JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Sicherheitsaspekte von Peer-to-Peer Netzwerken

Diplomarbeit zur Erlangung des akademischen Grades *Diplom-Ingenieur*

in der Studienrichtung *Informatik*

Angefertigt am Institut für *Informationsverarbeitung und Mikroprozessortechnik*

Betreuung:

o. Prof. Dr. Jörg R. Mühlbacher

Von:

Georg Linhard

Mitbetreuung:

Dipl.-Ing. Rudolf Hörmanseder

Linz, Oktober 2006

Zusammenfassung

Diese Diplomarbeit beschäftigt sich mit Peer-to-Peer Netzen, den Problemen und Risiken, welche bei der Nutzung für Unternehmen und Benutzer entstehen können. Die theoretischen Grundlagen der Technologie werden untersucht und die existierenden Peer-to-Peer Strukturen werden erklärt. Die Anwendungsbereiche von Peer-to-Peer werden genauer betrachtet.

Weiters wird ein Überblick über die Sicherheitsaspekte gegeben, wobei eine Kategorisierung der Konzepte zum Überwinden von Firewalls, welche diese Anwendungen verwenden, vorgenommen wurde. Anschließend werden die am weitest verbreiteten Tauschbörsen und Instant Messaging Netze mit ihrer Netzarchitektur und den zugehörigen Protokollen beschrieben. Darauf folgend werden mögliche Lösungsansätze für die beschriebenen Probleme angeführt. Ein mögliches Szenario für die Auswirkungen von Peer-to-Peer auf ein Unternehmen und für die Anwendung von Skype in einem Unternehmen wird beschrieben.

Abschließend wird vorhandene Open Source Software, welche der Lösung dient, getestet und ebenfalls vorgestellt.

Abstract

This diploma thesis deals with peer-to-peer networks, the problems and hazards arising from the use, for corporations or home users. The underlying theory of this technology is analysed and the existing peer-to-peer structures are being explained. A closer look is taken at the application range of peer-to-peer.

A survey of the security aspects is given, whereas a categorization of the concepts used by these applications to overcome firewalls , is conducted. Afterwards the most popular file sharing and instant messaging networks are described, including their network architecture and the associated protocols. Hereon, possible methods of resolution and solutions for the given problems are given. A possible scenario, describing the impact of peer-to-peer on enterprises, as well as a scenario for the application of Skype in an enterprise, is pictured.

Completing, the existing solutions in Open Source software are tested and introduced.

Inhaltsverzeichnis

1 Einleitung **9**

1.1 Motivation 10

1.2 Aufgabenstellung 11

2 Peer-to-Peer Grundlagen **12**

2.1 Definition 12

2.2 Peer-to-Peer Strukturen 13

2.2.1 Hybride Netze 13

2.2.2 Serverlose Peer-to-Peer Netze 14

3 Anwendungsbereiche von Peer-to-Peer **16**

3.1 Instant Messaging, Internet Telefonie 17

3.2 File Sharing 17

3.3 Verteiltes Rechnen, Grid Computing 18

3.4 Peer-to-Peer Groupware 18

4 Sicherheitsaspekte **20**

4.1 Risiko durch die Peer-to-Peer Software selbst 21

4.2 Einschleusen schädlicher Dateien 21

4.3 Überwinden von Sicherheitssystemen an den Unternehmensgrenzen 22

4.3.1 Konzepte zur Überwindung von Firewalls 23

4.4 Unerwünschte Erhöhung des übertragenen Datenvolumens . . 24

4.5 Unbeabsichtigtes Freigeben von Verzeichnissen 26

4.6 Datenschutz 27
4.7 Diebstahl der Identität, Social Engineering 28
4.8 Rechtliche Risiken 29

5 Beispielszenario **30**
5.1 Aufbau des Netzes 30
5.2 Das erste Auftreten von Peer-to-Peer 31
5.2.1 Die Suche nach der Ursache - Verwenden von Intrusion Detection 32
5.3 Umgehen der Standardports 33
5.3.1 Peer-to-Peer in der Sicherheitspolitik 33
5.3.2 Content Inspection 34
5.3.3 Offizieller Einsatz von Peer-to-Peer im Unternehmen . 36

6 Peer-to-Peer Filesharing Netze und Protokolle **37**
6.1 Gnutella 37
6.1.1 Die Verbindung zum Gnutella-Netz 38
6.1.2 Der Ultrapeer Modus 38
6.1.3 Suchanfragen 40
6.1.4 Der Dateitransfer 41
6.2 eDonkey / eMule 44
6.2.1 Kommunikation im eDonkey-Netz 44
6.3 BitTorrent 49
6.3.1 Netzarchitektur 49
6.3.2 Torrent-Dateien 50
6.3.3 Dateitransfer 51
6.4 FastTrack 53
6.4.1 FastTrack Netzarchitektur 54
6.4.2 Verbindungsaufbau 55
6.4.3 Suchanfragen 56
6.4.4 Dateitransfer 56
6.4.5 Verschlüsselung 56

7 Instant Messaging Netze **58**
7.1 Skype . . . 58
7.1.1 Netzarchitektur . . . 59
7.1.2 Verbindungsaufbau . . . 60
7.1.3 Benutzersuche . . . 61
7.1.4 Verbindung zu anderen Benutzern (Sprachtelefonie) . . . 62
7.2 AOL Instant Messenger und ICQ . . . 65
7.2.1 Netzarchitektur . . . 65
7.2.2 Kommunikation . . . 66
7.2.3 Verschlüsselung . . . 67
7.3 MSN Messenger . . . 69
7.3.1 Netzarchitektur . . . 69
7.3.2 Verbindungsaufbau . . . 71
7.3.3 Kommunikation . . . 72

8 Lösungen, Lösungsansätze **74**
8.1 Einbeziehen von Peer-to-Peer in die Sicherheitspolitik des Unternehmens . . . 74
8.2 Die Verwendung von Instant Messaging in Unternehmen am Beispiel Skype . . . 75
8.2.1 Sicherheitsaspekte von Skype . . . 75
8.2.2 Technische Massnahmen . . . 78
8.3 Peer-to-Peer Netzverkehr an den Unternehmensgrenzen erkennen . . . 80
8.3.1 Erkennen von Peer-to-Peer Protokollen anhand von TCP/UDP-Ports . . . 80
8.3.2 Erkennen von Peer-to-Peer Protokollen anhand der Analyse der Anwendungsschicht . . . 80
8.4 Open Source Projekte . . . 81
8.4.1 IPP2P . . . 81
8.4.2 Layer7-filter . . . 85
8.4.3 Snort . . . 87

8.4.4 Zusammenfassung 88

9 Fazit **90**

Literaturverzeichnis **92**

Internetquellen **96**

A Standardports von Peer-to-Peer Tauschbörsen und Instant Messengern **102**

B Protokoll-Analyse **104**

B.1 Gnutella 104

B.1.1 Handshake 104

B.1.2 Suchanfragen (Queries) 105

B.1.3 Dateitransfer 107

B.2 Bit Torrent 108

B.2.1 Bencoding 108

B.2.2 Dictionary für Torrent-Dateien 109

B.2.3 GET Anfragen an den Tracker 109

B.2.4 Tracker-Antworten 110

B.2.5 Verbindungsaufbau 111

B.2.6 Nachrichtenaustausch 111

B.3 FastTrack 112

B.3.1 UDP-Pakete 113

B.3.2 TCP-Pakete 114

B.3.3 Dateitransfer 117

C Snort Regeln **119**

C.1 Peer-to-Peer Filesharing 119

C.2 Instant Messaging 124

Lebenslauf **130**

Abbildungsverzeichnis

2.1 **Client-Server Struktur** (sinngemäß: [SW04]) 13
2.2 **Hybride (Serverbasierte) Struktur** (sinngemäß: [SW04]) . 14
2.3 **Serverlose (dezentrale) Peer-to-Peer Struktur** (sinngemäß: [SW04]) 15

4.1 **Push-Technik bei Kazaa ab Version 2** (sinngemäß: [JK03]) 25
4.2 **Überwindung des Double-Firewall Problems** 26

5.1 **Netzwerk des Beispielszenarios** 31
5.2 **Snort Beispielregel: Gnutella** 32
5.3 **IPP2P Modul bei Gibraltar** 35
5.4 **Gibraltar Firewall: Traffic Shaping** 36

6.1 **Gnutella Netzarchitektur** 39
6.2 **Ablaufdiagramm Gnutella Verbindungsaufbau** 40
6.3 **Suchanfragen und deren Weiterleitung im Gnutella Netz** 42
6.4 **Verbindungen im eDonkey/eMule Netzwerk (sinngemäß: [KB05])** 45
6.5 **Multisource File Transmission Protocol (MFTP) (sinngemäß: [Met])** 47
6.6 **BitTorrent Kommunikation (sinngemäß: [PBE+05])** . . 50
6.7 **Stuktur des FastTrack Netzes (sinngemäß: [PBE+05])** . 54

7.1 **Skype Netzarchitektur [PBE+05]** 60

7.2 **Verbindungsaufbau des Skype-Clients zu einem Supernode [BS04]** 64
7.3 **AIM und ICQ: Architektur und Kommunikation** (sinngemäß: [PBE+05]) 68
7.4 **Bedrohungen für Instant Messenger 2005** (Quelle: [Log06]) 70
7.5 **Verbindungsaufbau im MSN-Messenger Netz** (Quelle: [PBE+05]) 71

8.1 **Gibraltar Firewall: Verwendung des IPP2P-Moduls** . . 83

B.1 **Descriptor Paket Header (23 Bytes)** 105
B.2 **Query Descriptor Paket** 106
B.3 **QueryHits Descriptor Paket** 106
B.4 **QueryHits Results Set Format (Treffermenge)** 107
B.5 **FastTrack Paket: Node Ping** 113
B.6 **FastTrack Paket: Node Pong** 113
B.7 **FastTrack Paket: Supernode List** 114
B.8 **FastTrack Paket: User information** 114
B.9 **FastTrack Paket: Unshare file** 115
B.10 **FastTrack Paket: Search query** 115
B.11 **FastTrack Paket: Search term** 116
B.12 **FastTrack Paket: Search reply** 116
B.13 **FastTrack Paket: Search result** 116

Tabellenverzeichnis

8.1 **IPP2P Version 0.8.1: Erkennbare Peer-to-Peer Netze mit Schaltern** [www-20] 84

8.2 **Layer7-filter: aktuell erkennbare Protokolle [www-32]** . 86

8.3 **Vergleich: Open Source Software zur Erkennung von P2P** 89

A.1 **Standardports von Peer-to-Peer Protokollen** 102

B.1 **X-KazaaTags** (Quelle: [PBE+05]) 118

Kapitel 1

Einleitung

Peer-to-Peer (meist wird die Abkürzung P2P verwendet) Netze erfreuen sich, vor allem seit der Veröffentlichung der Anwendung *Napster* im Jahr 1998, unter Benutzern hoher Beliebtheit. Napster ermöglichte das Freigeben von Musikdateien für die gesamte Napster-Anwendergemeinschaft und das direkte Herunterladen der Dateien von deren Rechnern. Heute ist Napster ein kommerzieller Dienst, die ursprüngliche Tauschbörse musste nach mehreren Gerichtsprozessen und dem missglückten Versuch der Firma Bertelsmann, aus Napster ein legales Geschäft zu machen, 2002 geschlossen werden [Rö03].

Der Trend zu Peer-to-Peer Anwendungen ist jedoch ungebrochen, wobei die Technologie keineswegs neu ist, im Gegenteil, das frühe Internet bestand aus mehreren wenigen, gleichberechtigten Rechnern – im Grunde eine Peer-to-Peer Struktur. Weitere Beispiele für frühe Anwendungen von Peer-to-Peer sind das Usenet oder das Domain Name System (DNS) [Ora01].

Moderne Anwendungen der Peer-to-Peer Technologie finden sich beim schon erwähnten Filesharing, bei Instant Messaging (IM) Diensten, Verteiltem Rechnen (Distributed Computing/Grid) und Peer-to-Peer Groupware. Im Mittelpunkt steht hierbei immer ein Teilen von Ressourcen oder das Verbreiten von Informationen und Daten ohne eine zentrale Koordinationsinstanz [SF02], sowie der direkte Datenaustausch zwischen den Knoten [Bar01].

1.1 Motivation

Peer-to-Peer Filesharing Anwendungen werden oft für den illegalen Austausch von urheberrechtlich geschützten Dateien verwendet, was zu rechtlichen Konsequenzen für den Benutzer oder ein betroffenes Unternehmen führen kann.

Interoperabilität trotz vorhandener Firewalls ist meist sogar ein Designziel von Peer-to-Peer Anwendungen, die Grenzen eines Unternehmensnetzwerkes lassen sich somit nicht mehr lückenlos überwachen [Hur02]. Dabei verursachen diese Protokolle, vor allem jene der Tauschbörsen, oft starken und, zumindest für die Netzwerkadministration, unerwünschten Netzwerkverkehr. Ausserdem stammt die Peer-to-Peer Software meist aus wenig vertrauenswürdigen Quellen, was die Sicherheit der Arbeitsplatzrechner, etwa durch Ausspionieren von persönlichen oder firmeninternen Informationen, gefährdet.

Ein Unternehmen mag sich dieser Probleme durch ein Peer-to-Peer Verbot und dem Blockieren dieser Dienste noch relativ gut erwehren können, bei Universitätsnetzwerken liegt die Lösung allerdings nicht so nahe. Universitätsnetze sind üblicherweise mit weniger restriktiven Sicherheitsmaßnahmen ausgestattet, um für Forschungs- und Lehrzwecke so offen wie möglich zu sein.

Die Benutzerrechte auf Unternehmensrechnern lassen sich so einschränken, dass sich Fremdsoftware erst gar nicht installieren lässt. Denkt man beispielsweise an die Netzstruktur eines Studentenheimes, dann stellt man fest, dass hier die Rechner den Bewohnern des Heimes gehören, die Netzwerkinfrastruktur wird von der Heimleitung und der Universität zur Verfügung gestellt. Eine Kontrolle über die auf den Rechnern installierten Anwendungen ist somit unmöglich.

Die speziellen Eigenschaften der Peer-to-Peer Anwendungen und Netzwerke führen also nicht nur zu neuen Möglichkeiten in der Anwendung, sie stellen auch neue Anforderungen an bestehende Sicherheitsstrukturen von Unternehmens- und Universitätsnetzen.

1.2 Aufgabenstellung

Diese Diplomarbeit gibt einen Überblick über Peer-to-Peer im Allgemeinen, die verfügbaren Netze und Protokolle, und untersucht auch Gefahren und Probleme, welche durch die Nutzung auftreten können.

Zusätzlich werden die Möglichkeiten, P2P-Netzwerkverkehr aufzuspüren bzw. zu unterbinden, beispielsweise durch Intrusion Detection Systeme, Firewalls oder mit Hilfe von Traffic Shaping, aus Sicht des Netzwerkadministrators erläutert.

Kapitel 2

Peer-to-Peer Grundlagen

2.1 Definition

David Barkai definiert in [Bar01] *Peer-to-Peer Computing* mit Hilfe des Client-Server Modells (siehe Abb. 2.1): Im Client-Server Modell richtet der Client-Rechner (typischerweise der Rechner des Benutzers) seine Anfragen an einen Server, welcher auf diese antwortet. *Peer-to-Peer* bedeutet, dass jeder Peer (jeder teilnehmende Rechner) sowohl als Client als auch als Server agieren kann.

Eine formale Definition, ebenfalls in [Bar01] zu finden, lautet:

„Peer-to-peer computing is a network-based computing model for applications where computers share resources via direct exchanges between the participating computers."

