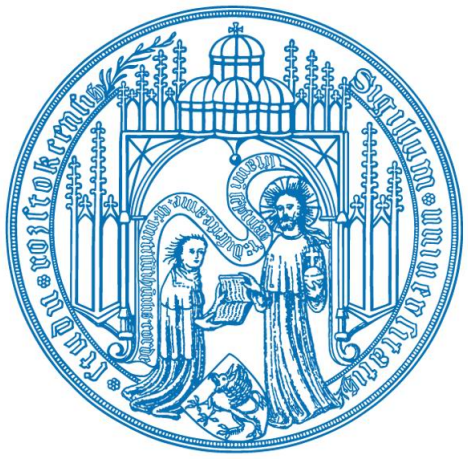
UNIVERSITÄT ROSTOCK

Institut für Nachrichtentechnik und Informationselektronik



Diplomarbeit

Analyse von IP-Datenverkehr im Transport Layer

vorgelegt am : 15. April 2005

von : cand. ing. Christian Groth
Anne-Frank-Weg 33
18069 Rostock

Matrikelnummer : 009820129
Studienrichtung : Informationstechnik

Betreuer : Dr.-Ing. Hans-Dietrich Melzer
Dipl.-Ing. Thomas Kessler

Ich erkläre hiermit, daß ich die vorliegende Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Rostock, den 14. 05. 2005

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AIM	AOL Instant Messenger
API	Application Program Interface
AS	Autonomous System
ASCII	American Standard Code for Information Interchange
CR	Carriage Return
DC++	Direct Connect PlusPlus
EBCDIC	Extended Binary Coded Decimal Interchange Code
ERF	Extensible Record Format
FTP	File Transfer Protocol
GBit	Gigabit
GPL	(GNU) General Public License
GNU	GNU's Not Unix
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ICQ	I Seek You
IEEE	Institute of Electric and Electronic Engineers
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol

IPFIX	Internet Protocol Flow Information eXport
IPv4	Internet Protocol version 4
IRC	Internet Relay Chat
ISO	International Standardization Organization
ISP	Internet Service Provider
LF	Line Feed
LLC	Logical Link Control
LINUX	Linux Is Not UniX
MAC	Media Access Control
MD5	Message Digest 5
MPLS	Multi Protocol Label Switching
MSNMS	MicroSoft Network Messaging Service
MiB	Megabyte
NIC	Network Interface Card
NFS	Network File System
OSCAR	Open System for CommunicAtion in Realtime
OPMAN	OPen IP-based Multiservice Access Network
OSI	Open Systems Interconnect
P2P	Peer to Peer
PERL	Practical Extraction and Report Language
PDU	Protocol Data Unit
POP	Post Office Protocol
QoS	Quality of Service
(R)ARP	(Reverse) Address Resolution Protocol
RFC	Request For Comments

RPC	Remote Procedure Call
SCTP	Stream Control Transmission Protocol
SMB	Server Message Block
SMTP	Simple Mail Transfer Protocol
SOHO	Small Office/ Home Office
SSH	Secure SHell
SPAN	Switched Port ANalyzer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UNIX (UNICS)	UNIpleXed Information and Computing System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VLAN	Virtual Local Area Network
WWW	World Wide Web

1. Einleitung

1.1 Aufgabenstellung

Analyse von IP Datenverkehr im Transport Layer

Die Analyse realer Kommunikationsaufkommen dient unter anderem der Optimierung von Kommunikationsnetzwerken und Datenströmen. Herr Groth soll im Rahmen der Diplomarbeit - aufbauend auf vorangegangene Arbeiten - die Aufzeichnung von Daten im Netz des Institutes für Nachrichtentechnik und Informationselektronik sowie im ComLab unter spezieller Berücksichtigung von Zeitpunkt und Dauer der Messungen optimieren und Aufzeichnungen durchführen. Um zu aussagefähigen Ergebnissen zu gelangen, sind folgende Teilaufgaben zu lösen:

- Vergleich, Bewertung und Auswahl von Methoden für die Bestimmung des Anwendungsschichtprotokolls anhand der Nutzdaten (Payload) der Layer 4 PDUs.
- Umsetzung/Anpassung ausgewählter Methoden in einer Messanwendung bzw. in einem Messsystem.
- Planung und Durchführung von Messungen in den IP Netzen des ComLab und des Institutes für Nachrichtentechnik und Informationselektronik.
- Auswertung der Messwerte bezüglich der Zusammensetzung von IP Datenverkehr auf Anwendungsebene.
- Vergleich, Auswahl und Bewertung geeigneter Methoden.

Die Analyseergebnisse sind in Hinblick auf die Nutzung zur Signalsynthese z.B. für die Generierung von Testfolgen zwecks Netzwerksimulation zu untersuchen und unter OPNET zu testen.

1.2 Zielsetzung der Arbeit

Mit der rasanten Zunahme des über das Internet transportierten Datenvolumens sind ebenfalls die Anforderungen an die Netzinfrastruktur gewachsen. Neue Kommunikationsdienste wie Audio- und Videoübertragung in Echtzeit sind gegenüber Verzögerungen und Jitter weitaus empfindlicher als „traditionelle“ Applikationen wie Dateitransfer oder interaktive Textbotschaften. Zusätzlich ist eine zunehmende Nutzung bandbreitenintensiver Peer-to-Peer-Dienste zu beobachten, wodurch die Netzlast weiter ansteigt. Die große Anzahl an netzwerkfähigen Applikationen erfordert eine genaue Auswertung des Nutzerverhaltens. Beim Auftreten nennenwerter Verkehrsvolumina mit zeitkritischen Inhalten ist u.U. eine Anpassung des Netzes erforderlich. Maßnahmen können beispielsweise aus Bandbreitenerhöhungen oder gezielten Eingriffen in den Durchsatz einzelner Protokolle, wie positive oder negative Priorisierung, bestehen. Derartige Handlungen erfordern im Vorfeld zwingend genaue Kenntnisse über die Zusammensetzung der transportierten Protokolle in der Applikationsschicht, wobei sich traditionelle, auf Portnummern zurückgreifende Methoden als zunehmend ungeeignet erweisen. Ziel der vorliegenden Arbeit ist es, eine skalierbare und wiederverwendbare Umgebung zum Zweck der Applikationsschichtanalyse des Kommunikationsaufkommens in modernen paketvermittelten Netzen zu entwickeln und anhand von in Netzwerken mit verschiedenen Dienstnutzungsprofilen aufgenommen Meßproben deren Leistungsfähigkeit zu beurteilen. Schwerpunkt wird auf die Heranziehung der Nutzdaten der Applikationsschicht zur Identifizierung der in Anspruch genommenen Dienste gelegt; es wird zusätzlich ein auf neuen Erkenntnissen beruhender Ansatz zur Identifikation von Applikationsschichtprotokollen anhand von Verbindungsmustern auf der Transportschicht verfolgt. Die vorliegende Arbei bettet sich in das Forschungsprojekt OPMAN (Open IP-based Multiservice Acess Network) des Instituts für Nachrichtentechnik und Informationselektronik an der Universität Rostock ein.

1.3 Gliederung der Arbeit

Vorliegende Arbeit ist in sechs Kapitel untergliedert. Es folgt eine Kurzbeschreibung der Inhalte der Kapitel.

- Kapitel 2 befaßt sich mit dem Begriff des Traffic Engineering und stellt seine Anwendungsgebiete sowie Hilfsmittel vor.
- Kapitel 3 stellt die Client-Server-Architektur und das Peer-to-Peer-Konzept anhand von ausgewählten Protokollen der Applikationsschicht im Kontext des TCP/IP-Schichtenmodells vor.
- Kapitel 4 geht auf die Durchführung von Datenverkehrsmessungen in Netzwerken ein und beschreibt die zur Anwendung gelangte Meßumgebung.
- Kapitel 5 umfaßt die Beurteilung, Auswahl und Anpassung von Analysewerkzeugen zur Bstimmung von Applikationsschichtprotokollen in Meßproben. Ferner wird ein Ansatz, der ohne Daten der Applikationsschicht auskommt, vorgestellt. Es sind ebenfalls Betrachtungen zur Leistungsfähigkeit der entwickelten Analyseumgebung enthalten.
- Kapitel 6 beurteilt die erzielten Ergebnisse und stellt weiterführende Aufgabenfelder vor.

2. Traffic Engineering

Die Disziplin des Traffic Engineering befaßt sich mit Methoden, die dem besseren Verständnis der Charakteristika des Verkehrsverhaltens von Netzwerken dienen. Das Ziel besteht in einer optimalen Lastverteilung mit minimalen Latenzzeiten, Jitter und Paketverlusten sowie maximalem Durchsatz für alle Anwendungen und maximaler Zuverlässigkeit des Netzes unter Berücksichtigung von vorgegebenen Randbedingungen [RFC3272]. Die Einhaltung dieser als Verkehrsvertrag[1] bezeichneten Bedingungen ist für eine zunehmende Zahl neuerer Applikationen wie Voice Over IP oder Video On Demand von essentieller Wichtigkeit. Traffic Engineering stellt gerade in Umgebungen mit hoher Bandbreitenüberbuchung eine nichttriviale Aufgabe dar.
Auf oberster Abstraktionsebene besteht ein IP-Netzwerk aus

- verteilten Ressourcen, die Transportdienste anbieten (Router, Switches, Übertragungswege),
- einem Anforderungssystem, das die offerierten Transportdienste in Anspruch nimmt (Endknoten) sowie
- Maßnahmen, die den Transport des Netzwerkverkehrs über die Ressourcen abwickeln (Protokolle)

[RFC3272]. Aktionen zur Verkehrsregulierung kann der Netzbetreiber lediglich an den in seinem Eigentum befindlichen Ressourcen vornehmen. Regulierungsmaßnahmen erfolgen im Kontext von drei allgemein akzeptierten Fakten:

- IP-Netze offerieren zunehmend Echtzeitdienste.
- Sie sind in der heutigen Geschäftswelt unverzichtbar[2].
- Sie operieren in sich schnell verändernden Umgebungen, vor allem bezüglich der Netzlast.

[1] engl. Service Level Agreements, SLA
[2] In der englischsprachigen Literatur wird die Vokabel *mission critical* verwendet.

Insbesondere kann eine Erhöhung der Netzlast zu einer Überlastung der Ressourcen führen, was erhöhte Antwortzeiten, Jitter und im Extremfall Paketverluste nach sich zieht. Die Hauptaufgabe des Traffic Engineering ist es, allgemein Überlastsituationen zu vermeiden bzw. ihnen entgegenzuwirken[3].
Das wichtigste Kriterium bei der Beurteilung der Qualität eines Netzes ist dessen Leistungsfähigkeit aus Sicht des Benutzers. Die von dem Benutzer verwendete Applikation legt fest, welche Anforderungen die Netzinfrastruktur zu erfüllen hat. Es ist die Aufgabe des Traffic Engineering, alle Schichten des verwendeten Netzwerkmodells zu nutzen, um den Anforderungen der Applikation und damit letztlich denen des Nutzers zu genügen. Der erste Schritt besteht darin, Qualitätsmerkmale zu definieren, nach denen das vorhandene Netz zu optimieren ist. Diese müssen präzise genug sein, um daraus Kriterien zur Gewinnung von Parametern durch Beobachtung zu gewinnen und schließlich eine Strategie zu Art, Dauer und Ort von Verkehrsmessungen zu formulieren. Im zweiten Schritt ist die Ist-Situation durch Verkehrsmessungen an geeigneten Orten wie Uplinks zu ISPs und „Flaschenhälsen“ in Erfahrung zu bringen. Abhängig von den definierten Gütekriterien sind entsprechend umfangreiche Analysen erforderlich. Mit den in der Analysephase gewonnenen Parametern kann ein Modell dimensioniert und einer Simulation unterzogen werden. Die gewonnenen Ergebnisse sind durch weitere Messungen zu verifizieren und auf das Netz anzuwenden. Die Analyse weiterer Meßdaten dokumentiert den Erfolg der Maßnahmen. Durch Parameterfeinabstimmung kann sukzessive die Leistungsoptimierung vor dem Hintergrund der Aufgabenstellung erfolgen. Bild 2.1 ist der iterative Charakter des Traffic Engineering zu entnehmen. Die angewandten Modelle können aufgrund der hohen Komplexität und Vielfalt der Protokolle und Netztopologien niemals exakt sein; vielmehr gehen durch den für die Modellbildung notwendigen Abstraktionsprozeß u. U. auch wichtige Details verloren.
Traffic Engineering läßt sich grob in vier Schritte unterteilen [Gall02], [RFC3272] :

- Messung (Measure)
- Analyse (Characterize)
- Simulation (Model)
- Planung/Umsetzung (Control).

Messung und Analyse sind dem Bereich des Traffic Monitoring zuzuordnen, während die *Performance Prediction* Simulation und Planung übernimmt. Sie hängt offensichtlich von der Verfügbarkeit aktueller Meßdaten ab. Der Meßvorgang umfaßt die Aufzeichnung des Verkehrs. Hier können schon erste Aussagen zu (zeitabhängiger) Netzlast, Trends, Zusammensetzung und Leistungsfähigkeit getroffen werden. Analog kann schon während der Aufzeichnung „uninteressanter“ Verkehr ausgefiltert werden.
Die Analyse nimmt in dieser Arbeit den größten Raum ein. Eine Unterteilung kann je nach untersuchter Protokollschicht u.a. nach IP-Adressen, AS-Pfaden, Port-Nummern der Transportschicht oder Art der transportierten Anwendungsprotokolle bzw. Kombinationen daraus erfolgen. Die Bestimmung von Applikationsschichtprotokollen von in unterschiedlichen Umgebungen aufgenommenen Meßproben steht im Fokus dieser

[3]Man spricht auch von proaktivem bzw. reaktivem Traffic Engineering.

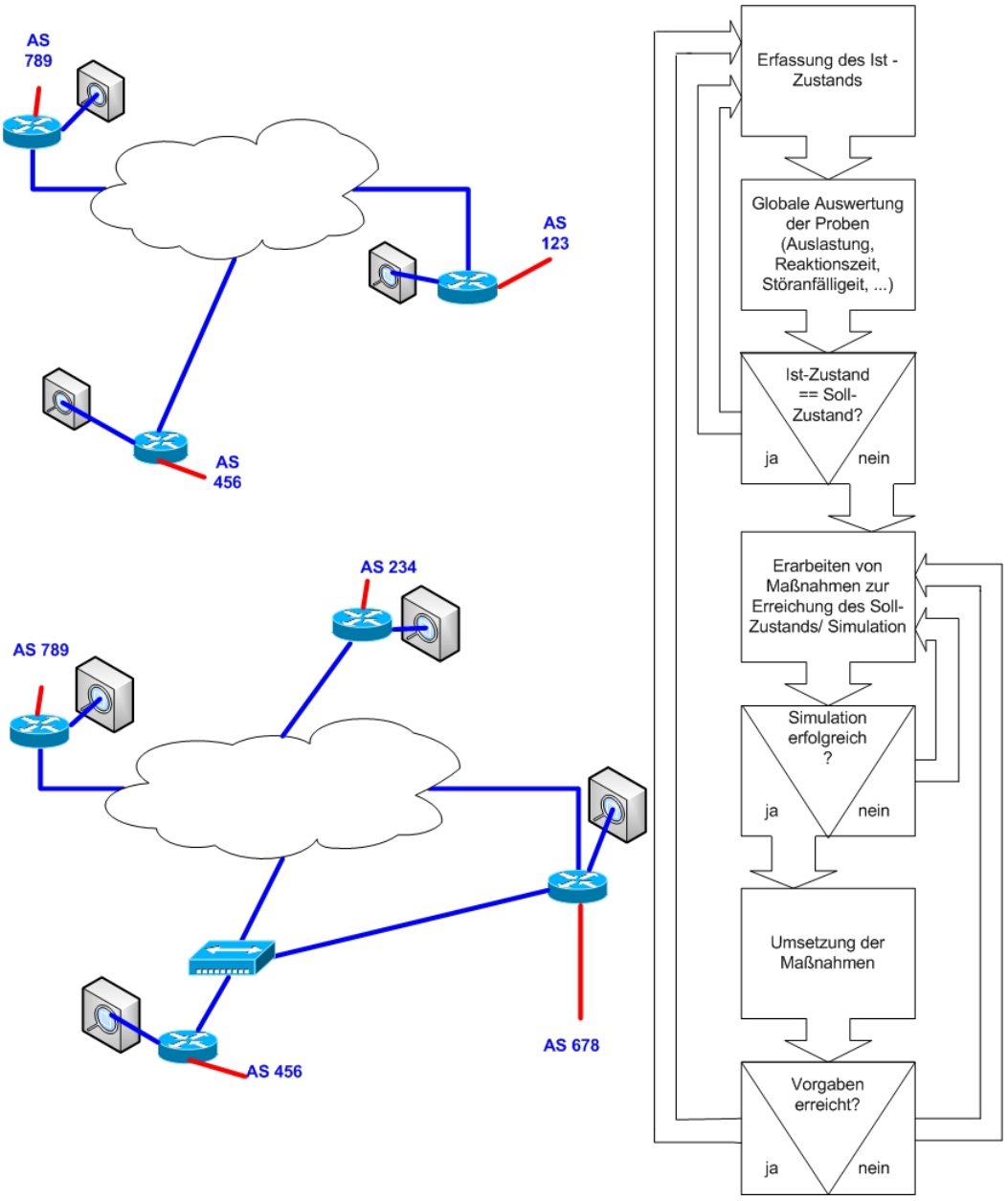


Abbildung 2.1: Schritte des Traffic Engineering. Eine Verbesserung der Antwortzeiten läßt sich z.B. durch die Erhöhung der Peerings und die Einführung von redundanten Topologien erreichen.

Arbeit. Der Überblick über den Protokollmix der Applikationsschicht kann für Netzwerkadministratoren ein wertvolles Hilfsmittel zur Feststellung von dominierenden Applikationsklassen und deren zeitlicher Entwicklung sein. Das Ziel ist die Aufbereitung der Daten für die Simulation der Charakteristika von durch von verschiedenen Applikationen geprägten Netzwerken. Die Simulation selbst findet im Rahmen dieser Arbeit keine Berücksichtigung.

2.1 Terminologie

Dieser Unterabschnitt erläutert die wichtigsten Begriffe der Disziplin des Traffic Enginering in dem Zusammenhang, in dem sie in dieser Arbeit gebraucht werden.

- Applikationsklasse
  Der Begriff Applikationsklasse umfaßt im Kontext dieser Arbeit die Zusammenfassung von Applikationsschichtprotokollen mit für den Nutzer vergleichbaren Eigenschaften. Die Applikationsklasse „Peer-to-Peer“ umfaßt u.a. die Protokolle eDonkey, Fasttrack, DirectConnect und Bittorrent.

- Flow
  Ein Flow (Fluß) ist die Zusammenfassung von Paketen eines Protokolls der Netzwerkschicht mit gleichen Eigenschaften. Im Rahmen dieser Arbeit besteht ein Fluß aus Paketen mit übereinstimmenden IP-Quell- und -Zieladressen, demselben Protokoll der Transportschicht und gleichen Quell- und Zielports. Es ist damit per Definition unidirektional. Flüsse werden von Flow Timeouts voneinander abgetrennt, d.h. ist die Kommunikation eines Flusses für mehr als ein festgelegtes Zeitlimit unterbrochen und findet danach Kommunikation in derselben Kombination aus IP-Quell- und -Zieladresse, Transportschichtprotokoll und Quell- und Zielports statt, wird dies als neuer Fluß betrachtet. Ein weithin akzeptierter Wert für das Flow Timeout ist 64s.

- Conversation
  Eine Konversation ist ein bidirektionaler Fluß.

- Backbone
  Der Backbone eines Netzwerks ist eine breitbandige Verbindung von Teilnetzen.

2.2 Traffic Monitoring

Traffic Monitoring hat als Disziplin des Traffic Engineering die Aufgabe, relevante Verkehrsparameter zu isolieren. Für die Kapazitätsplanung ist die Identifikation der lastverursachenden Protokolle der Applikationsschicht von Interesse. Der traditionelle Ansatz besteht in der Auswertung von Port-Informationen der Transportschicht. Als anerkannte Referenz dienen die *TCP and UDP Port Numbers* der [IANA]. Die immer schnellere Entwicklung von neuen Protokollen sowie die große Verbreitung von Peer-to-Peer-Applikationen mit frei wählbarer Portnummer führt diese Herangehensweise an ihre Grenzen. Seit einerseits einige Internet Service Provider die Bandbreite bekannter Peer-to-Peer-Portbereiche drosseln und andererseits die Peer-to-Peer-Clientsoftware den Wechsel sogar zum TCP-Port 80 (HTTP) ermöglicht, ist eine korrekte Identifizierung der Verkehsflüsse allein auf Transportschichtebene mit gedächtnisloser Analyse[4] als unzuverlässig zu betrachten. Des weiteren ist eine immer größere Anzahl von Verkehrsflüssen, welche nicht von der IANA zugeordnete Portkombinationen nutzen, zu beobachten. Eine mögliche Ursache ist bei Online-Spielen und neuen Peer-to-Peer-Protokollen zu suchen. Quellen, die das Verkehrsaufkommen eines USA-weiten IP-Transfernetzes seit 2002 wöchentlich auswerten, dokumentieren einen Anstieg des Anteils von unidentifiziertem Verkehr von 20% im Februar 2002 auf bis zu 33% im Dezember 2004 [ST04]. Zu berücksichtigen ist weiterhin der Umstand, daß die Messungen mit Cisco Netflow [Cis05] durchgeführt wurden, wobei für die Auswertung keine Daten der Applikationsschicht zur Verfügung standen. Die Zuordnung der Flüsse zu den Applikationsprotokollen ist dementsprechend kritisch

[4] Die gedächtnisbehaftete Analyse wird in Kap. 5.3.2.4 vorgestellt.

zu betrachten.
Um zu aussagekräftigen Ergebnissen bezüglich einer Zuordnung von Verkehrsflüssen zu denen sie hervorrufenden Applikationen zu gelangen, ist zumindest ein Teil der Nutzlast des Transportschichtprotokolls notwendig. Als de-facto-Standardwerkzeug zur Aufzeichnung von Netzwerkverkehr bis zum einzelnen Paket hinunter hat sich das frei verfügbare Programm [TCPdump] erwiesen, welches den Verkehr im lokalen Netz an der Netzwerkschnittstelle jedes handelsüblichen PCs aufzeichnen kann. Für hochgenaue professionelle Messungen an Backbones existieren verschiedene Spezialmeßkarten mit entsprechender Software, z.B. von Endace Measurement Systems [End05]. Kap. 4 geht näher auf die Durchführung von Messungen ein.
Diese „traditionelle“ Herangehensweise wird im professionellen Umfeld von automatisierten Umgebungen, welche dezentrale Aufzeichnung, zentrale Speicherung und Auswertung sowie Präsentation übernehmen, verdrängt. Dabei fungieren zentrale Netzelemente wie Router und Switches selbst als Meßgeräte, ohne daß dedizierte Hardware zur Aufzeichnung benötigt wird. Als zukünftiger Standard ist IPFIX [5] in Entwicklung. Als Grundlage dient das NetFlow-9-Format von Cisco Systems. Da NetFlow der Version 9 bereits verfügbar ist, wird es in Kap. 2.3 vorgestellt.

2.3 Cisco NetFlow v9

NetFlow ist wegen der weiten Verbreitung von Netzwerkkomponenten der Firma Cisco die derzeit wichtigste Technologie auf dem Gebiet der systematischen professionellen Netzwerküberwachung. Viele seiner Merkmale werden in den kommenden IETF-Standard IPFIX einfließen. NetFlow ist eine speziell auf Service Provider und Großunternehmen zugeschnittene Technologie. Als Einsatzgebiete werden

- Überwachung von Peerings
- Netzplanung
- Abrechnung bei Volumentarifmodellen
- Verfolgung von Angriffen
- Verkehrsflußanalysen
- Analyse der Protokoll- und Applikationszusammensetzung
- Überwachung des Datenverkehrs einzelner Nutzer

genannt [Cis04]. Entsprechend der Vielseitigkeit der Einsatzgebiete sind die verschiedensten Paket- und Flußcharakteristika von Interesse. In Version 5 sind u.a. Einträge über

- IP-Quell- und Zieladresse
- TCP- und UDP-Portnummern, Flags
- Paket- und Bytezahl je Fluß

[5] Internet Protocol Flow Information eXport

- IP-Type of Service
- ein- und ausgehende Subinterfaces
- Next Hop
- Quell- und Ziel-AS

verfügbar. Daneben sammelt die Meßinstanz Statistiken z.B. zur Paketgrößenverteilung, Anzahl der aktiven und abgelaufenen Flüsse sowie Durchsatz. Version 9 führt erstmalig das Konzept der erweiterbaren Templates (Schablonen) ein, mit denen der Administrator flexibel die Kriterien seiner Messungen bestimmen kann. Templates ermöglichen das Hinzufügen von neuen Einträgen auch nach Fertigstellung der Implementation. NetFlow selbst beschreibt Kommunikationsform und -Inhalt sowie die kommunizierenden Komponenten und deren Aufgaben im Netzwerk und ist in [RFC3954] dokumentiert. Durch diese Herangehensweise können Drittanbieter ebenfalls Software zur Speicherung, Analyse und Präsentation von NetFlow-Daten entwickeln.
Eine NetFlow-Umgebung besteht aus den Komponenten Exporter und Collector. Der Exporter läuft als eigenständiger Prozeß auf einem Router oder Switch. Seine Aufgaben umfassen

- Beobachtung aller passierenden Pakete
- Zusammenfassung dieser Pakete zu Flüssen
- Bildung von Flow Records aus den Flüssen entsprechend den Vorgaben des gewählten Flow Template
- periodisches Senden der Flow Records zum Collector

Der Collector ist die zentrale Sammelinstanz der Flow Records aller Exporter. Ihm obliegen

- Demultiplexen der Data Flow Sets nach Templates und Exporter
- Decodierung der Flow Records
- zentrale Speicherung der Flow-Informationen

Abb. 2.2 zeigt die Infrastruktur einer einfachen NetFlow-Umgebung. Die unidirektionalen Pfeile von den Routern zum Collector deuten an, daß NetFlow ein Push-Protokoll ist. Es setzt auf der Transportschicht auf, wobei frühere Versionen ausschließlich UDP verwendeten. Seit Version 9 besteht Transportschichtprotokollunabhängigkeit, so daß auch Protokolle wie das Stream Control Transmision Protocol (SCTP) verwendet werden können, welche die Vorteile von TCP und UDP vereinen.

In der NetFlow-Terminologie gibt es drei verschiedene Nachrichtentypen, die alle als Flow Sets bezeichnet und in Export Packets verschickt werden.

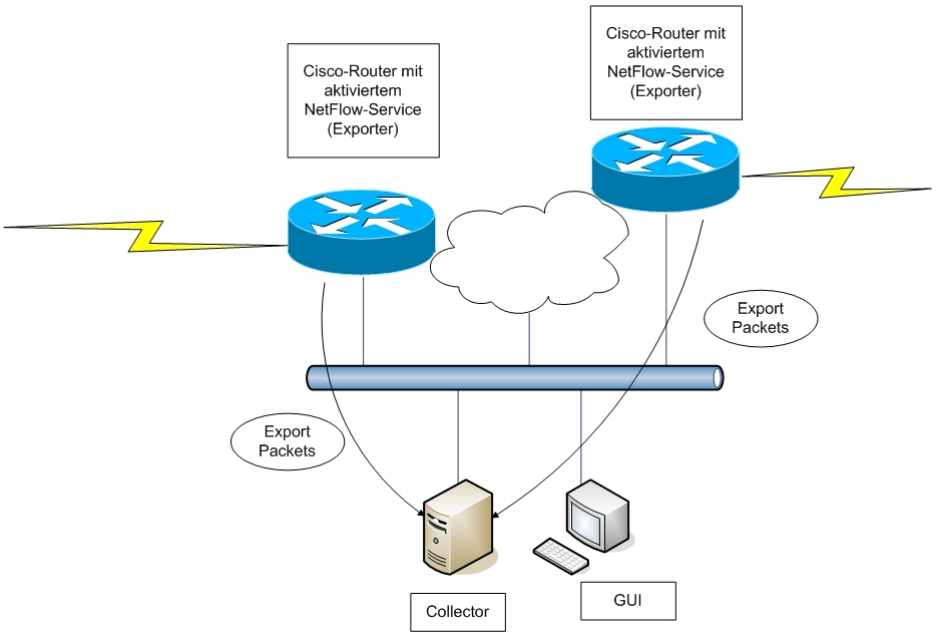


Abbildung 2.2: Beispiel einer einfachen NetFlow-Infrastruktur

- Das Template Flow Set enthält ein oder mehrere Template Records, die die Struktur der exportierten Flüsse beschreiben.
- Options Template Flow Set
- Das Data Flow Set besteht aus identisch strukturierten Datensätzen, deren Form durch das Template Flow Set definiert ist. Es kann sich sowohl um Flow Data Records als auch um Options Data Records handeln.

