Liu, Bo ; Cao, Yanchuan ; Cui, Yi ; Lu, Yansheng ; Xue, Yuan: Locality Analysis of BitTorrent-Like Peer-to-Peer Systems, 7th IEEE CCNC, 2010, S. 1–5

[LFM11] Liang, Zhichao ; Fan, Yulei ; Meng, Xiaofeng: A novel method to extend flash memory lifetime in flash-based DBMS. (2011), S. 190–201

[LLX+04] Liu, Y. ; Liu, X. ; Xiao, L. ; L.M.Ni ; Zhang, X.: Location-Aware Topology Matching in P2P Systems, INFOCOM, 2004, S. 2220–2230

[LMH+08] Leonardi, E. ; Mellia, M. ; Horvath, A. ; Muscariello, L. ; Niccolini, S. ; Rossi, D.: Building a cooperative P2P-TV application over a wise network: The approach of the European FP-7 strep NAPA-WINE, IEEE Communications Magazine 46 (4), 2008, S. 20+22

[M. 07] M. Karnstedt et al.: UniStore: Querying a DHT-based Universal Storage, International Conference on Data Engineering, 2007, S. 1503–1504

[Men05] Mennecke, T.: DSL Broadband Providers Perform Balancing Act. (2005)

[MJ09] Montresor, Alberto ; Jelasity, Márk: PeerSim: A Scalable P2P Simulator. In: *Proceedings of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, 2009, S. 99–100

[MM02] Maymounkov, P. ; Mazieres, D.: Kademlia: A peer-to-peer information system based on the xor metric, IPTPS, 2002

[Moc87] Mockapetris, P.: *Domain Names - Implementation and Specification*. RFC 1035, November 1987

[MZ05] Merugu, S. ; Zegura, E.: Adding Structure to Unstructured Peer-to-Peer Networks: The Use of Small-World Graphs, JPDC, 2005, S. 142–153

[NET09] NETCRAFT: *August 2009 Web Server Survey*. `http://news.netcraft.com/archives/web_server_survey.html`. Version: August 2009

[Nin11] Ninnemann, Matthias: *Freescale Semiconductor PowerQUICC II Pro - Performance und Auslastung.* Nokia Siemens Networks GmbH & Co. KG, Broadband Access Division, November 2011

[NL08] Niu, Di ; Li, Baochun: Circumventing server bottlenecks: Indirect large-scale P2P data collection, ICDCS, 2008, S. 61–68

[PD97] Peterson, Larry L. ; Davie, Bruce S.: *Computer networks: a systems approach.* Morgan Kaufmann Publishers, Inc., 1997

[PKAC08] Proudfoot, Ryan ; Kent, Kenneth ; Aubanel, Eric ; Chen, Nan: Flexible Software-Hardware Network Intrusion Detection System. In: *19th IEEE/IFIP International Symposium on Rapid System Prototyping*, 2008, S. 182–188

[PMTZ06] Pappas, V. ; Massey, D. ; Terzis, A. ; Zhang, L.: A Comparative Study of the DNS Design with DHT-Based Alternatives. In: *25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, 2006, S. 1–13

[Qin04] Qin Xin et al.: Availability in Global Peer-To-Peer Storage Systems, Distributed Data and Structures, Proceedings in Informatics, 2004

[RA10] Ribeiro, H.B. ; Anceaume, E.: DataCube: A P2P Persistent Data Storage Architecture Based on Hybrid Redundancy Schema. In: *18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, 2010, S. 302–306

[RD01] Rowstron, A. ; Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: *IFIP/ACM International Conference on Distributed Systems Platforms*, 2001, S. 329–350

[Riv92] Rivest, R.: *The MD5 Message-Digest Algorithm.* RFC 1321. `http://tools.ietf.org/html/rfc1321`. Version: April 1992

[RS04] Ramasubramanian, Venugopalan ; Sirer, Emin G.: The Design and Implementation of a Next Generation Name Service for the Internet, ACM SIGCOMM, 2004, S. 331–342

[RSR06] Rasti, A. ; Stutzbach, D. ; Rejaie, R.: On the Long-term Evolution of the Two-Tier Gnutella Overlay, INFOCOM, 2006

[RWHM03] Rosenberg, J. ; Weinberger, J. ; Huitema, C. ; Mahy, R.: *STUN - simple traversal of user datagram protocol (UDP) through network address translators (NATs)*. RFC 3489, March 2003

[san11a] sandvine: *Global Internet Phenomena Report, Fall 2011*. `http://www.sandvine.com/news/global_broadband_trends.asp`. Version: September 2011

[san11b] sandvine: *Global Internet Phenomena Report, Spring 2011*. `http://www.sandvine.com/news/global_broadband_trends.asp`. Version: März 2011