Aus den angeführten Definitionen lassen sich 3 wichtige Eigenschaften von Peer-to-Peer Strukturen extrahieren [Bar01, Mil01]:

1. **Client- und Serverfunktionalität**: alle Knoten sind gleichberechtigt und gleichwertig
2. **Direkter Austausch zwischen Peers**: es gibt keine zentrale Koordinationsinstanz, die Kommunikation findet direkt zwischen den Knoten statt

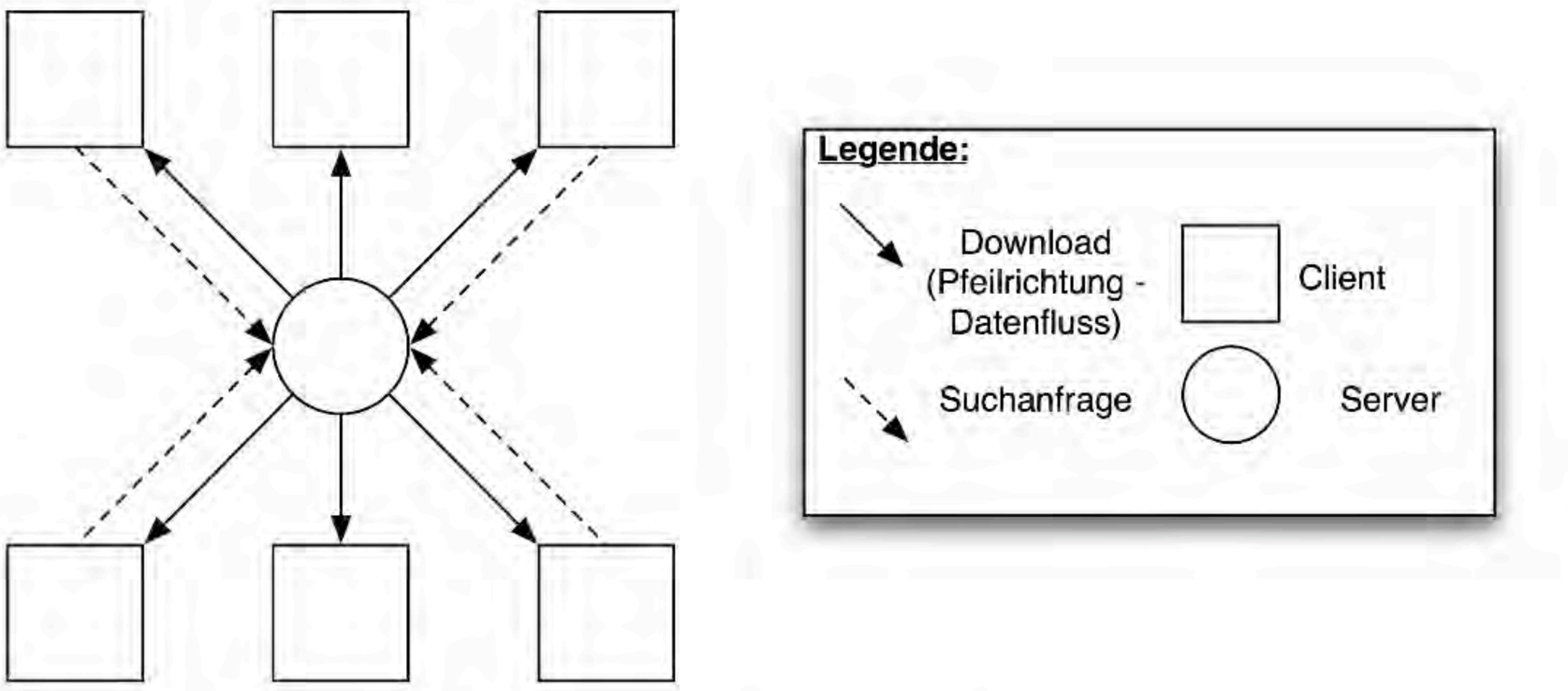


Abbildung 2.1: **Client-Server Struktur** (sinngemäß: [SW04])

3. **Autonomie**: die Knoten sind autonom in Bezug auf die eigene Aktivität (z.B. Ressourcen zur Verfügung zu stellen)

2.2 Peer-to-Peer Strukturen

Peer-to-Peer Strukturen lassen sich nach ihrer Organisation aufteilen in *hybride Netze*, welche zentrale Knoten bzw. Server für Anfragen verwenden und *serverlose Peer-to-Peer Netze*, welche Anfragen mit Hilfe von Routing-Algorithmen weiterleiten [HAS02].

2.2.1 Hybride Netze

Hybride Peer-to-Peer Netze werden in der Literatur manchmal auch als *Serverbasierte Peer-to-Peer Netze* bezeichnet [HAS02]: Dedizierte Server oder ausgezeichnete Peers verwalten eine Datenbank mit Metainformationen über die auf den einzelnen Rechnern verfügbaren Daten oder Dienste. Anfragen werden an die Server gestellt, die Dienste anschliessend in direkter Kommunikation zwischen den einzelnen Peers genutzt. Der Ablauf ist schematisch in Abb. 2.2 dargestellt.

Theodore Hong [Hon01] teilt Serverbasierte Peer-to-Peer Netze in *zentral koordinierte Netze*, welche dedizierte Server verwenden, und *hierarchische Peer-to-Peer Netze*, bei denen ausgezeichnete Knoten mit bestimmten Koordinationsaufgaben versehen sind, ein.

Beispiele für Serverbasierte Strukturen sind Napster [www-1] und SETI@home [www-2][1].

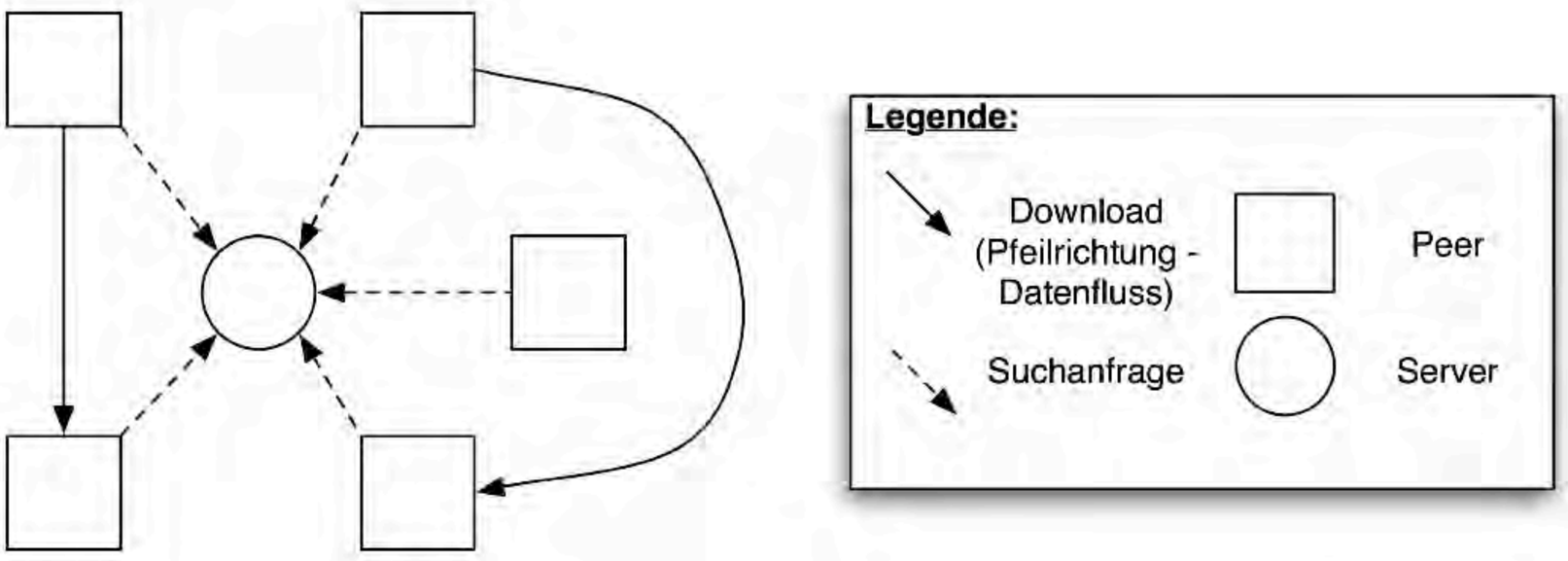


Abbildung 2.2: **Hybride (Serverbasierte) Struktur** (sinngemäß: [SW04])

2.2.2 Serverlose Peer-to-Peer Netze

Serverlose Peer-to-Peer Netze sind *dezentral* organisiert, es existiert keine koordinierenden Knoten [HAS02], die Methoden für das Weiterleiten von Suchanfragen und das Aufrechterhalten der Verbindung müssen in jedem Knoten lokal implementiert sein [Hon01]. Die *Kooperation*, ein zentrales Merkmal von Peer-to-Peer Netzen, steht hier, ganz im Gegensatz zur *Koordination* bei Client-Server Systemen, im Vordergrund [SW04].

Suchanfragen werden an die verbundenen Peers weitergeleitet, bis der gesuchte Dienst oder die gesuchten Daten gefunden wurden oder eine andere Abbruchbedingung erfüllt wurde [HAS02]. Eine Abbruchbedingung könnte beispielsweise erfüllt sein, wenn eine festgelegte Suchtiefe (maximale Anzahl

[1] Napster und SETI@home wurden mittlerweile mehrfach weiterentwickelt, gemeint sind hier die ersten Programmversionen, welche einen zentralen Server verwendeten.

der Weiterleitungen) erreicht wurde. Die Kommunikation findet nach einer erfolgreichen Suche direkt zwischen den Knoten statt. In Abbildung 2.3 wird eine Suchanfrage mit anschließendem Dateidownload dargestellt. Knoten A sendet eine Suchanfrage an den ihm bekannten Knoten, welcher diese wieder-um weiterleitet. Knoten B kann die gesuchten Daten zur Verfügung stellen, ein Dateidownload zwischen Knoten A und Knoten B wird initiiert.

Die dezentrale Struktur hat den Nachteil, dass Suchanfragen relativ ineffizient sind und sehr lange dauern können. Die effizienteste Peer-to-Peer Struktur ist, laut [PBE+05], die *hybride Struktur*.

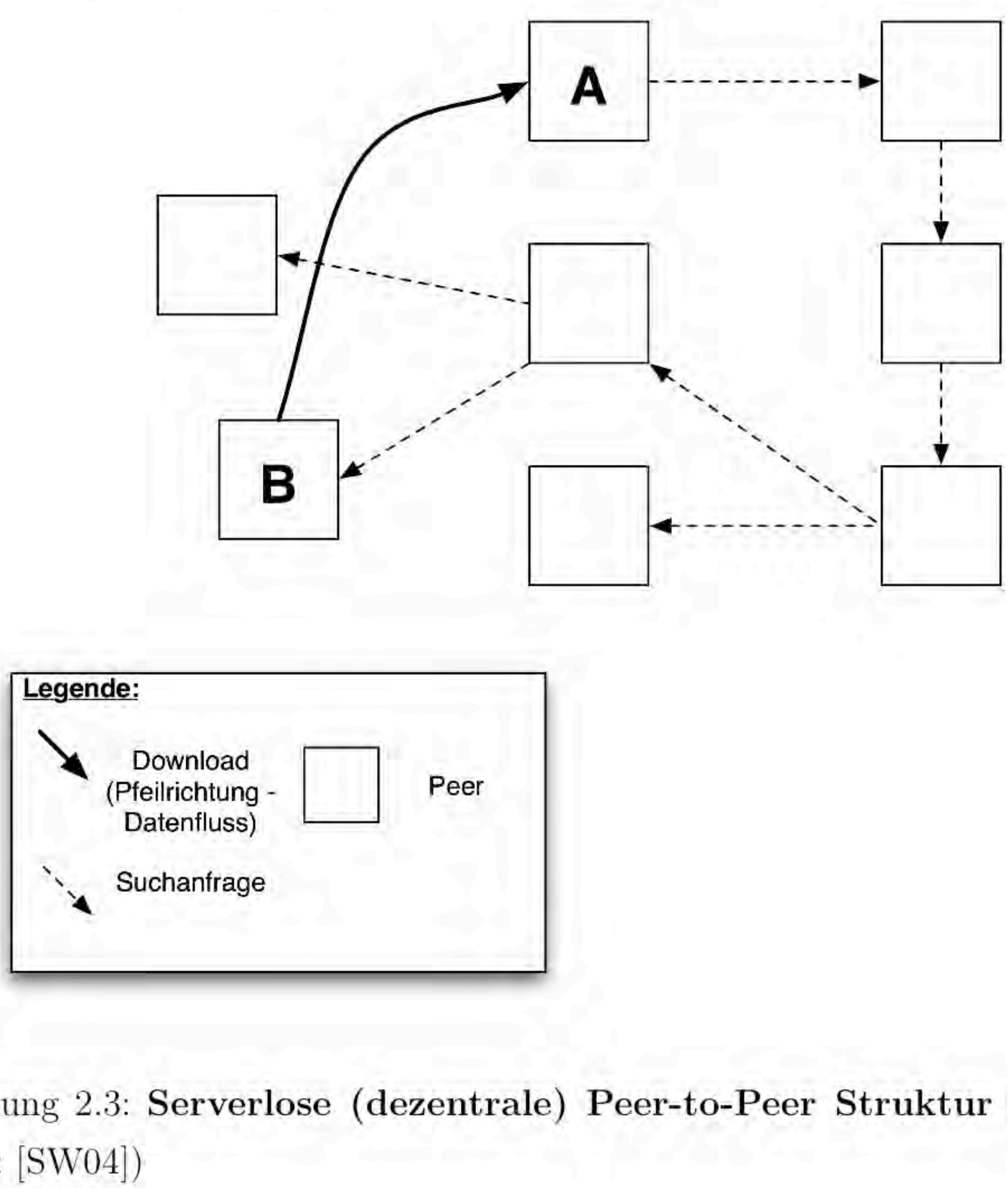


Abbildung 2.3: **Serverlose (dezentrale) Peer-to-Peer Struktur** (sinngemäß: [SW04])

Kapitel 3

Anwendungsbereiche von Peer-to-Peer

Das ursprüngliche Internet, das ARPAnet, war als Peer-to-Peer System aufgebaut, die wenigen damals verbundenen Rechner waren als gleichberechtigte Peers verbunden [Mil01]. Anwendungen wie das Usenet und das Domain Name System (DNS), welche aus dieser Zeit stammen, sind im Grunde Peer-to-Peer Anwendungen [Ora01]. Typische, moderne Anwendungsbereiche von Peer-to-Peer sind [Mil01]:

- Verteiltes Rechnen, Grid Computing
- File Sharing
- Collaboration, Peer-to-Peer Groupware
- Instant Messaging, Internet Telefonie

Der Fokus dieser Diplomarbeit liegt auf den Themen Instant Messaging/Internet Telefonie und File Sharing sowie deren Sicherheitsaspekte. Nachfolgend sollen die wichtigsten Anwendungsgebiete kurz vorgestellt werden.

3.1 Instant Messaging, Internet Telefonie

Instant Messaging (IM) Anwendungen sind auf den direkten Austausch von Nachrichten zwischen Benutzern ausgerichtet [SF02]. 1996 wurde das Instant Messaging-Programm ICQ [www-3] veröffentlicht, welches als erste kostenlose Windows-Software direkte Kommunikation ohne Chat-Server zwischen Benutzern ermöglichte [PBE+05]. Heute werden IM-Systeme von über 200 Millionen Benutzern weltweit verwendet, was Instant Messaging zur am meisten genutzten Peer-to-Peer Anwendung überhaupt macht [Mil01]. Viele Peer-to-Peer Anwendungen, wie z.B. die meisten File Sharing und Collaboration Applikationen, haben IM-Funktionen heute bereits integriert [SF02].

Der aktuelle Trend geht in Richtung Voice Messaging, wie beispielsweise der enorme Erfolg von Skype [www-4] zeigt. Skype kombiniert standard IM-Funktionen, wie Textnachrichten und Dateitransfers, mit Sprachkommunikation [PBE+05].

3.2 File Sharing

File Sharing Anwendungen dienen der dezentralen Speicherung und Verteilung von Daten. Suchalgorithmen werden verwendet, um aufzuspüren, welcher Benutzer die gesuchten Daten verfügbar hat, um diese anschließend direkt von dessen lokaler Festplatte zu laden [SF02].

Bekannte File Sharing Anwendungen sind Gnutella [www-5], BitTorrent [www-7], Kazaa [www-6] oder auch Freenet [www-8].

Bei *Gnutella* handelt es sich um ein Protokoll für den Datenaustausch in dezentralen Netzwerken [SF02]. Eine Überwachung von Gnutella-basierten Netzen ist schwer möglich, da keine zentrale Instanz existiert – im Gegensatz zur mittlerweile eingestellten Musiktauschbörse *Napster*, ein hybrides Peer-to-Peer System. Die Filesharing Software Kazaa, welche das FastTrack Protokoll verwendet, und BitTorrent werden in dieser Diplomarbeit in den Kapiteln 6.4 und 6.3 noch genauer betrachtet.

Das sich noch in der Betaphase befindliche *Freenet* geht bei der Dezentralisierung noch einen Schritt weiter: Peers wissen nicht, welche Daten verschlüsselt auf der Platte abgelegt werden. Ian Clarke entwickelte Freenet mit dem Ziel, Informationen anonym zur Verfügung stellen und beziehen zu können [SF02].

3.3 Verteiltes Rechnen, Grid Computing

Unter Grid Computing versteht man laut [SF02] „die koordinierte Nutzung geographisch verteilter Rechenressourcen".

Autonome Rechner werden zu einem logischen Rechner zusammengefasst, wodurch eine hohe Rechenkapazität erreicht werden soll [SF02].

Das bekannte SETI@Home-Projekt ist ein Pionier auf dem Gebiet des verteilten Rechnens. Die aufgezeichneten Daten eines Radioteleskops in Puerto Rico werden aufgeteilt und an private Rechner verteilt, auf denen dann zur Leerlaufzeit der CPU Berechnungen durchgeführt werden [SF02].

Laut [SF02] weist SETI@Home trotz der Client-Server Struktur bei der Verteilung der Rechenpakete Charakteristika eines Peer-to-Peer Netzes auf: „Die wesentlichen Dienste und Ressourcen werden von den Peers (Clients) zur Verfügung gestellt . . . ".

Weitere Informationen und Beispiele für solche Projekte findet man auf der BOINC-Homepage unter [www-9].

3.4 Peer-to-Peer Groupware

Peer-to-Peer Groupware, auch als Collaboration Software bezeichnet, kombiniert mehrere verschiedene Peer-to-Peer Technologien, um die Zusammenarbeit mehrerer Benutzer online und in Echtzeit zu ermöglichen [Mil01].

Wichtige Funktionen sind Instant Messaging, Audio/Video Telefonie und Konferenzen oder auch Unterstützung für das gemeinsame und gleichzeitige Bearbeiten von Dokumenten [Mil01].