Zum näheren Verständnis sei das Format des Export Packets kurz vorgestellt. Jedes Paket besitzt einen 20 Oktetts langen Header, gefolgt von einer beliebigen Kombination aus Template Flow Sets, Options Template Flow Sets und Data Flow Sets, vgl. Bild 2.3.

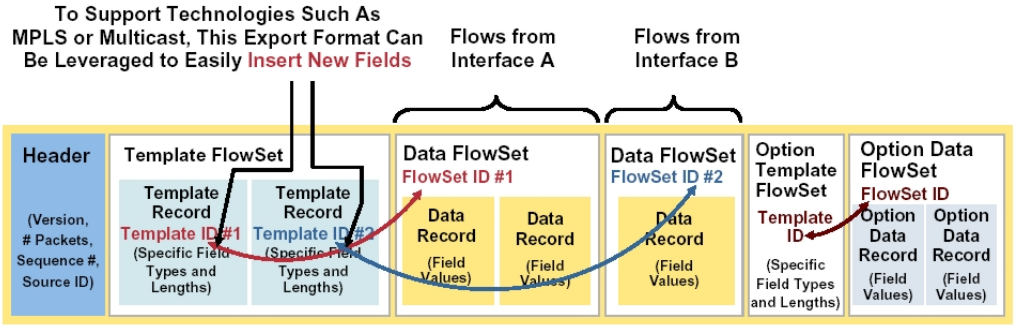


Abbildung 2.3: allgemeiner Aufbau eines Export Packet [Cis04]

Der Header ist wie in Abb. 2.4 aufgebaut.

Flow Templates haben die in Bild 2.5 gezeigte Form.

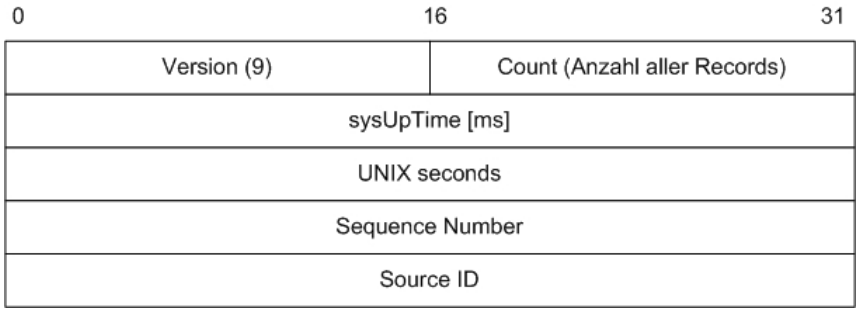


Abbildung 2.4: Aufbau des Export Packet Header

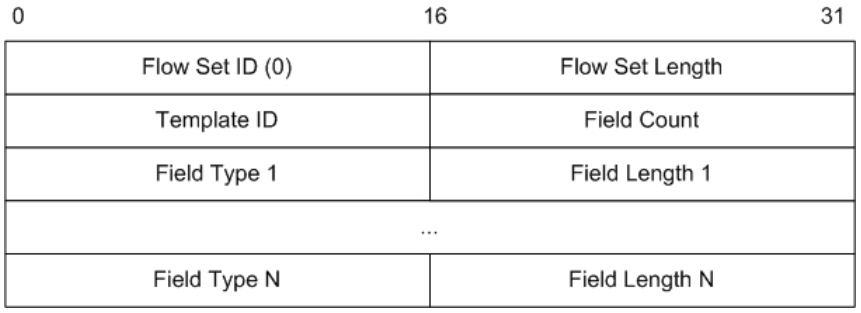


Abbildung 2.5: Aufbau des Template Flow Set

Data Flow Sets (vgl. 2.6) sind der am häufigsten auftretende FlowSet-Typ. Sie bestehen hauptsächlich aus einer Aneinanderreihung von Werten, die dem zugehörigen Feld im Template entsprechen.

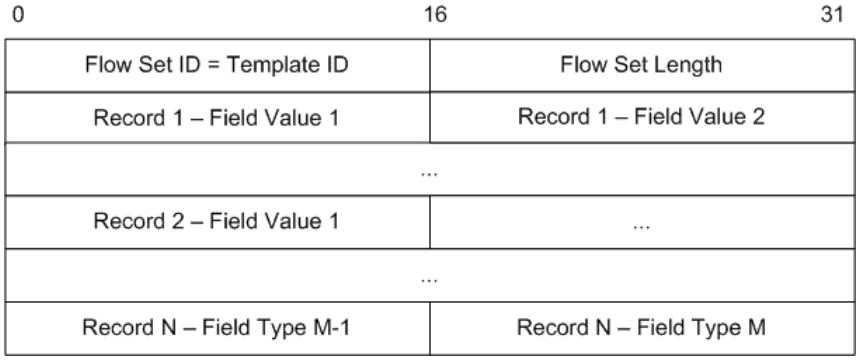


Abbildung 2.6: Data Flow Set

Auf den dritten Nachrichtentyp, das Options Template Flow Set, wird nicht eingegangen. Aus Abb. 2.5 und 2.6 ist ersichtlich, wie die Flexibilisierung durch das Template-Konzept umgesetzt ist. Ein Exporter kann Data FlowSets unter Verwendung verschiedener Templates an unterschiedliche Collectors senden. Die Assoziation zwischen Template Flow Set und Data Flow Set erfolgt am Collector lokal anhand Template ID des Template Flow Set und Flow Set ID des Data Flow Set.

3. Protokolle der Anwendungsschicht in Kontext des TCP/IP-Modells

Obwohl sich in der Literatur das 7-Schichten-OSI (Open Systems Interconnect)-Modell durchgesetzt hat, wird die Praxis weiterhin vom pragmatischeren TCP/IP-Schichtenmodell dominiert, welches zunächst betrachtet wird. Es folgt die Vorstellung der Client-Server- sowie der Peer-to-Peer-Architektur, ergänzt durch die Betrachtung typischer Vertreter beider Konzepte.

3.1 Das TCP/IP-Schichtenmodell

In Abb. 3.1 sind ISO-OSI-Modell und das TCP/IP-Modell vergleichend dargestellt. Das in den 1980er Jahren standardisierte OSI-Referenzmodell zielt auf größtmögliche Abstraktion von der Technik, die in den jeweiligen Schichten eingesetzt wird, ab. Nachteil der höheren Abstraktion ist eine stark zunehmende Komplexität, welche in der praktischen Umsetzung mit langen Entwicklungszeiten und hohen Kosten verbunden ist, ohne einen unmittelbar meßbaren „Return Of Investment“ zu bieten. Demgegenüber befand sich mit der TCP/IP-Protokollfamilie eine erprobte Technologie im praktischen Einsatz, die darüberhinaus durch die weite Verbreitung des UNIX-Betriebsystems im akademischen Umfeld ständig erweitert, verbessert und schließlich massenmarktkompatibel wurde, bevor OSI-konforme Referenzimplementierungen marktreif verfügbar waren. Alle maßgeblichen in der Informationstechnik verwendeten Protokolle setzen auf das TCP/IP-Modell auf oder bieten zumindest eine Schnittstelle. Die in dieser Arbeit betrachteten Applikationsschichtprotokolle basieren allesamt auf TCP/IP, weshalb ihm im Kontext der Aufgabenstellung der Vorzug gegeben wird.

Obwohl die hardwareabhängige Schicht nicht explizit zur TCP/IP-Protokollsuite gehört, wird sie, um bei der Mumerierung der Schichten Konsistenz mit dem OSI-Modell zu ereichen, in die meisten Betrachtungen mit einbezogen. Sie umfaßt die Bitübertragungsschicht sowie die MAC-Subschicht des OSI-Modells.

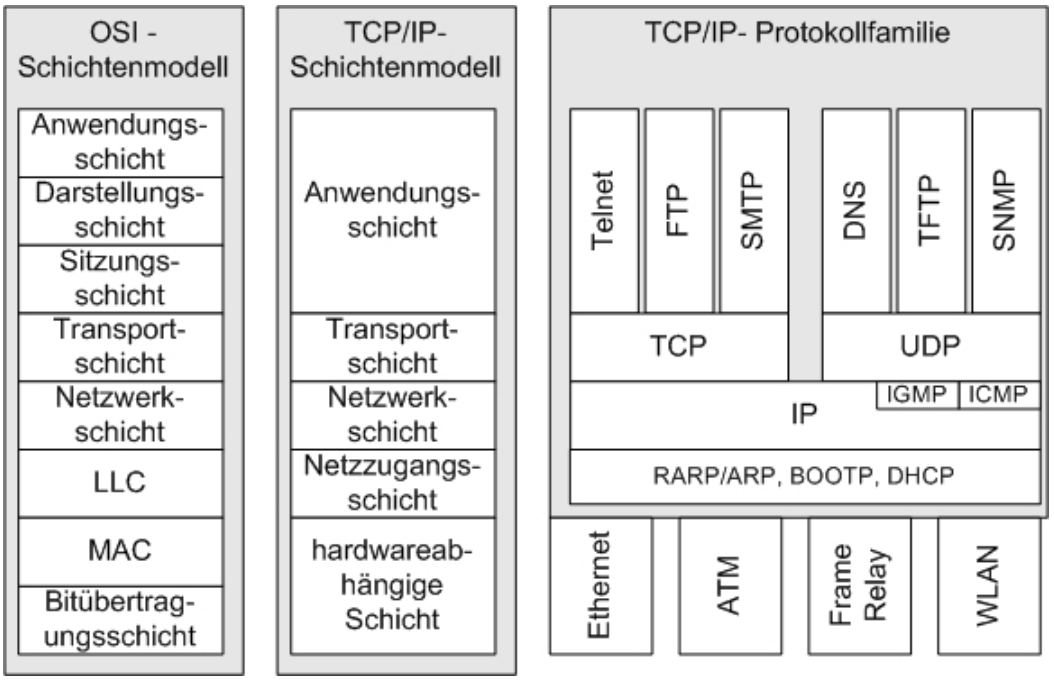


Abbildung 3.1: Gegenüberstellung der TCP/IP- und der OSI-Protokollsuite

3.1.1 Schicht 2 - Netzzugangsschicht (Host am Netz)

Die Host-am-Netz-Schicht setzt direkt auf den Treiber der Netzwerkhardware bzw. auf ein Subsystem mit eigenem Media Access Control (MAC)-Layer auf. Die MAC-Schicht ist nicht Teil der TCP/IP-Suite. Dadurch wird eine Unabhängigkeit vom Netzzugangsprotokoll erreicht, so daß TCP/IP-Verbindungen über eine Großzahl von Zugangstechnologien hergestellt werden können. Als Beispiele seinen

- die IEEE 802.3 Ethernet-Familie (z.B. 10base-T, 100base-T, 1000base-FX)
- PPP, PPPoE, SLIP, IEEE 802.14 (Internet over HFC)
- ISDN, HDLC, ATM
- IEEE 802.11 (WLAN), IEEE 802.16 (WIMAX)

genannt. Die Netzzugangsschicht hat ihr Äquivalent in der Logical-Link-Control (LLC)-Schicht des ISO-OSI-Modells. Ihr werden die auf der Netzwerk- und Netzzugangsschicht operierenden Protokolle ARP und RARP, deren Aufgaben in der Zuordnung von Hardware- zu IP-Adressen bestehen, zugeordnet. Da die Netzzugangsschicht konzeptuell der LLC-Schicht enspricht, ergeben sich kongruente Zuständigkeiten:

- flaches Adressierungsschema mit für jedes Gerät eindeutiger Kennung
- Definition einer Rahmenstruktur für die übertragenen Daten
- Segmentierung und Zusammensetzung der Rahmen

- Überprüfung der Rahmen auf Übertragungsfehler
- Versenden von Bestätigungsmeldungen, Anforderungen von Wiederholungssendungen (Hardware-Flußsteuerung).

Die Protokolle dieser Schicht sehen jede direkte physikalische Verbindung über ein Übertragungsmedium zu einer Netzwerkschnittstelle auf einer benachbarten Netzkomponente als eigenständiges Netzwerk an. Aufgrund dieser „lokalen“ Aufgaben kommt den Parametern dieser Schicht im Kontext dieser Arbeit keine Bedeutung zu.

3.1.2 Schicht 3 - Internet-Schicht

Die Internet-Schicht ist als eine Implementation der in der Netzwerkschicht des OSI-Modells geforderten Funktionalität anzusehen. Die konkreten Anforderungen bestehen in

- einem hierarchischen Adressierungsschema mit einer für jedes Gerät eindeutigen Kennung
- Definition einer Paketstruktur für die zu übertragenden Daten
- Bereitstellung eines unzuverlässigen, verbindungslosen (redundanten) Wegs von der Quelle zum Ziel
- Bestimmung des nach vorgebbaren Kriterien billigsten Weges von der Quelle zum Ziel anhand von im Paket enthaltenen Informationen[1]
- Feststellung von Übertragungsfehlern; Flußsteuerung in höheren Protokollschichten.

Ein IP-Paket besitzt die Maximalgröße von 1500 Bytes. Der kommende Standard IPv6 erweitert die obige Auflistung durch bessere Unterstützung von Quality of Service (QoS), Sicherheit (Authentifikation und Datenschutz) sowie Multicasting. Da in dieser Arbeit nur Traces aus IPv4-Netzwerken untersucht wurden, sei für eine ausführliche Beschreibung auf [RFC2460] verwiesen. Der Protokollkopf des Internet-Protokolls der Version 4 ist in Bild 3.2 dargestellt.

Die grau eingefärbten Felder sind für die Arbeit unbedeutend, die für Flüsse relevanten Parameter IP- Quell- und Zieladresse, sowie Transportschichtprotokolltyp weiß hervorgehoben. Neben IP existieren auf dieser Schicht das Internet Control Message Protocol sowie das Internet Group Management Protocol. Da diese jedoch keine Schnittstelle zur Transportschicht aufweisen, werden sie im weiteren vernachlässigt. Eine komplette Beschreibung des Internet Protocol der Version 4 ist in [RFC791] zu finden.

[1] Routing

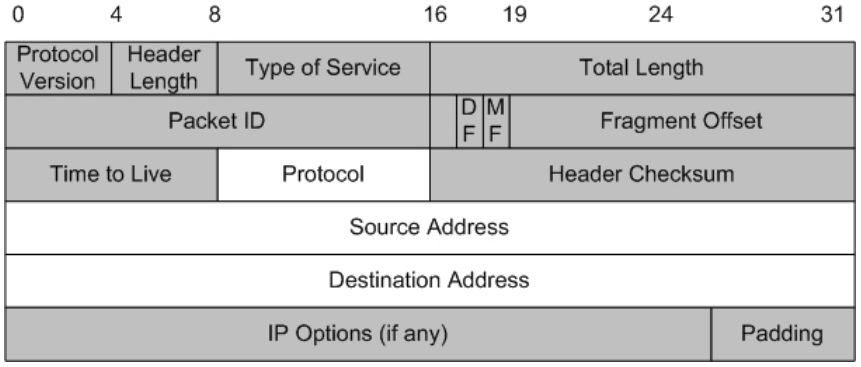


Abbildung 3.2: Aufbau des IPv4-Headers

3.1.3 Schicht 4 - Transportschicht

In der Hauptverantwortung der Transportschicht liegt es, eine gesicherte Ende-zu-Ende-Verbindung zwischen auf zwei miteinander kommunizierenden Geräten aktiven Applikationen bereitzustellen. Die daraus resultierenden Unteraufgaben bestehen aus

- Einführung des Begriffs des Transportstroms und Abstraktion der Ströme als logische Verbindungen
- Definition eines Datagramm- bzw. Segmentformats
- Bereitstellung des Konzepts der Portnummern zum Multiplexen von mehreren Transportströmen zwischen gleichen Netzwerkknoten
- Quittierung bzw. Neuanfoderung von Segmenten und Steuerung der Senderate des Kommunikationspartners
- Entdeckung von Fehlern und Neuanforderung von Segmenten
- Sortierung von Segmenten, die nicht in der Sendereihenfolge eintreffen.

TCP/IP stellt zwei Transportschichtprotokolle unterschiedlicher Komplexität zur Verfügung. Das User Datagram Protocol erfüllt lediglich die Bedingungen (1), (2) und (3) des Anforderungsprofils und wird verwendet, wenn geringe Datenmengen, die in wenigen Datagrammen Platz finden, oder zeitkritische Inhalte wie Voice-over-IP-Daten zu senden sind.
Das Transmission Control Protocol bietet den vollen Funktionsumfang einer gesicherten Datenfernübertragung und wird vor allem für Dateiübertragungen aller Art eingesetzt. Es weist gegenüber UDP eine deutlich höhere Komplexität auf. Beide Protokolle existieren unabhängig voneinander. Dies bedeutet insbesondere, daß identische TCP- und UDP-Portnummern keinesfalls bedeuten, daß dasselbe Programm der Applikationsschicht sie verwendet. Zur Erstellung von Flußtabellen sind auf dieser Schicht lediglich Quell- und Zielportnummern von Wichtigkeit, vgl. Abb. 3.3

Für eine vollständige Beschreibung von TCP sei auf [RFC793], von UDP auf [RFC768] verwiesen.

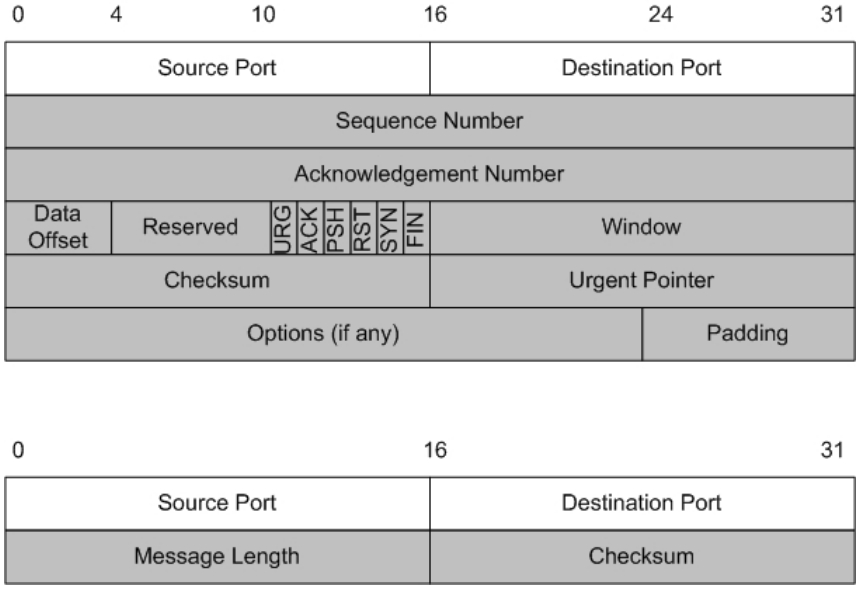


Abbildung 3.3: Aufbau von TCP-Header und UDP-Header

3.1.4 Schicht 5 - Anwendungsschicht

Die Aufgabe der Anwendungsschicht besteht darin, Anwendungsprogramme von der Komplexität der unteren Schichten des TCP/IP-Modells abzuschirmen und ihnen eine einheitliche Programmierschnittstelle zu bieten. Die Anwendungsschicht kommuniziert mit dem Application Program Interface (API) des jeweiligen Betriebssystems, um die Kommunikationsfähigkeit eines Programms mit Netzwerkfunktionalität herzustellen. Die API ist generell betriebssystemabhängig[2]. Den Applikationsprogrammen werden Systemaufrufe zum Erzeugen und Zerstören von Sockets[3], deren Binden an eine Netzwerkadresse, dem Aufbau einer verbindungsorientierten oder verbindungslosen Kommunikation, dem Senden und Empfangen von Daten, dem Setzen von IP-Adressen uvm. zur Verfügung gestellt.

3.2 Client-Server - Protokolle

Das Client-Server-Prinzip fußt auf der Verfügbarkeit einer hochleistungsfähigen, breitbandangebundenen zentralen Instanz, die Dienste anbietet (Server). Programme auf Stationen mit geringer Rechenleistung und Bandbreite (Clients) stellen Verbindungen zu den Servern her und nehmen die offerierten Dienste in Anspruch, typischerweise Berechnungen oder Download von Dateien. Server warten auf reservierten Ports der Transportschicht auf Anfragen von Clients. Dieses Konzept hat sich als skalierbar erwiesen. In erster Ausbaustufe existierten hauptsächlich Architekturen aus zwei Schichten: Benutzerschnittstelle (Client) sowie Datenzugriff und -Speicherung (Server). Mit dem zunehmenden Datenvolumen greifen immer mehr

[2]UNIX-Derivate kommunizieren über die Socket-Schnittstelle, Microsoft Windows verwendet Winsocks.

[3]Ein Socket ist ein ganzzahliger Platzhalter, der aus der Angabe von Netzwerkschichtprotokollfamilie (IP im TCP/IP-Modell), Verbindungsart (SOCK_STREAM, SOCK_DGRAM, SOCK_RAW) und Transportschichtprotokoll berechnet wird.

Applikationsserver ihrerseits auf Datenbankserver zurück. Die resultierende Architektur besteht somit aus drei Schichten: Client (Nutzerschnittstelle), Applikationsserver (Zugriff) und Datenbankserver (Speicherung). Eine Verallgemeinerung zur N-Schicht-Architektur, in der Applikationsserver untereinander kommunizieren, ist prinzipiell möglich.

3.2.1 Hypertext Transfer Protocol

Das Hypertext Transfer Protocol (HTTP) ist ein generisches, zustandsloses Protokoll zur verteilten Speicherung von Hypermedia-Dokumenten, die unter sog. *Uniform Resource Identifiers* (URI) internetweit eindeutig abgelegt sind. Seine Vielseitigkeit ermöglicht den Einsatz über das World Wide Web hinaus; mit seiner Hilfe kommunizieren auch einige Peer-to-Peer-Protokolle. Es nutzt TCP als Transportprotokoll. WWW-Server nehmen Verbindungen üblicherweise auf Port 80 entgegen.
HTTP basiert auf dem Anfrage-Rückmeldung (*request-response*)-Prinzip. Der Client sendet ein Anfrage der Form

Methode der Anfrage	URI
Protokollversion	Methodenerweiterung (optional)
Clientversion	Payload (optional)

Abbildung 3.4: prinzipieller Aufbau einer HTTP-Clientanfrage

Methode ist eine der in Tab. 3.1 aufgeführten Zeichenketten:

Methode	Bedeutung
OPTIONS	Bestimmung der Fähigkeiten des Servers bzw. der Bedingungen, unter denen eine Ressource zugänglich ist
GET	Anforderung der von der URI referenzierten Ressource
HEAD	Anforderung der Metainformationen über die von der URI referenzierten Ressource
POST	Übergabe der Dateneinheit im Argument als Erweiterung der durch die URI identifizierten Ressource
PUT	Ersetzung der durch die URI referenzierten Ressource durch mitgesendete Dateneinheit bzw. Erzeugen
DELETE	Löschen der URI-Ressource
TRACE	„Schleife“ für Diagnosezwecke
CONNECT	Aufbau von Verbindungen mittels Proxies zur Ressource

Tabelle 3.1: Methoden des HTTP

Die URI ist ein relativer

```
/pub/WWW/TheProject.html
```

oder absoluter

```
http://www.w3.org/pub/WWW/TheProject.html
```

Pfad zur Ressource. In HTTP 1.1 wird das Protokollversion-Feld als

```
HTTP/1.1
```

dargestellt, für Version 1.0 lautet die Zeichenkette entsprechend. Client-Informationen und Erweiterungen der Methoden werden in *Request Headers* übertragen. Möglich sind

```
Accept, Accept-Charset, Accept-Encoding, Accept-Language, Sec-
tion, Authorization, Expect, From, Host, If-Match, If-Modified-
Since, If-None-Match, If-Range, If-Unmodified-Since, Max-Forwards,
Proxy-Authorization, Range, Referer, TE, User-Agent[RFC2616].
```

Der Server antwortet mit

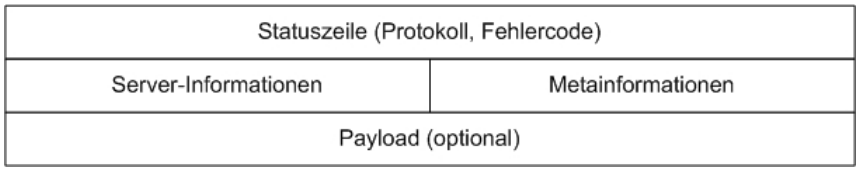


Abbildung 3.5: prinzipieller Aufbau einer HTTP-Servermeldung

Die Statuszeile besteht aus Protokollversion, einem dreistelligen Fehlercode sowie seiner textuellen Interpretation, z.B.

```
HTTP/1.1 200 OK
```

Die erste Ziffer des Fehlercodes dient zu dessen Einordnung; die restlichen Ziffern spezifizieren die genaue Fehlermeldung.

Fehlercode	Bedeutung
1	Anfrage erhalten
2	Aktion erfolgreich ausgeführt
3	Aktion teilweise erfolgreich, weitere Aktionen erforderlich
4	Fehler des Client
5	Server-Fehler

Tabelle 3.2: (Fehler-) Codes des HTTP

Server- und Metainformationen der Ressource werden analog zum Request Header im *Response Header* übertragen. Es existieren die Schlüsselwörter

```
Accept-Ranges, Age, ETag, Location, Proxy-Authenticate, Retry-
After, Server, Vary, WWW-Authenticate[RFC2616].
```

Der Payload ist nochmals in *Entity-Header* und *Entity-Body* unterteilt. der Entity-Header liefert weitere Informationen zum Inhalt und umfaßt

```
Allow, Content-Encoding, Content-Language, Content-Length, Content-
Location, Content-MD5, Content-Range, Content-Type, Expires,
Last-Modified, extension-header[RFC2616].
```

Alle Meldungen werden im Klartext ausgetauscht. Daher ist einfaches String-Matching die einfachste Erkennungsmethode. Eine Unterscheidung zu Peer-to-Peer-Programmen, die HTTP benutzen, ist lediglich durch genauere Auswertung des HTTP-Headers möglich, vgl. Kap 3.3.