[SBS10] Saïd Business School, Oxford U.: *Broadband Quality Study 2010*. `http://bcserdon.com/wp-content/uploads/2010/11/broadband-quality-study-20_10_2010-english-version.pdf`. Version: Oktober 2010

[Sch06] Schindelhauer, Prof. C.: *Peer-to-Peer Netzwerke*. Universität Freiburg, 2006

[SENB07] Steiner, Moritz ; En-Najjary, Taoufik ; Biersack, Ernst W.: A Global View of KAD. In: *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007, S. 117–122

[SFK8q] Srisuresh, P. ; Ford, B. ; Kegel, D.: *State of Peer-to-Peer (P2P) Communication across Network Address Translators (NATs)*. RFC 5128, März 2008q

[SG11] Srinivasan, Savitha ; Getov, Vladimir: Navigating the Cloud Computing Landscape - Technologies, Services, and Adopters. In: *IEEE Computer* 44 (2011), Nr. 3, S. 22–23

[SGR+11] Stingl, Dominik ; Gross, Christian ; Rueckert, Julius ; Nobach, Leonhard ; Kovacevic, Aleksandra ; Steinmetz, Ralf: PeerfactSim.KOM: A Simulation Framework for Peer-to-Peer Systems. In: *Proceedings of the 2011 International Conference on High Performance Computing & Simulation (HPCS 2011)*, 2011, S. 577–584

[Sim12] Simonite, Tom: *Live-Berichte aus dem P2P-Netz*. `http://www.heise.de/tr/artikel/Live-Berichte-aus-dem-P2P-Netz-1444318.html`. Version: Februar 2012

[SK11] Song, Yiting ; Koyanagi, Keiichi: Study on a hybrid P2P based DNS. In: *IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, 2011, S. 152–155

[SR06a] Stutzbach, Daniel ; Rejaie, Reza: Improving Lookup Performance Over a Widely-Deployed DHT. In: *INFOCOM*, 2006, S. 1–12

[SR06b] Stutzbach, Daniel ; Rejaie, Reza: Understanding Churn in Peer-to-Peer Networks, ACM SIGCOMM Internet Measurement Conference, 2006, S. 189–202

[SW05] Steinmetz, Ralf ; Wehrle, Klaus: *P2P Systems and Applications, Springer Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg, 2005

[Swi02] Swiss Education & Research Network (SWITCH): *Default TTL Values in TCP/IP*. 2002

[Tan03] Tanenbaum, Andrew S.: *Computer Networks*. 4. PH PTR, Pearson Education Internat., 2003

[TDM11] Toka, L. ; Dell'Amico, M. ; Michiardi, P.: Data transfer scheduling for P2P storage. In: *IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2011, S. 132–141

[Tel12] Telefónica Online Services: *Das international flächendeckende Netz der Telefónica*. `http://www.telefonica-online-services.de/Rechenzentren-Netz/Telefonica-Backbone`. Version: 2012

[The09] TheoryOrg: *Bittorrent Protocol Specification v1.0*. `http://wiki.theory.org/BitTorrentSpecification`. Version: 2009

[ver08] *The Network Simulator ns-2: Validation Tests*. `http://www.isi.edu/nsnam/ns/ns-tests.html`. Version: 2008

[Vix96] Vixie, P.: *A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)*. RFC 1996, August 1996

[Vix97] Vixie, P.: *Dynamic Updates in the Domain Name System (DNS UPDATE)*. RFC 2136, April 1997

[Vix00] Vixie, P.: *Secret Key Transaction Authentication for DNS (TSIG)*. RFC 2845, Mai 2000

[VNRS02] Verbeke, Jerome ; Nadgir, Neelakanth ; Ruetsch, Greg ; Sharapov, Ilya: Framework for Peer-to-Peer Distributed Computing in a Heterogeneous, Decentralized Environment. In: *In Proceedings of the 3rd International Workshop on Grid Computing*, Springer-Verlag, 2002, S. 1–12

[WD05] Wu, Shaomei ; Du, Thihui: GlobalStat: A statistics service for diverse data collaboration and integration in grid, HPC Asia, 2005, S. 594–599

[Wid08] Widiger, Harald: *Paketverarbeitende Systeme - Algorithmen und Architekturen für hohe Verarbeitungsgeschwindigkeiten*, Universität Rostock, Doktorarbeit, 2008

[WIH05] Wan, H. ; Ishikawa, N. ; Hjelm, J.: Autonomous Topology Optimization for Unstructured Peer-to-Peer Networks, ICPADS, 2005, S. 488–494

[wir10] wired.co.uk/: *Peter Sunde starts peer-to-peer DNS system.* `http://www.wired.co.uk/news/archive/2010-12/02/peter-sunde-p2p-dns`. Version: Dezember 2010