Der Marktführer auf dem Gebiet der P2P Groupware ist die mittlerweile von Microsoft aufgekaufte Firma Groove Networks [www-10]. Aber auch Microsoft selbst hat mit NetMeeting [www-11] eine Groupware Software im Angebot.

Kapitel 4

Sicherheitsaspekte

Welche Gefahren und Risiken ergeben sich durch die (erlaubte oder unerlaubte) Nutzung von Peer-to-Peer Software in Unternehmen? Auf über 50 % aller Arbeitsplatzrechner in Unternehmen läuft heute ein Instant Messaging Client und 40 % aller Viren und Würmer gelangten 2004 über die Kanäle von Peer-to-Peer Anwendungen in Unternehmensnetze [PBE+05].

Die meisten Unternehmen teilen ihre Dienste nach intern und extern verfügbaren Diensten auf die Firmenserver auf, meist laufen nur wenige Dienste auf einem Server – dies hat den Vorteil, dass im Falle der Korrumpierung eines von extern verfügbaren Servers, welcher einem höheren Risiko ausgesetzt ist, keine internen Dienste betroffen sind. Ein normaler Arbeitsplatzrechner nutzt meist interne und externe Dienste gleichzeitig. Nutzt ein Angreifer eine Schwachstelle in einem extern angebotenen Peer-to-Peer Dienst und kann sich so Zugriff auf einen Arbeitsplatzrechner verschaffen, so werden möglicherweise auch intern angebotene Dienste angreifbar [Dam02].

Dieses Kapitel gibt einen Überblick über die Risiken und Gefahren, in Kapitel 8 werden Lösungen und Lösungsansätze zur Minderung dieser Risiken angeführt.

4.1 Risiko durch die Peer-to-Peer Software selbst

Durch die grosse Zahl der laufenden Applikationen auf einem Arbeitsplatzrechner ist es sehr schwierig bis nahezu unmöglich, diesen immer frei von Sicherheitslücken zu halten [Dam02]. Auch Peer-to-Peer Software weist Sicherheitslücken auf, für die meisten Client-Programme werden regelmässig neue Sicherheitslücken bekannt. Der AOL Instant Messenger beispielsweise hat in der aktuellen Version 5 bisher 27 bekannte Sicherheitslücken. Unter [www-12] kann in einer Datenbank nach bekannten Sicherheitslücken für jegliche Art von Software gesucht werden.

Oftmals wird beim Installieren von Peer-to-Peer Software Malware oder Spyware mitinstalliert. Grokster, ein mittlerweile nach einem Rechtsstreit nicht mehr verfügbarer Filesharing Client, lieferte Werbesoftware von Drittfirmen mit, welche sich nach der Installation des Grokster Clients plötzlich am Rechner befand. Die Firma ClickTilUWin nutzte diese Möglichkeit und verbreitete so einen Desktop-Link mit Hilfe der Grokster-Software. Nach Aussage der Firma sollte es sich um einen Link auf ein Online-Casino handeln, bei dessen Aufruf wurde jedoch der Trojaner W32.DIDer.Trojan installiert [Cou02].

4.2 Einschleusen schädlicher Dateien

Peer-to-Peer Software, insbesondere Filesharing und Instant-Messaging Anwendungen, bieten neue Verbreitungsmöglichkeiten für Würmer und Viren. Viele Unternehmen untersuchen eingehende E-mails und von Webservern geladene Dateien auf Viren, doch Lösungen für Peer-to-Peer sind, auch aufgrund der Architektur dieser Netze, noch rar und werden selten genutzt [Fra02].

Die Authentizität der transportierten Daten kann in Tauschbörsen nicht überprüft werden, zumeist werden Dateien nur anhand der Dateinamen und

einiger weiterer Attribute (z.B. Dateilänge) identifiziert [Dam02]. Ausserdem weisen Benutzer beim Ausführen von aus Tauschbörsen geladenen Dateien nicht die selbe Sorgfalt auf wie beispielsweise beim Öffnen von E-mail Anhängen. Ein Benutzer könnte der Meinung sein, er würde gerade die aktuellste Antiviren-Software von einem Peer-to-Peer Portal herunterladen. In Wirklichkeit könnte das Installationspaket aber einen Virus enthalten, welcher gleich mitinstalliert wird [PBE+05].

4.3 Überwinden von Sicherheitssystemen an den Unternehmensgrenzen

Um die Gefahr für die, wie eingangs schon erwähnt, relativ schwach geschützten Arbeitsplatzrechner zu verringern, werden diese meist durch eine Firewall vom Internet getrennt. Diese stellt sicher, über welche Protokolle die Rechner mit der Aussenwelt in Verbindung treten dürfen. Zumeist handelt es sich dabei nur um etablierte Protokolle, wie beispielsweise HTTP [Dam02].

Ein Designziel von Peer-to-Peer Software ist es oft, die direkte Kommunikation zwischen Rechnern gerade über diese Unternehmensgrenzen hinweg zu ermöglichen [Hur02].

Durch Peer-to-Peer Software wird also möglich, was eigentlich durch eine Firewall verhindert werden soll – externe Rechner können Verbindungen zu internen Arbeitsplatzrechnern aufbauen. Ausserdem wird die Prüfung der übertragenen Daten durch das „Untertunneln“ der Firewall verhindert. Zusammenfassend lässt sich sagen, dass durch Peer-to-Peer Software die Durchsetzung von Security Policies an Unternehmensgrenzen komplett umgangen werden kann [Dam02].

4.3.1 Konzepte zur Überwindung von Firewalls

Für die Überwindung von Firewalls existieren verschiedene Konzepte, welche von Peer-to-Peer Anwendungen verwendet werden. Die Techniken können wie folgt klassifiziert werden.

Masquerading

Fast alle Peer-to-Peer Anwendungen können für ihre Protokolle HTTP als „Wrapper“ verwenden und so den eigenen Netzwerkverkehr quasi durch einen HTTP-Tunnel leiten [Dam02] – der Peer-to-Peer Verkehr wird als HTTP-Verkehr „verkleidet“.

Gnutella oder auch FastTrack verwenden für Dateitransfers sogar standardmässig das HTTP-Protokoll.

Port Hopping

Die ersten verfügbaren Peer-to-Peer Anwendungen verwendeten einen fixen TCP-Port, um Dateitransfers zu initiieren (z.B.: TCP-Port 1214 – Kazaa, TCP-Ports 6346/6347 – Gnutella). Das Blockieren dieser Ports reichte aus, um die Verwendung der Filesharing-Anwendungen im Netzwerk zu verhindern.

Aktuelle Filesharing-Anwendungen (z.B. Kazaa in Version 2) können *jeden beliebigen TCP-Port* für einen Dateitransfer verwenden (beispielsweise Port 80). Wird eine Verbindung auf einen bestimmten Port von der Firewall verhindert, so versucht der Client, die Verbindung über andere Ports herzustellen. Skype versucht beispielsweise zuerst, sich über einen zufälligen Port zu verbinden. Schlägt dies fehl, wird zunächst Port 80 (HTTP) und danach Port 443 (HTTPS) versucht.

Verschlüsselung

Durch die Verschlüsselung der übertragenen Daten macht es sehr schwierig, Pakete einem gewissen Protokoll zuzuordnen. Die Möglichkeit, nach be-

stimmten Klartext-Strings zu suchen welche das Protokoll identifizieren, besteht nicht mehr. Verschlüsselung in Kombination mit Masquerading (z.B. auf den TCP-Port 443, über welchen normalerweise HTTPS-Verkehr läuft) macht das Herausfiltern des Peer-to-Peer Verkehrs noch schwieriger.

Push-Transfers

Die Push-Technik kann dann zum Einsatz kommen, wenn sich nur einer der beiden Kommunikationspartner hinter einer Firewall befindet. Der andere Rechner kann zu dem Geschützten keine Verbindung aufbauen, also sendet er über das Peer-to-Peer Netz eine sog. Push-Nachricht an den anderen Client.

Dieser erhält diese Nachricht, sofern er zum Peer-to-Peer Netz eine bestehende Verbindung aufrecht erhält.Mit der Information aus dem Push-Paket kann dieser Rechner nun seinerseits die Verbindung aufbauen und der ursprünglich angestrebte Datenaustausch kann beginnen (siehe Abb. 4.1).

Verwenden von Proxy-Servern

Befinden sich zwei kommunizierende Peers hinter verschiedenen Firewalls, so wäre ein Verbindungsaufbau eigentlich nicht möglich. Um dieses sogenannte „Double Firewall Problem“ zu lösen, werden Relay-Server oder Peers, die nicht durch eine Firewall geschützt sind, für die Weiterleitung der Daten (im Prinzip wie ein Proxy-Server) verwendet (siehe Abb. 4.2) [Dam02].

4.4 Unerwünschte Erhöhung des übertragenen Datenvolumens

Mit der Verwendung von Peer-to-Peer Anwendungen in einem Netz geht meist auch eine Erhöhung des übertragenen Datenvolumens einher. Durch die zuvor erwähnte Überwindung der Sicherheitssysteme an den Unternehmensgrenzen lässt sich dies auch kaum einschränken und kontrollieren. Nach

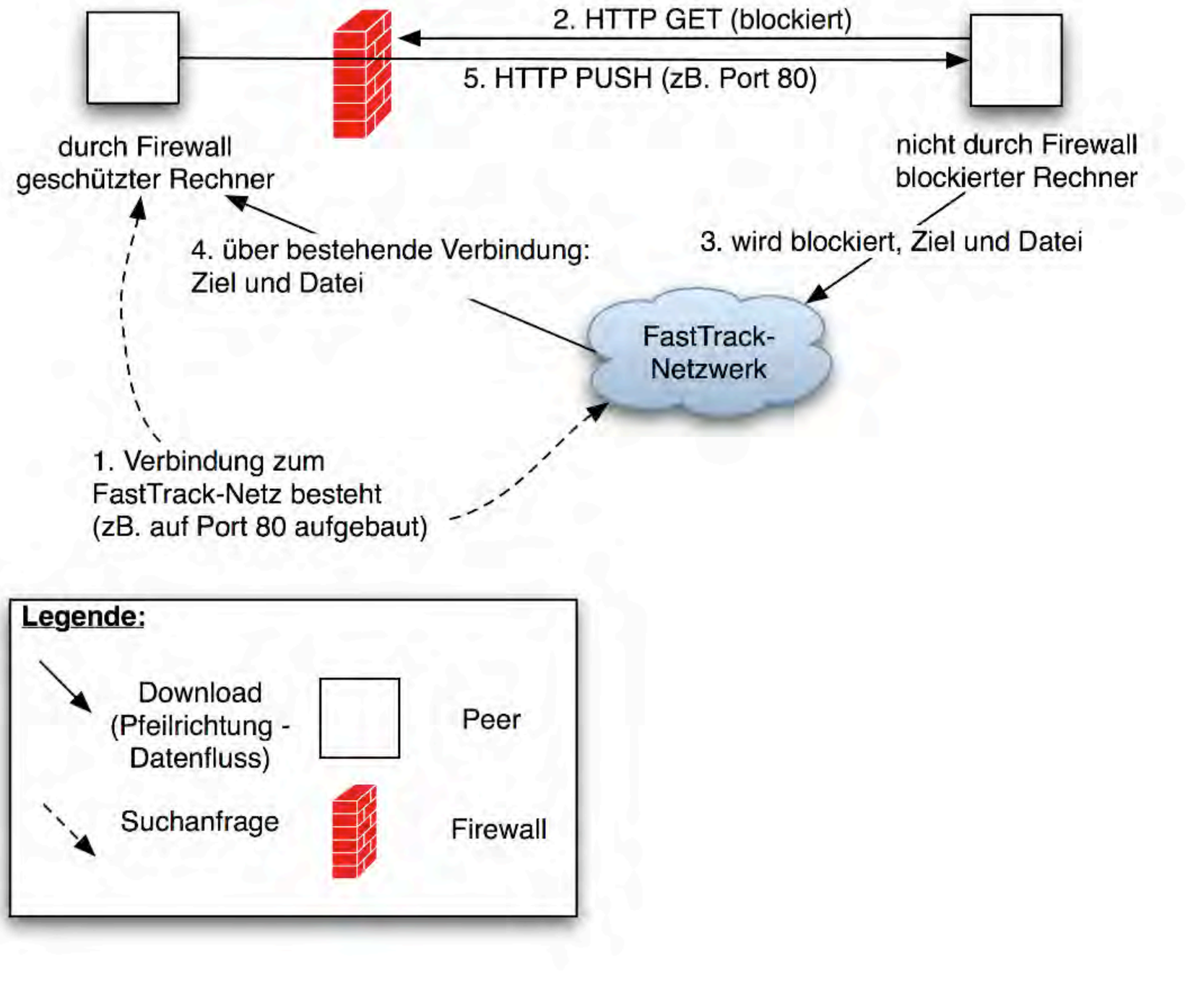


Abbildung 4.1: **Push-Technik bei Kazaa ab Version 2** (sinngemäß: [JK03])

Berichten der Universität von Florida [Gas03] betrug im Jahr 2003 der Anteil von Peer-to-Peer am gesamten Netzwerkverkehr 90 %.

Zusätzlich kann durch die hohe Anzahl der Suchanfragen in einem Peer-to-Peer Filesharing Netzwerk ein einzelner Peer relativ stark belastet werden – und somit auch sein Netzwerk [Dam02]. Aufgrund dieser Tatsache und die Sättigung der Bandbreite werden Unternehmensressourcen belegt, welche für andere, für den Unternehmenserfolg kritische Anwendungen dringend benötigt werden.

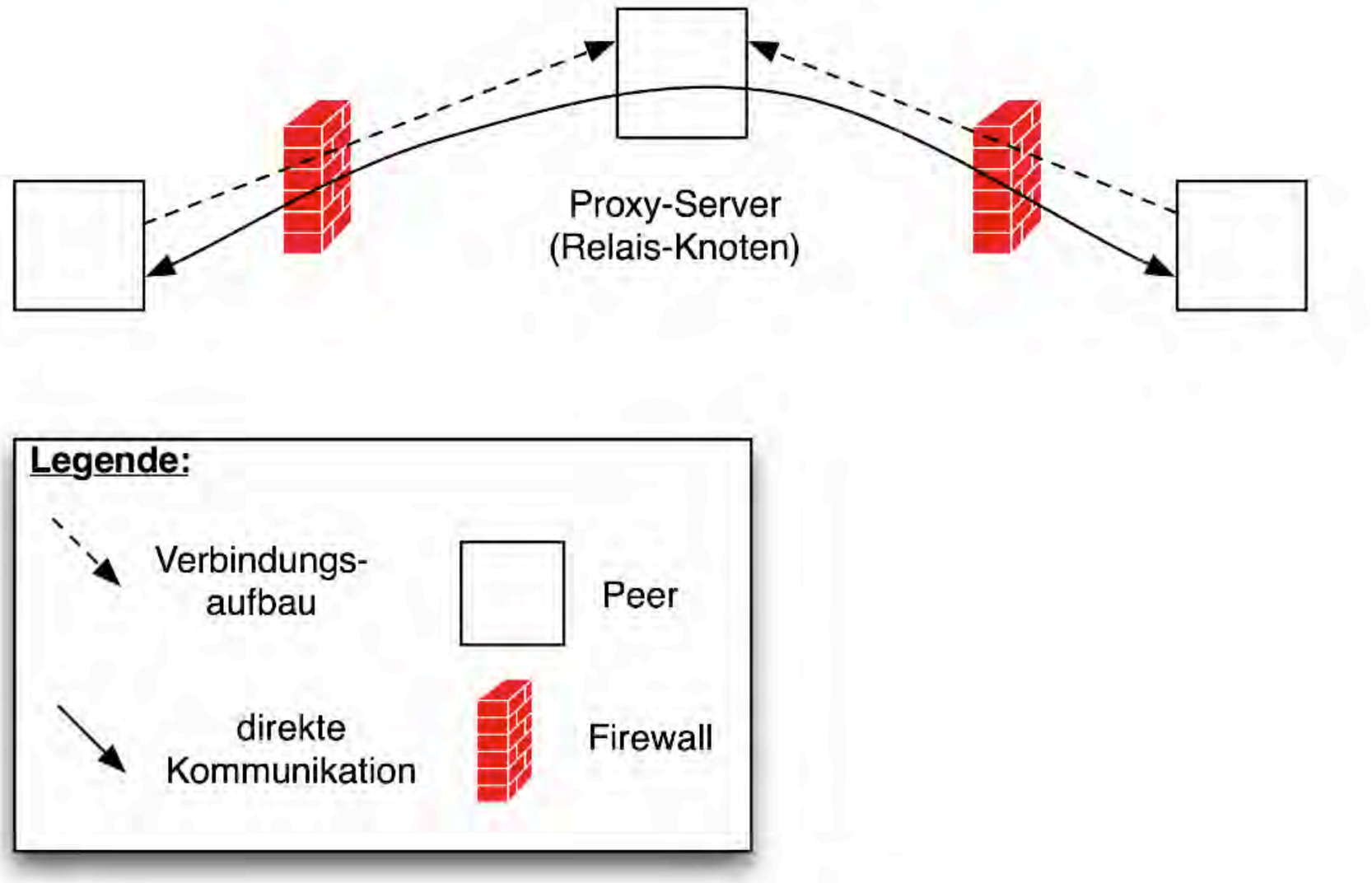


Abbildung 4.2: **Überwindung des Double-Firewall Problems**

4.5 Unbeabsichtigtes Freigeben von Verzeichnissen

Theoretisch hat der Benutzer Kontrolle über die Verzeichnisse, welche er in einer Filesharing-Applikation oder einen Instant Messenger für externe Rechner freigibt. Oft werden aber auch unabsichtlich Verzeichnisse freigegeben, welche eigentlich nicht für den Rest der Welt bestimmt sind. Dazu trägt einerseits das manchmal verwirrende oder einfach nur unterschiedliche Benutzerinterface der unzähligen Clients bei, andererseits werden in den Standardeinstellungen schon Verzeichnisse freigegeben, ohne dass dies explizit angegeben werden muss [Cou02]. Aus Unternehmenssicht sollten Datenfreigaben durch Benutzer auf Arbeitsplatzrechnern überhaupt nicht möglich sein.