3.2.2 File Transfer Protocol

Das File Transfer Protocol wurde 1973, zur Zeit des ARPANET, spezifiziert und ermöglicht den plattform- und betriebssystemunabhängigen Dateiaustausch über zuverlässige Ende-zu-Ende-Verbindungen wie beispielsweise das Transmission Control Protocol. Es bietet unverschlüsselte Authentifizierung und Dateiübertragung. Die meisten Implementierungen des FTP stellen eine interaktive, kommandozeilenbasierte Schnittstelle zur Verfügung, mit der der Nutzer unter anderem Dateien auflisten oder herauf- bzw. herunterladen kann.
FTP-Server erlauben gleichzeitige Sitzungen mit mehreren Clients. Hierbei kommt das Master-Slave-Konzept zur Anwendung, wobei ein Master nach Bedarf Slaves erzeugt, steuert und terminiert. Zunächst erwartet der Master-Server auf TCP-Port 21 eingehende Verbindungen. Da TCP Verbindungen als (Quell-IP:Quell-Port $\longrightarrow$ Ziel-IP:Ziel:Port) identifiziert, sind gleichzeitig mehrere Verbindungen auf dem lokalen Server-Port möglich. Akzeptiert der Server die Verbindung, antwortet er mit

```
220 <Klartextmeldung>
```

Nun erfolgt die Authentifikation mit

```
USER <Benutzername>
PASS <Passwort>
```

und die Sitzung beginnt. Sobald vom Client Daten angefordert werden (z.B. mittels des *get*- oder des *ls*-Kommandos, erzeugt der Master-Server auf dem lokalen TCP-Port 21 einen Slave-Server, welcher die weitere Kommunikation übernimmt. Um trotzdem die steuernde (mit TCP-Port 21 des Servers verbundene) Sitzung zu erhalten, muß der Client ein neues Socket erzeugen und den entsprechenden Port dem Server mit dem *port*-Befehl mitteilen. Der neu erzeugte Slave-Server kontaktiert über den lokalen Port 20 den Client auf dem mitgeteilten Port. Diese Technik wird als *Callback-Mechanismus* bezeichnet. Fordert der Client eine Datei an, erzeugt der Slave-Server einen Hilfsprozeß, der den eigentlichen Transfer übernimmt und zerstört ihn danach. Der letzte Schritt wiederholt sich für jede Datei. Bild 3.6 verdeutlicht das Konzept.

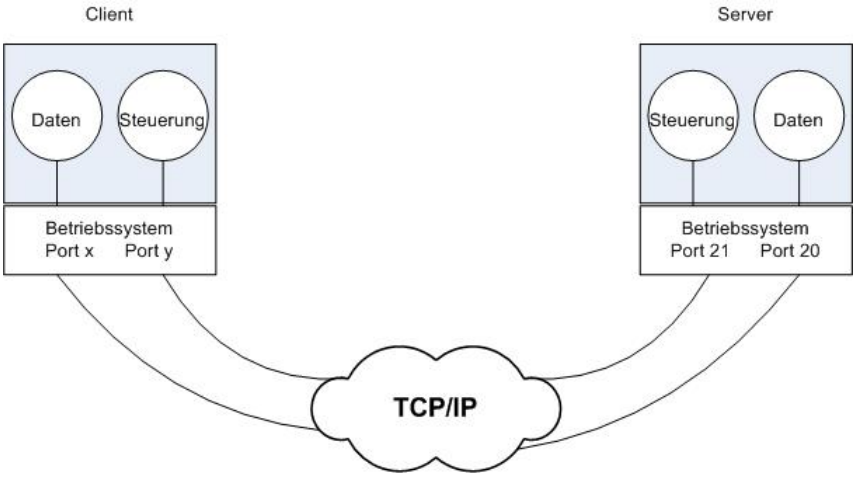


Abbildung 3.6: Funktionsweise von FTP

Wird die steuernde Sitzung unterbrochen, beenden Client und Server alle laufenden Transferprozesse zwischen beiden Kommunikationspartnern. Die Rückmeldungen vom Server erfolgt mittels dreistelliger Zahlencodes, welche im interaktiven Betrieb vom lokalen Client für den Benutzer in menschenlesbaren Text übersetzt werden, s. Tab 3.3.

Ziffer	primäre Bedeutung (Zeitpunkt)	sekundäre Bedeutung (Befehlsart)
0	nicht spezifiziert	Befehlssyntax
1	Befehl positiv begonnen	allgemeine Information
2	Befehl positiv beendet	Verbindungszustand
3	Zwischenstand positiv	Authentifizierung/ Accounting
4	transientes Problem, Befehl wiederholen	nicht spezifiziert
5	definitives Problem, Befehl nicht wiederholen	Dateisystem

Tabelle 3.3: (Fehler-) Codes des FTP

Die dritte Ziffer gibt weitere protokollspezifische Informationen an. FTP-Sitzungen lassen sich wie folgt erkennen.

- Eine Auswertung des Payload ist aufwendig, da Verwechslungsgefahr mit SMTP besteht. Typisch sind die USER- und PASS-Pakete.
- Es ist erfolgversprechend, FTP-Server zu identifizieren und alle involvierten Flüsse als FTP-Sitzungen zu deklarieren. Als FTP-Server werden alle Knoten identifiziert, die ausschließlich über die TCP-Ports 20 und 21 kommunizieren[4].

[4]Als denkbare (und implementierte) Erweiterung sind weitere wohlbekannte Ports (z.B. TCP-Port 80) ebenfalls zugelassen, um die Erkennung mehrerer Server auf einem Computer zu ermöglichen. Die Identifikation eines Flusses erfolgt dann anhand seiner Portnummern.

3.2.3 Electronic Mail

Elektronische Post ist das bevorzugte Medium zum schnellen, zuverlässigen und bequemen Austausch von personalisierten Nachrichten. Die Besonderheit des Prinzips von E-Mail liegt in dessen dezentraler Organisation, die eine Kommunikation der E-Mailserver untereinander erfordert. Dabei kommt das selbe Protokoll wie beim Absenden einer E-Mail durch den Benutzer zum Einsatz. Aus Gründen der Komplexität werden für Versand und Empfang von E-Mails verschiedene Protokolle verwendet.

3.2.3.1 Simple Mail Transfer Protocol

Das Simple Mail Transfer Protocol beschreibt ausschließlich einen Standard für den Austausch von Mail zwischen Netzknoten. Befehle werden, ebenso wie die zu übermitelnden Nachrichten, in ASCII codiert. Der erste Schritt des Absenders besteht im Aufbau einer TCP-Verbindung zum Mailserver, gewöhnlich auf Port 25. Ist der Server zur Annahme von Mail bereit, antwortet er mit der Meldung „220 (READY FOR MAIL)“ , eventuell gefolgt von einer Zeichenkette. Der Client kann sich nun mit dem Befehl HELO (hello) identifizieren; im Erfolgsfall bestätigt der Server mit dem Code „250 OK“ . Das Verschicken einer E-Mail beginnt mit dem MAIL - Befehl in der Form

```
MAIL <Sender-ID> FROM: <reverse-path> <CR><LF>.
```

An den Reverse-Path werden Fehler gemeldet. Akzeptiert der Server, sendet er

```
250 OK.
```

Im zweiten Schritt wird der Empfänger mit

```
RCPT <Sender-ID> TO:<forward-path> <CR><LF>
```

angegeben. Dieser Schritt kann für beliebig viele Empfänger ausgeführt werden. Mit den Forward- und Reverse-Pfaden können Source-Routing-Listen festgelegt werden, so daß Nachrichten z.B. nur über als vertrauenswürdig eingestufte Mailserver zur Ziel-Mailbox weitergeleitet werden. Der Server quitiert im Erfoglsfall wiederum mit

```
250 OK.
```

Der Mailkörper beginnt nach dem DATA - Befehl:

```
DATA <CR><LF>.
```

Vom Server wird die Antwort

```
354 Intermediate
```

erwartet. Je Zeile sind 1000 Zeichen erlaubt. Das Ende des DATA-Blocks bildet eine Zeile, die lediglich einen Punkt enthält. Alle weiteren Felder wie *Date, Subject, To, Cc, From* stehen ebenfalls im DATA-Block. SMTP verarbeitet lediglich 7-Bit-ASCII-codierte Zeichen. Im allgemeinen Fall befindet sich die Mailbox des Adressaten nicht auf dem Server, der die E-Mail vom Absender übernimmt. Dann wird der initiale Mailserver wiederum zum Client, der den Ziel-Mailserver kontaktiert.
Das heute mehrheitlich verwendete ESMTP (Extended SMTP) ermöglicht neben einer Erweiterung des Zeichensatzes auf 8-Bit-ASCII die Einbettung beliebiger Datentypen in E-Mails. Sicherstes Erkennungsmerkmal ist die Meldung EHLO anstelle von HELO.

- Charakteristisch sind die HELO- bzw. EHLO-Pakete am Anfang einer Konversation.
- Es ist erfolgversprechend, Mailserver zu identifizieren und alle involvierten Flüsse als E-Mail zu deklarieren. Als Mailserver werden alle Knoten identifiziert, die ausschließlich über den TCP-Port 25 kommunizieren oder deren Konversationspartner diesen Port benutzt.

3.2.3.2 Post Office Protocol

Das in [RFC1939] beschriebene Post Office Protocol der aktuellen Version 3 dient der Interaktion zwischen User Agent und Mailserver. Seine Aufgabe besteht lediglich in der Abholung von E-Mails vom Server; das Verschicken übernimmt das Simple Mail Transfer Protocol. Der POP-3-Server wartet per default auf TCP-Port 110 auf eingehende Verbindungen. Bei erfolgreichem Aufbau sendet der Server mittels des Network Virtual Terminal (NVT) eine Begrüßungsmeldung der Form

```
+OK POP3 server ready
```

Der Client muß sich nun mit der Nachricht

```
USER <Nutzername> PASS <Passwort>
```

identifizieren. Mit den Befehlen `LIST, RETR` und `DELE` kann der Benutzer E-Mails auflisten, abrufen und löschen. Der gesamte Befehlsvorrat umfaßt die Schlüsselwörter

```
USER, PASS, QUIT, STAT, LIST, RETR, DELE, NOOP, RSET, QUIT, APOP,
TOP, UIDL, +OK, -ERR.
```

Befehle werden mit der Kombination <CR><LF> abgeschlossen.

3.2.3.3 Internet Message Access Protocol

IMAP ähnelt dem Post Office Protocol im Befehlsformat und nutzt den TCP-Port 143. Es bietet dem Benutzer die Möglichkeit, Ordnerstrukturen auf dem Server anzulegen und E-Mails dort aufzubewahren. Da die Authentifikation an POP angelehnt ist, erfolgt keine Unterscheidung zwischen beiden Protokollen.

3.2.4 Remote Access

Gesicherte Ende-zu-Ende-Verbindungen auf der Transportebene ermöglichen interaktive Kommunikation zwischen zwei über ein Netzwerk verbundenen Computern. In diesem Abschnitt werden die Protokolle Telnet und Secure Shell vorgestellt.

3.2.4.1 Telnet

Obwohl Telnet als veraltet zu betrachten ist, da keinerlei Verschlüsselungsmechanismen implementiert sind, bildet sein Design die Grundlage von moderneren Terminalprogrammen wie SSH. Telnet erlaubt als Teil der TCP/IP-Suite die Fernsteuerung von Computern mittels einer textbasierten Schnittstelle über das Internet. Der Telnet-Client stellt eine TCP-Verbindung zu einem Telnet-Server her, welcher diese typischerweise auf Port 23 entgegennimmt. Die Eingaben des Clients können von jeder beliebigen Quelle stammen, erfolgen normalerweise jedoch von der Standardeingabe. Steuerbefehle und Statusmeldungen werden in Klartext übertragen. Telnet schickt jedes eingegebene Zeichen transparent zum Server und zurück. Es werden also auf Transportebene sehr viele kleine Segmente übertragen, wenn eine interaktive Sitzung besteht. Der Telnet-Service funktioniert ebenfalls nach dem Master-Slave-Prinzip, um mehrere Verbindungen gleichzeitig handhaben zu können. Bild 3.7 illustriert das Prinzip.

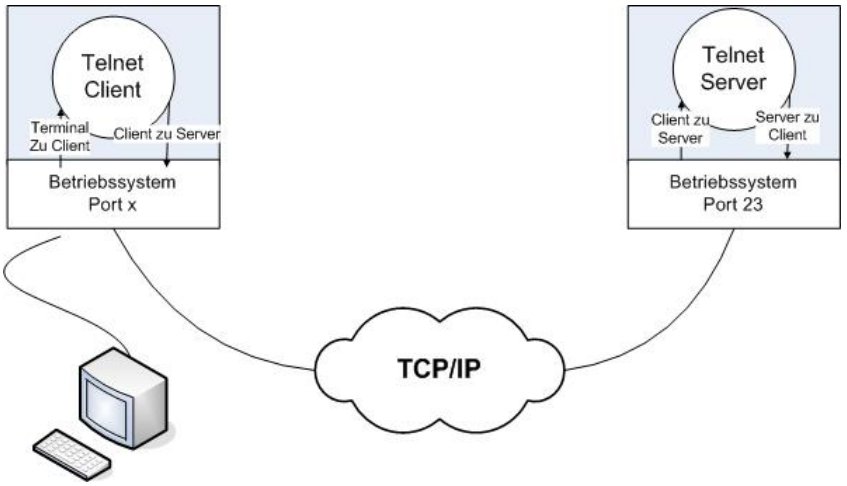


Abbildung 3.7: Funktionsweise von Telnet

Telnet besteht aus zwei für den Nutzer sichtbaren Diensten. Das *Network Virtual Terminal* definiert eine Standardschnittstelle für Clients. Diese reduziert die Komplexität der Clientsoftware und stellt ihr ein codierungsunabhängiges Textinterface bereit. Serverseitig geschieht die Formatumwandlung in das dort verwendete Format. Durch diesen Mechanismus wird die Eingabe geräteunabhängig, d.h., sie kann von beliebigen Quellen wie z.B. aus einer Datei stammen. Die *Options Negotiation* erlaubt den Austausch von Verbindungsoptionen, z.B. ob die übertragenen Daten als 7-Bit-ASCII- oder 8-Bit-ASCII-Codierung zu interpretieren sind.

3.2.4.2 Secure Shell

Die Secure Shell ist als ein um Verschlüsselung und abhörsichere Authentifizierung erweitertes Telnet anzusehen. Der SSH-Terminalserver nimmt Verbindungen typi-

scherweise auf TCP-Port 22 entgegen. Nach dem Drei-Wege-Handshake sendet der Server die Zeichenkette

```
SSH-<Protokollversion-Programmname>
```

.

Der Client antwortet nach dem gleichen Muster. Im nächsten Schritt bietet der Server dem Client eine Liste von Authentifizierungsmethoden im kommasepariertem Klartext an. Das allgemeine Paketformat ist

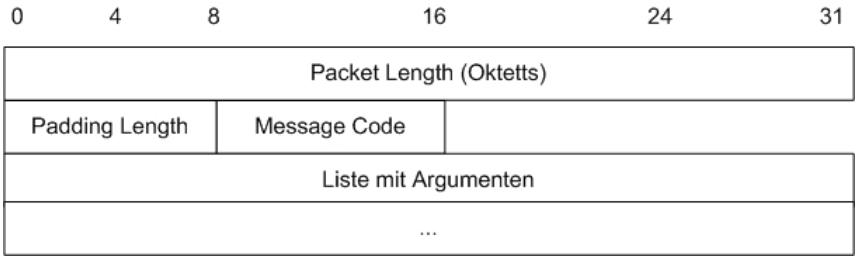


Abbildung 3.8: Paketformat des SSH-Dienstes

In der Längenangabe werden die vier Oktetts dieses Feldes nicht berücksichtigt. Ein Paket kann mehrere Nachrichten beinhalten. Mittels Padding wird die Länge jeder Nachricht auf ein Vielfaches von 16 gebracht. Der Client stößt die Authentifizierung an, indem er den dem Algorithmus entsprechenden Message Code an den Server sendet. Dann werden die nach dem ausgewählten Algorithmus gebildeten Schlüssel ausgetauscht. War dieser Schritt erfolgreich, beginnt die verschlüsselte Sitzung. Die zur Erkennung nutzbaren Merkmale sind

- der serverseitig benutzte Port 22
- die Zeichenkette „SSH-‘im ersten Paket vom Client zum Server und umgekehrt.

3.2.5 Server Message Block

Server Message Block ist die Implementation der Datei- und Druckerfreigabe unter den Betriebssystemen von Microsoft. Es nutzt die TCP-Ports 445 und 135. Unter Linux stellt das Samba-Protokoll die Interoperabilität mit Computern, die Microsoft Windows verwenden, her. SMB ermöglicht auf Servern alle Dateisystemoperationen, die auch lokal möglich sind, z.B.

- Öffnen, Schreiben, Lesen und Schließen von Dateien
- Erzeugen, Suchen und Löschen von Verzeichnissen und Dateien
- Drucken

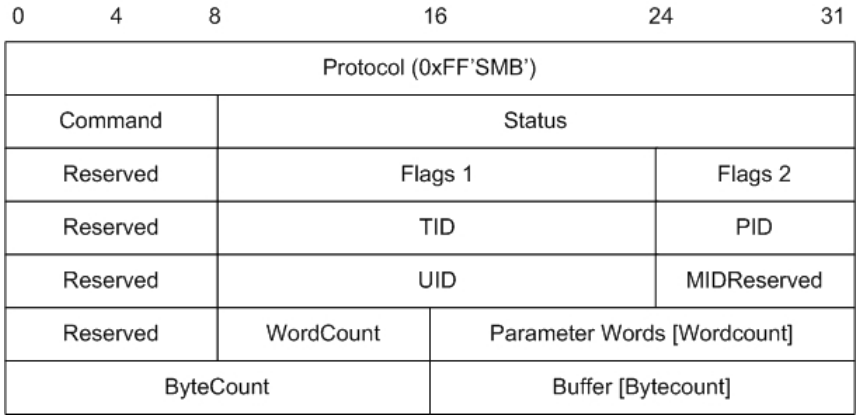


Abbildung 3.9: Aufbau des SMB-Protokollkopfes[MSDN05]

Bei Unternehmensnetzen, die über mehrere Standorte verteilt sind, ist das Auftreten des SMB-Protokolls zu erwarten. Bild 3.9 skizziert den Aufbau eines SMB-Pakets.

Unmittelbar nach dem Header steht eine unterschiedliche Zahl von Bytes, die ein SMB-Kommando oder eine Antwort darstellen. Jedes Kommando wie Open File (COM Feldkennung: SMBopen) oder Get Print Queue (SMBsplretq) hat seinen eigenen Satz von Parametern und Daten. Es müssen nicht alle Kommandofelder gefüllt werden; das hängt vom einzelnen Kommando ab.
SMB-Pakete sind aufgrund

- der Verwendung der TCP-Ports 445 und 135 und
- der Protokollkennung 0xFF'SMB'

erkennbar.

3.2.6 Network File System

Analog zu SMB ermöglicht das von SUN entwickelte NFS den netzwerktransparenten Dateizugriff. Im Unterschied zu anderen Client-Server-Protokollen verwendet NFS keine wohlbekannten Portnummern, da für jeden Client aus Performancegründen eine eigene Serverinstanz erzeugt wird. Die Kommunikation zwischen Client und Server findet zunächst über einen Portmapper, der auf TCP- oder UDP-Port 111 auf Verbindungen wartet, statt. Der Portmapper verwaltet mehrere Dienste, darunter NFS. Request- und Reply-Pakete sind in Abb. 3.10 und 3.11 angegeben.

Die Zuordnung von Anfrage und Antwort findet mit Hilfe der *Transaction-ID*, welche übereinstimmen muß, statt. Das *Program ID*-Feld codiert die geforderte Applikation[5]. Der Portmapper veranlaßt den Start einer entsprechenden Serverinstanz und informiert den Client über den entsprechenden Port. Da aktuelle Implementationen von NFS/Portmap sowohl TCP- als auch UDP-Verbindungen nutzen, und diese erst zur Laufzeit festgelegt werden, ist die Gefahr einer Verwechslung mit Peer-to-Peer-Protokollen gegeben (vgl. Kap. 5.3.2.4). Als Workaround in einer Analyse ohne Payloaduntersuchung ist es lediglich möglich, die IP-Adressen aller Flüsse mit

[5]NFS hat den Code 100003, das Mount-„Unterprotokoll“ 100005.

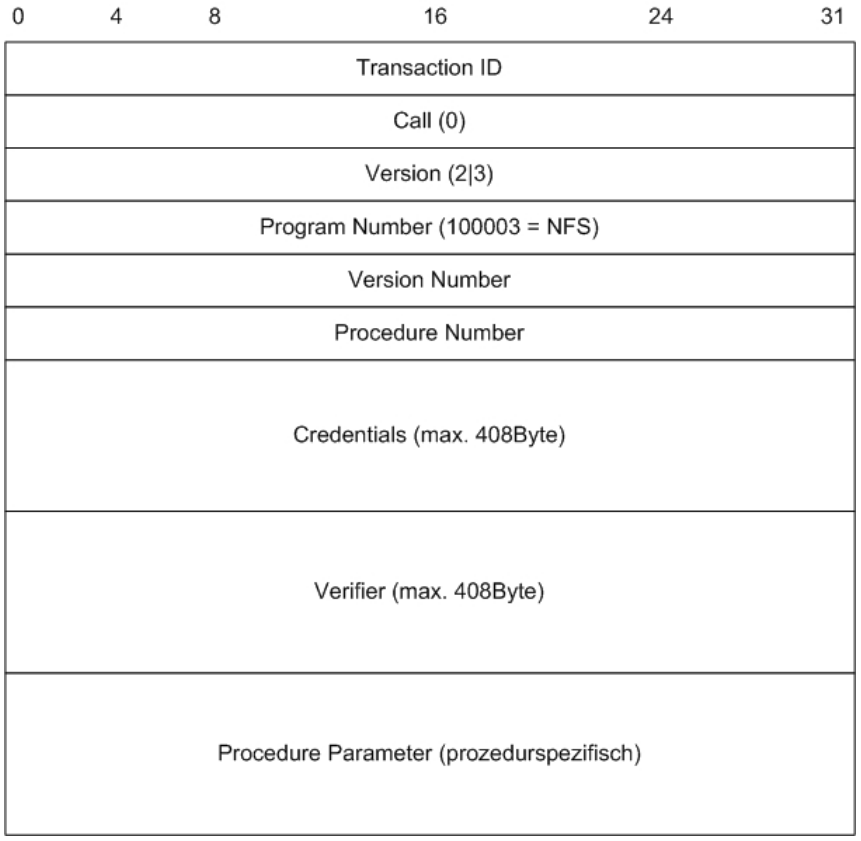


Abbildung 3.10: Aufbau einer Portmap-Anfrage[Ste00]

dem Zielport 111 in eine NFS-Paarliste aufzunehmen und Kombinationen dieser IP-Adressen als NFS-Flüsse zu kennzeichnen. Der Unterschied zu einem ebenfalls auf TCP- oder UDP-Port 111 laufendem Peer-to-Peer-Programm ist mit diesem Ansatz jedoch nicht möglich. Da die Nutzdatenübertragung jedoch meist auf UDP aufsetzt (Version 2), sind große UDP-Flüsse ein zusätzliches Indiz für das NFS-Protokoll.

3.2.7 Online-Gaming

Online-Gaming ist ein relativ neues Phänomen, dessen Verkehrsvolumen stetig anwächst. Der Trend geht in Richtung *Massive Multiplayer Online Role-Playing Games*. Als wichtige Vertreter gelten derzeit u.a. „World of Warcraft“ und „Dark Age of Camelot“. Der Großteil des erzeugten Gaming-Verkehrs terminiert auf dedizierten Servern. Da die große Anzahl von Neuveröffentlichungen und der schnelle Wechsel aktueller Spieletitel eine Identifikation der einzelnen, meist proprietären, Protokolle stark erschwert und auch nur für Marketingzwecke interessant sein dürfte, wurde eine Analyse auf der Applikationsschicht verworfen. Eine Betrachtung der von Online-Spielen häufig verwendeten UDP- und TCP-Ports sowie evtl. der Paketgrößen verspricht mehr Erfolg. Dedizierte Gaming-Server können aufgrund von Verbindungsmustern erkannt werden, wenn nur auf den für Online-Gaming festgelegten Ports einer IP-Adresse Datenverkehr stattfindet.
Andererseits existieren einige nichtkommerzielle netzwerkfähige Spiele, die ohne zentrale Server auskommen.

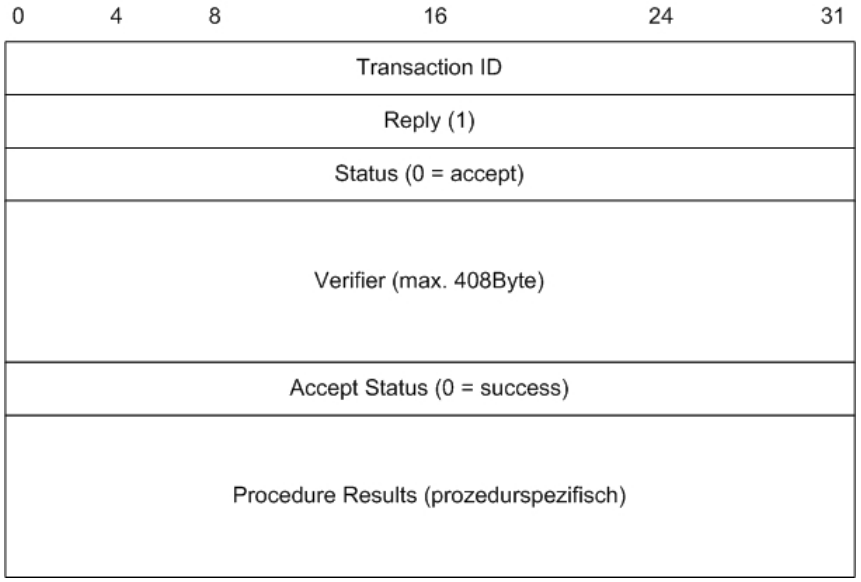


Abbildung 3.11: Aufbau einer Portmap-Antwort[Ste00]

3.2.8 Malware

Unter Malware ist jegliche Software mit expliziter oder impliziter Schadfunktion, welche sich über ein Netzwerk verbreitet oder es zur Ausführung seiner Schadfunktion nutzt, zusammengefaßt. Als Beispiele sind Würmer, Trojaner aber auch Portscans zu nennen. Eine Payloadanalyse ist aufgrund der relativen Kurzlebigkeit der meisten Schädlingsprogramme und ihrer großen Anzahl zum Scheitern verurteilt. Hier kann die Nonpayloadanalyse ihre Stärken ausspielen, indem die Verbindungsmuster auszuwerten sind. Portscans verraten sich durch sehr viele Verbindungen zwischen zwei IP-Adressen mit wechselndem Zielport, bzw. durch Verbindungen zwischen derselben Quell- und vielen Zieladressen bei einer geringen Zielportanzahl. Würmer ähneln Adreßscans insofern, als daß sie meist nur zu einem Zielport verbinden. Zur Identifikation dieser Erscheinungen sind umfangreiche Betrachtungen der Verbindungsmuster nötig, wobei eine entsprechend große Flußtabelle und Speicheraustattung der Analysehardware erforderlich sind.

3.3 Peer-to-Peer-Protokolle

Insbesondere bei bandbreitenintensiven Applikationen wie FTP und rechenzeitintensiven Aufgaben ergeben sich beim Client-Server-Modell teils gravierende Nachteile. Stellt ein Anbieter neue Software oder Patches bereit, entsteht kurzfristig eine Hoch- oder sogar Überlastsituation. Eine Aufstockung von Rechenkapazität und Bandbreite ist insbesondere für kleine Unternehmen sowie Non-Profit-Organisationen nicht finanzierbar. Eine Alternative besteht darin, jeden Konsumenten (Downloader) gleichzeitig zum Anbieter der gerade heruntergeladenen Datei zu machen und damit die Probleme Rechnerleistung und Bandbreite an den „Rand“ des Netzwerks auf viele Computer zu verteilen. Mit Hashing-Mechanismen (z.B. MD5) ist ein Schutz des angebotenen Inhalts gegen Manipulationen möglich. Vor allem das Bittorrent-Netzwerk wurde im Hinblick auf diese Einsatzmöglichkeit entwickelt.
Es existieren reine und hybride Formen von Peer-to-Peer-Netzen. Die hybriden Peer-to-Peer-Netze sind am häufigsten anzutreffen. Sie benötigen weiter die Dienste von

Servern, um Aufgaben wie die Suche von Dateien oder anderen Peers zu bewerkstelligen. Die Mehrheit der in den untersuchten Proben identifizierten Peer-to-Peer-Protokolle (z.B. eMule, Bittorrent, DirectConnect) ist hybrider Natur. Peer-to-Peer-Netzwerke im engeren Sinn kommen völlig ohne Server aus. Statt Clients und Servern gibt es lediglich Peers, die alle Aufgaben übernehmen. Als Beispiele sind GNUtella und KaZaA[6] zu nennen. Der Begriff des Peer-to-Peer-Netzwerks wird im weiteren Verlauf dieser Arbeit auf beide Formen angewandt.
Die Anmeldung in einem Peer-to-Peer-Netzwerk läuft nach einem festen Muster ab. Beim Start wählt die Software IP-Adresse und Port eines Superpeers[7] und verbindet sich mit ihm. Der Superpeer verbreitet die Erreichbarkeitsinformationen (IP,Port) sowie evtl. die Liste der freigegebenen Dateien im Netzwerk (Abb. 3.12).