[W.K04] W.K. Lin et al.: Erasure Code Replication Revisited, IEEE P2P, 2004, S. 90–97

[XIL10] XILINX: *ML405 Documentation.* `http://www.xilinx.com/support/documentation/ml405.htm`. Version: 2010

[XIL11a] XILINX: *JTAGPPC Controller (v2.01c).* `http://www.xilinx.com/support/documentation/ip_documentation/jtagppc_cntlr.pdf`. Version: 2011

[XIL11b] XILINX: *LightWeight IP (lwIP) Application Examples.* `http://www.xilinx.com/support/documentation/application_notes/xapp1026.pdf`. Version: 2011

[XIL11c] XILINX: *LogiCORE IP Multi-Port Memory Controller (MPMC) (v6.03.a).* `http://www.xilinx.com/support/documentation/ip_documentation/mpmc.pdf`. Version: 2011

[XIL12] XILINX: *CoreConnect Architecture - Processor Local Bus (PLB).* `http://www.xilinx.com/ipcenter/processor_central/coreconnect/coreconnect_plb.htm`. Version: 2012

[XN99] Xiao, X. ; Ni, L.: Internet QoS: A Big Picture, IEEE Network Magazine, 1999, S. 8–18

[XYKS08] Xie, H. ; Yang, Y. R. ; Krishnamurthy, A. ; Silberschatz, Y. L. A.: P4P: Provider Portal for Applications, ACM SIGCOMM, 2008, S. 351–362

[YLL09] Yu, Jie ; Lu, Liming ; Li, Zhoujun: KadCache: Employing Kad to Mitigate Flash Crowds and Application Layer DDoS Attacks Against Web Servers. In: *ACM SIGCOMM09 Poster Session* (2009), S. 8–9

[YYXT08] Ye, Yunqi ; Yen, I-Ling ; Xiao, Liangliang ; Thuraisingham, B.: Secure, Highly Available, and High Performance Peer-to-Peer Storage Systems. In: *11th IEEE High Assurance Systems Engineering Symposium*, 2008, S. 383–391

[Zat07] Zattoo Europa AG: *Zattoo - TV to Go.* `http://zattoo.com/`. Version: 2007

[ZDH+02] Zhao, Ben Y. ; Duan, Yitao ; Huang, Ling ; Joseph, Anthony D. ; Kubiatowicz, John D.: Brocade: Landmark Routing on Overlay Networks. In: *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002, S. 34–44

[Zim80] Zimmermann, Hubert: OSI Reference Model - The ISO Model of Architecture for Open System Interconnection. In: *IEEE Transactions on Communications* Bd. 28, 1980, S. 452–432

[ZSZ03] Zhang, Zheng ; Shi, Shu-Ming ; Zhu, Jing: SOMO: Self-Organized Metadata Overlay for Resource Management in P2P DHT. In: *Lecture Notes in Computer Science* 2735 (2003), S. 170–182

Liste der Veröffentlichungen und Fachvorträge auf Tagungen

[DGT+10] Danielis, Peter ; Gotzmann, Maik ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: A Peer-To-Peer-based Storage Platform for Storing Session Data in Internet Access Networks. In: *World Telecommunications Congress (WTC 2010)*, 2010. – ISBN 978–3–8007–3303–3, S. 5–10

[DKT07] Danielis, Peter ; Kubisch, Stephan ; Timmermann, Dirk: Realisierung und Implementierung eines Algorithmus zur Echtzeit-Mustererkennung in einem Ethernet-Datenstrom. In: *12. Symposium Maritime Elektrotechnik, Elektronik und Informationstechnik* IEF, 2007, S. 191–196

[DKW+08a] Danielis, Peter ; Kubisch, Stephan ; Widiger, Harald ; Schulz, Jens ; Duchow, Daniel ; Bahls, Thomas ; Timmermann, Dirk: A Conceptual Framework for Increasing Physical Proximity in Unstructured Peer-To-Peer Networks. In: *IEEE Sarnoff Symposium 2008*, 2008. – ISBN 978–1–4244–1843–5, S. 1–5

[DKW+08b] Danielis, Peter ; Kubisch, Stephan ; Widiger, Harald ; Schulz, Jens ; Duchow, Daniel ; Bahls, Thomas ; Timmermann, Dirk ; Lange, Christian: Trust-by-Wire in Packet-switched IP Networks: Calling Line Identification Presentation for IP (Hardware Prototype Demonstration). In: *Design, Automation and Test in Europe Conference and Exhibition (DATE 2008), University Booth* DATE, 2008

[DKW+08c] Danielis, Peter ; Kubisch, Stephan ; Widiger, Harald ; Schulz, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: IPclip - An Innovative Mechanism to Reestablish Trust-by-Wire in Packet-switched IP