Das Verhalten mancher Peer-to-Peer Anwendungen, wie beispielsweise das von LimeWire [www-13], trägt ebenfalls zur Verwirrung der Benutzer bei [Cou02]: Wird diese Software beendet, so bleibt sie solange offen, bis alle Uploads und Downloads beendet wurden – dieses Verhalten ist sogar in den Frequently Asked Questions von Limewire dokumentiert[?] und kann auch abgeschalten werden. Es bleibt jedoch zu befürchten, dass viele Benutzer dieses Problem gar nicht wahrnehmen und somit Netzwerkverkehr verursachen, obwohl sie dies gar nicht beabsichtigen.

Aber nicht nur durch die Anwender oder die Software selbst können solche Probleme entstehen. 2002 geriet der Wurm „Win32.Worm.Duload.A/B“ in Umlauf. Dieser Wurm verursachte die Freigabe verschiedener Verzeichnisse eines Rechners für die Tauschbörse KaZaA [www-6] [Cou02].

Werden durch Fehlkonfiguration, Würmer oder Trojaner Verzeichnisse auf einem Firmen-PC für externe Rechner freigegeben, so kann die Vertraulichkeit der Daten nicht mehr gewährleistet werden. Ein Nachweis im Schadensfall ist schwierig bis unmöglich [Dam02].

4.6 Datenschutz

Das Verwenden von Instant Messaging Anwendungen am Arbeitsplatz führt dazu, dass persönliche Informationen nach außen preisgegeben werden (beispielsweise die Firmen e-mail Adresse), welche der Arbeitgeber vielleicht gar nicht öffentlich machen möchte [Fra02].

Ein weiteres Risiko ergibt sich aus der Natur von Peer-to-Peer Netzen: die Internet Protocol Adresse ist den Kommunikationspartnern bekannt [Cou02]. Ein potentieller Angreifer hat es so einfacher, seine Attacken zu planen [Fra02].

Wie zuvor schon erwähnt, ist das Risiko, vertrauliche, unternehmensinterne Daten zu verlieren bzw. preiszugeben, kein geringes. An einen Kollegen versendete, firmeninterne Nachrichten, werden durch die meisten Instant Messaging Programme zwei Mal unverschlüsselt durch die Firewall an

den Dienstanbieter geschickt [Fra02]. Das Problem hierbei ist, dass viele Instant Messaging Anwendungen die Kommunikation zweier Client-Rechner über einen Server abwickeln. Verschlüsselung ist bei kaum einem Instant Messaging Client implementiert [PBE+05] – somit wird aus einer, als firmenintern betrachteten Unterhaltung zwischen zwei Angestellten, unbeabsichtigt ein abhörbares Gespräch im Internet.

Viele IM-Anwendungen speichern den Verlauf aller Gespräche als Textdatei auf dem Rechner – einem erfolgreichen Angreifer, einem neugierigen Kollegen oder auch dem Arbeitgeber offenbart sich die Gesamtheit aller über die Peer-to-Peer Software geführten Gespräche [PBE+05].

4.7 Diebstahl der Identität, Social Engineering

Benutzernamen und Passwörter von Instant Messaging Diensten können auf der lokalen Arbeitsstation entschlüsselt werden oder bei der Anmeldung am Server mitgehört werden [PBE+05]. Hat ein Angreifer diese Informationen gesammelt, so kann er von jedem beliebigen Rechner aus die Identität des Benutzers (in IM-Netzen der sog. *screen name*) annehmen. In sogenannten *buddy lists* werden die Kontakte eines Benutzers gespeichert – ein Benutzer vertraut darauf, dass die Nachrichten von anderen Benutzern auf seiner Kontaktliste auch von diesen persönlich stammen [Dam02].

Mit etwas Geschick kann sich der Angreifer nun als eine andere Person ausgeben und somit möglicherweise an vertrauliche Daten gelangen. Für weiterführende Informationen zum sogenannten *Social Engineering* sei an dieser Stelle auf das Buch des Ex-Hackers Kevin Mitnick, „The Art of Deception“ verwiesen [MS02].

4.8 Rechtliche Risiken

Peer-to-Peer Tauschbörsen dienen dem Tausch von Dateien, eine Kontrolle der Urheberrechte einer Datei gibt es nicht. Der überwiegende Teil der in den Tauschbörsen verfügbaren Dateien ist jedoch mit Urheberrechten behaftet, wodurch sich rechtliche Probleme ergeben.

Es gibt eine Reihe von Prozessen, welche gegen Peer-to-Peer Tauschbörsen geführt wurden. An dieser Stelle seien der Prozess der RIAA (Recording Industry Association of America) gegen Napster Inc. [www-14] und der Prozess der MGM Studios gegen Grokster Ltd. [www-15] erwähnt. Beide Prozesse haben die Tauschbörsen verloren, Napster und Grokster mussten ihre Server schließen [PBE+05].

Kapitel 5

Beispielszenario

Das in diesem Kapitel beschriebene Szenario könnte mit dem Untertitel „Aus dem Leben eines Administrators“ versehen werden. Anhand einer alltäglichen Situation sollen die Auswirkungen von Peer-to-Peer Software auf ein mittelgroßes Firmennetz sowie die möglichen Gegenmaßnahmen eines verantwortlichen Administrators dargestellt werden. Um die Funktion der verwendeten Werkzeuge zu dokumentieren, beziehen sich die Beispielkonfigurationen auf das Peer-to-Peer Netz *Gnutella* (siehe Kapitel 6.1).

Eine allgemeinere Beschreibung der möglichen Vorgehensweise und der verfügbaren Werkzeuge findet sich in Kapitel 8 ab Seite 74.

5.1 Aufbau des Netzes

Das in diesem Szenario verwendete fiktive Firmennetz beinhaltet ungefähr 200 Client-Rechner, welche durch eine Firewall mit dem Internet verbunden sind. Die Firewall schützt vor Zugriffen aus dem Internet. Außerdem wird ein HTTP-Proxyserver verwendet, um den Zugriff auf Webseiten zu beschleunigen und Bandbreite zu sparen (siehe Abb. 5.1).

Als Firewall verwendet die Netzwerkadministration die Linux-Distribution *Gibraltar* [www-16]. Die Sicherheitspolitik sieht vor, dass Zugriffe vom internen Netz in das Internet freizuschalten sind, während ein Verbindungsaufbau

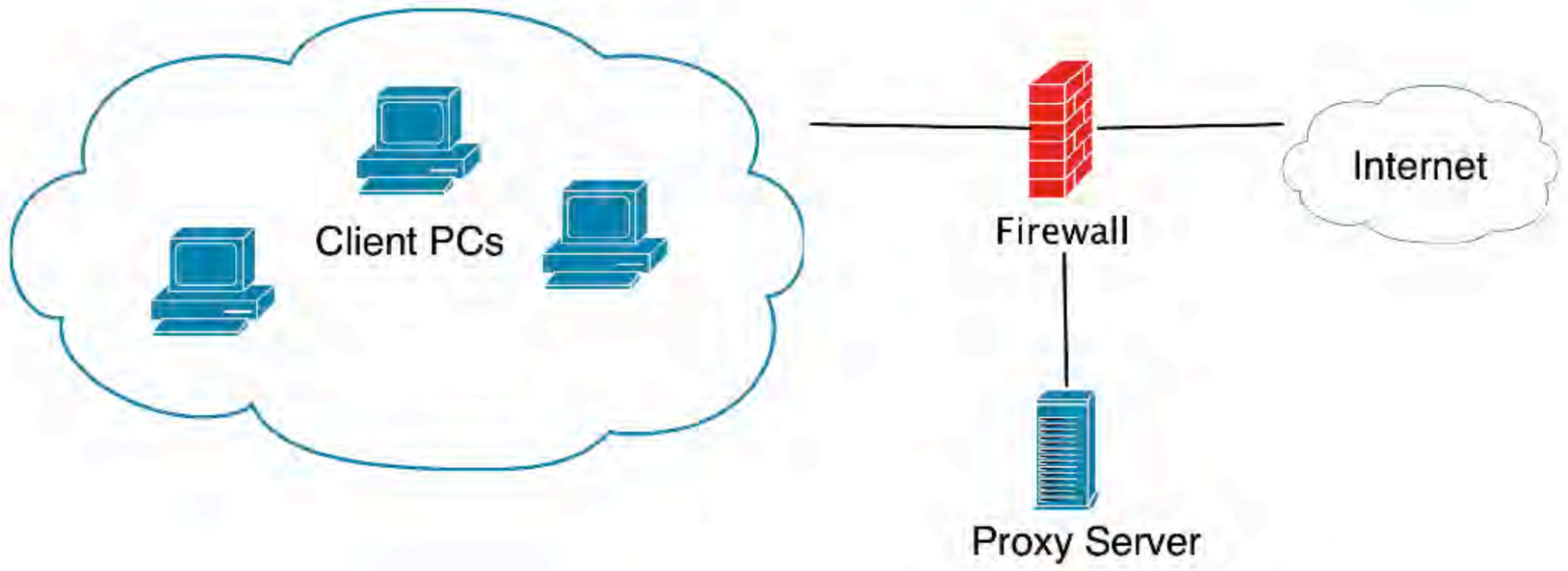


Abbildung 5.1: **Netzwerk des Beispielszenarios**

von extern blockiert wird. Der Aufbau des Netzes wurde bewusst möglichst einfach gehalten, interne Server und Server in der DMZ[1] wurden, bis auf den Proxy, nicht berücksichtigt.

5.2 Das erste Auftreten von Peer-to-Peer

Eines Tages häufen sich in diesem fiktiven Szenario die Benutzer-Beschwerden über lange Antwortzeiten und langsame Downloads aus dem Internet. Da der Netzwerkadministrator die Anbindung an das Internet als ausreichend dimensioniert betrachtet, vermutet er zuerst Probleme beim Provider oder das Umgehen des Proxy-Servers und untersucht mit Hilfe von Netzwerk-Monitoring Werkzeugen, wie *Multi Router Traffic Grapher (MRTG)*[2], die Auslastung der Internet-Anbindung. Er kommt zu dem Schluss, dass die Leitung tatsächlich vollständig ausgelastet ist, allerdings nicht nur zu den Geschäftszeiten, sondern auch nach Dienstschluss und sogar nachts lässt sich erheblicher Netzverkehr feststellen.

[1] DMZ . . . Demilitarized Zone

[2] Nähere Informationen zu MRTG sowie die Software selbst können unter [www-17] bezogen werden

5.2.1 Die Suche nach der Ursache - Verwenden von Intrusion Detection

Besonders seltsam erscheint, dass nicht nur sehr große Datenmengen heruntergeladen werden, sondern dass auch der Upload ins Internet ständig Maximalwerte erreicht. Der Administrator kommt zu der Vermutung, dass es sich hierbei um Peer-to-Peer Netzverkehr handeln könnte und installiert das Intrusion Detection System Snort (Snort wird in Kapitel 8.4.3 in Bezug auf Peer-to-Peer nochmals genauer behandelt). Aktiviert werden die Peer-to-Peer Regeln von [www-18], da diese um einiges umfangreicher sind als das standardmäßig mit Snort mitgelieferte Regelwerk. Als Beispiel sei an dieser Stelle eine Snort-Regel zur Erkennung von Gnutella (die Beschreibung zum Gnutella-Protokoll findet sich in Kapitel 6.1) angeführt:

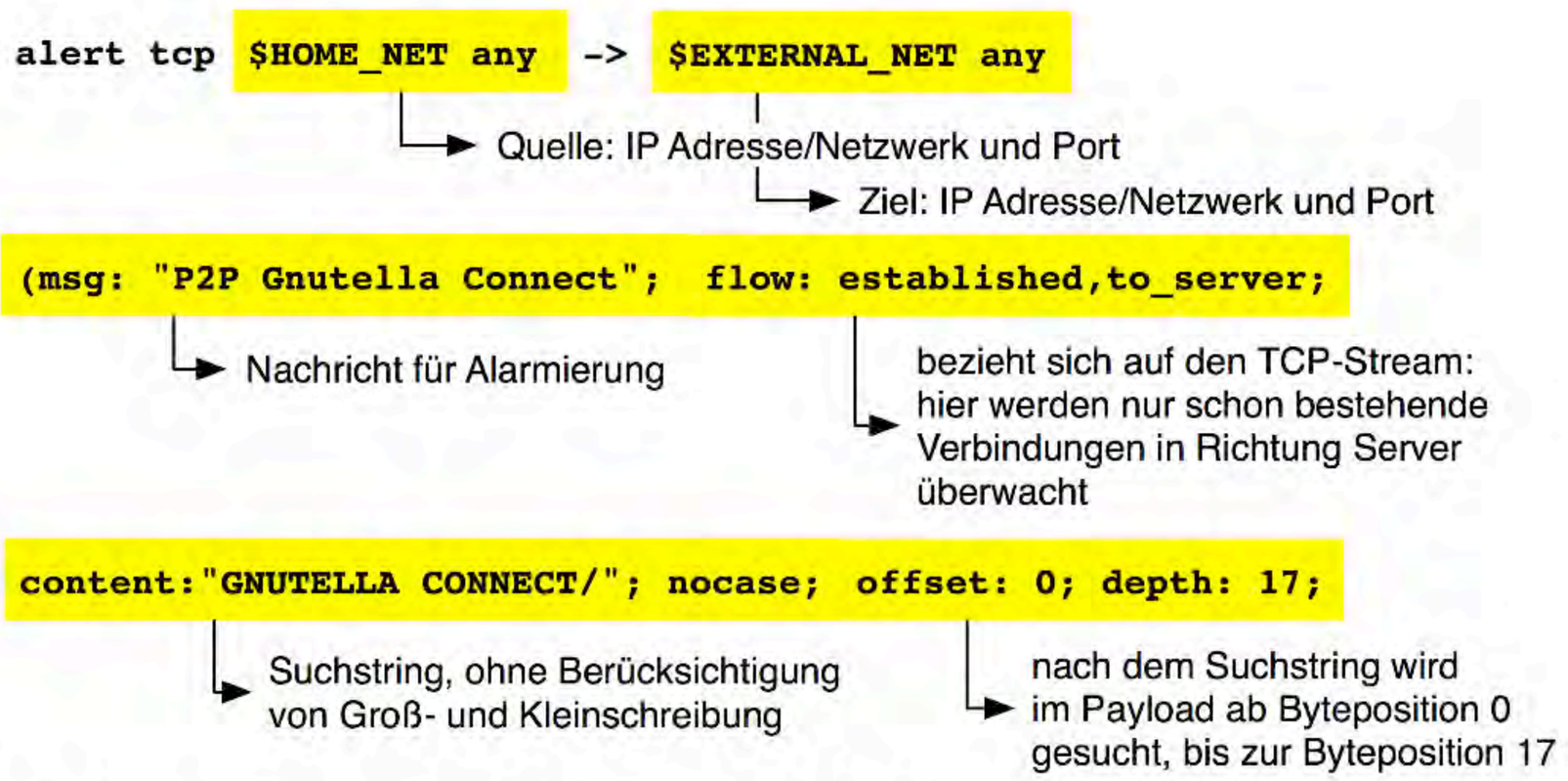


Abbildung 5.2: **Snort Beispielregel: Gnutella**

Tatsächlich kann der Administrator mit Hilfe von Snort die Verwendung von Peer-to-Peer Tauschbörsen-Programmen nachweisen.

Sperren von Standardports

Das Problem scheint zunächst einfach lösbar: der Administrator sperrt die Standardports der verwendeten Peer-to-Peer Programme (ausgehend). Bei Gnutella sind dies beispielsweise die Ports 6346 und 6347 (TCP und UDP). Eine Portliste der gängigsten Peer-to-Peer Programme findet sich in Anhang A.1 ab Seite 102.

Tatsächlich sinkt der Netzverkehr wieder auf ein normales Maß, das Intrusion Detection Sytem zeigt keine Gnutella-Verbindungen mehr an. Das Problem ist gelöst - jedoch nur temporär.

5.3 Umgehen der Standardports

Einige Tage nach dem Sperren der Filesharing-Ports schlägt das Intrusion Detection System wieder an. Diesmal werden die Verbindungen nicht mehr über die Standardports aufgebaut, sondern über den nach außen freigegebenen Port 80. Da dieser Port für die Verbindung zu externen Webservern dringend benötigt wird, lässt sich das Problem dieses Mal nicht einfach durch das Sperren des Ports lösen. Die Techniken, welche Peer-to-Peer Programme verwenden, um Firewalls zu umgehen, wurden in Kapitel 4.3.1 ab Seite 23 beschrieben und kategorisiert.