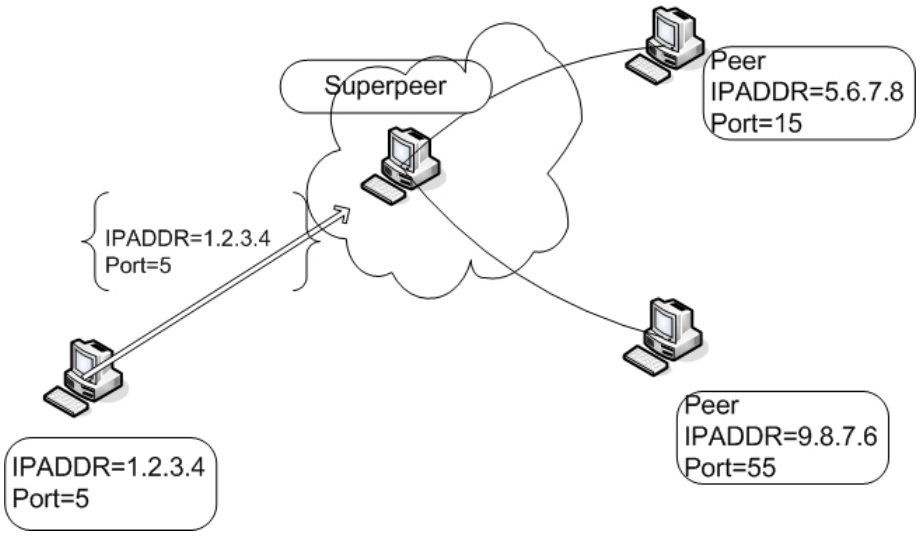


Abbildung 3.12: Verbindungsherstellung zu einem Peer-to-Peer-Netzwerk - Anmeldung

Kein Superpeer besitzt die vollständigen Informationen über alle Teilnehmer, da die Anforderungen an Übertragungsbandbreite und Rechenkapazität nicht zu bewältigen wären. Es wird ein teilvermaschtes Netz unter den Superpeers aufgebaut, in das der einzelne Teilnehmer entsprechend geringen Einblick hat, vgl. Abb. 3.13.

Eine vollständige Durchsuchung des Netzes ist wegen der großen Verbreitung von begehrten Dateien meist nicht nötig. Wird eine bestimmte Trefferzahl nicht erreicht, veranlassen manche Protokolle die Weiterleitung der Anfrage an andere Superpeers. Die Stärke von Peer-to-Peer-Netzwerken liegen in der sehr kostengünstigen Verbreitung von Dateien, da jeder Konsument gleichzeitig Anbieter zumindest von Teilen der gerade von ihm heruntergeladenen Datei(en) ist.
Der größte Anteil der über Peer-to-Peer-Netzwerke ausgetauschten Inhalte umfaßt z. Zt. urheberrechtlich geschützte Inhalte. Aufgrund des in jüngster Zeit verschärften Vorgehens von Film-, Musik- und Softwareindustrie gegen Peer-to-Peer-Tauschbörsen

[6]Beim Anmelden im Kazaa-Netzwerk wird eine HTTP-Verbindung zu einem Content-Server aufgebaut. Ebenso wie vom in der originalen Kaza-Distribution enthaltenen Programm p2pnetworking.exe unterhaltene Verbindungen bleibt sie unberücksichtigt

[7]Superpeers verfügen über Informationen, die freigegebene Dateien, IP-Adressen und Portnummern teilnehmender Peers umfassen.

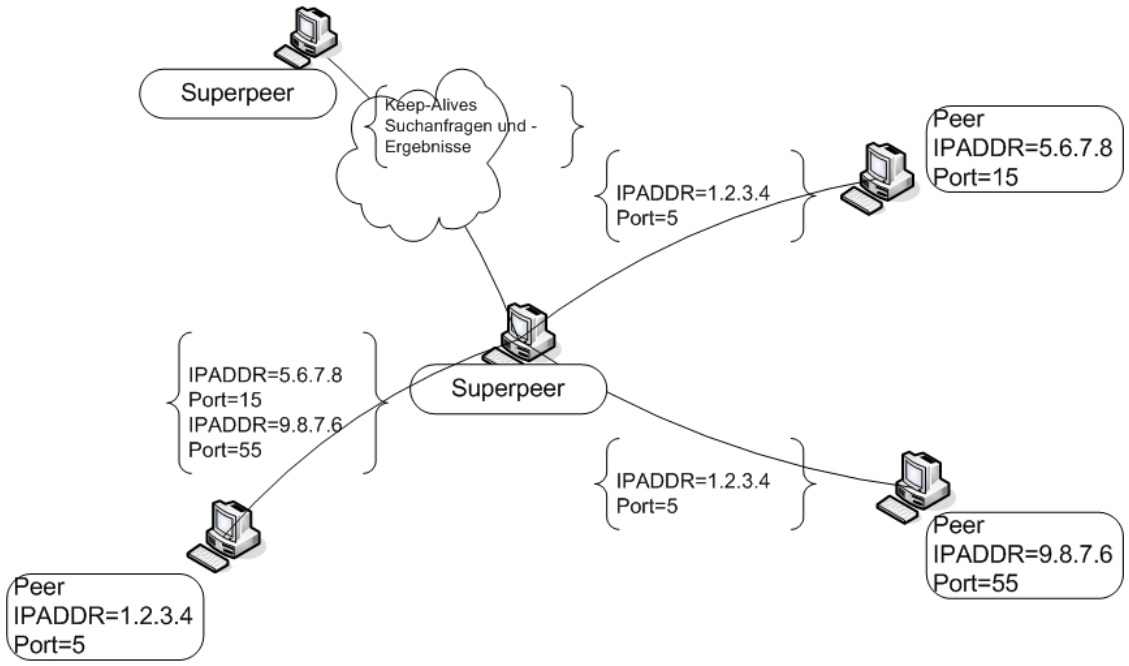


Abbildung 3.13: Verbindungsherstellung zu einem Peer-to-Peer-Netzwerk - Suche

sind diese verstärkt ins Zwielicht der Illegalität gerückt worden. Versuche, diese Technologie auch kommerziell zu nutzen, werden z.B. von der Firma Arvato Systems [Hei05] vorbereitet.
Aus im Literaturverzeichnis angegebenen Quellen übernomme Protokollspezifika wurden mittels des Netzwerkanalyseprogramms Ethereal auf einer Windows 2000-Plattform verifiziert bzw. ergänzt.

3.3.1 Fasttrack

Das Fasttrack-Protokoll war vor dem Aufstieg von eDonkey und Bittorrent das meistgenutzte Peer-to-Peer-Protokoll mit bis zu 2,5Mio. Nutzern [Hei04]. Die verbreitetsten Clients sind *Kazaa* und das spywarefreie Kazaa lite. Beide Implementationen sind proprietär, nicht offiziell dokumentiert und kommunizieren teilweise verschlüsselt. Das Fasttrack-Protokoll bildet ein verteiltes, selbstorganisierendes Netzwerk. Clients mit leistungsfähiger Hardware und breitbandiger Netzanbindung werden als *Supernodes* ausgewählt. Sie koordinieren Suchanfragen. Jeder Client muß eine Verbindung zu mindestens einem Supernode herstellen; dazu befinden sich einige IP-Adressen von Supernodes im Host Cache der Clients. Der Client sendet ein ICMP-ECHO-REQUEST an jede IP-Adresse in seinem Host-Cache. Dann sendet er an antwortende Hosts je ein UDP-Paket mit wechselnden Zielports und dem Inhalt

```
0x27 0x00 0x00 0x00 0x29 0x80 KaZaA 0x00
```

TCP-Verbindungen werden für Suche, Anforderung und Übertragung von Dateien verwendet. Typische Signaturen lauten

```
Get /.hash
Get /.files
```

```
UserAgent: KazaaClient
Server: KazaaClient
X-Kazaa-
```

Diese Signaturen wurden unter Verwendung des Programms KaZaA-Lite mit Hilfe von Ethereal ermittelt. Ein Test des Payload auf diese Zeichenketten verspricht Erfolg.

3.3.2 eDonkey/ eMule

eDonkey-Netze bestehen aus Clients und Servern. Clients halten lokal eine Liste mit Servern vor, von denen sie beim Programmstart zufällig einen zur Verbindung via TCP auswählen. Serverlisten werden über HTTP aktualisiert und enthalten auch Informationen über die Ports, auf denen die Server Verbindungen akzeptieren. Der eDonkey-Server nimmt Suchanfragen entgegen und fordert von den Clients gleichsam Informationen über freigegebene Dateien und vorhandene Dateiteile an. Eine Protokollerweiterung ermöglicht Suchanfragen an mehrere Server via UDP. Die Server liefern Listen der Peers zurück, die über gesuchte Dateiteile verfügen. Der suchende Peer baut dann TCP-Verbindungen zu jedem anderen Peer auf, der gesuchte Dateiteile anbietet. Dateien werden unter eDonkey in sog. *Chunks* der Größe 9,32MiB eingeteilt, welche über das Warteschlangenprinzip mit Priorisierung ausgetauscht werden. Signalisierungsinformationen werden mittels eines fixen, 6 Byte langen Header-Musters ausgetauscht, vgl. 3.14.

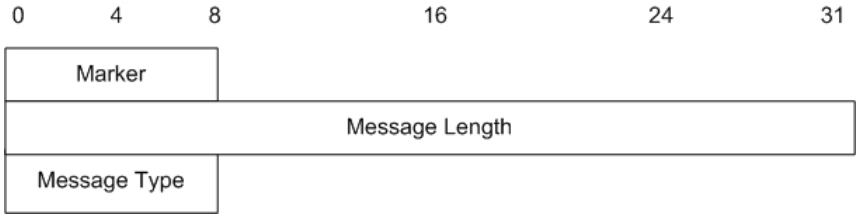


Abbildung 3.14: Aufbau eines eDonkey-Pakets mit Signalisierungsinformationen

Der Marker hat den Wert 0xE3 für das eDonkey-Protokoll bzw. 0xC5 für die eMule-Erweiterungen (TCP/UDP), 0xE4 (UDP) sowie 0xD4 für komprimierte eMule-Daten. Das nächste Feld, welches die Paketlänge in Network Byte Order angibt, schließt sich und den Marker von der Angabe aus. Das letzte Feld enthält die Art der Meldung. Einige dieser Message Types sind in Tab. 3.4 zu finden.

Dem Message-Type-Feld folgen kontextabhängig Felder variabler Länge und Anzahl. Die Identifizierung von eDonkey-Signalisierungspaketen kann folgendermaßen zuverlässig erreicht werden.

- Das erste Byte des Transportschicht-Payload enthält eines der Marker-Bytes.
- Das sechste Byte wird mit einer Liste, die alle Message-Typen enthält, abgeglichen.
- Der Inhalt des Packet-Length-Feldes wird mit der Länge des Payload abzüglich 5 auf Gleichheit geprüft. Es muß kleiner als die theoretisch maximal mögliche Länge eines IP-Pakets von 65535Byte sein.

Code	Bedeutung
0x01	OP_LOGINREQUEST OP_HELLO
0x38	OP_SERVERMESSAGE
0x40	OP_IDCHANGE
0x15	OP_OFFERFILES
0x14	OP_SERVERSTATUS
0x32	OP_SERVERLIST
0x41	OP_SERVERIDENT
0x16	OP_SEARCHREQUEST
0x46	OP_SENDINGPART
0x4C	OP_HELLOANSWER OP_SEARCHRESULT

Tabelle 3.4: Message-Types des eDonkey-Protokolls. Einige Codes besitzen eine Doppelbelegung, abhängig davon, ob eine Client-Server- oder eine Client-Client-Verbindung besteht.[KB05]

3.3.3 Bittorrent

Einer kürzlichen Meldung in [Hei04] zufolge belegt das in [Coh05] ausführlich dokumentierte Bittorrent bezüglich verursachtem Verkehrsvolumen den ersten Platz unter den Peer-to-Peer-Protokollen. Dieser Umstand findet in einer ausführlicheren Betrachtung des Protokolls seine Berücksichtigung.
Das Bittorrent-Protokoll besteht aus folgenden Komponenten:

- Datei, die Metainformationen über die bereitgestellte Datei(en) enthält (Metadatei, meist *.torrent)
- Webserver, auf dem die Metadatei abgelegt ist
- Web-Browser, der die Metadatei vom Webserver abholt
- Tracker, der Informationen über freigegebene und zum Download angeforderte Dateien der einzelnen Peers sammelt und weitergibt
- Bittorrent-Software, die die Einzelheiten des Up- und Downloads mit anderen Peers abwickelt.

In der Bittorrent-Terminologie wird zwischen *Serving Host* und *Client Host* unterschieden. Serving Hosts können Tracker, Webserver, Speicherort von Metadateien, oder sog. *Original Downloaders* sein. Original Downloaders, auch als *Origins* oder *Seeders* bezeichnet, sind Knoten, die erstmals eine Datei zum Download anbieten. Alle anderen Knoten, die das Bittorrent-Protokoll ausführen, werden als Client Hosts bezeichnet. Die Nutzerschnittstelle beschränkt sich darauf, den Speicherort für die angeforderten Dateien festzulegen. Heruntergeladene Dateifragmente dienen als Quellen für andere Client Hosts.
Das Bittorrent-Protokoll besteht aus zwei Unterprotokollen. Das *Tracker Protocol*

ist für die Kommunikation zwischen Downloader und Tracker, das *Peer Wire Protocol*[8] für den Nutzdatenaustausch zwischen den Peers zuständig. Der Download einer Datei läuft folgendermaßen ab. Durch die MIME-Verknüpfung des Metadateityps mit der Bittorent-Software wird diese durch Anklicken der Metadatei im Webbrowser auf dem Client Host gestartet. Die Software nimmt mit dem in der Metadatei angegebenen Tracker Kontakt auf. Der Tracker als vermittelnde Instanz hat Kenntnis von allen Peers, die in den Download einer Datei involviert sind und gibt ihnen eine zufällig generierte Liste ihrer Downloadpartner via HTTP oder HTTPS zurück. Die Downloader kommunizieren periodisch mit dem Tracker, um ihn über den Fortschritt ihres Downloads zu informieren. Der Nutzdatenaustausch zwischen den Downloaders hingegen erfolgt dezentral mittels Bittorrents *Peer Protocol*, welches auf TCP aufsetzt.
Bittorrent-Flüsse lassen sich entsprechend obiger Erläuterungen in die bidirektionalen Flüsse Downloader ↔ Tracker (Signalisierung) und Downloader ↔ Downloader (Signalisierung und Nutzdatenaustausch) einteilen.
Nachrichten zwischen Downloader und Tracker werden in einem sog. *bencoded*-Format ausgetauscht. Dabei handelt es sich um verschachtelte Listen (List) und Verzeichnisse (Dictionary), die Zeichenketten (Strings) und Ganzzahlen (Integer) enthalten. Tab. A.1 im Anhang gibt einen Überblick. Schlüssel von Verzeichnissen sind nach ASCII-Konventionen sortierte Strings.
Trackeranfragen sind bidirektional. Der Tracker fordert Informationen mit HTTP-GET an und antwortet im Bencoded Format. Der Tracker benötigt einen eigenen Webserver. Die GET-Anforderungen beinhalten die in Tab. 3.5 genannten Schlüssel.

Schlüsel	Wert
info_hash	20 Byte langer SHA1-Hash der bencodierten Form des info-Feldes aus der Metadatei
peer_id	20-stelliger String, den der Downloader für seine Identifikation benutzt
ip	IP-Adresse oder DNS-Name des Peers
port	Port, auf dem der Peer eingehende Verbindungen akzeptiert (TCP 6881..6889)
uploaded	Anzahl der hochgeladenen Bytes in ASCII-Codierung
downloaded	Anzahl der heruntergeladenen Bytes in ASCII-Codierung
left	Anzahl der Bytes, die der Peer noch herunterladen muß, codiert in ASCII
event	optionaler Schlüssel, der die Werte *started, completed* oder *stopped* enthalten kann, um den Tracker zu informieren

Tabelle 3.5: Schlüssel in GET-Anfragen des Trackers

Bittorrents *Peer Protocol* dient der Kommunikation zwischen den Peers. Es existieren vier globale Zustände, jeweils zwei für Uploader und Downloader, wobei ein Peer beide Rollen übernehmen kann. Die Zustände für den Uploader sind in Tab. 3.6 zusammengefaßt.

[8] auch Peer Protocol

Zustand	Bedeutung
choked	Uploader ist nicht zum Upload bereit
unchoked	Uploader ist zum Upload bereit
interested	Downloader benötigt Dateiteile des Uploaders
uninterested	Downloader benötigt keine Dateiteile des Uploaders

Tabelle 3.6: Zustände der Peers in Bittorrent

Eine Übertragung kommt nur in der Kombination *interested* - *unchoked* zustande. Nach der Zuweisung der Peers durch den Tracker ist der Downloader zunächst im Zustand *uninterested* und der Uploader *choked*.
Die Kommunikation im Peer Wire Protocol beginnt mit einem Handshake, dem ein Strom von (<Länge>:<Daten>)-Paaren in bencodierter Form folgt. Der Handshake beginnt mit der Zeichenkette „19 BitTorrent Protocol “. Es folgt ein 8 Byte langes mit „0 “ ausgefülltes Feld, dem sich der 20 Bit lange Hashwert des info-Schlüssels der Metadatei anschließt (vgl. A.2). Am Ende des Handshake steht die peer_id aus Tab. 3.5. Die Message beginnt mit einem Wert im Bereich 0..8 mit folgenden Bedeutungen.

Code	Name	Bedeutung
0	choke	Uploader ist nicht zum Upload bereit
1	unchoke	Uploader ist zum Upload bereit
2	interested	Downloader benötigt Dateiteile des Uploaders
3	not interested	Downloader benötigt keine Dateiteile des Uploaders
4	have	Index des fertiggestellten Dateiteils
5	bitfield	- nur als erste Message gesandt - enthält in Bitfeldern Indizes der bereits gesendeten Dateiteile
6	request	- gefolgt von *index, begin, length* - Anfordern eines Dateiteils
7	piece	- gefolgt von *index, begin, piece* - ...
8	cancel	- gefolgt von *index, begin, length* - am Ende von Downloads gesendet, um Verkehrsaufkommen zu reduzieren

Tabelle 3.7: Bittorrent Wire Protocol

Der Choking-Algorithmus in Bittorrent wird in dieser Arbeit nicht behandelt.
Die aus der Betrachtung des Protokolls gewonnenen Erkenntnisse werden kurz aufgeführt.

- Es wird ausschließlich TCP als Transportprotokoll eingesetzt.
- Die benutzten TCP-Ports liegen im Bereich 6881..6889.
- Die Signalisierung des Bittorrent Wire Protocol beginnt mit der Sequenz „0x13BitTorrent“.
- Es ist aufwendig, das Tracker-Protokoll zu identifizieren. Gleichzeitig wird von diesem nur wenig Verkehrsaufkommen verursacht.

3.3.4 GNUtella2

Bei GNUtella handelt es sich um ein selbstorganisierendes Ad-Hoc-Netzwerk, das teilnehmende Knoten verbindet. Für diese Knoten sind die Operationsmodi *Hub* (Radnabe) und *Leaf* (Blatt) vorgesehen. Blätter sind für die unmitelbare Funktionalität des Gnutella-Netzwerks nicht erforderlich, sondern stellen lediglich Dateien bereit bzw. fordern sie zum Download an. Sie verbinden typischerweise zu drei Hubs. Hubs sind als lokale Server für eine Anzahl von Blättern anzusehen, wobei auf ihnen dieselbe Software laüft wie auf den Blättern. Sie sind jeweils für ca. 300..500 Blätter zuständig und halten Verbindungen zu 5..30 anderen Hubs aufrecht. Diese vermaschte Hub-Struktur bildet ein *Hub Cluster*, die kleinste durchsuchbare Einheit. Die Hubs in einem Cluster tauschen untereinander Cache-Einträge, Filtertabellen und Statistiken aus. Die Gnutella-Software entscheidet selbsttätig, ob der Knoten, auf dem sie läuft, als Hub geeignet ist. Kriterien sind Hardwareausstattung, Betriebssystem, Laufzeit und verfügbare Bandbreite. Die Aufgaben eines Hubs umfassen

- Austausch von aktuellen Informationen mit allen Hubs im Cluster und deren benachbarten Hubs
- Verwalten einer Routingtabelle zur Gewährleistung der kürzesten Verbindung zwischen Knoten
- Aufbau von Hashtabellen zur Bearbeitung von Suchanfragen
- Beobachtung der lokalen Verbindungen und ggf. Änderung des Betriebsmodus auf Leaf

Der Verbindungsaufbau der Knoten beginnt mit einem 3-Wege-Handshake. Der Knoten, der eine neue Verbindung aufbauen will, sendet einen Header der Form

```
GNUTELLA CONNECT/0.6
Listen-IP: 1.2.3.4:6346
Remote-IP: 6.7.8.9
User-Agent: Shareaza 1.8.2.0
Accept: application/x-gnutella2
X-Ultrapeer: False
```

Die Antwort des kontaktierten Knoten lautet

```
GNUTELLA/0.6 200 OK
Listen-IP: 6.7.8.9:6346
Remote-IP: 1.2.3.4
User-Agent: Shareaza 1.8.2.0
Content-Type: application/x-gnutella2
Accept: application/x-gnutella2
X-Ultrapeer: True
X-Ultrapeer-Needed: False
```

Der Initiator akzeptiert die Antwort mit

```
GNUTELLA/0.6 200 OK
Content-Type: application/x-gnutella2
X-Ultrapeer: False
```

Verbindungen können auch abgelehnt werden, beispielsweise mit der Meldung

```
GNUTELLA/0.6 503 Too many connections
```

[Sto03] Die Argumente variieren je nach Software und Version sowie IP-Adressen. Gnutella der Version 1 ist durch die Zeichenkette GNUTELLA/0.4 gekennzeichnet. Download-Anfragen beginnen mit der Zeichenkette

```
GET /<string>/<string>/HTTP
Connection: Keep-Alive
Range: byte=0-
User-Agent: <string>
```

Aus diesem Beispiel wird die Ähnlichkeit des Formats zu HTTP deutlich. Signalisierungspakete via TCP sind an den Zeichenketten „GNUTELLA“ , „GET“ , oder „HTTP“ erkennbar. In den letzteren Fällen sind die Pakete weiter auszuwerten. Es bieten sich die Zeichenketten

```
User-Agent: <Name>
UserAgent: <Name>
Server: <Name>
```

an. Das Argument <Name> umfaßt die Menge

```
LimeWire, BearShare, Gnucleus, Morpheus, MorpheusOS, XoloX, Mor-
pheusPE, gtk-gnutella, Aquisition, Mutella-0.3.3, Mutella-0.3.9b,
Mutella-0.4, Mutella-0.4.0, Mutella-0.4.1, MyNapster, Qtella,
AquaLime, NapShare, Comeback, Go, PHEX, SwapNut, Shareaza, Free-
Wire, Openext, Phex.
```

Die Suche in einer Liste und der Abgleich jedes Pakets ist aufwendig und fehleranfällig, da eine Erweiterung um zusätzliche Clients nur durch eine Vergrößerung der Liste herbeigeführt werden kann.
Sollen kleine Datenmengen in großen Zeiträumen übertragen werden, was für Signalisierung typisch ist, kommt die UDP-Erweiterung zur Anwendung. Der Header eines Gnutella-Datagramms umfaßt 8 Bytes, die wie folgt codiert sind:

```
#pragma pack(1)
typedef struct
{
CHAR szTag[3];
```

```
BYTE nFlags;
WORD nSequence;
BYTE nPart;
BYTE nCount;
}
GND_HEADER;
```

szTag besteht aus der Sequenz „GND (GNutella Datagram). Für nFlags sind bisher die Werte 0x01 (deflate) und 0x02 (acknowledge me) definiert. nSequence ist nur senderseitig eindeutig und dient zur Zusammensetzung fragmentierter Pakete, wobei nPart die Nummer des Fragments enthält. nCount enthält die Anzahl der Fragmente. Ist nCount 0, handelt es sich um ein Acknowledgement.
Die Erkennung von Gnutella-induziertem Verkehr kann wie folgt verlaufen

- Die standardmäßig benutzten TCP- und UDP-Ports sind 6346 und 6347.
- Die Zeichenkette „GNUTELLA“ identifiziert ein Paket eindeutig.
- Pakete, deren Payload mit „GET“ oder „HTTP“ beginnt, sind auf die Strings „UserAgent“ , „User-Agent“ und „Server“ mit den entprechenden Argumenten zu untersuchen.
- Der Payload von Gnutella-UDP-Paketen beginnt mit „GND“ .

3.3.5 Soulseek

Das Soulseek-Netzwerk verwendet - ähnlich Napster der Version 1 - lediglich einen zentralen Server und proprietäre Client-Software. Der Client baut nach dem Start eine TCP-Verbindung zum Server unter der URL www.slsknet.org auf und sendet ein HTTP-GET-Request.

```
GET /slskinfo HTTP /1.1
User-Agent: SoulSeek/<Version>
Host: www.slsknet.org
```

Akzeptiert der Server, antwortet er mit

```
HTTP/1.1 200 OK
<weitere Meldungen>.
```

Auf diesem Weg erfolgt die Prüfung des Clients auf Aktualität. Soulseek-Signalisierungspakete haben das in Bild 3.15 dargestellte Format.

Das Message-Length-Feld ist in Little-Endian-Notation codiert und schließt sich von der Längenangabe aus. Nach dem Message-Code-Feld folgt ein variabler Teil, der je nach Message Type Zeichenketten, einzelne Buchstaben und Ganzzahlen der Breite 32Bit enthalten kann, wodurch die Erkennung erschwert wird. Einige bekannte Message Codes und ihre Argumente sind in [Sole03] zu finden.

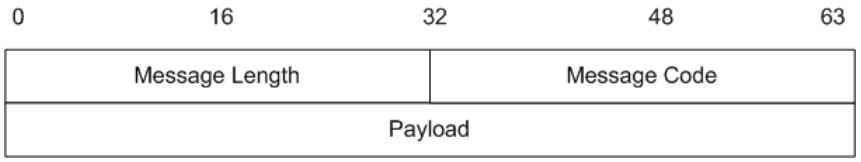


Abbildung 3.15: Grundsätzlicher Aufbau eines SoulSeek-Pakets mit Signalisierungsinformationen

Die Vorgehensweise zur Erkennung des SoulSeek-Protokolls ist aufgrund des Nachrichtenformats kompliziert, da typische Marker wie etwa bei eDonkey und Direct Connect fehlen und das Paketformat keine einheitliche Form aufweist. Darüberhinaus können mehrere Nachrichten in einem Paket zusammengefaßt sein, so daß die Auswertung des Message-Length-Feldes keinen Erfolg verspricht. Andererseits besteht keine Möglichkeit, die Porteinstellungen im Client zu ändern.

- SoulSeek verwendet die TCP-Ports 2234..2240 sowie 5534. UDP wird nicht verwendet.
- Für eine Payloadanalyse wären alle bekannten Message Types und deren Argumente gegen jedes Paket zu prüfen.