Networks. In: *3. Essener Workshop Neue Herausforderungen in der Netzsicherheit (Prospective Challenges in Network Security)"*, 2008, S. 1–2

[DKW+09] Danielis, Peter ; Kubisch, Stephan ; Widiger, Harald ; Rohrbeck, Jens ; Altmann, Vladyslav ; Skodzik, Jan ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: Trust-by-Wire in Packet-Switched IPv6 Networks: Tools and FPGA Prototype for the IPclip System. In: *6th IEEE Consumer Communications and Networking Conference*, 2009. – ISBN 978–1–4244–2309–5, S. 1–2

[DSA+bm] Danielis, Peter ; Skodzik, Jan ; Altmann, Vlado ; Rohrbeck, Jens ; Wegner, Tim ; Timmermann, Dirk: P-DONAS: A P2P-based Domain Name System in Access Networks. In: *Journal of Internet Services and Applications (JISA)* (2012 subm.)

[DSL+10] Danielis, Peter ; Skodzik, Jan ; Lorenz, Carsten ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: P-DONAS: A Prototype for a P2P-based Domain Name System in Access Networks. In: *Design, Automation and Test in Europe Conference and Exhibition (DATE 2010), University Booth*, 2010

[DSR+11] Danielis, Peter ; Skodzik, Jan ; Rohrbeck, Jens ; Altmann, Vlado ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: Using Proximity Information between BitTorrent Peers: An Extensive Study of Effects on Internet Traffic Distribution. In: *International Journal on Advances in Systems and Measurements* 4 (2011), Nr. 3&4, S. 212–221. – ISSN 1942–261x

[DSTB11] Danielis, Peter ; Skodzik, Jan ; Timmermann, Dirk ; Bahls, Thomas: Impacts of Improved Peer Selection on Internet Traffic in BitTorrent Networks. In: *6th International Conference on Internet Monitoring and Protection (ICIMP)*, 2011. – ISBN 978–1–61208–004–8, S. 8–13

[DT10] Danielis, Peter ; Timmermann, Dirk: Use of Peer-To-Peer Technology in Internet Access Networks and its Impacts. In: *IPDPS 2010 PhD Forum*, 2010. – ISBN 978–1–4244–6533–0, S. 1–3

[KWD+07] Kubisch, Stephan ; Widiger, Harald ; Danielis, Peter ; Duchow, Daniel ; Bahls, Thomas ; Timmermann, Dirk ; Lange, Christian ; Röwer, Oliver: Configuration Tool and FPGA-Prototype of a Hardware Packet

Processing System. In: *Design, Automation and Test in Europe Conference and Exhibition (DATE 2007), University Booth* DATE, 2007

[KWD+08a] Kubisch, Stephan ; Widiger, Harald ; Danielis, Peter ; Schulz, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: Complementing E-Mails with Distinct, Geographic Location Information in Packet-switched IP Networks. In: *MIT 2008 Spam Conference*, 2008, S. 1–25

[KWD+08b] Kubisch, Stephan ; Widiger, Harald ; Danielis, Peter ; Schulz, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: Countering Phishing Threats with Trust-by-Wire in Packet-switched IP Networks - A Conceptual Framework. In: *22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS), 4th International Workshop on Security in Systems and Networks (SSN 2008)*, 2008. – ISBN 978–1–4244–1694–3, S. 1–8

[KWD+08c] Kubisch, Stephan ; Widiger, Harald ; Danielis, Peter ; Schulz, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: Trust-by-Wire in Packet-switched IP Networks: Calling Line Identification Presentation for IP. In: *1st ITU-T Kaleidoscope Conference: Innovations in Next Generation Networks - Future Network and Services*, 2008. – ISBN 92–61–12441–0, S. 375–382

[RAP+11] Rohrbeck, Jens ; Altmann, Vlado ; Pfeiffer, Stefan ; Danielis, Peter ; Skodzik, Jan ; Timmermann, Dirk ; Ninnemann, Matthias ; Rönnau, Maik: The Secure Access Node Project: A Hardware-Based Large-Scale Security Solution for Access Networks. In: *International Journal On Advances in Security* 4 (2011), Nr. 3&4, S. 234–244. – ISSN 1942–2636

[SDA+11a] Skodzik, Jan ; Danielis, Peter ; Altmann, Vlado ; Rohrbeck, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: DuDE: A Distributed Computing System using a Decentralized P2P Environment. In: *36th IEEE LCN, 4th International Workshop on Architectures, Services and Applications for the Next Generation Internet*, 2011. – ISBN 978–1–61284–927–0, S. 1060–1067