5.3.1 Peer-to-Peer in der Sicherheitspolitik

Da sich das Problem um die Tauschbörsen mittlerweile zu einer erheblichen Verschwendung von Zeit und Ressourcen entwickelt hat, wird die Sicherheitspolitik (*Security Policy*) des Unternehmens in Bezug auf Peer-to-Peer aktualisiert. Im *Acceptable Use* Teil der Policy wird die Verwendung von Peer-to-Peer Tauschbörsen Software explizit untersagt.

Durch das Einbeziehen in die Sicherheitspolitik hat der Administrator nun die Möglichkeit, Benutzer von Peer-to-Peer Software abzumahnen. Außerdem

geht die Änderung der Policy per E-mail an alle Angestellten des Unternehmens und soll so das Bewusstsein in Bezug auf diese Software schärfen.

5.3.2 Content Inspection

Um dem Problem Herr zu werden, sieht der Administrator keine andere Möglichkeit, als Content Inspection (Analyse der Anwendungsschicht, Layer 7 des ISO/OSI Referenzmodells) auf der Firewall zu verwenden. Bei dieser Technik wird der Payload der übertragenen Datenpakete auf Peer-to-Peer Protokolle untersucht. Glücklicherweise bietet die eingesetzte Firewall die Möglichkeit, Content Inspection auf Basis des iptables-Moduls [www-19] *Ipp2p* [www-20] für Peer-to-Peer Tauschbörsen durchzuführen. In der Weboberfläche der Firewall lässt sich die Funktion relativ einfach aktivieren (siehe Abb. 5.3). Hier wurde die Peer-to-Peer Filter-Funktion in einer DROP-Regel aktiviert, es wird also jedes Paket, in welchem Peer-to-Peer Signaturen gefunden werden, sofort verworfen. Dadurch kann eine Peer-to-Peer Verbindung gar nicht erst zustande kommen.

Die anfängliche Befürchtung, die Content Inspection könnte sich auf die Leistung der Firewall stark negativ auswirken, bestätigt sich nicht. IPP2P führt die Content Inspection zuverlässig und relativ schnell durch.

Traffic Shaping

Da die Möglichkeit besteht, dass das IPP2P Modul sog. „falsche Positive“ liefert, also Pakete als Peer-to-Peer klassifiziert, auf welche dies nicht zutrifft, kann es von Vorteil sein, Peer-to-Peer Netzverkehr nicht komplett zu blockieren. Als Alternative bietet sich an, eine (mit sehr wenig Bandbreite versehene) Traffic-Shaping Klasse zu definieren und den Peer-to-Peer Netzverkehr somit quasi lahm zu legen. Ein fälschlich als Peer-to-Peer klassifiziertes Paket kann passieren, für die den Download von Dateien mittels Tauschbörsen-Software ist die Bandbreite jedoch sehr gering und damit für die Benutzer nicht attraktiv.

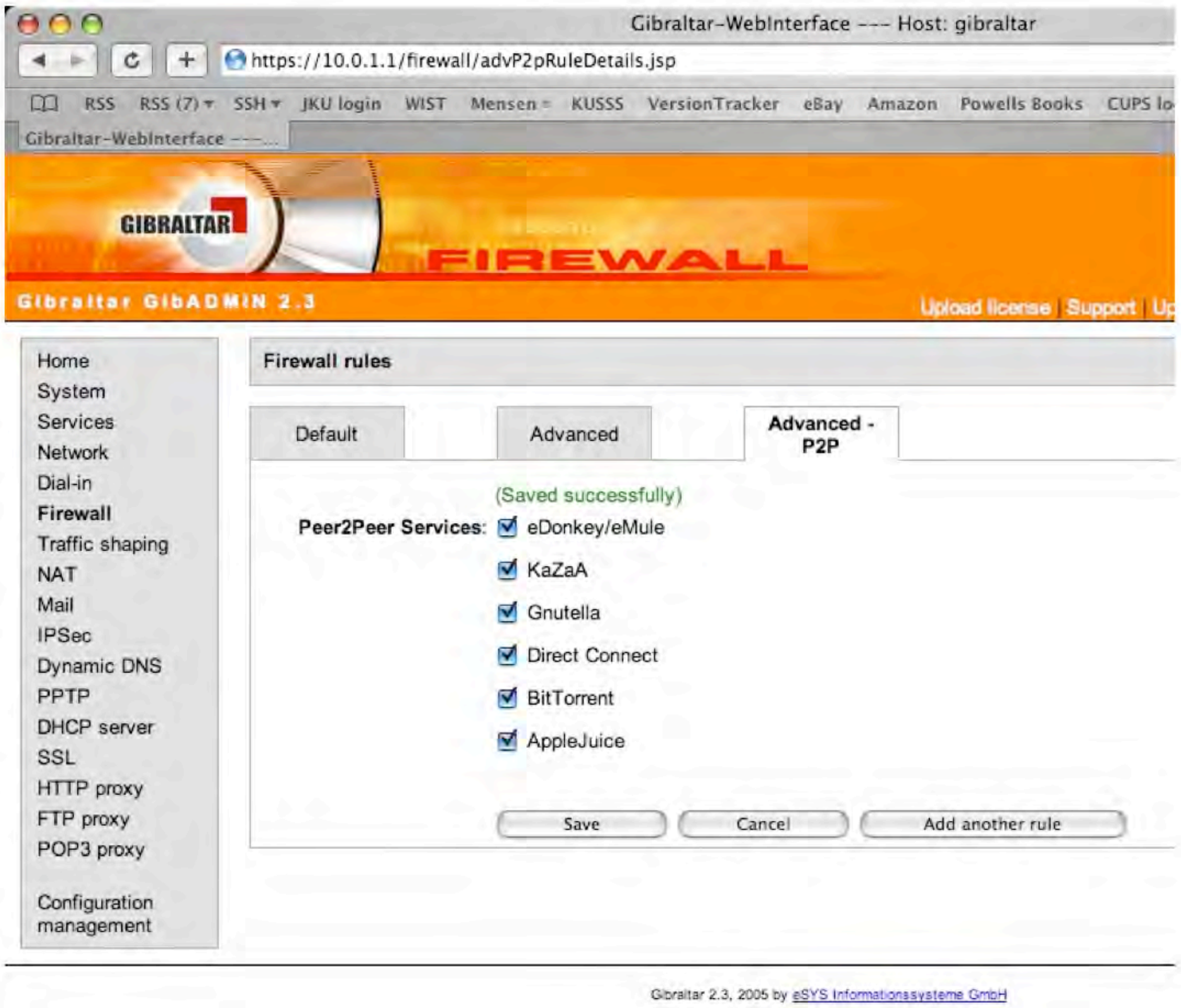


Abbildung 5.3: **IPP2P Modul bei Gibraltar**

Geht der Administrator diesen Kompromiss ein, dann muss er sich auch bewusst sein, dass die Bedrohungen, welche durch Peer-to-Peer für das von ihm betreute Netzwerk entstehen, dadurch nicht abgeschwächt werden können. Der verursachte Netzverkehr durch Peer-to-Peer ist jedoch minimal.

In diesem Beispiel verwendet die Administration wieder die Gibraltar-Firewall. In Abbildung 5.3.2 zeigt die Regel, welche abgehenden Peer-to-Peer Netzverkehr auf 1 KByte/s reduziert.

Soll die Regel für das Traffic Shaping von Peer-to-Peer „von Hand“ eingepflegt werden, so müssen nicht nur Pakete, welche den Suchstring (die Signatur) „GNUTELLA“ enthalten, berücksichtigt werden, sondern alle Pa-

kete, die zu dieser Verbindung gehören. Unter Linux (Netfilter) kann der Administrator hierfür auf das connmark-Modul [www-21] zurückgreifen.

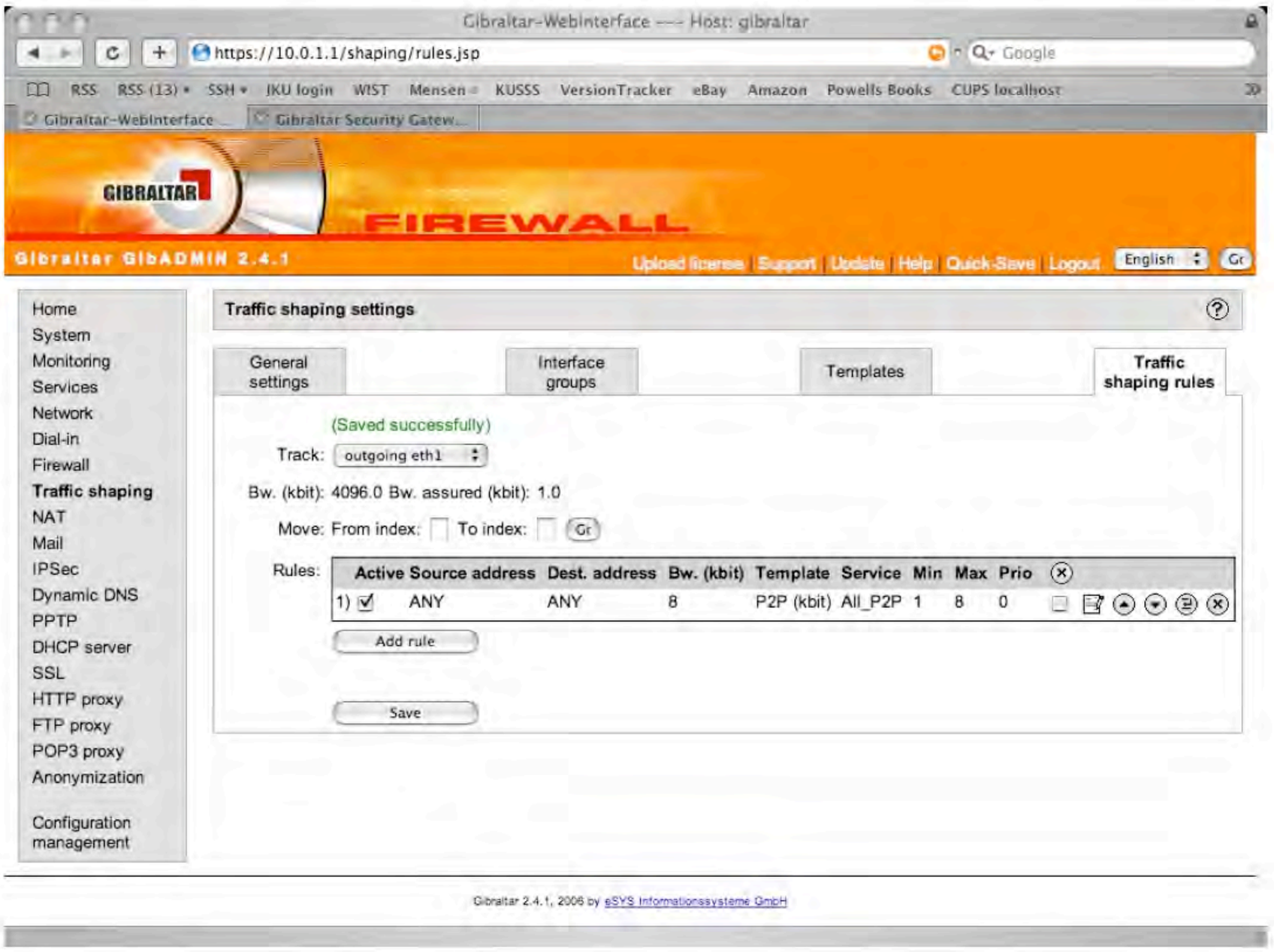


Abbildung 5.4: **Gibraltar Firewall: Traffic Shaping**

5.3.3 Offizieller Einsatz von Peer-to-Peer im Unternehmen

Bei der Diskussion über die Nachteile und Gefahren von Peer-to-Peer Anwendungen darf man nicht vergessen, dass durchaus auch für Unternehmen nützliche Anwendungen, wie beispielsweise Skype, ein Instant Messaging Programm mit Sprachtelefonie-Funktion, existieren. Was bei der Verwendung dieser Software in einem Unternehmensnetz zu beachten ist und wie sich eine solche Lösung gestaltet, wird in Kapitel 8.2.1 (Seite 75) beschrieben.

Kapitel 6

Peer-to-Peer Filesharing Netze und Protokolle

6.1 Gnutella

Das Gnutella-Protokoll wurde von Justin Frankel, welcher damals für AOL tätig war, entwickelt. Im März 2000 wurde die erste Software für die Nutzung des Netzes veröffentlicht. AOL zwang Frankel jedoch dazu, die Software nach kurzer Zeit wieder vom Netz zu nehmen. Zu diesem Zeitpunkt war es aber bereits zu spät, das Programm hatte sich verbreitet und es dauerte nicht lange, bis das Protokoll entschlüsselt wurde und viele weitere Clients für das Gnutella-Netz veröffentlicht wurden [gnu]. Das Gnutella-Netz zählte (mit Stand vom 14. Juni 2006) 2.2 Millionen Benutzer weltweit [www-22].

Gnutella ist ein serverloses Peer-to-Peer System, die Peers werden in der Gnutella-Protokoll Spezifikation [Kir03] als *Servents* bezeichnet. Nach einer Erweiterung des Protokolls werden im Netz ausgezeichnete Knoten, sogenannte *Ultrapeers*, für bestimmte Zwecke verwendet. Es handelt sich bei diesen Knoten zwar nicht um Server, die Netzarchitektur wurde dadurch jedoch *hybrid*. Nach [Hon01] handelt es sich bei Gnutella um ein hierarchisches serverloses Peer-to-Peer Netz.

Gnutella dient hauptsächlich dem Austausch von Dateien, obwohl in der Spezifikation auch von anderen Ressourcen wie kryptographischen Schlüsseln oder Metainformationen die Rede ist. Es wird allerdings nicht darauf eingegangen, wie diese zu handhaben sind.

6.1.1 Die Verbindung zum Gnutella-Netz

Für den Start eines Verbindungsaufbaus zum Gnutella-Netz werden IP Adressen von Rechnern, die im Netz online sind, benötigt. Diese können einerseits lokal am Rechner gespeichert sein oder von sogenannten *Gnutella WebCaches* bezogen werden. Die WebCaches speichern die Adressen von online Hosts in einer Datenbank und geben diese bei Bedarf an die Servents weiter. Adressen, welche dem Servent während der Kommunikation durch Suchanfragen o.ä. zur Kenntnis gelangen, werden lokal gespeichert. Somit wird klar, dass es sich bei Gnutella zwar um ein prinzipiell serverloses Protokoll handelt; um die Verbindung zum Netzwerk zu erleichtern, wird aber trotzdem auf ein *hybrides Modell* zurückgegriffen.

Ist eine IP-Adresse bekannt, so versucht der Servent, die Verbindung aufzubauen. Der genaue Ablauf des **Handshakes** ist im Anhang ab Seite 104 nachzulesen. Die Architektur des Netzes ist in Abb. 6.1 dargestellt.

6.1.2 Der Ultrapeer Modus

Gnutella verwendet eine hierarchische Einteilung der Knoten in *Ultrapeers* und in deren Blätter, die *Leaf Nodes*. Ein Blattknoten kann nur Verbindungen zu Ultrapeers haben. Ein Ultrapeer verhält sich vom Prinzip her wie ein Proxy für das Gnutella Netzwerk [Kir03], er kümmert sich um die Nachrichtenweiterleitung und das Routing (z.B. der Suchanfragen). Normalerweise hält ein Ultrapeer Verbindung zu 16 weiteren Ultrapeers. Das Netz kann dadurch besser skalieren und überflüssiger Netzverkehr wird vermieden.