3.3.6 Direct Connect

Der ursprüngliche Direct-Connect-Client wurde von NeoModus unter Visual Basic entwickelt, ist mittlerweile jedoch fast vollständig von einer in Visual C++ geschriebenen Open-Source-Variante namens DC++ verdrängt worden. Das Direct-Connect-Netz besteht aus einigen Superhubs (Webservern), vielen Hubs und Clients. Hubs laufen unter einer speziellen Hubsoftware und benötigen eine breitbandige Internetverbindung. Sie skalieren bis zu Nutzerzahlen von ca. 5000. Ihre Verwaltung obliegt - ähnlich IRC-Kanälen - Operatoren. Hubs sind für die Koordination von Suchanfragen und Antworten zuständig und nehmen selbst nicht am Dateiaustausch teil. Einem Client muß beim ersten Start mindestens eine Superhub-Adresse bekannt sein, von der er eine aktuelle Hub-Liste im XML-Format beziehen kann. Die Kommunikation läuft unter Benutzung von HTTP ab; daher ist es schwierig, sie von regulärem Web-Verkehr zu unterscheiden.

```
GET /PublicHubList.config.bz2 HTTP/1.1
User-Agent: DC++ v0.688
Host: www.hublist.org
Connection: close
Cache-Control: no-cache
```

Die Antwort des Servers lautet

```
HTTP/1.1 200 OK
Date: Fri, 04 Mar 2005 14:12:14 GMT
```

```
Server: Apache/<Version>
<weitere serverspezifische Informationen>
<bz2-komprimierte Hubliste>
```

Der verwendete Client war DC++ Version 0.688. Nach dem Download der Hubliste kann der Client mittels TCP zu beliebig vielen Hubs unter Angabe eines eindeutigen Benutzernamens verbinden. Nach dem Drei-Wege-Handshake sendet das Hub die Sequenz

```
$Lock <Hub-Softwarename und -Version>|
```

Der Client sendet dem Hub die Erweiterungen des Direct-Connect-Protokolls, die er unterstützt.

```
$Supports <Liste der Kommandos>|$Key <Schlüssel>
|$ValidateNick <Nutzername>|
```

Akzeptiert das Hub, antwortet es mit

```
$Supports <Liste der Kommandos>|$HubName <Name> |
<Begrüßungsmeldung>|<Begrüßungsmeldung|>
```

Für Suchen ist das Kommando $Search zuständig. Suchanfragen werden an alle Hubs, mit denen der anfragende Client verbunden ist, ausgesendet und von diesen an alle verbundenen Clients weitergeleitet. Stellt der Client Treffer fest, sendet er die Meldung $SR gefolgt vom jeweiligen Benutzernamen, dem absoluten Pfad, ggf. dem Hashwert, sowie IP-Adresse und Port in einem UDP-Paket an den suchenden Client. Bei Interesse öffnet der Client eine TCP-Verbindung zum Peer.

```
$MyNick <Benutzername>|$Lock EXTENDEDPROTOCOL
ABCABCABCABCABCABC Pk=DCPLUSPLUS0.668ABCABC|$Supports <Erwei-
terungen von DC++>|$Direction Download <Zahl>|$Key <Wert>|
```

Der Uploader bestätigt mit

```
$Supports <Erweiterungen von DC++>|$Direction Upload <Zahl>|$Key
<Wert>|.
```

Der anfordernde Client sendet anschließend

```
$AdcGet tthl TTH/<TTH-Wert>|
```

an den Uploader, sofern AdcGet beiderseitig unterstützt wird. Für ältere Versionen ist $Get vorgesehen. Der Uploader bestätigt mit

```
$AdcSnd tthl TTH/<TTH-Wert>|,
```

wenn ein freier Upload-Slot vorhanden ist. Der Downloader sendet den $GetZBlock - Befehl, um den Download endgültig zu starten. Der Uploader antwortet mit $Sending und beginnt den Upload der angeforderten Datei.
Folgende Merkmale lassen sich zur Erkennung des Direct-Connect-Protokolls heranziehen.

- Die standardmäßigen TCP-Ports sind 411 und 1411, die UDP-Ports 412 und 1412.
- Die Signalisierung nutzt den ASCII-Zeichensatz und hat die Form „$<Befehl1> <Argument1> <Argument2> ...|$<Befehl2> <Argument1> ...|“ . In [KBFc04] ist eine unvollständige Liste der Befehle zu finden. Anstelle eines kompletten Stringvergleichs ist ein vereinfachtes Suchmuster, das den Beginn des Payload auf „$ “ gefolgt von einem Großbuchstaben und das Ende auf „|“ untersucht, empfehlenswert.
- Da die Kommunikation mit dem Superhub nur ein sehr geringes Verkehrsaufkommen verursacht, ist der Aufwand zur Unterscheidung von regulärem Web-Traffic als ungerechtfertigt anzusehen.

3.3.7 Chat

Chat ist eine Peer-to-Peer-Anwendung, wobei zumindest der Anmeldevorgang auf einem zentralen Server stattfindet. Da Chatprotokolle für den Austausch von Textnachrichten konzipiert sind und eine dementsprechend geringe Netzlast erzeugen, werden sie nur kurz umrissen. Besonders im IRC existieren jedoch Chaträume, die auf Dateiaustausch spezialisiert sind. Weiterhin bieten alle vorgestellten Chatprotokolle den Austausch von Dateien an. Da hierzu eine Direktverbindung zwischen den Komunikationspartnern aufgebaut wird, ist eine Erkennung nur über den Befehlssatz des jeweiligen Chatprotokolls möglich, da die Aushandlung der Verbindungsparameter noch über die Chatserver abläuft. Aufgrund der auf 12 Byte begrenzten Applikationsschichtdaten wurde dieser Versuch nicht unternommen.

3.3.7.1 Internet Relay Chat

Das 1989 entwickelte auf TCP aufsetzende *Internet Relay Chat* ist in den RFCs 2810 bis 2813 [FAQ03] beschrieben. Es besteht aus Servern, die ein IRC-Netzwerk bilden, und Clients. Die gesamte Kommunikation zwischen den Clients läuft auf ihren jeweiligen Servern zusammen. Nachdem der Client auf einem IRC-Server eingeloggt ist, kann der Benutzer einen Chatraum (Channel) betreten und dort mit anderen Teilnehmern des Channel Textnachrichten in Konferenzform austauschen. Jede Nachricht besteht in IRC aus bis zu drei Teilen: optionalem Präfix, Befehl und bis zu 15 Parametern, die jeweils durch ein Leerzeichen (0x20) getrennt sind. Der

Präfix wird durch einen Doppelpunkt eingeleitet (0x3B) und nur von Servern verwendet. Befehle sind im ASCII-Format codiert und bestehen aus dem Klartextbefehl oder seinem aus drei Ziffern zusammengesetzten Pendant. Die gesamte Nachricht besteht aus maximal 512 Zeichen und ist mit der Sequenz Carriage Return-Line Feed abgeschlossen. Zur Erkennung des Protokolls sind die IRC-Befehle geeignet, wobei die Analyse durch das am Anfang der Nachricht befindliche optionale Präfixfeld erschwert wird. Die Liste der Befehle lautet

```
PASS, NICK, USER, OPER, MODE, SERVICE, QUIT, SQUIT, JOIN, PART,
TOPIC, NAMES, LIST, INVITE, KICK, PRIVMSG, NOTICE, MOTD, LU-
SERS, VERSION, STATS, LINKS, TIME, CONNECT, TRACE, ADMIN, IN-
FO, SERVLIST, SQUERY, WHO, WHOIS, WHOWAS, KILL, PING, PONG, ER-
ROR, AWAY, REHASH, DIE, RESTART, SUMMON, USERS, WALLOPS, USER-
HOST, ISON.
```

3.3.7.2 OSCAR

Seit der Übernahme des ICQ-Herstellers Mirabilis durch AOL nutzen sowohl der AOL Instant Messenger (AIM) als auch ICQ das OSCAR (Open System for Communication in Realtime)-Protokoll. Es nutzt TCP als Transportmechanismus und kommuniziert mittels Port 5190. OSCAR-Nachrichten beginnen immer mit einem FLAP[9] -Header, der das Marker-Byte (0x2A) enthält (FLAP-ID), gefolgt vom Channel-Byte, das die Werte 0x01, 0x02, 0x03, 0x04 oder 0x05 annehmen kann (vgl. Abb. 3.16). Die nächsten zwei Bytes enthalten eine Sequenznummer, die - analog TCP - die Reihenfolge der Textnachrichten gewährleistet. Die Länge des FLAP-Payload wird durch das letzte Element des FLAP-Header beschrieben. Es schließt den Header von der Längenangabe aus.

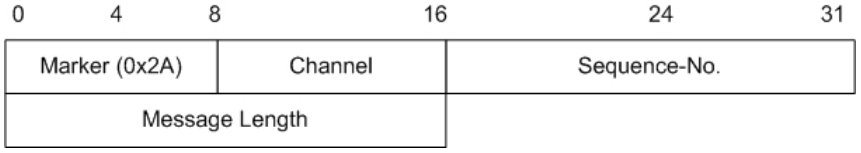


Abbildung 3.16: Aufbau des FLAP-Headers in OSCAR

Weitere Angaben zum OSCAR-Protokoll sind in [OSC1] und [OSC2] enthalten.

3.3.7.3 MSN Messaging Service

Der Microsoft Network Messaging Service (MSNMS) ist ein weitverbreitetes Chatprotokoll von Microsoft. Der Login-Server akzeptiert Verbindungen per default auf TCP-Port 1863. Die Signalisierung ist ASCII-codiert. Der Nachrichtenkopf besteht aus einem dreistelligen Befehl, dem eine *Transaction ID* folgt. Diese stellt die richtige Reihenfolge der Nachrichten bei mehreren Chatsitzungen sicher. Sie beginnt bei 1 und wird für jedes Request-Response-Paar bis zum Maximalwert von $2^{32} - 1$ um 1 erhöht. Die Befehle sind [ML99]

[9]FLAP ist das Transportprotokoll in OSCAR; es stellt die Reihenfolge der Nachrichten sicher und multiplext Nachrichtentypen in verschiedene Kanäle (Channel). Die Abkürzung FLAP wird von AOL nicht erklärt.

```
VER, CVR, USR, MSG, SYN, GTC, BYE, CAL, CHG, CHL, ILN, JOI, LSG,
LST, PNG, PRP, QNG, XFR, ACK, ADD, ANS, BLP, FLN, INF, IRO, NAK,
NLN, OUT, REM, RNG.
```

Da sowohl das OSCAR-als auch das MSNMS-Protokoll die komplette Chatsitzung über die Loginserver abwickeln, ist bei nicht anonymisierten Flußtabellen eine Filterung nach deren IP-Adressen denkbar. Bei einer Filterung nach TCP-Ports würden Webchat-Konversationen wie icq2GO [icq2GO] als HTTP-Verkehr identifiziert. Es ist anzunehmen, daß Microsoft und AOL die zur Zeit für diese Dienste verwendeten Portnummern beibehalten.

4. Durchführung von Messungen

Meßdaten, welche zu Simulationszwecken aufgenommen werden, sollten stets die „Normalsituation“ des Netzes widerspiegeln, also weder in Über- noch Niedriglastsituationen entstehen. Es empfiehlt sich daher, sowohl das zu beobachtende Netzsegment hinsichtlich Eignung zu beurteilen, als auch die zeitliche Auslastung in Erfahrung zu bringen. Ein Meßpunkt, an den hauptsächlich Firmennetze angeschlossen sind, ist für eine freitägliche Messung weit weniger geeignet als ein Meßpunkt, auf dem Privatkundenanschlüsse auflaufen. Aufmerksamkeit sollte ebenfalls der Frage nach der Leistungsfähigkeit der Meßhardware gewidmet werden, um Paketverluste durch Überlastung zu vermeiden. Weiterhin ist es nötig, Anforderungen an die Meßprobe zu formulieren; insbesondere spielen Zeitauflösung und benötigte Payloadlänge eine Rolle. Betrachtungen bis hinunter zur Sicherungsschicht sind vor der Messung anzustellen. Es muß bekannt sein, ob VLAN- und/oder MPLS-Technologie verwendet wird, ob Source Routing zum Einsatz kommt, ob und auf welcher Ebene Verkehr verschlüsselt wird und welche Applikationsprotokolle zu erwarten sind. Dementsprechend ist die aufzunehmende Paketlänge festzulegen. Weiterhin ist mit Hinblick auf die Auswertung in Erfahrung zu bringen, ob asymmetrisches Routing auftritt. Es ist ebenfalls zu evaluieren, wie die Messung physikalisch realisierbar ist, z.B. ob ein passives Tap verwendet werden kann, oder physikalische bzw. logische Netzwerkverbindungen auf Meßanschlüsse der vorhandenen Infrastruktur gespiegelt werden.

4.1 Vorbereitung

Die Aufnahme von Meßwerten erfolgte zum einen im internen Netz des Instituts für Nachrichtentechnik und Informationselektronik an der Universität Rostock, zum anderen im Backbone eines in Ostdeutschland ansässigen Internet Service Providers; mit Hinblick auf die Entwicklung einer Analyseumgebung lag damit ein umfangreicher Satz an Eingangsdaten unterschiedlicher Charakteristika vor. Die Meßdatei aus dem Institut für Nachrichtentechnik der Universität Rostock wurde bereits im Rahmen von [Bru05] aufgenommen. In den beiden Fällen der ISP-Messung wurden Ethernet-Rahmen indirekt an gespiegelten Trunk-Ports zentraler Switches abgegriffen. Ethernetrahmen wurden mit VLAN-Kapselung nach IEEE 802.1q versehen. Die Meßtechnik bestand aus einem unter SuSe Linux 9 betriebenen Barebone-PC mit

einer Spezialmeßkarte vom Typ DAG 3.5E der Firma Endace Measurement Systems. Die technischen Parameter des Meßsystems wurden über das proc-Dateisystem bestimmt und sind in Tab. 4.1 aufgeführt.

Chipsatz	VIA K8M800
CPU	AMD Athlon64 3200
RAM	2x512MB DDR-400
HDD	250GB Maxtor DiamondMax10 6B250S0 (SCSI)
Betriebssystem	SuSe Linux 9.0
Kernel	2.4.21-99-athlon
Dateisystem	ReiserFS
Meßkarte	Endace DAG 3.5E passive Tap
Bus	PCI 2.2 (533MBit/s)
Meßports	2x100MBit/s vollduplex
Zeitauflösung	64Bit $\simeq$ 0,48 μs

Tabelle 4.1: Konfiguration des Meßrechners

Der Vorteil bei der Verwendung einer Meßkarte besteht in der hohen Zeitgenauigkeit und der geringen Paketverlustrate bei gleichzeitig geringer Belastung des Hostrechners. Bei der Verwendung von Sniffer-Programmen wie TCPdump ist nicht sichergestellt, daß das eingetroffene Paket sofort verarbeitet wird; ist der Betriebssystemkernel von einem höherpriorisierten Prozeß belegt, wird die Ankunftzeit verfälscht. Des weiteren kann es bei starker Belastung der CPU des Hostrechners zu Paketverlusten kommen, da alle Operationen (Setzen des Zeitstempels, Isolierung der Header, Transfer in den Hauptspeicher) parallel zu den übrigen Aufgaben des Betriebssystems abgearbeitet werden müssen. Zusammen mit dem kleinen Pufferspeicher einer Ethernet-NIC wächst die Paketverlustrate proportional zum Durchsatz des beobachteten Teilnetzes. Demgegenüber besitzt die verwendete Netzwerkmeßkarte einen integrierten Prozessor und einen hochgenauen Taktgenerator, der z.B. über GPS synchronisiert werden kann. Eine eingehende Vorstellung und Bewertung der Leistungsfähigkeit des verwendeten Meßsystems ist in [Bru05] zu finden.
Der wichtigste Aspekt im Zusammenhang mit der Gewinnung aussagefähiger Resultate ist die nötige zu erfassende Länge der Pakete. Aus Gründen des Datenschutzes und um die zu verarbeitende Datenmenge gering zu halten, ist es weder erforderlich noch gewünscht, die kompletten Ethernet-Rahmen aufzuzeichnen. Andererseits muß genug Payload vorhanden sein, um eindeutige Aussagen über die jeweiligen Protokolle der Applikationsschicht treffen zu können. Da die TCP-Header wegen ihrer Options-Felder eine variable Länge von bis zu 36 Oktetts haben können, sind im Vorfeld von Messungen konkrete Aussagen über die Auftretenshäufigkeit der Optionen zu treffen. Informationen über die maximale Segmentgröße werden während des Drei-Wege-Handshake ausgetauscht; hier tritt kein Payload auf. Die 12 Byte lange Timestamp-Option muß in jedem Fall berücksichtigt werden, wenn ihr Auftreten bekannt ist.
Ethernet- und VLAN-Header nach IEEE 802.1q besitzen eine konstante Größe. Die optionalen Felder des IP-Headers werden aufgrund ihrer seltenen Verwendung vernachlässigt, Source Routing findet in den Netzen nicht statt. Bild 4.1 zeigt den generellen Aufbau eines Ethernetrahmens mit einer Strukturierung bis zum TCP-Header. Das typische Darstellungsformat von 32Bit pro Zeile wurde beibehalten.

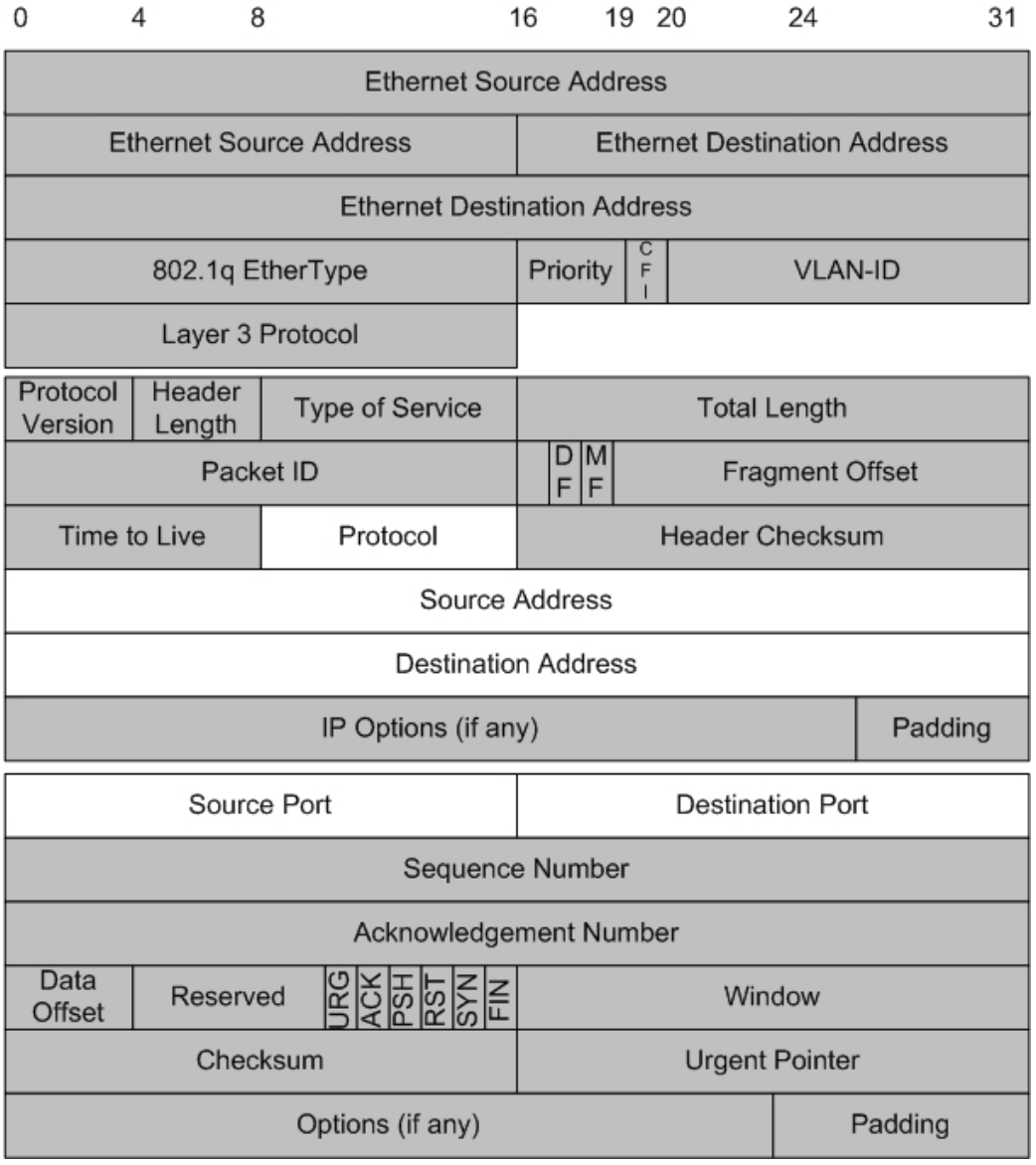


Abbildung 4.1: Aufbau eines Ethernet-Frame mit VLAN-Header sowie IP- und TCP-Header als Payload

Zur eindeutigen Identifizierung des Anwendungsschichtprotokolls wurde eine zu erfassende Payload-Länge von 12Bytes festgelegt. In Tabelle 4.2 wird die Bestimmung der zu erfassenden Rahmenlänge durchgeführt.

Protokoll	Größe des Headers	Zwischensumme
802.3u	14Byte	14Byte
802.1q	4Byte	18Byte
IP	20Byte	38Byte
TCP	24Byte	62Byte
Applikation	12Byte	74Byte

Tabelle 4.2: Bestimmung der aufzuzeichnenden Mindestgröße der Pakete

Fr den IP-Header wurde die standardmäßige Größe von 20Byte angesetzt, da die einzigen hier zu erwartenden Optionen eine Liste von IP-Adressen im Rahmen von *Loose Source Routing* oder *Strict Source Routing* sind. Die Auftretenshäufigkeit derartiger Pakete wurde als vernachlässigbar angenommen.
Die Beschränkung der Segmentgröße ist im Options-Feld des TCP-Headers möglich. Mit Overhead nimmt diese Option 32Bit ein, so daß der TCP-Header auf 24Byte anwächst.

4.2 Aufnahme der Messwerte

Die Messung des Datenverkehrs über den Backbone des Internet Service Providers wurde zunächst in zwei Sitzungen zu jeweils 24h aufgeteilt. Die Gründe bestehen einerseits in der resultierenden Größe des Tracefile, andererseits in der Tatsache, daß die in den Meßsitzungen beobachteten VLANs durch einen Router getrennt sind, wodurch für eine spätere Datenaufbereitung evtl. wichtige VLAN-Zugehörigkeitsinformationen verlorengingen. Ein weiterer wichtiger Gesichtspunkt besteht darin, daß auf diese Weise unterschiedliche Nutzerprofile aufgezeichnet werden konnten. Die getrennte Untersuchung beider Proben ermöglicht direkte Aussagen zur Leistungsfähigkeit der zu entwickelnden Analysewerkzeuge in zwei nahezu distinkten Umgebungen und liefert indirekt Anhaltspunkte bezüglich der Erkennungsrate in gemischten Umgebungen. Die Meßanordnung ist in Abb. 4.2 dargestellt.

In Kooperation mit dem Service Provider wurden die Aufzeichnungen der Proben für den Beginn des Monats Dezember terminiert. Die Wochentage Dienstag und Donnerstag ließen einen regulären Betrieb des Netzwerks erwarten. In der ersten, am 7. 12. 2004 begonnen, Sitzung erfolgte die Spiegelung von fünf VLANs auf den Meßport. Drei dieser VLANs gehören zu Level-4-Kabelnetzbetreibern mit Bandbreiten von (6...24)MBit/s und voller Auslastung, eines zu einem Geschäftskundenanschluß mit 2MBit/s und ein weiteres zu einer Server-Kollokation mit geringer Last. Wegen des großen Anteils von Privatkundenanschlüssen war von viel Peer-to-Peer-Datenverkehr auszugehen, wodurch die Auslastung auch über Nacht hoch bliebe. Die DAG-Meßsoftware wählt als Namen für die Meßdatei eine Kombination aus Datum und Uhrzeit des Messungsbeginns nach dem Schema YYYYMMDD-HHMMSS-0[1]. Der Name der ersten Meßdatei lautet 20041207-154133-0.
Die zweite, am 9. 12. 2004 gestartete Messung unter dem Dateinamen 20041209-150130-0 zielte auf das Monitoring von Geschäftskunden ab. Der beobachtete Stamm umfaßte vorwiegend kleine Firmen aus Gewerbeansiedlungen. Typische Bandbreiten betrugen (1..4)MBit/s. Die Gewerbeansiedlungen sind mittels Dark Fiber angebunden, wodurch den Kunden die Möglichkeit gegeben ist, ihre Bandbreite im Up- und Downstream kurzzeitig zu erhöhen. Bei einigen Kunden handelt es sich um Außenstellen größerer Unternehmen. Es war demnach auch Verkehr, der durch LAN-LAN-Kopplung hervorgerufen wird, zu erwarten. Der typische Vertreter ist das Server-Message-Block-Protokoll. Beide Messungen wurden in einem statisch gerouteten Netz unter symmetrischem Routing durchgeführt.
Der Aufbau der DAG-Meßkarte erfordert den beidseitigen Abschluß des Geräts. In beiden Fällen war eine Unterbrechung der Verbindungen zwischen Router und Switches zwecks Einbringung des Tap ausgeschlossen, so daß der Datenverkehr direkt

[1] Y - year, M - month, D - day, H - hour, M - minute, S - second

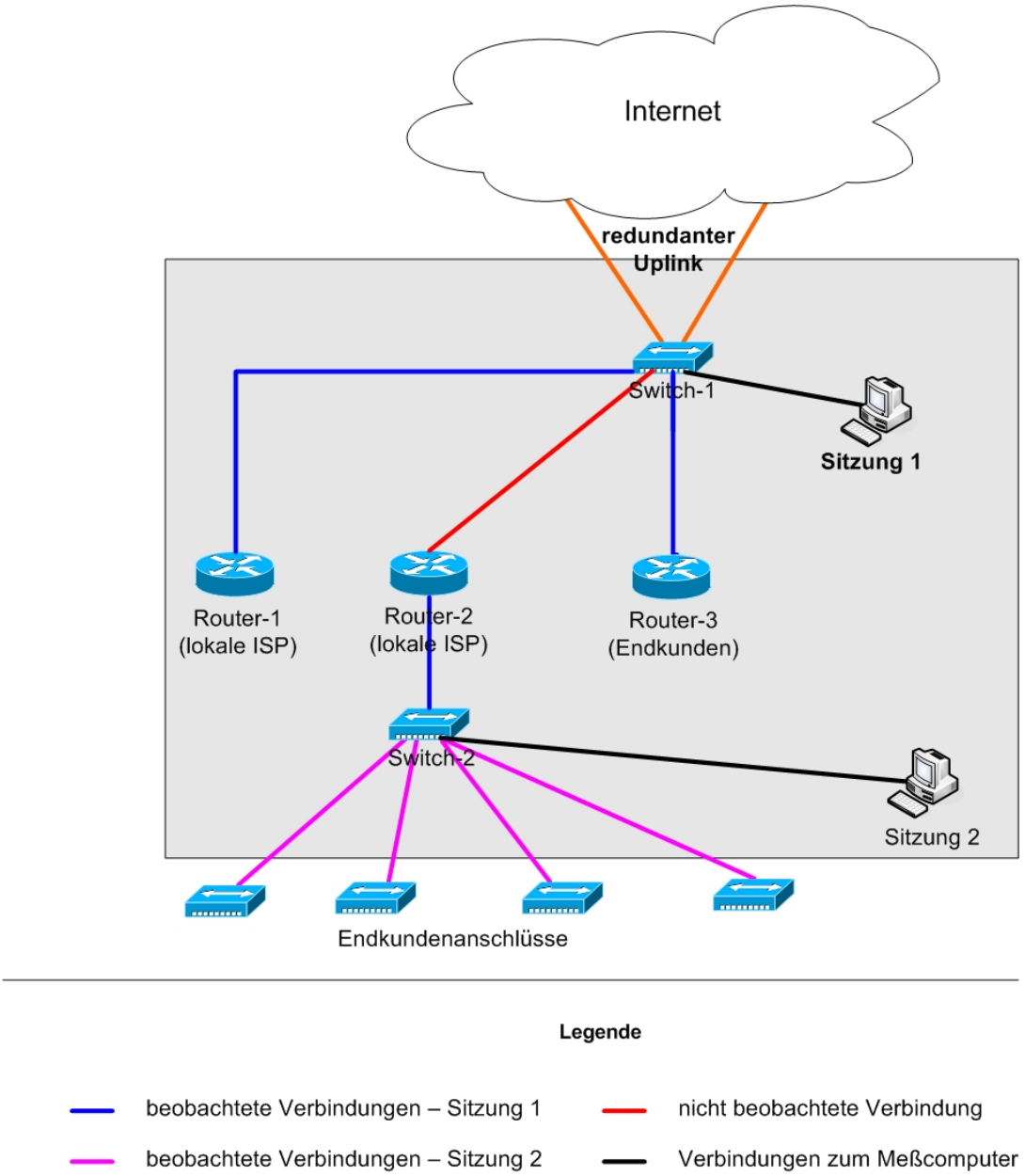


Abbildung 4.2: relevante Netzkomponenten und Meßpunkte im PoP des ISP

am Switch abzugreifen war. Um die Pakete auf einem bestimmtem Port abzugreifen, kann bei Switches von Cisco eine sog. SPAN (Switched Port ANalyzer)-Sitzung eingerichtet werden. Modelle vom Typ Catalyst 3550 ermöglichen sowohl Port- als auch VLAN-basierte Messungen. Die durchgeführte Sitzung wurde folgendermaßen eingerichtet

```
switch(config)# monitor session 1 source vlan vlan-id rx
switch(config)# monitor session 1 destination interface dest.-interface
encapsulation dot1q
```

Die Angaben in Kursivschrift waren entsprechend einzusetzen. Der in die zweite Messung einbezogene Switch vom Typ Cisco Catalyst 2900 unterstützt lediglich Monitor-Sitzungen per Port. Aus diesem Grund und wegen des verhältnismäßig ge-

ringen Verkehrsaufkommens in dem betrachteten Subnetz wurde der Uplink gespiegelt. Die Befehlssyntax lautet

```
switch(config)# monitor session 1 source interface int-id both
switch(config)# monitor session 1 destination interface int-id encapsulation dot1q
```

Für Source-VLAN bzw. Source-Interface können jeweils Listen mit durch Kommata getrennten Werten angegeben werden. Für eine genaue Beschreibung von SPAN sei auf [Cis01] verwiesen.
Ein Anschluß ist mit dem SPAN-Port des Switches zu verbinden, der andere mit einem ‚Dummy-Interface'. Dazu kann ein beliebiger Rechner mit 100Base-Tx-Netzwerkkarte an den zweiten Port der Meßkarte angeschlossen werden. Anschließend wurde dieses Interface über die mitgelieferte Software deaktiviert. Bild 4.2 erläutert diesen Schritt.