[SDA+11b] Skodzik, Jan ; Danielis, Peter ; Altmann, Vlado ; Rohrbeck, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: DuDE: A Proto-

type for a P2P-based Distributed Computing System. In: *36th IEEE LCN, 4th International Workshop on Architectures, Services and Applications for the Next Generation Internet*, 2011

[TDT11] Tockhorn, Andreas ; Danielis, Peter ; Timmermann, Dirk: A Configurable FPGA-Based Traffic Generator for High-Performance Tests of Packet Processing Systems. In: *6th International Conference on Internet Monitoring and Protection (ICIMP)*, 2011. – ISBN 978–1–61208–004–8, S. 14–19

[WKD+08] Widiger, Harald ; Kubisch, Stephan ; Danielis, Peter ; Schulz, Jens ; Timmermann, Dirk ; Bahls, Thomas ; Duchow, Daniel: IPclip: An Architecture to restore Trust-by-Wire in Packet-switched Networks. In: *33rd Annual IEEE Conference on Local Computer Networks (LCN)*, 2008. – ISBN 978–1–4244–2413–9, S. 312–319

Betreute studentische Arbeiten

[Ang09] Angierski, André: *Codes zur Fehlererkennung und -korrektur*, Universität Rostock, Projektseminar, 2009

[Beh11] Behboudi, Noushin: *Power-Aware Application Specific Instruction-Set Processors Design*, University of Teheran, Iran/Universität Rostock, Master-Arbeit, 2011

[Cel07] Celik, Ebru: *Commissioning, programming and connection of a PowerPC to a hardware module on an FPGA*, Universität Rostock, Praktikum, 2007

[Got09] Gotzmann, Maik: *Untersuchung und Implementierung von Erasure Resilient Codes für eine P2P-basierte Speicherplattform*, Universität Rostock, Bachelor-Arbeit, 2009

[Hag11] Haghi, Zahra N.: *QCA design and fault models*, University of Teheran, Iran/Universität Rostock, Master-Arbeit, 2011

[Kar12a] Karow, Johannes: *Peer-to-Peer-basierte Speicherlösungen im Kontext mit Cloud Computing*, Universität Rostock, Literaturarbeit, 2012

[Kar12b] Karow, Johannes: *Portierung und Evaluierung eines Peer-to-Peer-basierten Speichersystems für den Einsatz auf einem Xilinx Evaluation Board*, Universität Rostock, Bachelor-Arbeit, 2012

[Kru11] Kruse, Sebastian: *Lookup-Deterministik in DHT-basierten P2P-Netzwerken*, Universität Rostock, Literaturarbeit, 2011

[Kru12] Kruse, Sebastian: *Untersuchung und Implementierung einer dynamischen Suchtoleranz für das DHT-basierte P2P-Netzwerk Kademlia/Kad*, Universität Rostock, Bachelor-Arbeit, 2012

[Lip07] Lipowski, Tilo: *Struktur, Aufbau und Funktionalität des P2P-Netzwerkprotokolls FastTrack*, Universität Rostock, Kleiner Beleg, 2007

[Mun09] Munoz, Beatriz C.: *Analysis and Implementation of a Peer-to-Peer-based Storage Solution for Log Files and Statistics in Internet Access Networks*, Universität Rostock, Bachelor-Arbeit, 2009

[Obr09] Obradovic, Anja: *Design and Implementation of a Concept for a Dynamic Search Tolerance in a P2P-based DNS*, Universität Rostock, Praktikum, 2009

[Pas09] Passow, Peter: *Anpassung und Implementierung eines Konzepts zur Umgehung von Firewalls und NAT in einem P2P-basierten DNS*, Universität Rostock, Bachelor-Arbeit, 2009

[Pfe09] Pfefferkorn, Daniel: *Untersuchung und Implementierung des Peer-to-Peer-Netzwerks Kademlia sowie Anwendung in einem Peer-to-Peer-basierten Domain Name System*, Universität Rostock, Projektarbeit, 2009

[Pin12] Pingel, Jessica: *Cloud Computing in der Standardisierung*, Universität Rostock, Literaturarbeit, 2012

[Sam11] Samara, Hani: *Konzipierung und Implementierung eines Hardware-Caches für den beschleunigten Zugriff auf Speicherdaten*, Universität Rostock, Bachelor-Arbeit, 2011

[Sha11] Shabana, Mohammed: *Aktuelle Entwicklungen und Trends der Peer-to-Peer-Technologie*, Universität Rostock, Literaturarbeit, 2011

[Sko08] Skodzik, Jan: *Auswirkungen erhöhter netzwerktechnischer Nähe des Peer-to-Peer-Datenverkehrs am Beispiel von BitTorrent*, Universität Rostock, Bachelor-Arbeit, 2008