Peers mit hoher Bandbreite und hoher Uptime können zu Ultrapeers werden. Die Entscheidung, ob ein Knoten im Ultrapeer-Modus oder als nor-

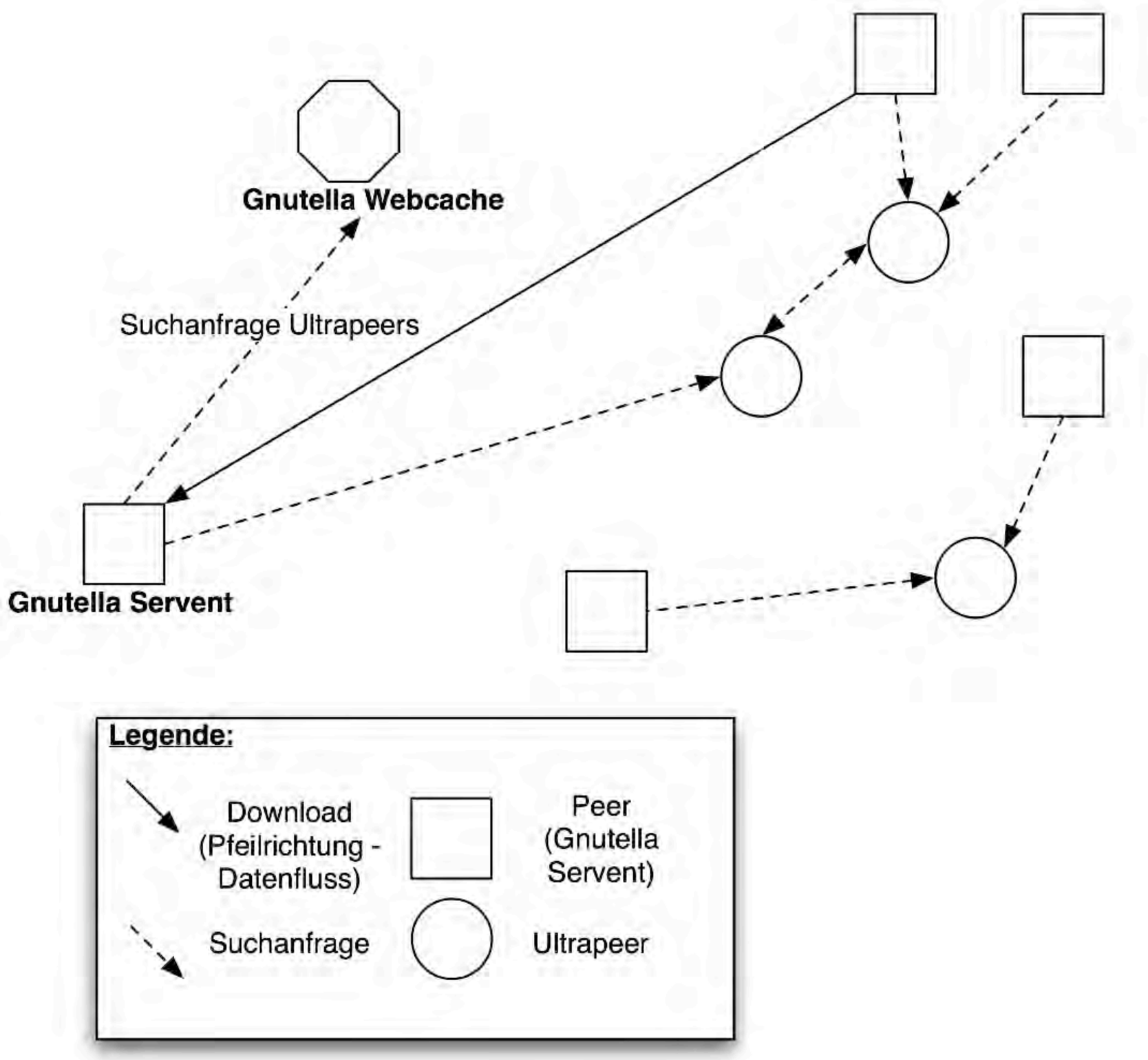


Abbildung 6.1: **Gnutella Netzarchitektur**

males Blatt fungiert, wird beim Handshake, also beim Aufbau einer neuen Gnutella-Verbindung, getroffen (siehe B.1.1). Hierzu sendet der Knoten den `X-Ultrapeer` Header mit dem Wert *true* oder *false*.

Der Verbindungsaufbau wird vereinfacht in einem Ablaufdiagramm in Abb. 6.2 dargestellt. Ist eine Verbindung zu einem Ultrapeer aufgebaut, so wird eine Liste mit den Hash-Werten der am Rechner freigegebenen Dateien an den Ultrapeer gesendet, welche dieser in seiner Datenbank abspeichert.

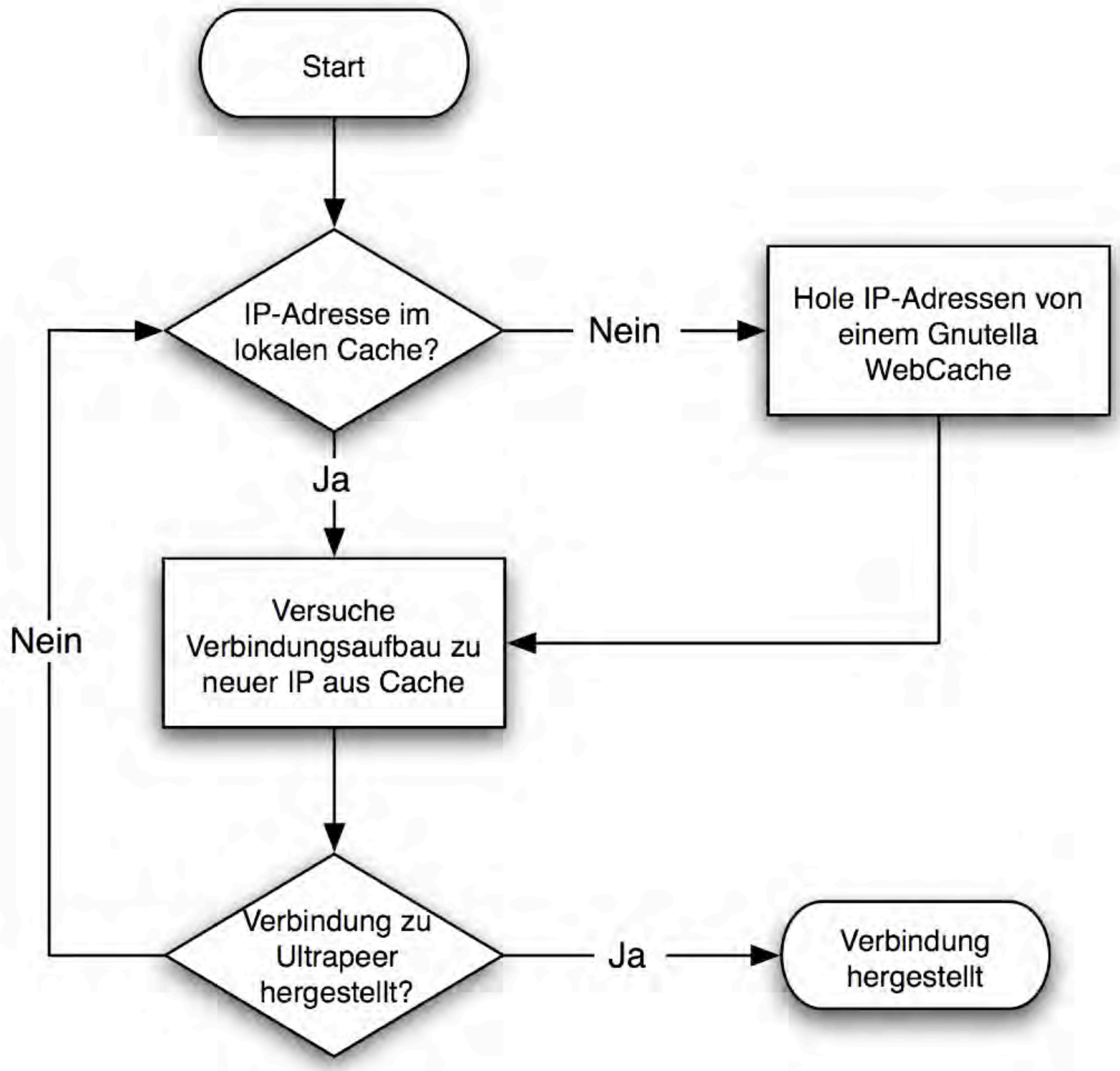


Abbildung 6.2: **Ablaufdiagramm Gnutella Verbindungsaufbau**

6.1.3 Suchanfragen

Um Suchanfragen möglichst effizient zu gestalten, wurde das *Query Routing Protocol* [Kir03] entwickelt. Das Konzept sieht vor, dass die (Hash-)Listen mit den freigegebenen Dateien nicht nur von den Blattknoten zu den Ultrapeers gesendet werden, sondern dass die Ultrapeers diese Listen auch untereinander austauschen.

Eine von einem Blattknoten aus gestartete Suchanfrage wird zunächst an den Ultrapeer weitergegeben. Dieser sucht in seiner Hashtabelle nach möglichen Treffern und sendet die Anfragen an die betroffenen Blattknoten, bzw. an die betreffenden Ultrapeers weiter (siehe Abb. 6.3). So wird ein unnötiger Broadcast von Suchanfragen vermieden, Bandbreite und Ressourcen der Blattknoten werden geschont.

Erhält ein Knoten eine Suchanfrage und hat er einen Treffer zu vermelden, so wird eine positive `Query Hit`-Nachricht direkt an den ursprünglich anfragenden Knoten gesendet [Kir03].

Eine weitere Verbesserung für die Effizienz von Suchanfragen stellt das *Dynamic Querying* dar [Fis06]. Beim Dynamic Querying werden die Suchanfragen vom Client zunächst nur an seinen eigenen Ultrapeer gesendet (die TTL der Suchanfrage ist 1). Liefert der Ultrapeer genügend Treffer zurück (normalerweise wird ein Wert von 250 als ausreichend angesehen), so ist die Suche zu Ende. Ist das Ergebnis nicht zufriedenstellend, so wird die Suche mit einer TTL von 2 fortgesetzt. Das geht so lange, bis genügend Treffer eingelangt sind. Dynamic Querying dauert zwar etwas länger, es werden aber unnötige Suchanfragen vermieden und die Wahrscheinlichkeit, seltene Dateien zu erhalten, wird erhöht [Fis06].

6.1.4 Der Dateitransfer

Das Protokoll für den Dateitransfer ist HTTP (normalerweise in der Version 1.1, es kann aber auch in Version 1.0 verwendet werden).

Der Servent, welcher den Download starten möchte, sendet einen HTTP-GET Request an den Zielserver [Kir03]. Der Servent (in diesem Fall fungiert dieser als Server) antwortet und beginnt mit der Datenübertragung.

Gnutella bietet die Möglichkeit, Teile einer Datei parallel von mehreren Peers gleichzeitig herunterzuladen. Der anfragende Peer hat die Möglichkeit anzugeben, welche Teile er von einer Datei erhalten möchte. Im Gegensatz zu Protokollen wie eMule ist diese Funktion jedoch nicht Teil des Protokolls, die Intelligenz dafür muss im verwendeten Client implementiert werden.

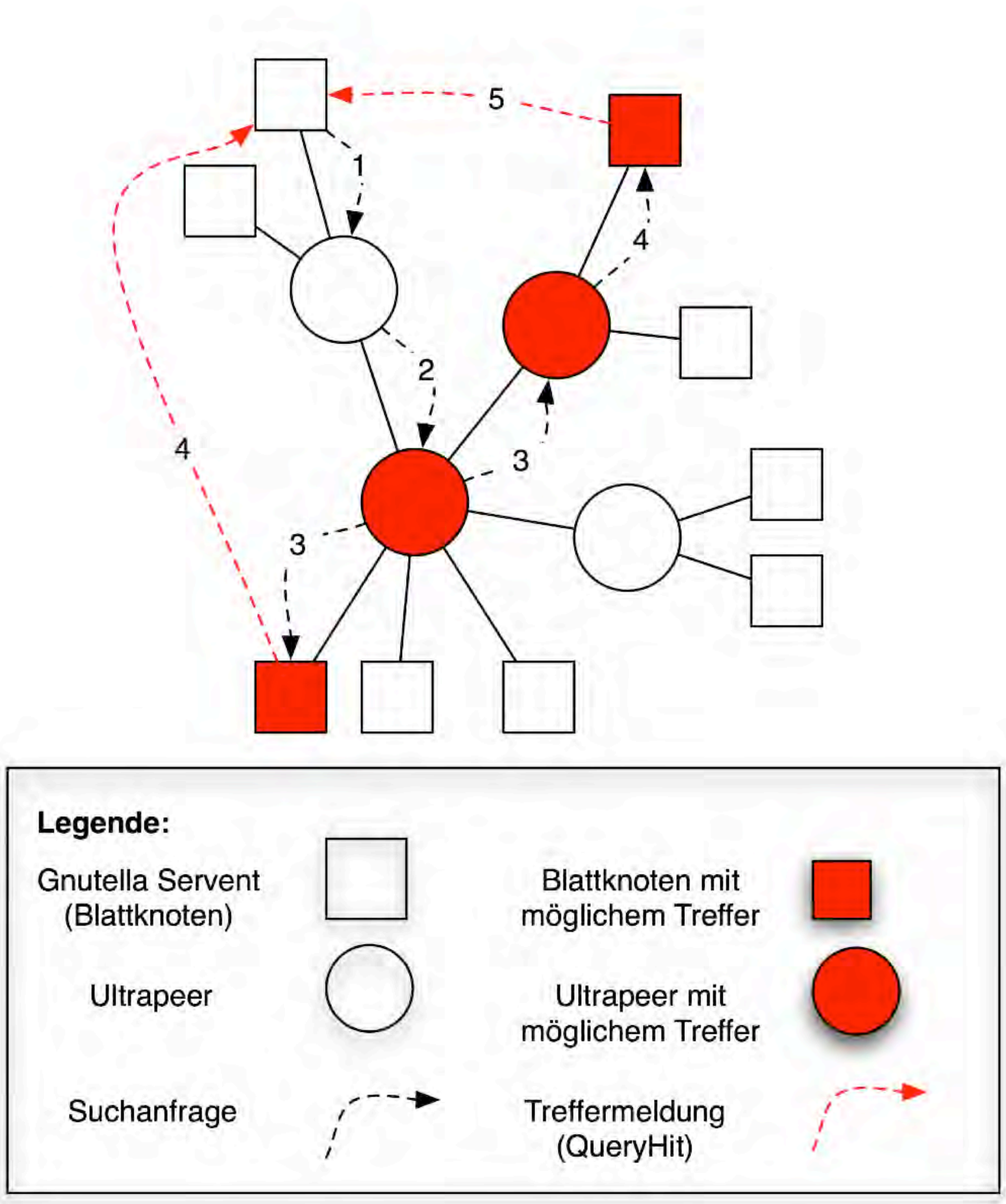


Abbildung 6.3: **Suchanfragen und deren Weiterleitung im Gnutella Netz**

Es ist möglich, dass ein Servent nicht für einen Datentransfer erreichbar ist. Dies kann durch die Verwendung einer Firewall herbeigeführt werden, welche keinen Verbindungsaufbau von außen zulässt. In einem solchen Fall wird über die bereits bestehende Verbindung zum Gnutella-Netz eine

sog. *Push*-Nachricht gesendet. Erhält ein Servent eine solche Nachricht, dann baut er selbsttätig eine TCP-Verbindung zum anfragenden Peer auf und der Dateitransfer kann stattfinden. Sollten beide Peers keine Verbindungen von außen zulassen, so kann der Transfer nicht durchgeführt werden.

Genaueres zu den ausgetauschten Nachrichten findet sich im Anhang B.1 ab Seite 104.

6.2 eDonkey / eMule

Der erste eDonkey-Client wurde im Juni 2000 unter dem Namen *eDonkey 2000* von der Firma MetaMachine veröffentlicht. Der Client war kommerziell und das Protokoll nicht öffentlich. 2002 wurde der OpenSource Client *eMule* veröffentlicht, das Protokoll konnte durch Reverse Engineering nachgebildet werden [PBE+05].

Das eDonkey Netz ist das derzeit größte aktive Peer-to-Peer Netz, 3.8 Millionen Benutzer [www-22] waren im Mai 2006 in der Tauschbörse online. Nach mehreren Polizeiaktionen gegen grosse eDonkey Server (siehe Pressemeldungen [Hei06] und [Kle06]) waren im Juni 2006 nur noch 3.1 Millionen Benutzer in der Tauschbörse online.

eDonkey ist ein hybrides Peer-to-Peer Netz, es basiert auf dedizierten Servern, welche die Dateilisten in einer Datenbank verwalten und die Suche in dieser durchführen [KB05]. [Hon01] würde eDonkey als ein zentral koordiniertes Peer-to-Peer Netz bezeichnen. Neuerdings gibt es Bestrebungen, auf ein serverloses Netz umzustellen (nähere Informationen hierzu gibt es auf der Homepage von Overnet [www-23]).

6.2.1 Kommunikation im eDonkey-Netz [KB05]

Ein eDonkey Client muss die TCP-Verbindung während der gesamten Sitzungsdauer im eDonkey-Netz offenhalten. Mit anderen Servern kann der Client UDP-Nachrichten für die Verbesserung der Datei- und Quellensuche austauschen. Die Server halten untereinander keine Verbindung (siehe Abb. 6.4).

Verbindungsaufbau

1. Verbindungsaufbau zum Server.
2. Der Server sendet eine ClientID an den Client.
3. Der Client sendet die Liste seiner freigegebenen Dateien, welche der Server in der zentralen Datenbank speichert.