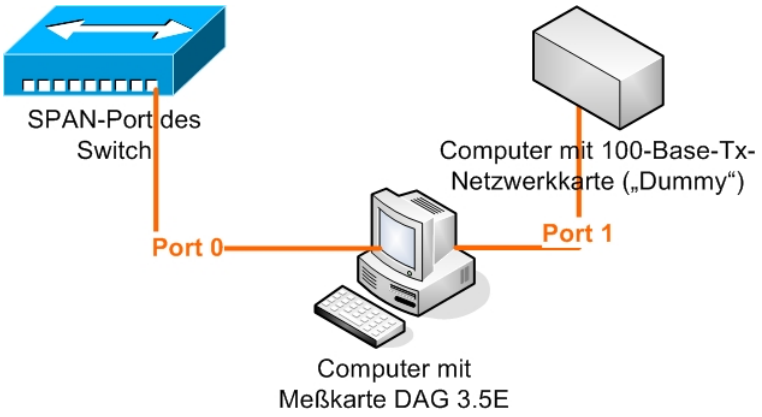


Abbildung 4.3: Verwendung einer Meßkarte mit Tap an einem SPAN-Port

Tabelle 4.3 gibt eine Zusammenfassung der gemessenen Proben.

Name der Probe	20041025-140917-0	20041207-154133-0	20041209-150130-0
Dateigröße [Bytes]	2496083940	39991044632	7376984120
übertragene Bytes	11791839138	162459529152	21370995266
aufgenommene Rahmenlänge	70Byte	74Byte	74Byte
kleinste auftretende Flußgröße	48Byte	48Byte	48Byte
res. Länge des Payload der Applikationsschicht	8Byte	12Byte	12Byte
Zeitraum der Messung	190,9h	24h	24h
durchschnittl. Durchsatz (unidirektional)	153,2kBit/s	15,0MBit/s	2,0MBit/s
Profil	Forschung	Privatnutzer	Geschäftsnutzer

Tabelle 4.3: Charakteristika der aufgenommenen Proben

5. Analyse

In diesem Kapitel werden zunächst die Anforderungen an eine leistungsfähige Analyseumgebung formuliert, sodann erfolgt eine Bewertung verfügbarer Applikationen nach den zuvor festgelegten Maßstäben. Die Analyse der gesammelten Daten erfolgt in mehreren Stufen:

- Erstellung der Flußtabellen
- Payload-Analyse der aufgezeichneten Pakete
- Zuordnung der erfolgreich identifizierten Pakete zu den Flüssen
- entsprechende Markierung der nicht identifizierten Flüsse
- hybrider Ansatz mit transportschichtportbasierter Analyse der „Well-Known Ports “ von Peer-to-Peer - Protokollen und Verbindungsmuster
- Analyse nach Verbindungsmustern der Transportschicht ohne Berücksichtigung der Applikationsschicht
- portnummernbasierte Analyse der „Well-Known Ports “ der Flußtabelle ohne Payloadanalyse
- Gegenüberstellung der Verfahren und Vergleich

Dabei zeigte sich, daß es - zumindest im Open-Source-Bereich - keine Software gibt, die alle Aufgaben am effizientesten löst. Insbesondere für die zweite Hälfte der Analyse wurde auf selbst geschriebene Tools zurückgegriffen. Zudem war das Problem der Koordination und Integration der einzelnen Verarbeitungsschritte zu lösen.
Im abschließenden Teil dieses Kapitels werden die Analyseergebnisse der im Kapitel 4 aufgenommenen Proben vorgestellt und die Leistungsfähigkeit der vorgestellten Lösung betrachtet.

5.1 Payloadanalyse und Nonpayloadanalyse

Der naheliegende Ansatz, Netzverkehr zu identifizieren, besteht in der Untersuchung der Nutzlast eines jeden TCP-Segments bzw. UDP-Datagramms. Insbesondere am Anfang von Flüssen lassen sich mittels Mustervergleichen spezifische Signaturen identifizieren (vgl. Kap. 3.2 und 3.3). Dieses Prinzip findet nicht nur für die statistische Auswertung im Rahmen von *Offline-Analysen*, sondern zunehmend auch in Echtzeit Anwendung. Mit *Layer-7-Switches* z.B. der Hersteller [F5] (BIG-IP) oder [Packeteer] (Packeteer) ermöglichen Analyse und applikationsschichtgestütztes Traffic-Shaping bis zu Bandbreiten von 1GBit/s bei 1,4Mio. Flüssen (Packeteer 10000, vgl [Packeteer]).
Ein weiteres Anwendungsfeld der Analyse von Applikationsschichtinhalten ist die *Intrusion Detection*.
Die Non-Payloadanalyse ist derzeit gering verbreitet (vgl. [KBFc04], S. 1) und z. Zt. lediglich für das akademische Umfeld von Interesse. Sie berücksichtigt ausschließlich die (TCP-)Verbindungsmuster, um Aussagen zur Zugehörigkeit zu einer Applikationsklasse[1] zu treffen. Von der Tendenz des Anstiegs des Anteils an verschlüsselter Kommunikation ist der vorgestellte Algorithmus demnach nicht betroffen ([KBFc04], Kap. 5.5). Es ist ebenfalls vor dem Hintergrund des i. allg. gleichen Transportschichtverhaltens der Peer-to-Peer-Protokolle (vgl. Kap. 3.3) möglich, bislang unbekannte bzw. nicht rückentwickelte (disassemblierte) Protokolle zuzuordnen. Ein Geschwindigkeitsvorteil gegenüber der Payloaduntersuchung ist wegen der Notwendigkeit des Aufbaus großer Tabellen und des mehrmaligen Duchlaufens der Flußtabelle nicht zu erwarten. Als Hauptnachteil ist mangelnde Echtzeitfähigkeit des Algorithmus anzusehen, da die Flow Table mehrmals durchlaufen werden muß (vgl. [KBFc04], Kap. 5.5). Tabelle 5.1 gibt einen Überblick bezüglich ausgewählter Kriterien.

Kriterium	Payloadanalyse	Nonpayloadanalyse
Echtzeitfähigkeit	mit Spezialhardware	nein
Schutz der Privatsphäre	nein	ja

Tabelle 5.1: Eigenschaften von Payload- und Nonpayloadanalyse

5.2 Beurteilung ausgewählter Programme für die Payloadanalyse

Das Analysetool der Wahl hatte verschiedenen Anforderungen gerecht zu werden.

- Freeware bzw. vorhandene Lizenz
  Die Nichtverfügbarkeit finanzieller Mittel setzte diesen Punkt zwingend voraus.

- Lauffähigkeit unter LINUX
  Da die Meßumgebung auf dem Betriebssystem LINUX aufsetzt und nur die Hardware des Meßrechners über genügend Leistungsreserven für die Verarbeitung großer Datenmengen verfügt, war diese Voraussetzung ebenfalls zwingend

[1] Der zugrundeliegende Artikel konzentriert sich auf die Erkennung von Peer-to-Peer-Aktivitäten. Da nur Verkehrsmuster untersucht werden, sind Aussagen zu den Protokollen der Applikationsschicht nicht möglich.

zu erfüllen.

- Unterstützung des ERF - Dateiformats
  Dies ist das Speicherformat der Meßanwendung.

- Erkennung der gängigsten Applikationen im Payload des Transportprotokolls
  Die Erstellung eines eigenen Programms, welches diese Aufgabe übernimmt, ist zwar möglich, jedoch als „Insellösung“ anzusehen, da es bei vorhandener Personalstärke und Zeit nicht möglich ist, ein ausreichend umfangreiches Analysetool zu entwickeln und zu verifizieren. Des weiteren würde das Problem der Erweiterung, Wartung und Pflege einen längerfristigen Einsatz über die vorliegende Arbeit deutlich erschweren.

- Erweiterbarkeit bzw. Weiterentwicklung durch den Hersteller
  Das Aufgabenfeld der Netzwerkanalyse sollte von dieser Aufgabe abgekoppelt sein.

Diesen Kriterien entspricht ein freies Open-Source-Projekt. Das unter der GPL freigegebene, zum kostenlosen Download bereitgestellte Netzwerkanalysetool *Ethereal* [Eth05] erkennt fast alle Protokolle von Bedeutung auf jeder Protokollebene. Diese große Leistungsfähigkeit wird durch den Einsatz vieler Freiwilliger unter Federführung des Chefentwicklers Gerald Combs erreicht. Für jedes Protokoll existiert ein sog. *Dissector*, der als C-Quellcode in die Programmstruktur eingefügt ist.
Für die nichtinteraktive Analyse ist das zur Ethereal-Suite gehörende Tethereal geeignet. Es bietet alle Merkmale und Filter, die Ethereal unterstützt. Ein Nachteil von Ethereal ist die geringe maximale Größe der analysierbaren Tracedatei. Als Faustregel gilt eine maximale Dateigröße von 35% des hardwaremäßig vorhandenen freien Arbeitsspeichers. Obwohl Ethereal aus Netzwerkproben auch *Flußtabellen* bzw. *Konversationstabellen*[2] erstellen kann, ist es für umfangreiche Traces aufgrund seines hohen Zeitbedarfs schlecht geeignet. In der zum freien Download verfügbaren CoralReef-Suite ([CR04]) ist ein Programm enthalten, welches ein deutlich niedrigeres Laufzeitverhalten aufweist. Sowohl Ethereal als auch CoralReef bieten in den aktuellen Versionen native Unterstützung für das ERF-Format des Meßkartenherstellers Endace. Alternativ ist auch das kostenpflichtige Open-Source-Tool nProbe [nP04] zur Erstellung der Flußtabellen einbindbar[3]. Das Ausgabeformat kann mittels Formatierungsanweisungen dem von crl_flow angepaßt werden, so daß eine reibungslose Weiterverarbeitung gewährleistet ist, vgl. Kap. 5.3.1. Bei der Verwendung von nProbe ist jedoch zu beachten, daß lediglich das PCAP-Format gelesen werden kann.

5.3 Anpassung der Analyseumgebung

Nach erfolgreichen Tests des Analyseprogramms Ethereal trat die Notwendigkeit einer externen Steuerung dieses Programms zutage, um den Einträgen der Flußta-

[2] bidirektionale Flußtabellen

[3] Dies erfordert jedoch die Modifikation der Quelldatei nprobe.c, um den Namen der Flußdatei frei wählen zu können. Es ist erforderlich, die Zeilen 2170ff auszukommentieren und Zeilen 553ff entsprechend anzupassen. Das trifft auf Version 3.0 (01/2004) zu.

bellen die entsprechenden Protokolle der Applikationsschicht zuzuordnen. Als Minimalforderung wurde die nicht-interaktive Erstellung von Flußtabellen sowie eine erste Auswertung nach eindeutigen Treffern mittels Filterung nach Protokollen der Applikationsschicht mit Ausgabe der Ergebnisse in (Text-) Dateien formuliert. Diese Aufgabe stellt gewisse Anforderungen an die Automatisierungsumgebung:

- Unkomplizierte Stringmanipulation
  Das automatische Erstellen von Dateien nach einfachem Muster sowie einfach mögliche Textmanipulationen sollen eine geringe Komplexität des Programms sicherstellen.
- Unkomplizierte Erweiterbarkeit
  Es sollen z.B. Berechnungen zum Erstellen von Statistiken nach Wahl implementierbar sein.
- Gutes Laufzeitverhalten beim Verarbeiten großer Datenmengen
- Plattformunabhängigkeit

Pattern Matching mit regulären Ausdrücken, die native Unterstützung von Hashtabellen sowie Verfügbarkeit auf fast allen Betriebsystemen machen Perl zum Mittel der Wahl. Eine kompilierte Sprache würde Konfigurationsänderungen erschweren[4], während ein einfaches Shellskript weder portabel noch im geforderten Umfang erweiterbar wäre.

5.3.1 Automatisierung der Payloadanalyse mit Perl

Die Erstellung der Flow Tables sowie die Payload-Analyse durch das Programm Tethereal wurde mit Hilfe eines Perl-Skriptes automatisiert. Das *analyse_ethereal.pl* benannte Skript ist in einen Konfigurations- und einen Ausführungsteil gegliedert und greift auf die Programme *dDagconvert* der DAGtools, die CoralReef-Suite, nProbe sowie auf das terminalbasierte Programm Tethereal der Ethereal-Distribution zurück. Die Aufgabe von Dagconvert besteht in der Aufteilung einer großen Meßdatei in kleinere und damit leichter zu verarbeitende Fragmente sowei optional einer Umwandlung des Dateiformats. Die wesentlichen Optionen wie Quell- und Zieldateiformat, Arbeitsverzeichnis sowie Zieldateigröße sind in analyse_ethereal.pl frei wählbar. Für Tethereal ist eine Auswahl der Protokolle der Transportschicht, die berücksichtigt werden sollen, möglich. Der Ausführungsteil übergibt die Teiltraces sukzessive an die Programme nProbe bzw. crl_flow.pl der CoralReef-Suite, welche die Flußtabelle berechnet. Ein weiteres Programm der CoralReef-Suite, crl_print_pkt, gibt, sofern vorhanden, den Payload der Transportschicht-PDUs mit Zeitstempel sowie Netzwerk- und Transportschichtinformationen in eine gesonderte Textdatei aus. Das von crl_flow ausgegebene Format bildet die Grundlage der weiteren Analyse. Es hat die Form

- IP-Adresse (Quelle)

[4] In den Perl-Skripten wird mit durch den Benutzer erweiterbaren Datenstrukturen gearbeitet.

- IP-Adresse (Ziel)
- Transportschichtprotokoll
- OK-Flag
- Quellport
- Zielport
- Anzahl der Pakete
- Anzahl der Bytes
- Anzahl der Flüsse
- Zeistempel des Flußbeginns
- Zeitstempel des Flußendes.

Die Ausgabeparameter von nProbe sind dementsprechend konfiguriert. Im zweiten Schritt wird das Teil-Trace an Ethereal übergeben. Die letzte Aktion besteht in der Zuordnung der identifizierten Pakete zu den Einträgen in der Flußtabelle und der Ausgabe der aufgezeichneten Applikationsschichtinformationen des ersten Pakets eines Flusses, wenn verfügbar. Der generelle Ablauf ist in Bild 5.1 festgehalten. Da Ethereal die Analysen nur getrennt nach TCP und UDP durchführt, wird jedes Teiltrace zweimal durchlaufen.

5.3.2 Non - Payloadanalyse mit Perl

Die mittels der reinen Payloadanalyse erreichbare Identifizierungsquote liegt - je nach Zusammensetzung der Netzwerkprobe - zwischen 0,4% und 15% aller Flüsse (vgl. Tab. 5.5). Damit erbringt die Identifizierung von Paketen nur anhand ihrer Applikationsschichtsignatur noch keine akzeptablen Ergebnisse. Ein Mittel zur Erhöhung des identifizierten Verkehrsanteils liegt in der Auswertung der Verbindungsmuster unter gleichzeitiger Berücksichtigung der Ergebnisse der Payloadanalyse. Des weiteren kann im Bereich des Peer-to-Peer-Datenverkehrs davon ausgegangen werden, daß viele Clients weiterhin die wohlbekannten Ports verwenden. Ein Wechsel auf z.B. TCP-Port 80 erfolgt üblicherweise zu dem Zweck, vom eigenen Provider gedrosselte Ports (z.B. 4661..4665 für das eDonkey-Protokoll) zu umgehen bzw. das eigene Nutzerverhalten zu tarnen. Der im Perlscript analyse_hybrid.pl umgesetzte Algorithmus geht teilweise auf [KBFc04] zurück. Alle verfügbaren Skripte, evtl. Kommandozeilenparameter und Aufgaben werden kurz vorgestellt.

- analyse_ethereal.pl
  Argumente: keine, Konfiguration im Skript
  Ausgabedateien: Flußtabellen der Teiltraces, getrennt nach Transportschichtprotokoll im Unterverzeichnis „results“
  Aufgaben:
  - Aufteilung von großen Meßdateien

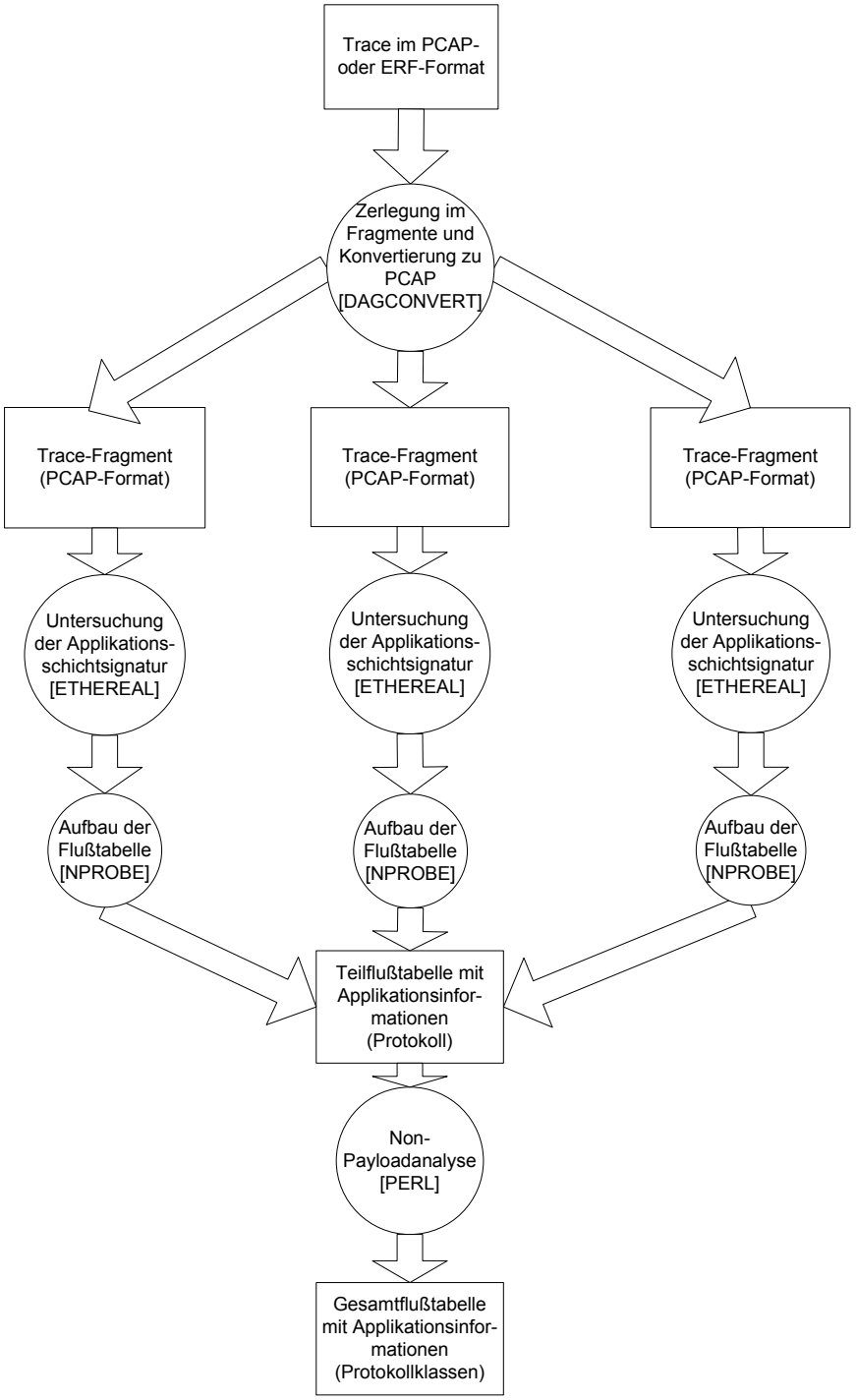


Abbildung 5.1: Erster Schritt der Analyse: Übergabe an Tethereal und Erstellung der Flußtabellen. Die für den jeweiligen Schritt verwendeten Programme stehen in eckigen Klammern.

  - Übergabe der so erzeugten Teiltraces an Tethereal
  - Erstellung der Flußtabellen
  - Zuordnung der identifizierten Applikationsschichtprotokolle zu den entsprechenden Flüssen

- analyse_hybrid.pl
  Argumente: keine, Konfiguration im Skript
  Ausgabedateien: conv_temp_h0.text (Ergebnis Tethereal), conv_final_h0.text (Gesamtergebnis)
  Aufgaben:
  - Zusammenfügen der Flußtabellen der Teiltraces
  - Analyse der „Well-Known Ports“ für Peer-to-Peer-Protokolle und SMB
  - Verifikation von Web-, Mail-, DNS-, Online-Gaming-, Streaming- und FTP-Servern und Identifizierung der entsprechenden Flüsse
  - Auswertung der Ergebnisse von Tethereal
    Beschreibung in Kap. 5.3.2.1

- analyse_table.pl
  Argumente: Zielverzeichnis von analyse_ethereal.pl, Index der ersten zu analysierenden Flußtabelle, Index der letzten zu analysierenden Flußtabelle
  Ausgabedateien: conv_temp_n0.text (Ergebnis der Portanalyse für Peer-to-Peer-Protokolle, SMB), conv_final_n0.text (Gesamtergebnis)
  Aufgaben:Aufgaben:
  - Zusammenfügen der Flußtabellen der Teiltraces
  - Analyse der „Well-Known Ports“ für Peer-to-Peer-Protokolle und SMB
  - Verifikation von Web-, Mail-, DNS-, Online-Gaming-, Streaming-Content, Chat-, NFS- und FTP-Servern und Identifizierung der entsprechenden Flüsse
  - Identifikation von Peer-to-Peer-Flüssen nach Kap. 5.3.2.3, 5.3.2.3

- analyse_ports.pl
  Argumente: keine, Konfiguration im Skript
  Ausgabedateien: conv_final_p0.text (Gesamtergebnis)
  Aufgaben:
  - Zusammenfügen der Flußtabellen der Teiltraces
  - Analyse der „Well-Known Ports“

- summarize_all.pl
  Argument: Datei, die Flußtabelle enthält (conv_{final|temp}_{n|p|h}[Index].text)
  Ausgabedatei: Standardausgabe
  Aufgaben:
  - Durchlaufen der Flußtabelle und Sammeln von Statistiken über die Zusammensetzung der Flüsse unter Berücksichtigung aller Protokolle der Applikationsschicht Beschreibung in Kap. 5.3.2.5

- summarize.pl
  Argument: Datei, die Flußtabelle enthält (conv_{final|temp}_{n|p|h}[Index].text)
  Ausgabedatei: Standardausgabe
  Aufgaben:
  - Durchlaufen der Flußtabelle und Sammeln von Statistiken über die Zusammensetzung der Flüsse unter Berücksichtigung vorgegebener Verkehrsklassen
    Beschreibung in Kap. 5.3.2.5

In den folgenden Unterabschnitten werden die angewandten Heuristiken beschrieben.

5.3.2.1 Auswertung der Payloaduntersuchung

Die Konversationstabelle muß zur eindeutigen Klassifizierung der Konversationen mehrmals durchlaufen werden. Da durch Aufteilung der Trace-Datei mit hoher Wahrscheinlichkeit einige (identifizierte) Konversationen ebenfalls aufgetrennt wurden, es jedoch nicht auszuschließen ist, daß im weiteren Verlauf der Konversation keine Steuerungsinformationen mehr ausgetauscht werden, so daß die Identifikation verlorenginge, wird im ersten Durchlauf jede bereits identifizierte Verbindung in einer Hashtabelle abgelegt. Als individueller Schlüssel dient die Kombination aus (IP-Adresse1:Port1.IP-Adresse2:Port2), den Wert bildet das jeweilige Protokoll der Applikationsschicht. Des weiteren wird die Erkennung aufgrund der wohldefinierten Ports aus Tab. 5.2 vorgenommen. Gleichzeitig kommt eine weitere Heuristik zur Anwendung.

> Zwischen zwei Netzknoten wird in einem kurzen Beobachtungszeitraum nur eine Applikationsart verwendet. Die Fälle, in denen zwischen zwei Peer-to-Peer-Knoten gleichzeitig eine FTP- oder WWW-Sitzung stattfindet, sind vernachlässigbar.

Es sei ausdrücklich darauf hingewiesen, daß diese Annahme aufgrund der Dynamik des Internet nur für kurze Zeiträume gilt. Die in dieser Arbeit untersuchten Proben mit Meßdauern von 24h fallen in diese Katergorie. Eine Ausnahme besteht lediglich an Grenzen zu Netzen, wo statische IP-Adressen verwendet werden bzw. über DHCP-Server die Hardwareadressen von Netzwerkkarten statisch auf IP-Adressen abgebildet werden.

5.3.2.2 Well-Known Ports

Die wohldefinierten Ports laut *Internet Assigned Numbers Authority* (IANA) sind nur begrenzt aussagekräftig. So wird z.B. der vom weitverbreiteten eDonkey-Protokoll per Voreinstellung genutzte Bereich 4662..4665 (TCP) genau wie Ports 6881..6889 (BitTorrent) als „unassigned“ geführt. In den Perlscripts wurden zur Identifizierung von Peer-to-Peer Verkehr sowie von Verkehr, der durch Online-Gaming hervorgerufen wird, zwei Listen mit zum Zeitpunkt der Messungen verbreiteten Protokollen angelegt, vgl 5.2.