[Sko09] Skodzik, Jan: *Modifikation und Portierung eines Peer-to-Peer-basierten Domain Name Systems für den Einsatz auf einem Xilinx Evaluation Board*, Universität Rostock, Projektarbeit, 2009

[Sko10] Skodzik, Jan: *Entwurf eines Peer-to-Peer-basierten Systems zur verteilten Berechnung von Statistiken*, Universität Rostock, Master-Arbeit, 2010

[Weg08] Wegner, Tim: *Entwurf und Implementierung eines Peer-to-Peer-basierten Domain Name Systems*, Universität Rostock, Projektarbeit, 2008

[Wol11] Wolff, Johann-Peter: *Entwurf und Implementierung einer Softwarelösung zur Erfassung und Analyse von Logdaten*, Universität Rostock, Bachelor-Arbeit, 2011

[Zie12] Zierke, Robert: *Untersuchung und Bewertung Peer-to-Peer-basierter IPTV-Lösungen*, Universität Rostock, Literaturarbeit, 2012

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die Dissertation zum Thema

Peer-to-Peer-Technologie in Teilnehmerzugangsnetzen

vollkommen selbst verfasst habe und zu ihrer Anfertigung keine anderen als die angegebenen Hilfsmittel und Quellen verwendet habe.

Rostock, den 9. Oktober 2012

Thesen

1. Der hohe Anteil von P2P-Datenverkehr durch File-Sharing-Anwendungen erfordert die Lenkung dieser Datenmengen in geordnete Bahnen, um zu vermeiden, dass die Kernnetze der Netzbetreiber überflutet werden.
2. Da P2P-Daten unabhängig vom physikalischen Underlay auf dem logischen P2P-Overlay geroutet werden, müssen Maßnahmen ergriffen werden, das Routing auf dem logischen Overlay besser an das physikalische Underlay anzupassen.
3. Der neue Algorithmus für das BitTorrent-Netzwerk verbessert die Peer-Auswahl durch Hop Count-Nutzung. Ein Leistungsvergleich von Standard- und neuem BitTorrent-Algorithmus stellt die Vorteile sowohl für den Nutzer als auch für Netzbetreiber dar.
4. P2P stellt abgesehen von seinen illegalen Anwendungen wie dem Herunterladen von nicht lizenzierten Filmen vor allem aber ein innovatives Netzwerkparadigma dar. Es hat sich bereits für viele legale Anwendungen wie P2P-basierte IPTV-Lösungen wie Zattoo und Joost bewährt.
5. Zahlreiche Internet-Basisdienste wie DNS und ISP-basierte Dienstleistungen sowie Administrierungs- und Wartungsdienste weisen heutzutage schwerwiegende Probleme bezüglich Skalierbarkeit und Fehlertoleranz auf. Diese Dienste sind abhängig von alter zentralisierter Service-Infrastruktur, die nicht für die Dimension und Komplexität des heutigen und zukünftigen Internets ausgelegt ist. So muss kostenintensive neue Infrastruktur zur Verfügung gestellt werden, um die unzureichende Skalierbarkeit und Fehlertoleranz zu kompensieren.
6. Für die Realisierung effizienter dezentralisierter Netzwerkinfrastruktur bietet P2P-Technologie eine exzellente technische Basis. Hierfür ist die Auswahl eines geeigneten P2P-Protokolls essentiell.

7. Distributed Hash Table-basierte P2P-Systeme bieten den besten Kompromiss bezüglich Such- und Speicherkomplexität. Sie sind wenig fehleranfällig und weisen unter der Voraussetzung einer entsprechend eingestellten Suchtoleranz eine deterministische Suche auf, das heißt es kommt zu keinen False Negatives. Sie werden deshalb als P2P-Protokollform für den Entwurf P2P-basierter Netzwerkinfrastruktur ausgewählt.

8. Da das Distributed Hash Table-basierte Kademlia-Netzwerk insbesondere hinsichtlich der Suche eine geringe Komplexität und weitere Vorteile gegenüber ähnlich leistungsstarken Netzwerken aufweist, bietet es die besten Voraussetzungen für die Realisierung P2P-basierter Infrastrukturen und wird deshalb dafür ausgewählt. Im Speziellen fällt die Wahl auf die Kademlia-Implementierungsvariante Kad, die die komplette Kademlia-Funktionalität umsetzt.

9. Trotz aller Vorteile des Kad-Protokolls bedarf dieses insbesondere hinsichtlich der Behandlung von Firewalls und NAT sowie der Gewährleistung einer deterministischen Suche der Erweiterung, um die reibungslose Funktionalität des Netzwerks gewährleisten zu können.