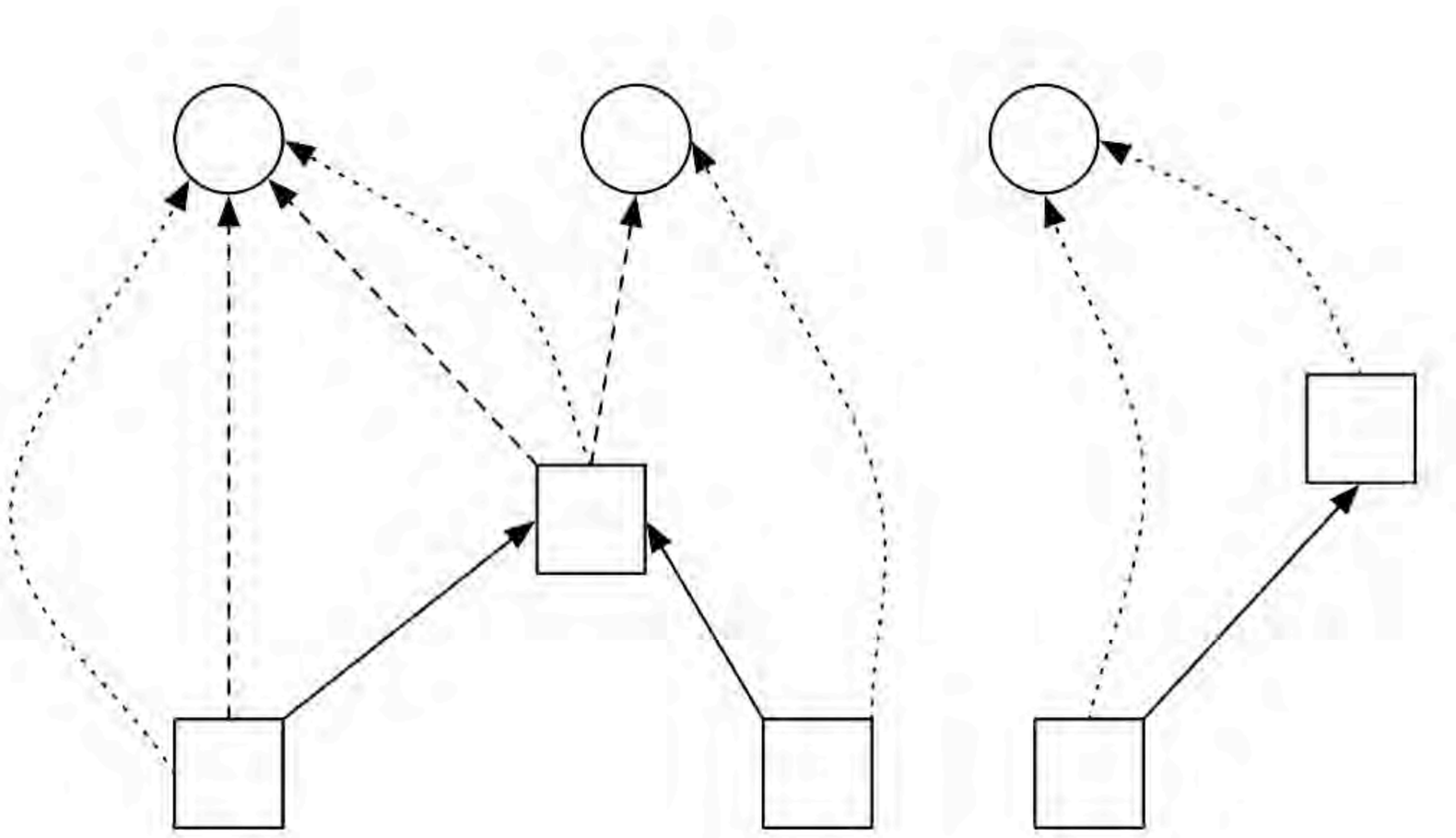


Abbildung 6.4: **Verbindungen im eDonkey/eMule Netzwerk (sinngemäß: [KB05])**

ClientID

Die ClientID ist eine 4 Byte lange ID-Nummer, welche der Server beim Verbindungsaufbau vergibt. Sie ist prinzipiell nur für die Dauer der TCP-Sitzung gültig und dient der Identifizierung des eDonkey-Clients, wobei es zwei Arten gibt [KB05]:

low ID Eine *low ID* wird vom Server vergeben, wenn der Client keine Verbindungen annehmen kann, weil er sich hinter einer Firewall befindet oder die Verbindung durch einen Proxy-Server bzw. mittels NAT hergestellt wird.

Manche Server lehnen Verbindungen mit low ClientIDs ab oder lassen nur eine bestimmte Anzahl zu, da hierbei der Server durch Nachrichtenweiterleitung stärker belastet wird.

high ID *High IDs* werden vergeben, wenn der Client TCP-Verbindungen annehmen kann. Die High ID wird aus der IP-Adresse berechnet.

Die Berechnung der High ID für die IP-Adresse $X.Y.Z.W$ wird wie folgt durchgeführt [KB05]: $HighID = X + 2^8 * Y + 2^{16} * Z + 2^{24} * W$ (Big Endian Repräsentation)

LowIDs sind immer niedriger als 16777216 (0x1000000), es kann nicht nachvollzogen werden, wie diese berechnet werden.

Um trotz einer Low ID, bei der der Client ja keine ankommenden Verbindungen akzeptiert, Dateitransfers durchführen zu können, verwendet eDonkey einen *Callback* Mechanismus. Dieser funktioniert ähnlich wie bei Gnutella, der Client sendet den Callback Request an den Server, dieser leitet ihn an den Client mit der Low ID weiter, welcher daraufhin die Verbindung aufbaut.

Credit System

Um für die Benutzer einen Anreiz zu schaffen, Dateien zum Upload freizugeben, bedient sich eMule eines Credit Systems. Je mehr Dateien ein Clients an andere Clients übertragen hat, desto mehr Credits erhält er. Diese wiederum bewirken ein schnelleres Aufsteigen in den Upload Queues (siehe Dateidownload, nächster Punkt).

Zur Identifikation berechnet der Client einmalig eine 128 Bit lange Zufallszahl, die immer gleich bleibt, durch Verschlüsselung wird das System vor Manipulation geschützt.

Dateidownload

Dateien werden fragmentiert in sog. „parts“, wodurch der Download einer Datei von mehreren Quellen gleichzeitig möglich ist. eDonkey bezeichnet dieses Verfahren als *Multisource File Transmission Protocol (MFTP)*. In Abbildung 6.5 ist MFTP schematisch dargestellt.

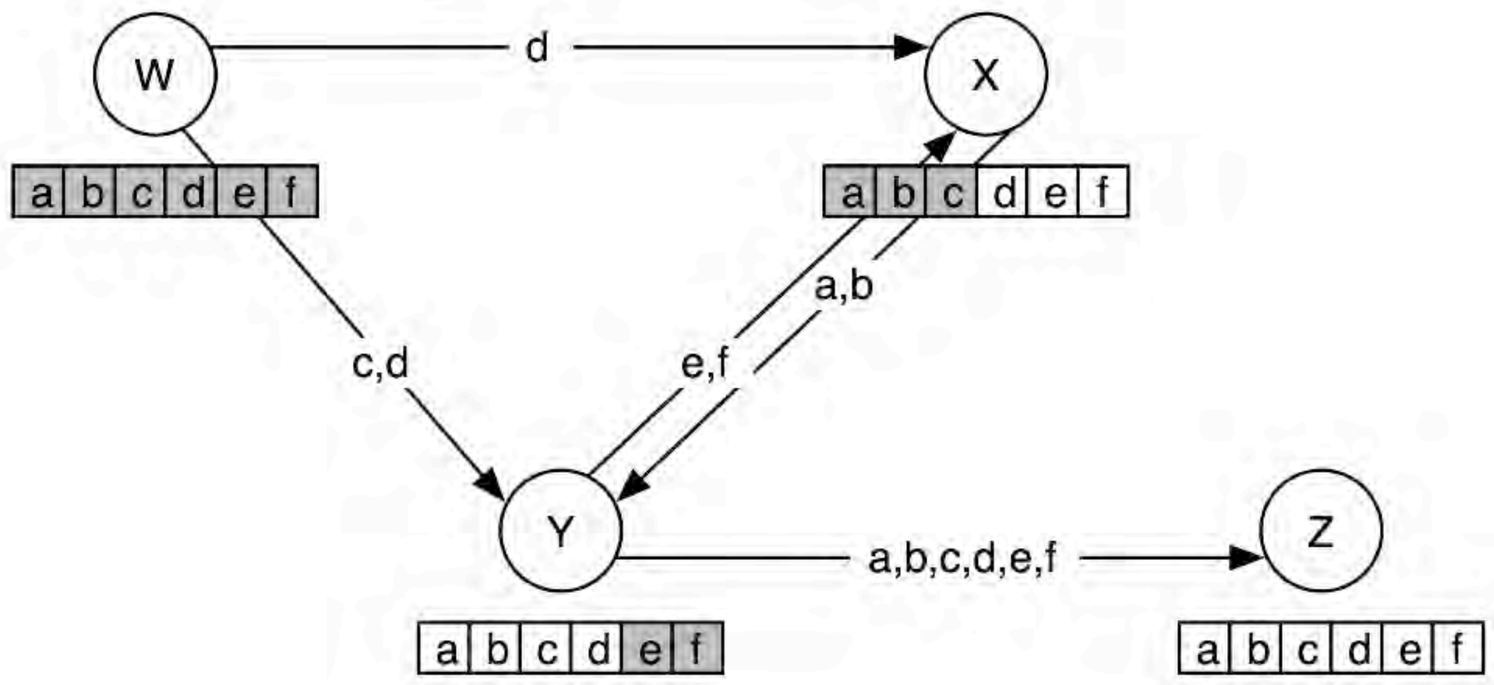


Abbildung 6.5: **Multisource File Transmission Protocol (MFTP) (sinngemäß: [Met])**

Der Dateidownload läuft nach folgendem Schema ab:

1. Der Client sendet die Liste seiner gewünschten Downloads.

2. Der Server sendet eine Liste von Clients, welche die gewünschten Dateifragmente online freigegeben haben („Sources“).

3. Der Client baut Verbindungen zu den anderen Clients auf, um die Dateiteile herunterzuladen.

eDonkey Clients arbeiten alle Download-Anfragen in einer Liste, der *Upload Queue*, der Reihe nach ab. Jeder Eintrag resultiert in einer Upload-Anfrage. Ein Client kann die gleichen Dateiteile von mehreren verschiedenen Clients anfordern, hat er den Teil schon erhalten, wenn ein Client diesen Teil senden möchte, dann lehnt er den Transfer einfach ab.

File ID

Zur eindeutigen Identifikation und zum Entdecken von fehlerhaften Dateien wird ein Hashwert als FileID verwendet. Dabei gibt es zwei Arten:

File Hash Der 128 Bit lange File Hash wird mittels des MD4-Algorithmus [Riv92] aus dem Inhalt der Datei errechnet. Dabei wird die Datei in 9.28 MByte große Teilstücke aufgeteilt, für jedes Teilstück der Hash errechnet und danach kombiniert [KB05].

Root Hash Der Root Hash ist ein SHA1-Hashwert [Jon01] über jeden Block eines 9.28 MByte großen Teilstücks, wobei die Blockgrösse 180 KByte beträgt). Der Hash dient einer höheren Verlässlichkeit und der Fehlervermeidung [KB05].

Nach der Übertragung eines Teilstücks wird der Hashwert auf Übereinstimmung geprüft, ist ein Fehler vorhanden, dann werden jeweils 180 KByte der Datei neu geladen, danach wird der Wert wieder überprüft, bis der Fehler ausgebessert wurde.

6.3 BitTorrent

BitTorrent wurde von Bram Cohen, einem oft als exzentrisch bezeichneten, freien Softwareentwickler designt und Ende 2002 veröffentlicht. Der innovative Ansatz der Tauschbörse wurde von den Benutzern mit offenen Armen empfangen und bald tauchten die ersten, sog. „Tracker"-Seiten, welche für den Dateiaustausch benötigt werden, im Internet auf. Als Ende 2003 mehrere *Distributed Denial of Service*-Angriffe auf die größten Tracker-Seiten ausgeübt wurden, nahmen viele dieser Tracker ihre Server wieder vom Netz [PBE+05].

BitTorrent und die der Tauschbörse zugrundeliegende Technologie wird heute in vielen legalen Anwendungen verwendet. Die Firma Red Hat [www-24] benutzte BitTorrent im März 2003, um die damals neue Version 9.0 von Red Hat Linux zu verbreiten. Innerhalb von 4 Stunden wurden 1.5 Terabyte an Daten heruntergeladen, 3 Tage nach der Release waren es bereits 21.15 Terabyte, wobei nur ein sehr geringer Anteil von der RedHat-Seite direkt heruntergeladen wurde [PBE+05]. Hält man sich vor Augen, dass die durchschnittliche Übertragungsrate in diesen 3 Tagen bei über 85 MByte/s lag, dann wird schnell klar, dass dieser Ansturm von einer einzelnen Internetseite wohl nur sehr schwierig bewältigt werden hätte können, von den Kosten, die diese enorme Datenmenge verursachen würde, ganz zu schweigen.

Eine weitere Anwendung dieser Technik verwendet der Videospiel-Hersteller Blizzard Entertainment [www-25], um die Aktualisierungen für die eigene Software zu verbreiten.

6.3.1 Netzarchitektur

Während sich ein Benutzer bei den meisten Tauschbörsen in einem Netz von Peer-to-Peer Rechnern befindet, in welchem verschiedene Dateien gesucht und ausgetauscht werden können, geht BitTorrent hier einen anderen Weg. BitTorrent bildet kein grosses Netz für alle Dateien, sondern für einzelne

Downloads einen „Schwarm“(Swarm), welcher nur für den Austausch dieses einen Downloads existiert (siehe Abb. 6.6).

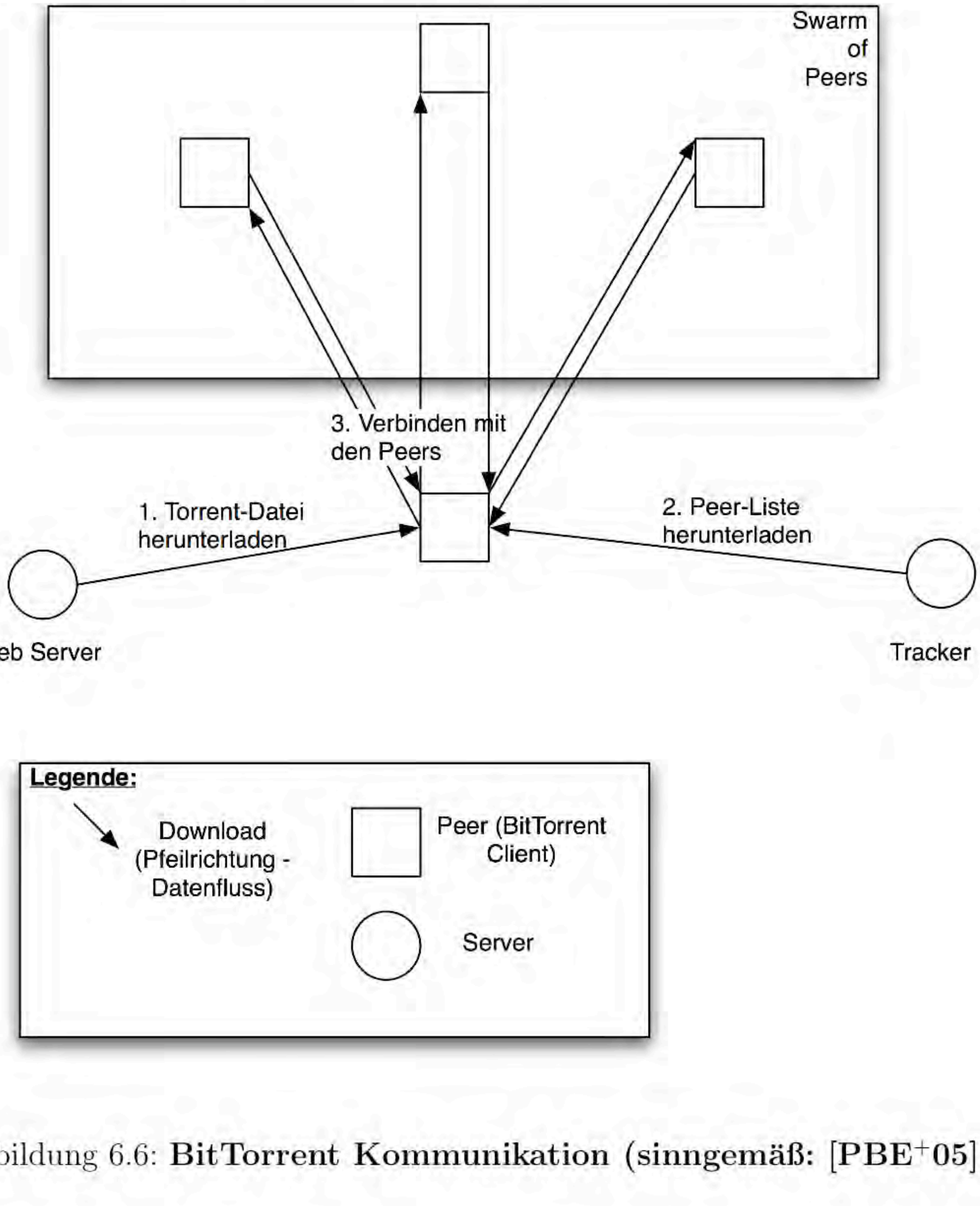
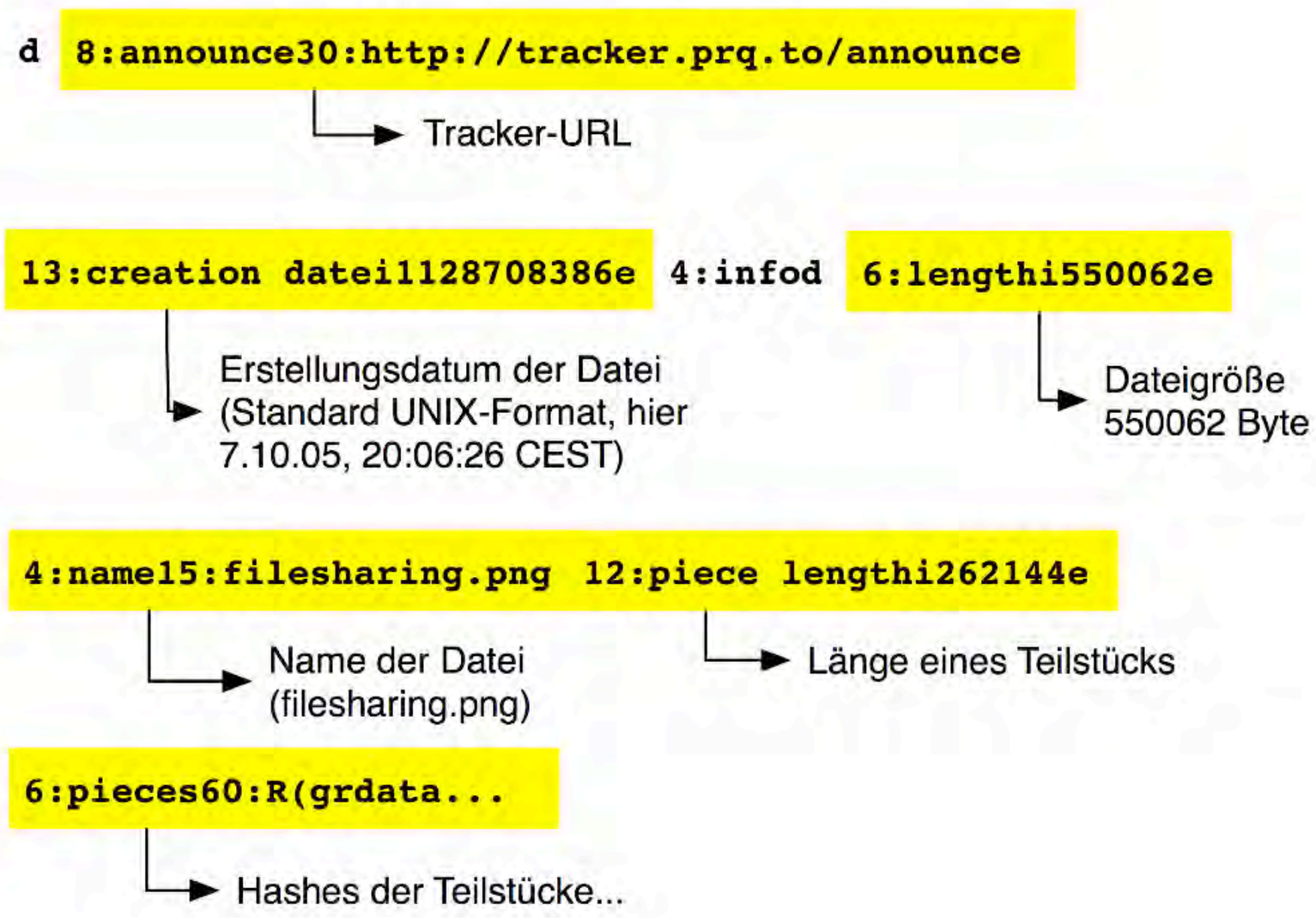