Die von Spielen genutzten Ports sind u.a. [MS05] entnommen. Die Ports der Peer-to-Peer-Applikationen sind auf den im Anhang aufgelisteten Homepages der jeweiligen

Applikation	TCP-Port(s)	UDP-Port(s)
P2P		
eMule (eDonkey/Kademlia)	4662..4665	4672
BitTorrent	6881..6889	-
Kazaa	1214	wechselnd
Direct Connect	411, 1411	412, 1412
GNUtella	6346, 6347	6346, 6347
SoulSeek	2234..2239	-
WinMX	6699	-
Gaming		
HalfLife, CounterStrike	27015..27019, 28900	27015..27019
Medal Of Honour	18020, 28910, 29900, 29901	13300, 27900, 29910
Battlefield 1942	-	14567, 22000, 23000..23009, 27900, 28900
Diablo	4000, 6112	-
Dark Age Of Camelot	1280, 10500..10504, 10622..10624	-
Need For Speed Hot Pursuit 2	8511,28900	1230, 8512, 27900, 61200..61230
Rainbow Six	2346..2348	-
Streaming Media		
Windows Streaming Media	1755	-
Real Time Streaming Protocol	554	6970
RealAudio	7070	6970

Tabelle 5.2: berücksichtigte Ports

Hersteller/ Entwickler zu finden. Die für Streaming einbezogenen Portnummern wurden von [KBFc04] übernommen. Da gerade im Spielebereich durch die hohe Zahl an Neuveröffentlichungen eine hohe Fluktuation der verwendeten Portnummern besteht, kann die Filterliste erweitert werden. Die große Anzahl der genutzten Ports bei Online-Spielen birgt das Risiko, daß eine entsprechend „getarnte“ Peer-to-Peer-Verbindung als Gaming-Konversation mißklassifiziert werden könnte.

5.3.2.3 Analyse ohne Payloaduntersuchung

Eine Payloaduntersuchung ist zur alleinigen Identifikation vor allem von Peer-to-Peer-Verkehr nicht ausreichend. Zum einen finden zum Beginn der Messungen bereits Up- und Downloads, die keine Signalisierung mehr benötigen, statt; andererseits kann eine Payloadanalyse nur über bereits bekannte und untersuchte Protokolle Auskunft geben. Die proprietären Protokolle Fasttrack Kap. (vgl. Kap. 3.3.1) und SoulSeek (vgl. Kap. 3.3.5) sowie das Direct-Connect-Protokoll (vgl. Kap. 3.3.6) werden in der verwendeten Version Ethereal 0.10.8 nicht erkannt.
Die Erkennung von Peer-to-Peer-Verkehr basiert im Skript analyse_hybrid auf den Ergebnissen der Payloadanalyse und macht sich darüberhinaus die Verbindungsmuster dieser Verkehrsart zunutze. Alle an Peer-to-Peer-Transfers beteiligten Knoten werden mit der betreffenden Portnummer in einem weiteren Hash gespeichert.

> Tritt dieselbe (IP:Port:Transportschichtprotokoll)-Kombination aus einem P2P-Fluß in einer bisher unidentifizierten Konversation nochmals auf und ist der Port des Kommunikationspartners > 1024, wird diese Konversation ebenfalls als Peer-to-Peer-Verkehr markiert.

Ein hervorstechendes Merkmal, welches Peer-to-Peer-Verkehrsmuster von anderen Applikationsprotokollen unterscheidet, ist die Verwendung von UDP und TCP auf der Transportebene, vgl. Abschnitt 3.3 und [KBFc04]. Folglich werden IP-Paare, die über TCP und UDP-Flüsse kommunizieren und keine Online-Gaming- oder RPC-Ports nutzen, als Peer-to-Peer gekennzeichnet.
Eine ähnliche Heuristik kommt zur Identifikation von FTP- und WWW-Servern zum Einsatz. Zunächst werden alle IP-Adressen, welche über die wohldefinierten Ports 20, 21 und 80 in Hashes als mögliche FTP- bzw. WWW-Server abgelegt.

> Sobald ein als WWW-Server klassifizierter Knoten Verbindungen auf anderen Ports als den wohlbekannten Serverports unterhält, wird er aus dem Hash der möglichen WWW-Server entfernt. Dies gilt analog auch für alle anderen Server.

Die Betrachtung von DNS- und Mailservern ist insofern etwas komplexer, als daß diese auch untereinander kommunizieren, um E-Mails bzw. DNS-Anfragen und -Antworten weiterzuleiten. Sie können also sowohl als Client als auch als Server auftreten. Eine Erweiterung des obigen Ansatzes dient zur Erfassung von DNS- und E-Mail-Konversationen.

> Ein als E-Mailserver zwischengespeicherter Knoten gilt dann als verifiziert, wenn er oder seine Kommunikationspartner ausschließlich die vordefinierten TCP-Ports 25 (SMTP), 110 (POP), 113(SMTP-Authentification Service) oder 143 (IMAP) verwenden. Analog gilt ein DNS-Server als verifiziert, wenn er oder sein Partner TCP- oder UDP-Port 53 verwendet.

Das für diese Aufgabe erstellte Skript analyse_hybrid.pl legt zuerst eine Hilfsdatei mit dem Namen conv_temp_{n|h}[Index].text an, in der die von Ethereal durch den Aufruf aus analyse_ethereal.pl ermittelten Protokollzugehörigkeiten den Flüssen zugeordnet sind. Treten in der Flußtabelle weitere Flüsse mit gleicher (Quell-IP-Adresse, Quellportnummer, Ziel-IP-Adresse, Zielportnummer, Transportschichtprotokoll) auf, werden sie diesem Applikationsschichtprotokoll zugeordnet. Ein Auszug aus einer entsprechenden Datei hat die Form

#src. ip dst. ip L4proto src. port dst. port octets L4proto payload L7proto
106.204.33.118 106.204.221.29 6 4335 135 52 TCP –no payload available– UNKNOWN
116.175.148.105 106.204.41.252 6 1863 2650 52 TCP 51 4e 47 20 34 34 0d 0a |QNG 44..| CHAT
106.204.37.213 34.185.249.15 17 4672 60 50 UDP c5 91 47 00 |..G.| eDonkey
143.203.143.88 146.194.5.102 6 80 64591 460 TCP 48 54 54 50 2f 31 2e 31 20 33 |HTTP/1.1 3| HTTP
102.25.233.179 106.204.37.174 6 6881 1480 50 TCP –no payload available– P2P
146.194.29.71 102.25.222.67 17 4672 4672 50 UDP c5 91 a1 10 |....| P2P

106.204.34.35 144.77.168.207 6 4662 4834 100 TCP –no payload available– P2P
106.204.32.142 145.108.122.252 17 4672 4672 50 UDP c5 91 00 00 73 01 |....s.| P2P
106.204.33.124 113.223.210.77 6 4448 6881 50 TCP 00 00 00 00 |....| P2P
113.223.210.77 106.204.33.124 6 6881 4448 50 TCP –no payload available– P2P
146.194.28.157 127.104.138.29 6 63379 445 112 TCP –no payload available– SMB

Alle IP-Adressen wurden anonymisiert.
Des weiteren werden die Flüsse auf die Verwendung der von Peer-to-Peer-Protokollen verwendeten Well-Known Ports untersucht und entsprechend zugeordnet. Gleichzeitig kommt die erste Stufe der vorgenannten Heuristiken zum Tragen. Jede IP-Adresse, die eine der erwähnten Portnummern verwendet, wird als „möglicher Server“ des entsprechenden Dienstes zwischengespeichert. Zur Verifikation dieser Server muß die Konversationstabelle nochmals durchlaufen werden. Im letzten Duchlauf erfolgt die Verifikation der Server wie oben beschrieben. Kann ein Fluß nicht zugeordnet werden, wird er im letzten Durchlauf als „UNKNOWN“ markiert. Da nun Kenntnisse über alle Server vorliegen, werden die Flüsse mit der zugehörigen Verkehrsart in die Ergebnisdatei conv_final_h0.text geschrieben.
Das Programmablaufdiagramm ist in Bild A.1 und A.2 beschrieben.

5.3.2.4 Auswertung der Verbindungsmuster auf der Transportschicht

Alternativ zur Auswertung auf Basis der Payloadanalyse wurde ein Skript, das zur Identifikation von Applikationsschichtprotokollen lediglich Netzwerk- und Transportschichtinformationen verwendet, entwickelt. Es basiert auf [KBFc04]. Die Erkennung von Mail-, Web-, Online-Gaming- und FTP-Servern erfolgt wie in Kap. 5.3.2.3. Peer-to-Peer-Verkehrsmuster werden im ersten Schritt anhand der Well-Known Ports (vgl. Tab 5.2) identifiziert. Des weiteren sind Peer-to-Peer-Protokolle durch viele Verbindungen zu vielen IP-Adressen gekennzeichnet, während Client-Server-Protokolle viele Verbindungen zu einer IP-Adresse aufbauen. Darüberhinaus verwenden die meisten populären Peer-to-Peer-Protokolle (eDonkey, KaZaA, Direct Connect, GNUtella) UDP zur Signalisierung, während die Nutzdaten in TCP-Segmenten transportiert werden (vgl. [KBFc04]).

> Bestehen zwischen zwei IP-Adressen (1) sowohl TCP- als auch UDP-Verbindungen, (2) ist die Anzahl dieser Verbindungen kleiner als ein vordefinierter Wert (z.B. 10) und ist (3) weder ein Online-Gaming-, SMB- oder NFS-Server an der Konversation beteiligt, werden diese Flüsse als Peer-to-Peer-Verkehr betrachtet.

Bedingung (3) stellt sicher, daß Online-Gaming-Sitzungen, die ebenfalls TCP und UDP als Transportprotokolle nutzen, nicht mißidentifiziert werden. Die Kombination aus den Bedingungen filtert auch Portscans heraus, wie sie z.B. von Würmern hervorgerufen werden. Als Ausnahme sind Online-Gaming-Sitzungen anzusehen, wenn diese innerhalb des lokalen Netzwerks, in dem die Messungen erfolgen, stattfinden. Das Programmablaufdiagramm ist im Anhang unter A.3 und A.4 zu finden. Da bei dieser Art der Analyse die Anforderungen an den Arbeitsspeicher durch die mehrfache Speicherung aller in der Meßdatei vorhandenen IP-Adressen sehr hoch sind, können nur verhältnismäßig kleine Flußtabellen untersucht werden. Aus diesem Grund können nicht, wie bei der hybriden Untersuchung, alle Teiltraces zusammengefügt

werden. Durch die Kommandozeilenargumente 2 und 3 sind die Indizes der ersten und der letzten zu untersuchenden Teil-Flußtabellen angebbar. Für große Meßdateien ist die Steuerung über ein separates Shellskript empfehlenswert.
Durch die Konzentration auf einen Dienst je IP-Adresse würden alle Knoten, die mehr als einen Dienst anbieten - also vor allem Server in SOHO-Umgebungen - nicht erkannt. Um diesen Effekt zu verhindern, werden Server - wenn sie lediglich über wohlbekannte Ports kommunizieren - nicht aus der entsprechenden Servertabelle gelöscht. Auf diese Weise wird erreicht, daß z.B. Webserver, die gleichzeitig E-Maildienste anbieten, nicht durch das Erkennungsraster fallen.

5.3.2.5 Zusammenfassung der Protokollübersicht

Die Darstellung der Zusammensetzung von Netzwerkverkehr erfolgt vorzugsweise tabellarisch und graphisch. Aufgrund der großen Anzahl der auftretenden Applikationsschichtprotokolle wäre eine Darstellung aller Protokolle unübersichtlich und dadurch in seiner Aussagekraft eingeschränkt. Weiterhin ermöglicht die Non-Payloadanalyse im Fall von Online-Gaming und Peer-to-Peer-Verkehr keine Rückschlüsse auf das Protokoll. Folglich bietet es sich an, Protokollklassen zu bilden, welche Protokolle mit jeweils hohem Verbreitungsgrad sowie vergleichbaren Eigenschaften zusammenfassen. Sie sind im Perlskript summarize.pl in Form von Arrays angegeben und frei modifizierbar. Es sind die in Ethereal verwendeten Abkürzungen für die Protokolle zu verwenden. Folgende Klassen sind vorgegeben:

Klasse	Protokoll
Peer-to-Peer	EDonkey/Kademlia, Bittorrent, SoulSeek, GNUtella
CHAT	ICQ, AIM, MS-Messenger, Yahoo, IRC, Jabber
REMOTE	SSH, Telnet, Rsh, X11, Remote Procedure Call Netbios, LDAP, SMB, NFS
E-MAIL	SMTP, POP, IMAP
ENCRYPTED	Encapsulating Security Payload, SSL, SSLv3
FTP	FTP, TFTP
E-Business	(MY)SQL, Internet Content Adaptation Protocol
REALTIME	RTP, RSVP, SIP, Skinny, GRYPHON
HTTP	HTTP
DNS	DNS
OTHER	alle anderen identifizierten Protokolle (not UNKNOWN)

Tabelle 5.3: Untersuchte Verkehrsklassen

Für jede Art von Protokollen der Anwendungsschicht wird in einem Array ein Feld mit einem in der Notation von Ethereal formulierten Filter angelegt.Das Array läßt sich durch Auskommentieren bzw. Hinzufügen von Feldern beliebig verändern, soweit die Unterstützung durch Ethereal gegeben ist.
Zusätzlich werden die identifizierten und nicht identifizierten Bytes gezählt. Die Ergebnisse werden tabellarisch auf der Standardausgabe gelistet. Die Umleitung in eine Textdatei ermöglicht das Einlesen in Tabellenkalkulationsprogramme wie OpenOffice Calc, wo eine graphische Ausgabe in Diagrammform möglich ist.
Das Skript summarize_all.pl berechnet eine Tabelle mit allen festgestellten Protokollen. Beide Skripte nehmen auf der Kommandozeile den Dateinamen der Flußtabelle

Protokoll	Anzahl der erkannten Bytes
ASAP	280
BOOTP	46
CHAT	49495
DCCP	104
DNS	83392
ENTTEC	140
FTP	10656386
GAMING	139326
HTTP	13529459
ICMP	256404
ICP	21570
ICQ	207
IEEE	55
IP	46
IPv6	1452
KRB5	97
LLC	139
MAIL	1246075
MDNS	160
NBDS	2221
NBNS	1271
NTP	1596
P2P	157069850
RX	140
SMB	269196
SNMP	318
SSDP	210
STREAMING	390234
STUN	102
SliMP3	51707
TIME	199
TPCP	51
UNKNOWN	29838701
eDonkey	25512
recognized	183797487
total	213636188

Tabelle 5.4: Beispielausgabe von summarize_all.pl. Die Tabelleneinträge CHAT, GAMING, MAIL, P2P und STREAMING wurden durch Nonpayloadanalyse erzeugt.

entgegen und geben das Resultat auf der Standardausgabe aus. Die Beispielausgabe für eine Probe ist in Tab. 5.4 angegeben.

Das Skript summarize_all.pl kann über mehrere Dateien verteilte Flußtabellen zusammenfassen. Voraussetzung ist, daß sich die Teiltabellen in demselben Verzeichnis befinden und auf [Index].text enden.

5.4 Beurteilung der Leistungsfähigkeit der Meßumgebung

Die Meßumgebung kam bei der Applikationsprotokollanalyse der drei in Kap. 4 beschriebenen Proben unter der Zielstellung einer rechenzeiteffizienten, umfassenden Identifizierung der jeweils enthaltenen Applikationsschichtprotokolle zum Einsatz. Bereits vor der Verfügbarkeit konkreter Ergebnisse ist ersichtlich, daß die Qualität der Resultate zwangsläufig stark von der vorgelagerten Analysesoftware - im konkreten Fall Ethereal - abhängt. Wegen dessen ständiger Weiterentwicklung - insbesondere der erkannten Applikationsschichtprotokolle - ist sichergestellt, daß Protokolländerungen und -Neueinführungen zeitnah und - abgesehen von der Neuinstallation der Software - ohne Aufwand für den Operator stattfinden. Weiterhin steht es jedem Nutzer frei, im Rahmen der GPL eigenen Analysecode hinzuzufügen und damit einen systematischen Ansatz in der Analyse zu verfolgen. Der Code im Skript analyse_hybrid.pl bedient sich, Peer-to-Peer-Protokolle betreffend, zum Teil der Textmustererkennung. Die Gründe bestehen darin, daß einige der Protokolle derzeit von Ethereal für den konkreten Zweck der Protokollerkennung nicht oder nur unzureichend erkannt werden; andererseits ließ der Zeitrahmen die Entwicklung eigenen Codes nicht zu.
Der in 5.3.2 - 5.3.2.3 verfolgte Ansatz wird einerseits mit der in 5.3.2.4 vorgestellten Methode verglichen, andererseits wird auch eine Gegenüberstellung mit einer rein portbasierten Analyse durchgeführt.

Name der Probe	20041025-140917-0	20041207-154133-0	20041209-150130-0
Anzahl der aufgezeichneten Bytes	13157940140	162459529152	21370995266
Ethereal	2050018527	625616860	415911845
analyse_hybrid.pl	6472482321	150382047656	17623440722
analyse_table.pl	6539222313	116688749189	17657753394
analyse_ports.pl	4551354221	100182625237	17002164653
dominierende Anwendungsklasse	HTTP (27%)	P2P (81%)	HTTP (51%)
Profil	Forschung	Privatnutzer	Geschäftsnutzer

Tabelle 5.5: Anzahl der erkannten Bytes für jede Methode

Die detaillierten Ergebnisse per Applikationsklasse sind im Anhang (A.3) zu finden. Die hohe Erkennungsrate der von Privatnutzern dominierten Probe resultiert aus der überwiegenden Verwendung von wohlbekannten Portnummern von Peer-to-Peer-Programmen.

5.4.1 False Positives

Als *False Positives* werden falsch identifizierte Flüsse bezeichnet. Die größte Fehlerquelle besteht bei Peer-to-Peer-Verkehr mit auf wohlbekannte Dienste veränderten Portnummern. Eine rein portbasierte Analyse würde die resultierenden Peer-to-Peer-Flüsse als WWW-Verkehr mißidentifizieren. Zur Vermeidung dieser Fehler wurden die in Kap. 5.3.2.3 und 5.3.2.4 vorgestellten Heuristiken implementiert. Als weitere Fehlerquelle sind Portscans anzusehen. Dabei handelt es sich um kurze Flüsse, welche TCP- uder UDP- Verbindungen zu vielgenutzten Ports in einem IP-Adreßbereich

aufbauen. Diese Flüsse werden ausgeschlossen, wenn sie weniger als 100 Bytes umfassen[5] Die Größe des Flußlimits kann in analyse_hybrid.pl und in analyse_table.pl angepaßt werden.
Einige Peer-to-Peer-Clients, z.B. eMule, ermöglichen den Download über HTTP-Proxyserver. Während die Signalisierung direkt zwischen beiden Peers via UDP erfolgt, werden die Nutzdaten mittels des Proxyservers ausgetauscht, s. Abb5.2.

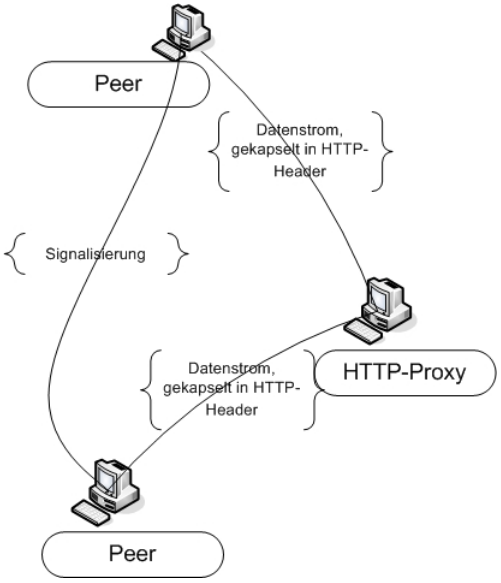


Abbildung 5.2: Dateientausch in Peer-to-Peer-Netzen mittels eines Proxyservers

Aufgrund der Kapselung des Datenstroms in einen HTTP-Header wird eine Konversation über einen Proxy als HTTP-Strom identifiziert. Da die Proxyunterstützung im Programm verankert ist, bleibt auch die Untersuchung des Payload von HTTP-Paketen ohne Erfolg. Die Nonpayloadanalyse kann diese Peer-to-Peer-Flüsse ebenfalls nicht erkennen, da die Bedingung von gleichzeitigen TCP- und UDP-Flüssen zwischen zwei IP-Adressen nicht erfüllt ist.

5.4.2 False Negatives

False Negatives sind nicht identifizierte (als „unbekannt“) markierte Flüsse, die zu einem zu identifizierenden Protokoll gehören. Nach diesem Kriterium sind alle als unbekannt eingestuften Peer-to-Peer-Flüsse False Negatives, wenn nach Peer-to-Peer-Flüssen gesucht wird. Im Kontext dieser Arbeit sind alle in den untersuchten Proben nicht identifizierten Flüsse False Negatives. Zur Vermeidung von False Positives ergriffene Maßnahmen führen zwangsläufig zu False Negatives, so daß eine vorherige Abwägung nötig ist, welche den Anteil von False Positives und False Negatives minimiert. Die Heuristik zur Identifizierung von Servern und das 100-Byte-Flußlimit beeinflussen maßgeblich die Ergebnisse. Während das Auslassen der Serverheuristik auf eine Portnummernanalyse hinausliefe, wurde das 100-Byte-Flußlimit auf 0 und 50 Byte geändert. Die kleinste auftretende Flußgröse beträgt 48Byte und tritt im ersten Schritt des 3-Wege-Handshake auf (20Byte-IP-Header, 20Byte-TCP-Header, 8Byte TCP-Optionen). Bei dieser Flußgröse ist der Verdacht eines Portscans mit

[5]Dieses Kriterium sei als 100-Byte-Flußlimit bezeichnet.

halboffener Verbindung gegeben[6]. Bei kompletter Öffnung der Verbindung bestätigt der initiierende Knoten das TCP-Acknowledgement des Servers mit ebenfalls einem TCP-Acknowledgement. Das Paket besitzt im Normalfall (ohne TCP-Optionen) eine Größe von 40Byte[7], so daß die Gesamtgröße des Flusses bis zu diesem Zeitpunkt bei 88Byte liegt. Als Berechnungsgrundlage diente die Probe 20041209-150130-0; die Berechnung wurde mit den Skripten analyse_hybrid.pl und analyse_table.pl durchgeführt.

Limitgröße (Bytes)	0	50	100
erkannte Bytes gesamt	17780291525	17763191185	17623421422
davon Peer-to-Peer	1116462178	1116462734	1116729314
davon HTTP	10946424378	10946424378	10946424378
davon DNS	113246119	113246119	113246119
davon FTP	2679125640	2679125640	2679125640
davon E-Mail	1653150020	1653150020	1653150020
davon Remote Traffic	975440816	958306386	818072979
davon Chat	12695042	12698060	12717178
davon Gaming	2863885	2890611	3043537
davon E-Business	44844	44844	44844
davon verschlüsselt	639	639	639
davon Echtzeitprotokolle/ Streaming	67511548	67511548	67511548
andere erkannte Protokolle	280837964	280841754	280866774
nicht erkannt	3590703741	3607804081	3747573844

Tabelle 5.6: Ergebnisse bei verschiedenen Flußgrößenlimits (hybride Methode)

Wie Tab. 5.6 zeigt, wird bei der Hybridanalyse auschließlich die Verkehrsklasse „Remote Traffic“ nennenswert beeinflußt. Das ist dadurch zu erklären, daß das SMB-Protokoll in dieser Gruppe geführt wird und die Großzahl an Würmern Verbindungsversuche auf die zugeordneten TCP-Ports 135 und vor allem 445 starten.

Ähnlich Tab. 5.6 zeigt Tab. 5.7 lediglich bei der Gruppe REMOTE eine Abhängigkeit vom gewählten Transferlimit. Um Fehler durch Portscans auszuschließen, wird eine Mindestflußgröße von 100Byte empfohlen.

5.4.3 Berechnungszeit

Die berücksichtigte Berechnungszeit schließt lediglich die Auswerung der Flußtabellen ein; die Erstellung der Flußtabellen sowie die Payloadanalyse durch Tethereal wird ausgeklammert, da die Berechnungsgeschwindigkeit der eingebundenen Programme nicht beeinflußbar ist.

Alle Zeiten wurden auf dem Meßsystem ermittelt (vgl. Tab. 4.1). Aufgrund des hohen Speicherbedarfs und der damit verbundenen längeren Speicherzugriffszeit benötigt die Nonpayloadanalyse die meiste Zeit für die Berechnung der Tabelle. Betreffs der

[6] Andere Gründe wie z.B: ein Programmabsturz oder die verbindung zu einer inaktiven IP-Adresse sind ebenfalls denkbar

[7] 20Byte IP-Header, 20Byte TCP-Header

[8] blockweise Betrachtung von je 20 Teildateien zu 200MiB

Limitgröße (Bytes)	0	50	100
erkannte Bytes gesamt	17816104587	17798754730	17657753394
davon Peer-to-Peer	560581166	560598108	560636724
davon HTTP	11935298465	11935277130	11935261544
davon DNS	99639772	99634594	99634594
davon FTP	2678493594	2678485664	2678440023
davon E-Mail	1313833212	1313835618	1313833572
davon Remote Traffic	1040491561	1023154303	882920896
davon Chat	13015493	13023505	13022865
davon Gaming	1472137	1467819	736803
davon Echtzeitprotokolle/ Streaming	16291924	16290726	16279110
andere erkannte Protokolle	156987263	156987263	156987263
nicht erkannt	3554890679	3572240536	3713241872

Tabelle 5.7: Ergebnisse bei verschiedenen Flußgrößenlimits (Methode ohne Payloaduntersuchung)

Skript	analyse_hybrid.pl	analyse_table.pl[8]	analyse_ports.pl
20041025-140917-0	320s	329s	102s
20041209-150133-0	2021s	2151s	551s
20041207-154133-0	39751s	20515s	6131s

Tabelle 5.8: Berechnungszeiten der einzelnen Methoden bei verschiedenen Tracegrößen

für die Probe 20041207-154133-0 ermittelte Berechnungszeiten ist die Partitionierung der Gesamtflußtabelle für analyse_table.pl zu berücksichtigen. Ebenfalls zu beachten ist die bei der Payloadanalyse zusätzlich benötigte Zeit für die Auswertung durch Ethereal. Auf dem Meßcomputer sind je nach Zusammensetzung der Meßprobe ca. (15...20)min für ein 200MiB großes Teiltrace zu veranschlagen.

Einer Verbesserung der Leistung ist durch Berechnung der Flußtabelle während der Aufzeichnung der Pakete, wie crl_flow sie ermöglicht, erreichbar. Dieser Vorschlag zielt auf die Beschleunigung der Nonpayload-Analyse ab. Die Schnittstellen des Skripts analyse_table.pl wären entsprechend zu ändern, ergänzend wäre zu untersuchen, bis zu welcher Bitrate eine schritthaltende Verarbeitung möglich ist. Weiterhin müßten die TCP- und UDP-Verbindungstabellen periodisch gelöscht werden. Auf diese Weise würde der erste Durchlauf durch die Flußtabelle in Echtzeit erfolgen, die beiden weiteren nötigen Durchläufe würden weiter offline stattfinden.
Zur Beschleunigung der Payloadanalyse ist es vorstellbar, die Meßdateigröße auf von Tethereal zu verarbeitende Größen zu beschränken und rotierend zu speichern. Parallel zur Aufnahme der Meßdaten könnte bei ausreichender Rechenleistung die Payloadanalyse durch Tethereal zeitversetzt nach Vervollständigung einer Teilmeßdatei erfolgen.