10. Da bei Provider-übergreifender Kommunikation von Zugangsknoten ggf. Netzbetreiberseitige Firewalls und NAT Kommunikationshemmnisse darstellen, wurde ein erweitertes Erkennungsverfahren entwickelt. Wird so eine Firewall oder NAT festgestellt, kann der betroffene Knoten mit Hilfe eines sogenannten Buddys wieder vollständig an der Kommunikation im Kad-Netzwerk teilnehmen, sofern ein geeigneter Buddy existiert.

11. Das Kad-Protokoll muss die Anforderung eindeutiger Suchanfragen gewährleisten können, um einen wirklichen Ersatz zum Client-Server-Paradigma darzustellen. Die Deterministik einer Suche hängt bei Kad von der Suchtoleranz ab. Es wurde ein Verfahren für die dynamische Einstellung der Suchtoleranz im Kad-Netzwerk zur Laufzeit entwickelt und seine Gültigkeit bewiesen.

12. Zugangsknoten des Zugangsnetzwerks haben einen gewissen Anteil an verfügbaren Speicher- und Rechenressourcen, der ohne zusätzliche Kosten genutzt werden kann. Sie werden deshalb als Mitglieder des P2P-Netzwerks organisiert und durch das P2P-Netzwerk Kad verbunden. Jeder Zugangsknoten trägt dabei einen Teil seiner RAM-Speicher- und Rechenkapazität zum Kad-Netzwerk bei.

13. Zugangsknoten verbrauchen einen bedeutenden Teil der in Telekommunikationsnetzwerken benötigten Energie. Bei der üblichen Nutzung wird Energie verschwendet. Indem aber ungenutzte Ressourcen der Zugangsknoten für neue Aufgaben eingesetzt werden, wird die Energieeffizienz erhöht. An anderer Stelle können zudem Netzwerkkomponenten eingespart werden, so dass weitere Energie eingespart werden kann.

14. ISP müssen die Sitzungsdaten ihrer Kunden speichern, da diese zu Betriebs-, Verwaltungs- und Kontrollzwecken benötigt werden. Um die Persistenz der Sitzungsdaten sicherzustellen, wird heutzutage Flash-Speicher eingesetzt. Allerdings ist dieser in seiner Verfügbarkeit und Wiederbeschreibbarkeit limitiert und wird außerdem für andere Zwecke benötigt.

15. Es wurde die P2P-basierte Speicherplattform PSP für die Speicherung von Sitzungsdaten in Internet-Zugangsnetzwerken konzipiert und implementiert, um die verfügbaren flüchtigen RAM-Speicher- sowie Rechenkapazitäten von Zugangsknoten zu nutzen und Sitzungsdaten mit Hilfe von Reed-Solomon-Codes mit gleich hoher Verfügbarkeit wie Flash-Speicher vorzuhalten.

16. Statistiken sind für ISP von grundlegender Bedeutung, um die Dienstgüte zu verbessern, Flaschenhälse in ihren Netzwerken und Attacken auf Netzwerkkomponenten zu erkennen. Existierende zentralisierte Rechensysteme sind nicht in der Lage, die ständig anwachsenden Log-Datenmengen zu verarbeiten, um daraus Statistiken zu erzeugen. Zudem kann lediglich ein Standardsatz an Statistiken erstellt werden.

17. Es wurde ein verteiltes Rechensystem auf der Grundlage einer verteilten P2P-Umgebung (DuDE) entworfen und umgesetzt. Durch die Nutzung bereits vorhandener Ressourcen von Zugangsknoten für die Realisierung von DuDE stellt dieses System eine kostengünstige Möglichkeit für Netzbetreiber dar, die heutigen Engpässe bei der Statistikerstellung zu beheben.

18. Das traditionelle DNS reicht auf Grund seiner limitierten Skalierbarkeit nicht mehr für die ständig wachsende Anzahl von Domainnamen und Nutzern im Internet aus. Für eine hohe Skalierbarkeit müssen ISP in extra Ausrüstung, d.h. zusätzliche DNS-Serverfarmen investieren. Dies ist mit hohen Kosten für das Betreiben, die Verwaltung sowie den Energieverbrauch dieser zusätzlichen Infrastruktur verbunden.