6. Zusammenfassung und Ausblick

Für Netzwerkadministratoren und Internet Service Provider gewinnt die Notwendigkeit der Kenntnis über die Zusammensetzung von Protokollen der Anwendungsschicht in Zeiten einer wachsenden Anzahl von netzwerkfähigen Applikationen stetig an Bedeutung. Die Bearbeitung der gestellten Aufgaben erfolgte bewußt mit Hinsicht auf die Verwendung kostenlos und frei verfügbarer Software. Die Delegierung von Analyse der Applikationsschichtprotokolle und der Erstellung der Flußtabellen begründen sich auf die Ausgereiftheit der ausgewählten Programme gegenüber möglichen Eigenentwicklungen. Weiterhin ist sichergestellt, daß eine Wartung der entwickelten Umgebung weitgehend entfallen und eine Anpassung nach den individuellen Bedürfnisen des Benutzers erfolgen kann. Die Verarbeitung großer Traces ist ebenfalls gewährleistet.
Im Verlauf der Arbeit wurden die verbreitetsten Protokolle der Applikationsschicht vorgestellt. Engegen den Behauptungen in [KBFc04] stellte sich heraus, daß die untersuchten Peer-to-Peer-Protokolle bis auf Fasttrack und SoulSeek durchaus gut dokumentiert sind. Erwartungsgemäß erzielt die Payloadanalyse die höchste Erkennungsrate des untersuchten Probenmaterials. Bei Umsetzung der vorgestellten Verbesserungsvorschläge in Kap. 5.4.3 kann die Nonpayloadanalyse bei der Untersuchung der meistgenutzten Dienste des Internet zu einer zeiteffizienteren Alternative gegenüber der Payloadanalyse werden. Es wurden ebenfalls Möglichkeiten zur Verbesserung des Laufzeitverhaltens der payloadgestützten Analyseumgebung umrissen. Ergänzende Schritte zur Entwicklung einer ganzheitlichen Lösung im Rahmen des Traffic Engineering können aus

- der Integration in eine IPFIX- oder NetFlow-Umgebung,
- einer verbesserten graphischen Aufbereitung der Analyseergebnisse z.B. unter MATLAB,
- der Laufzeitoptimierung, wie in 5.4.3 vorgeschlagen,
- der Ersetzung von Ethereal durch eine schnellere Alternative bzw. dessen Anpassung durch Umgehung des Dissector-Overhead[1]

[1] Es kommt lediglich auf die Erkennung der Protokolle der Applikationsschicht an, nicht auf die Funktion des spezifischen Pakets innerhalb des Protokolls.

bestehen. Eine Untersuchung in [KBFc04] vorgestellten und in Kap. 5.3.2.4 aufgegriffenen Heuristiken bezüglich der Erkennung von Peer-to-Peer-Flüssen auf die Eignung für eine Simulation bleibt ebenfalls offen.

A. Anhang

A.1 Das Bencoded-Format in Bittorrent

Datentyp	Form	Beispiel	Interpretation
String	<Länge>:<String>	19:Bittorrent Protocol	Bittorrent Protocol
Integer	i<Zahl>e	i23e	23
List	l<Länge Element 1>: <Element 1> <Länge Element 2>: ...e	l7:leecher7:releasee	['leecher', 'release']
Dictionary	d<Länge Schlüssel 1>: <Schlüssel 1><Länge Wert 1> :<Element 1> <Länge Schlüssel 2> :<Wert 2>...e	d5:leechl2:peer:a1:releasee	'leech': ['pe', 'r']

Tabelle A.1: Datentypen im Bencoded Format. Die Längenangaben werden hexadezimal codiert ($19 \equiv 0x13$).

A.2 Aufbau von Dictionaries in Bittorrent

Dictionaries (Verzeichnisse) haben den Aufbau Schlüssel:Wert. Metadateien enthalten Verzeichnisse mit den Schlüsseln in Tab. A.2.

Schlüsel	Wert
announce	URL des Trackers
info	verschachteltes Dictionary
info:name	Default-Speichername der Datei
info:piece length	Zahl von Bytes eines Dateifragments, in die eine Datei für die Überttragung zerlegt wird (Chunkgröße) Voreinstellung: 2^{18}
info:pieces	String, dessen Länge ein ganzzahliges Vielfaches von 20 ist. Jeder der 20-elementigen Substrings ist der SHA1-Hash des indexierten Dateifragments
info:length	Dateigröße in Bytes (1 Datei) oder Verzeichnis mit files als Unterverzeichnis (mehrere Dateien)
info:length:files	enthält Liste der Dateien, vgl. Tab. A.1

Tabelle A.2: Inhalt einer Bittorrent-Metadatei

A.3 Analyseergebnisse der aufgezeichneten Proben

Protokoll	Payloadanalyse	Nonpayloadanalyse	Portnummernanalyse
P2P	84777	47866938	84777
CHAT	8378016	8344586	8546746
REMOTE[1]	2004159901	2045909050	110132673
EMAIL	1185744366	1141872829	1141058302
ENCRYPTED	100	0	0
FTP	20918825	20917917	20918355
EBUSINESS	348	0	0
HTTP	3215981007	3260016260	3211867398
DNS	8736667	8556339	8435027
GAMING	157295	10837	4351153
STREAMING/ REALTIME	75136	938	45959790
OTHER	28245883	5726619	0
missed	5319356817	5252616825	7240484917
recognized	6472482321	6539222313	4551354221
total	11791839138	11791839138	11791839138

Tabelle A.3: Ergebnisse der Analyse für die Probe 20041025-140917-0 in Bytes. Es besteht weitgehende Übereinstimmung zwischen den Ergebnissen von Payload- und Nonpayloadanalyse.

Protokoll	Payloadanalyse	Nonpayloadanalyse	Portnummernanalyse
P2P	132496445113	88018466138	73325510131
CHAT	110272301	172954079	207936976
REMOTE[2]	322817402	2249912615	322743221
EMAIL	1867647327	1531283565	1626565230
ENCRYPTED	35522	0	0
FTP	3324457202	6332813396	6427686386
EBUSINESS	161494	0	0
HTTP	11178689233	16939680201	16570160083
DNS	86516228	88768105	79360426
GAMING	208446927	715343398	884449045
STREAMING/ REALTIME	507871939	503924750	738213739
OTHER	278686968	135602942	0
missed	12043876733	45737175200	62243299152
recognized	150382047656	116688749189	100182625237
total	162425924389	162425924389	162425924389

Tabelle A.4: Ergebnisse der Analyse für die Probe 20041207-154133-0. Die Nonpayloadanalyse erkannte deutlich weniger Peer-to-Peer-Flüsse als die Payloadanalyse. Das ist darin begründet, daß nicht alle Peer-to-Peer-Protokolle (z.B. SoulSeek) UDP und TCP als Transportprotokolle nutzen. Außerdem konnten aus Speichergründen nur Teile der Flußtabelle unabhängig voneinander untersucht werden, was zu Ungenauigkeiten führt (vgl. Kap. 5.3.2.4).

Protokoll	Payloadanalyse	Nonpayloadanalyse	Portnummernanalyse
P2P	1116729314	560636724	419745014
CHAT	12717178	13022865	13595065
REMOTE[3]	818072979	882920896	818073702
EMAIL	1653150020	1313833572	1315855342
ENCRYPTED	639	0	0
FTP	2679125640	2678440023	2678921054
EBUSINESS	44844	0	0
HTTP	10946424378	11935261544	11561390518
DNS	113246119	99634594	106739767
GAMING	3043537	736803	17610851
STREAMING/ REALTIME	67511548	16279110	70233340
OTHER	213355226	156987263	0
missed	3747573844	3713241872	4368830613
recognized	17623421422	17657753394	17002164653
total	21370995266	21370995266	21370995266

Tabelle A.5: Ergebnisse der Analyse für die Probe 20041209-150130-0. Die Nonpayloadanalyse erkannte nur ca. 50% der mittels der Payloadanalyse identifizierten Peer-to-Peer-Bytes.

A.4 Programmablaufpläne für die Analyse mit und ohne Payloaduntersuchung

Informationen werden in Hashtabellen gespeichert. Die Notation lautet *Tabellenname*{*Schlüssel*}=*Wert*. Variablennamen sind kursiv, Befehlskonstrukte in Großbuchstaben und Konstanten in kursiver Großschrift dargestellt.

[1]beinhaltet Telnet (TCP-Port 23), SSH (TCP-Port 22), SMB (TCP-Ports 135, 445), PORTMAP (TCP-Port 111), NFS

[3]beinhaltet Telnet (TCP-Port 23), SSH (TCP-Port 22), SMB (TCP-Ports 135, 445), PORTMAP (TCP-Port 111), NFS

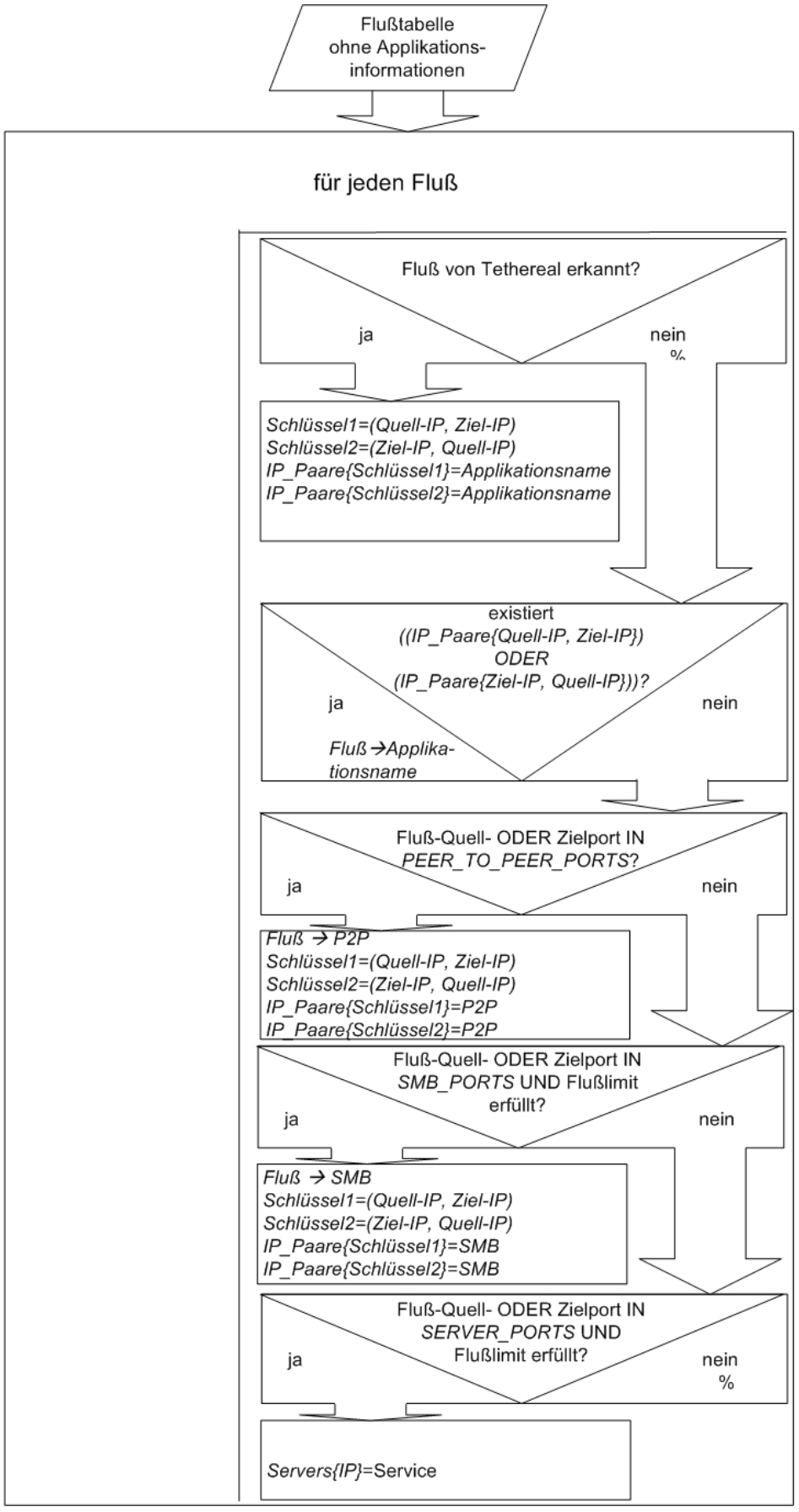


Abbildung A.1: Programmablaufplan der hybriden Analyse mit Payloaduntersuchung, Schritt 1

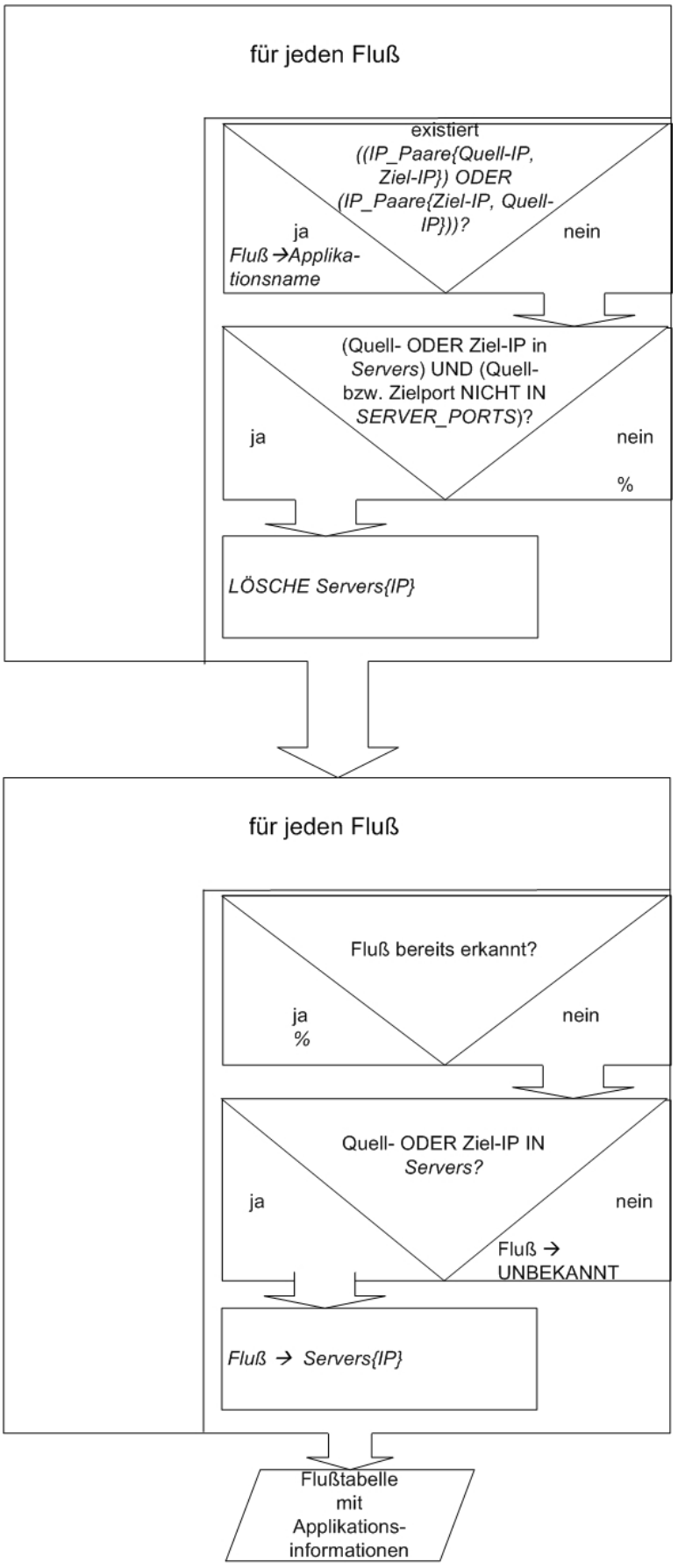


Abbildung A.2: Programmablaufplan der hybriden Analyse mit Payloaduntersuchung, Schritt 2

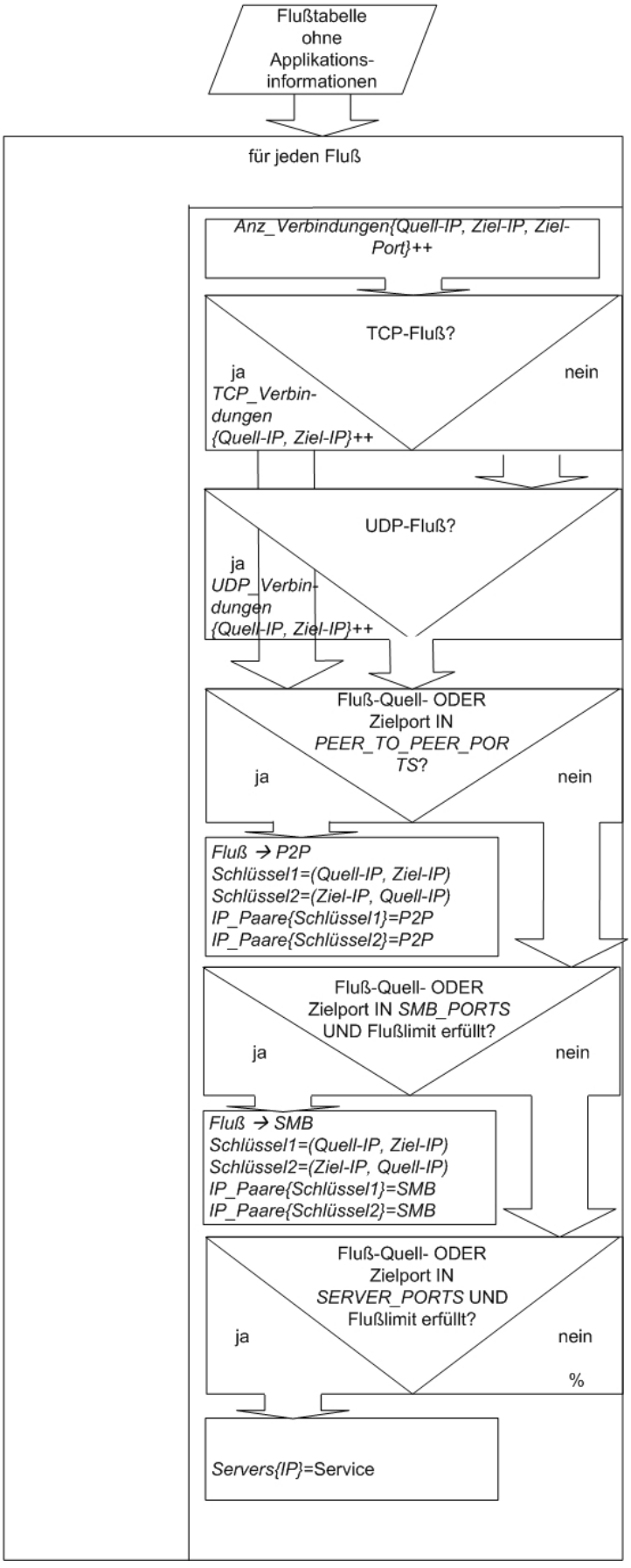


Abbildung A.3: Programmablaufplan der Analyse ohne Payloaduntersuchung, Schritt 1

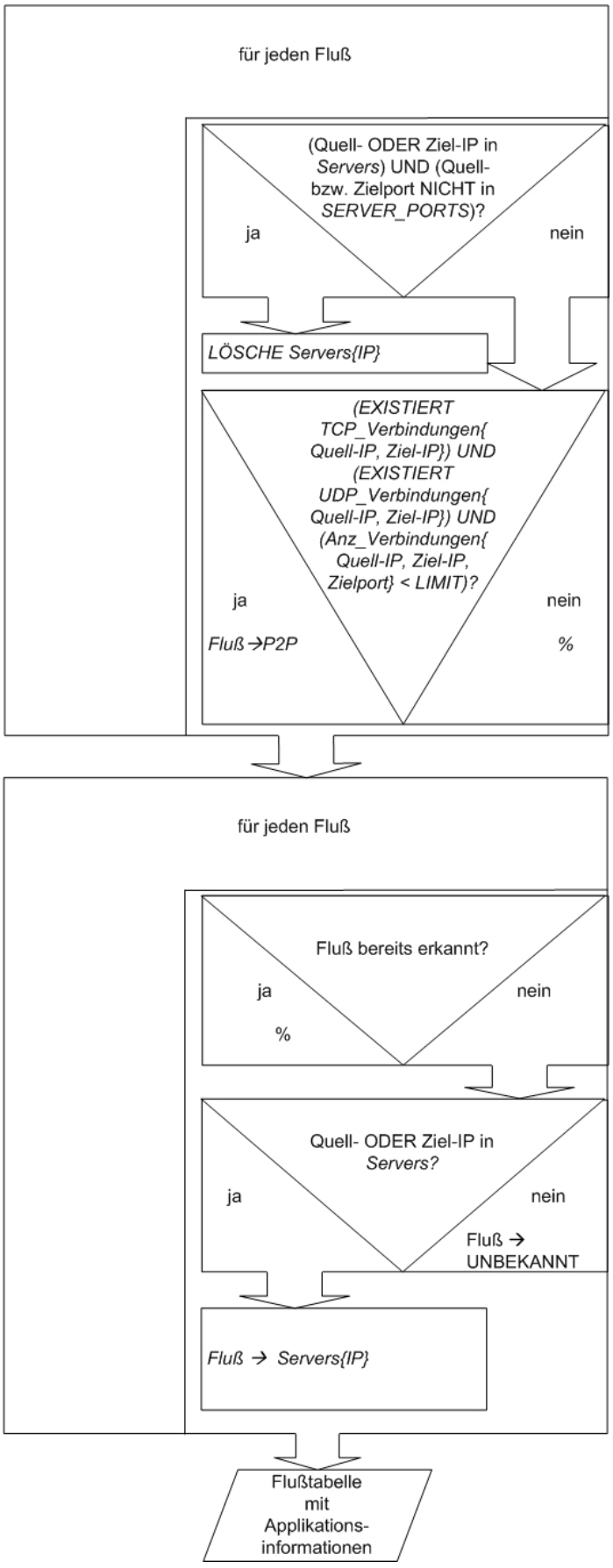


Abbildung A.4: Programmablaufplan der Analyse ohne Payloaduntersuchung, Schritt 2

Literaturverzeichnis

[Com00] Douglas E. Comer
Internetworking with TCP/IP Volume 1.
Prentice Hall, Inc., 2000.

[Ste00] W. Richard Stevens
TCP/IP Illustrated Vol. 1.
Addison-Wesley Longman, Inc., 2000.

[Bru05] Udo Brunswig
Analyse und Visualisierung des Kommunikationsaufkommens in paketorientierten Netzen.
Diplomarbeit
Universität Rostock, Institut für Nachrichtentechnik und Informationselektronik, 2005

[Coh05] Bram Cohen
Bittorrent Protocol Specification.
http://www.bittorrent.com/protocol.html
2005

[KBFc04] Thomas Karagiannis, Andre Broido, Michailis Faloutsos, Kc claffy
Transport Layer Identification of P2P Traffic. IMC'04
CAIDA, UC Riverside, 2004

[Sto03] Michael Stokes
Gnutella2 Specification Document First Draft. Shareaza Pty. Ltd., 2003
http://dsl.upc.es/netscout/doc/gnutella2_draft.htm

[Sole03] Brian Enigma, Robert Sayre, Arend van Beelen jr.
SoulSeek Protocol Documentation.
The SoleSeek Project, 2003
http://cvs.sourceforge.net/viewcvs.py/soleseek/SoleSeek/doc/protocol.html

[RFC2616] R. Fielding, J. Gettys, J. Mogul, et. al.
RFC 2616 - Hypertext Transfer Protocol – HTTP/1.1
Network Working Group, ISOC, 1999
http://www.faqs.org/rfcs/rfc2616.html

[IANA] IANA
PORT NUMBERS.
IANA, 2005
http://www.iana.org/assignments/port-numbers

[TCPdump] TCPdump
tcpdump/libpcap
2004
http://www.tcpdump.org

[ST04] Stanislav Shalunov, Benjamin Teitelbaum
Internet2 NetFlow Statistics.
Internet2, 2004
http://netflow.internet2.edu/

[Cap01] Prof. Clemens Cap
Skriptum zur Vorlesung „Internet-Protokolle und -Dienste.“
Universität Rostock
2002

[Gall02] Rick Gallaher
MPLS Traffic Engineering.
Converge! Network Digest
http://www.convergedigest.com/tutorials/mpls5/page1.asp
2002

[Hei04] Heise Online
Edonkey überholt Kazaa in der Nutzergunst.
Heise Online
http://www.heise.de/newsticker/meldung/52060
2004

[MS05] Microsoft Corporation
Wie gehe ich vor, um Ports in der Internetverbindungsfirewall von Windows XP zu öffnen?
Microsoft Corporation
http://www.microsoft.com/germany/ms/security/ports.mspx
2005

[RFC3272] D. Awduche et. al.
Overview and Principles of Internet Traffic Engineering.
IETF, 2002
RFC3272 Informational

[Cis01] Cisco Systems, Inc.
Catalyst 3550 Multilayer Switch Software Configuration Guide, 12.1(6)EA1 - Configuring SPAN
Cisco Systems, Inc., 2001

[Cis04] Cisco Systems, Inc.
Netflow for Accounting, Analysis and Attack.
http://www.cisco.com/warp/public/732/Tech/nmp/netflow/presos/
2004

[Cis05] Cisco Systems, Inc.
Cisco IOS NetFlow.
http://www.cisco.com/warp/public/732/Tech/nmp/netflow/
2005

[End05] Endace Measurement Systems
Homepage
http://www.endace.com/
2005

[F5] F5 Networks
BIG-IP Traffic Management.
http://www.f5.com/f5products/products/bigip/
2004

[Packeteer] Packeteer, Inc.
Homepage
http://www.packeteer.com/
2004

[Eth05] Gerald Combs, et. al.
Ethereal - The world's most popular network protocol analyzer.
http://www.ethereal.com
2005

[CR04] The CoralReef Team
CoralReef Homepage
http://www.caida.org/tools/measurement/coralreef/
2004

[FAQ03] Advameg, Inc.
Internet RFC/STD/FYI/BCP Archives.
http://www.faqs.org/rfcs/

2003

[OSC1] Ryan Tenney et. al.
The OSCAR protocol documentation project.
http://joust.kano.net/wiki/oscar/
2005

[OSC2] A. V. Shutko
OSCAR (ICQ v7/v8/v9) protocol documentation.
http://iserverd.khstu.ru/oscar/
2005

[nP04] Luca Deri
nProbe: a NetFlow v5/v9/nFlow Probe for IPv4/v6.
http://www.ntop.org/ntop.html
2004

[ML99] R. Movva, W. Lai
MSN Messenger Service 1.0 Protocol.
http://www.hypothetic.org/docs/msn/ietf_draft.txt
1999

[icq2GO] ICQ, Inc.
icq2GO-Your Web-based ICQ!
http://www.icq.com/icq2go/
2005

[Hei05] Heise Online
Bertelsmann-Tochter plant Distributions-Plattform mit P2P-Technik.
http://www.heise.de/newsticker/meldung/57805
2005

[RFC3954] B. Claise
Cisco Systems NetFlow Services Export Version 9.
http://www.faqs.org/rfcs/rfc3954.html
2004

[RFC2460] S. Deering, R. Hinden
Internet Protocol, Version 6 (IPv6) Specification.
http://www.faqs.org/rfcs/rfc2460.html
1998

[RFC791] J. Postel
Internet Protocol Specification.
http://www.faqs.org/rfcs/rfc791.html
1981

[RFC793] J. Postel
Transmission Control Protocol.
http://www.faqs.org/rfcs/rfc793.html
1981

[RFC768] J. Postel
User Datagram Protocol.
http://www.faqs.org/rfcs/rfc768.html
1980

[RFC1939] J. Myers, M. Rose
Post Office Protocol - Version 3.
http://www.faqs.org/rfcs/rfc1939.html
1996

[MSDN05] Microsoft Developer Network
SMB Packet Header.
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cifs/protocol/smb_header.asp
2005

[KB05] Yoram Kulbak, Danny Bickson
The eMule Protocol Specification.
School of Computer Science and Engineering
The Hebrew University of Jerusalem, Israel
http://www.cs.huji.ac.il/labs/danss/presentations/emule.pdf
2005

Soulseek
http://www.slsknet.org.

Bittorrent
http://bitconjurer.org/BitTorrent.

eMule
http://www.emule-project.net.

KaZaA
http://www.kazaa.com.

GNUtella
http://www.gnutella.com.

DirectConnect
http://www.neo-modus.com.
http://dcplusplus.sourceforge.net.