19. Um vor allen Dingen die Kosten des ISP für zusätzliche DNS-Serverfarmen zu sparen, wurde ein P2P-basiertes DNS (P-DONAS) entwickelt. P-DONAS verbindet mittels Kad Zugangsknoten miteinander und nutzt deren verfügbare Ressourcen. Es stellt somit eine vertrauenswürdige, ressourcensparende, hoch skalierbare und widerstandsfähige Lösung dar, die kompatibel zum traditionellen DNS ist.
20. Mit dem vollständig implementierten P-DONAS-Prototypen wurde bewiesen, dass mit Hilfe eines P2P-basierten DNS bestehende DNS-Lösungen um bis zu 53 % entlastet werden können, wenn es als Ergänzung eingesetzt wird. Potentiell kann es das traditionelle DNS vollständig ersetzen.
21. Das Netzwerk aus Zugangsknoten stellt eine Private Cloud dar, die Dienste zur Verfügung stellt und die notwendige Datenbasis mit garantierter Datenverfügbarkeit vorhält. So sind die Ansätze dieser Forschungsarbeit nutzbar, um enorme Speicher- und Rechenkapazitäten zu mobilisieren, die über Schnittstellen virtualisiert angeboten werden könnten. Wenn in Zukunft Cloud-Ressourcen standardisiert angefragt werden können, ist somit eine Kombination der beiden Prinzipien Cloud Computing und P2P möglich.

Kurzreferat

In den letzten Jahren haben verschiedene P2P-Applikationen wie z. B. das File-Sharing im Bereich der privaten Internet-Nutzung eine weite Verbreitung erlangt. P2P-Datenverkehr nimmt daher einen großen Teil des gesamten Internetdatenverkehrs ein. Konkret wird deshalb zunächst untersucht, inwiefern gezieltes P2P-Routing hilft, diese P2P-Datenmassen besser zu bewältigen und die Netzwerkinfrastruktur zu entlasten. Da P2P-Routing üblicherweise die zu Grunde liegende physikalische Netzwerktopologie nicht berücksichtigt, werden herkömmliche Mechanismen von Netzbetreibern für die Lenkung von Datenverkehr außer Kraft gesetzt. Es wurde ein Algorithmus entwickelt, der die Peer-Auswahl in dem P2P-Netzwerk BitTorrent durch Hop Count-Nutzung verbessert. Eine Auswertung des neuen BitTorrent-Algorithmus stellt dabei deutlich seine Vorteile gegenüber dem Standard-Algorithmus sowohl für den Nutzer als auch für Netzbetreiber heraus.

Wenngleich die Nutzung der P2P-Netze oftmals sofort mit dem illegalen Tausch lizenzrechtlichen Materials assoziiert wird, so sind die zugrunde liegenden Technologien vom Verwendungszweck losgelöst zu betrachten. Die vorliegende Forschungsarbeit widmet sich daher der P2P-Netzwerktechnologie als solcher und studiert Einsatzmöglichkeiten in Teilnehmerzugangsnetzen. Es wird untersucht, inwiefern mit geeigneten Mitteln Kosten der Netzbetreiber reduziert werden können, wenn P2P-Technologien im Teilnehmerzugangsnetz eingesetzt werden und Zugangsknoten mit verfügbaren Speicher- und Rechenressourcen selber Teilnehmer eines P2P-Netzes werden. Auf diese Weise können vorhandene Internet-Dienste, die Schwachstellen besitzen, ersetzt bzw. komplementiert werden. Wie die Resultate dieser Forschungsarbeit beweisen, konnten diese Dienste verbessert und um neue Funktionalitäten erweitert werden. Kostenintensive zusätzliche Infrastruktur kann somit eingespart werden. Zudem wird die Energieeffizienz der vorhandenen Zugangsknoten durch eine bessere Ausnutzung gesteigert. Als Ergebnisse wurden voll funktionsfähige Prototypen für eine P2P-basierte Speicherplattform, ein verteiltes Rechensystem sowie ein P2P-basiertes DNS entwickelt.

Abstract

Recently, various P2P applications like, e.g., file sharing have achieved wide dissemination. Therefore, P2P data volumes have a significant share in Internet data traffic.

At first, it has been investigated how well-directed P2P routing helps to handle these large P2P data volumes and to disburden network infrastructure. As P2P routing does usually not consider the physical network underlay, common traffic engineering mechanisms of ISPs are invalidated. An algorithm has been developed, which improves the peer selection in the P2P network BitTorrent by using the hop count. An evaluation of the new BitTorrent algorithm highlights its benefits over the standard algorithm both for users and the ISP.

Although P2P networks are often solely associated with the illegal sharing of legal licensing material, the underlying technology of P2P networks has to be considered unrelated to their application. This research work addresses the network technology of P2P networks itself and researches into its applications in access networks with respect to various aspects. Investigations are carried out on how the expenses of ISPs can be reduced with appropriate actions if P2P technology is deployed in access networks. Thereby, access nodes with their available memory and computing resources become members of a P2P network themselves. In this way, existing Internet services, which show weak spots, can be complemented or replaced. Results prove that these services could be improved and extended by new functionality. Thereby, cost-intensive additional infrastructure can be saved and moreover, the energy efficiency of existing access nodes is increased by an improved utilization. As proof of concept, fully operational prototypes of a P2P-based storage platform, a distributed data processing system, and a P2P-based DNS have been developed.