Abbildung 6.6: **BitTorrent Kommunikation (sinngemäß: [PBE+05])**

6.3.2 Torrent-Dateien

BitTorrent arbeitet mit sogenannten Torrent-Dateien (eine genauere Beschreibung der Torrent-Dateien findet sich im Anhang ab Seite 108), welche z.B.

auf einem Webserver abgelegt werden, aber auch auf anderem Wege an die Clients gelangen können. Diese Torrent-Dateien dienen zur eindeutigen Identifizierung einer oder mehrerer Dateien und enthalten unter anderem die URL eines Trackers, welcher für die Verwaltung der Verbindungen unter den Peers zuständig ist [PBE+05]. Als Beispiel sei der Inhalt folgender Torrent-Datei angeführt (entnommen aus [PBE+05]):



Ein Client benötigt zwingend eine Torrent-Datei, um die Verbindung zu einem Tracker herstellen und einen Download beginnen zu können.

6.3.3 Dateitransfer

Der nächste Schritt beim Download einer Datei ist die **Verbindung zum Tracker**. Der Tracker hält eine Liste aller Peers (deren IP-Adressen und offene Ports), welche die Datei zum Herunterladen anbieten, und verteilt diese Information an die anfragenden Clients [PBE+05].

Ein Client kann nun direkt bei den Peers Teilstücke der Datei anfordern und diese dann herunterladen. Um Fehler bei der Übertragung auszuschliessen, ist für jedes Teilstück der Datei in der Torrent-Datei ein Hashwert abgelegt.

Die Clients senden auch Informationen über die übertragenen Datenmengen an den Tracker, welcher nun aufgrund seiner zentralen Stellung auch die Möglichkeit hat, das Verhältnis von Download zu Upload eines Clients zu errechnen. Dieses Verhältnis kann genutzt werden, um reine Downloader zu bestrafen, indem diese entweder weniger oder keine IP-Adressen von Peers aus der Liste erhalten bzw. gute Uploader mit mehr Adressen zu belohnen [PBE+05]. Peers, welche die komplette Datei für den *Swarm* zur Verfügung stellen, werden als *Seeder* bezeichnet, Peers, welche die Datei noch nicht komplett heruntergeladen haben, als *Leecher*.

Eine genaue Beschreibung der verwendeten Nachrichten und deren Codierung findet sich im Anhang ab Seite 108.

6.4 FastTrack

Das FastTrack Netzwerk, welches weniger durch seinen Namen als durch die Namen der dafür verfügbaren Client-Programme wie *Kazaa*, *Morpheus* oder *Grokster*, bekannt ist, wurde im März 2001, zeitgleich mit der Veröffentlichung des ersten Clients, Kazaa, ins Leben gerufen [PBE+05].

FastTrack Clients zeichneten sich schon von Beginn an durch die Beigabe von Spyware und Adware aus. Spyware ist Software, welche versucht, Benutzerverhalten auszuspionieren, Adware ist unerwünschte Werbesoftware [PBE+05].

Im Gegensatz zum damals schon relativ weit verbreiteten Gnutella-Netz, bei welchem das Protokoll offen und frei erweiterbar gehalten ist, war das FastTrack Protokoll von Beginn an ein geschlossenes Protokoll, welches die ausgetauschten Nachrichten verschlüsselt übertrug. Zur Zeit der Veröffentlichung war FastTrack das erste Peer-to-Peer Netz, welches das Konzept der *Supernodes* verwendete. Supernodes übernehmen im FastTrack-Netz wichtige Funktionen wie die Indizierung der verfügbaren Dateien. Gnutella übernahm das Konzept in Form der *Ultrapeers*.

Vorteile des hybriden Ansatzes gegenüber den zentralisierten Peer-to-Peer Netzen der ersten Generation, wie beispielsweise Napster, waren einerseits die bessere Skalierbarkeit und Verlässlichkeit und andererseits die rechtliche Situation. Zentrale Server können einem Besitzer zugeordnet werden, welcher für etwaige Urheberrechts-Verletzungen zur Verantwortung gezogen werden kann. Da bei einem hierarchischen, serverlosen Peer-to-Peer Netz prinzipiell jeder Knoten des Netzes die Rolle eines Supernodes einnehmen kann, von welchen zudem eine grosse Anzahl im Netz existieren, sind diese sehr schwierig greifbar [PBE+05].

Das FastTrack Netz zählte im April 2002 schon 4.4 Millionen Benutzer [fas], derzeit sind es 2.8 Millionen (Stand: 14.6.2006 [www-22]).

6.4.1 FastTrack Netzarchitektur

Wie zuvor schon erwähnt kann jeder Knoten im Netz die Rolle eines **Supernodes** einnehmen. Das Konzept der Supernodes ist ähnlich dem der Ultrapeers bei Gnutella. Supernodes übernehmen die Indizierung der durch die Benutzer freigegebenen Dateien und sind verantwortlich für die Suche im Netz [Har04].

Standardknoten kommunizieren untereinander lediglich zum Zwecke des Dateiaustauschs, die gesamte restliche Kommunikation wird zwischen den Supernodes abgewickelt. Es ergibt sich die in Abb. 6.7 dargestellte Netzstruktur, in welcher die Supernodes den zentralen Ring des Netzes bilden [PBE+05].

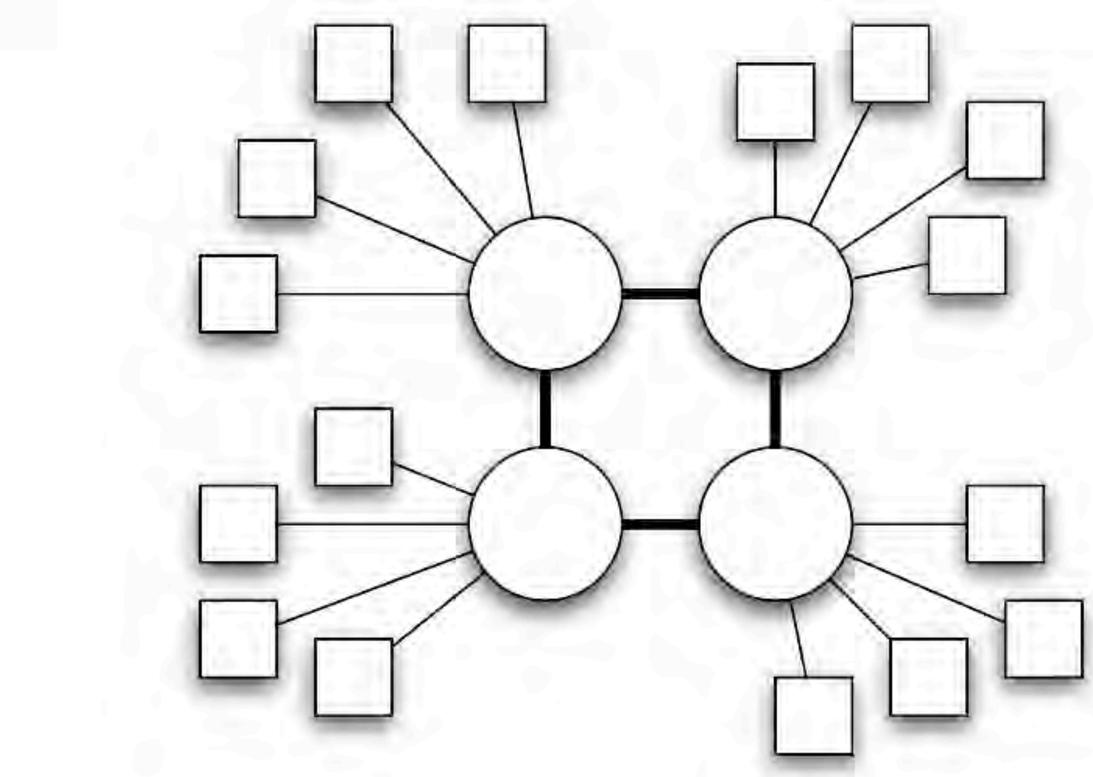


Abbildung 6.7: **Stuktur des FastTrack Netzes (sinngemäß: [PBE+05])**

Supernodes

Jeder Knoten ist zu Beginn ein Standardknoten und kann jederzeit zu einem Supernode werden, es sei denn, der Benutzer hat dies im Client-Programm deaktiviert. Die „Beförderung“ zum Supernode läuft autonom ab, genaueres über die Algorithmen ist nicht bekannt; es ist jedoch wahrscheinlich, dass eine hohe Bandbreite, niedrige Latenz und genug verfügbare Rechenleistung eine Rolle spielen [PBE+05]. Das Verhältnis von Standardknoten zu Supernodes im gesamten Netz ist in etwa 100:1 [Har04].

6.4.2 Verbindungsaufbau

Für den Verbindungsaufbau zum FastTrack Netz benötigt ein Client die IP-Adresse und den Port eines Supernodes. Eine Liste von Supernodes wird lokal gehalten und während der Verbindung im Netz stetig aktualisiert [PBE+05].

Der Standardablauf beim Verbindungsaufbau ist der folgende [PBE+05, Har04]:

1. Der Client sendet ein UDP Paket mit der Verbindungsanfrage (*node ping*) an mehrere Supernodes.

2. Akzeptiert ein Supernode die Verbindung, so antwortet er mit einem ähnlichen UDP Paket, dem *node pong*.

3. Kann der Client die Verbindung mittels UDP nicht herstellen, so versucht er es mittels des TCP-Protokolls.

4. Hat ein Supernode geantwortet, so wird vom Client der Handshake eingeleitet, auf welchen der Supernode antwortet.

5. Von diesem Zeitpunkt an ist die Kommunikation verschlüsselt (näheres zur Art der Verschlüsselung in Punkt 6.4.5), der Name des Netzwerkes (z.B. „KaZaA“ oder „Grokster“) wird ausgetauscht. Ab diesem Zeitpunkt ist die Verbindung zum FastTrack Netz hergestellt.

Kann der Client zu keinem Supernode eine Verbinung aufbauen, dann war der Verbindungsaufbau mit dem FastTrack Netz erfolglos. Die Kenntnis der Adresse mindestens eines Supernodes ist zwingend erforderlich!

6.4.3 Suchanfragen

Suchanfragen werden sofort verschlüsselt und zum Supernode gesendet, welcher diese an die mit ihm verbundenen Supernodes weiterleitet. Die Ergebnisse werden an den Client zurückgesendet [PBE+05].

Der Client kann angeben, nach welcher Dateiart (z.B. Audio, Video oder Software) er suchen möchte und wie viele Suchresultate maximal zurückgeliefert werden dürfen. Die Antwort des Supernodes gibt für jedes Suchergebnis die IP-Adresse, den Port und die Bandbreite des Clients an. Ausserdem wird der Benutzer- und Netzwerkname, die Dateigrösse und der Dateiname sowie ein Hashwert der Datei zurückgeliefert [Har04].

Eine genaue Beschreibung der Kommunikation des FastTrack-Protokolls findet sich in Anhang B.3 ab Seite 112.

6.4.4 Dateitransfer

Ein Dateitransfer wird bei FastTrack, genau wie bei Gnutella, mittels einer `HTTP GET` Anfrage gestartet. Die IP-Adresse und der Port sind aus den Ergebnissen der Suchanfrage bekannt, der Dateitransfer findet über eine direkte TCP-Verbindung statt. Auf eine nähere Beschreibung wird hier verzichtet, im Anhang ab Seite 112 wird nochmals genauer auf den Dateitransfer eingegangen.

6.4.5 Verschlüsselung

Beim Verbindungsaufbau wird von beiden Kommunikationspartnern ein 63 Byte langes Pad mithilfe eines Seed-Wertes, welcher im Klartext ausgetauscht wird, initialisiert. Mit Hilfe des Pads wird eine Pseudo-Zufallszahlenfolge

erzeugt und Byte für Byte mittels einer XOR-Operation mit dem Klartext verknüpft [Har04].

Kapitel 7

Instant Messaging Netze

7.1 Skype

Skype, einer der am meisten verbreiteten Clients für Sprachtelefonie, bietet zusätzlich Unterstützung für Instant Messaging und Dateitransfers. Die Software wurde im August 2003 von den Kazaa-Entwicklern veröffentlicht und hatte bereits ein Jahr nach Herausgabe der ersten Beta-Version, also im Oktober 2004, über 1 Million gleichzeitig am Netz angemeldete Benutzer. Im Mai 2005 waren es schon über 3 Millionen aktive Benutzer [PBE+05] und aktuell (August 2006) sind teilweise über 6 Millionen Benutzer im Skype-Netz online. Der grosse Erfolg von Skype ist einerseits auf die Tatsache zurückzuführen, dass Skype Sprachtelefonie kostenlos anbietet, andererseits auf die einfache Handhabung. Ausserdem funktioniert Skype auch über Proxy-Server und auf Geräten, welche über NAT oder Firewalls an das Internet angebunden sind, fast problemlos. Die Firma bewirbt das Produkt auf seiner Homepage [www-4] mit den Worten „it just works“. Tatsächlich lässt sich Skype im Vergleich zu anderen Sprachtelefonie-Diensten sehr einfach einrichten und bietet eine gute Sprachqualität.

Der Quellcode des Skype Clients ist nicht öffentlich verfügbar und die Kommunikation läuft verschlüsselt ab. Konnte das FastTrack Protokoll aufgrund der schwachen Verschlüsselung noch relativ schnell mittels Reverse-

Engineering Methoden entschlüsselt werden (siehe hierzu [Har04]), ist dies bei Skype aufgrund der eingesetzten AES[1]-Verschlüsselung schwieriger.

Zusätzlich bietet Skype die kostenpflichtigen Dienste „*Skype In*“ und „*Skype Out*“ an [PBE+05]. „Skype In“ bietet die Möglichkeit, von jedem Telefonanschluss im Festnetz aus einen Benutzer im Skype-Netz anzurufen. „Skype Out“ ermöglicht Telefonanrufe vom Skype-Netz aus in das Telefon-Festnetz.

7.1.1 Netzarchitektur

Die Architektur des Skype-Netzes ist der des FastTrack-Netzes (siehe Kapitel 6.4) sehr ähnlich, was nicht weiter überraschend ist, da beide Systeme von denselben Personen entwickelt wurden.

Im Skype-Netz gibt es drei verschiedene Arten von Knoten [PBE+05, BS04]:

Skype Server Der zentrale Skype Server (siehe Abb. 7.1) ist für die Anmeldung neuer Benutzer und die Authentifizierung von Benutzern zuständig. Daher werden Benutzernamen und Passwörter sowie die Kontaktliste der Clients auf diesem Server gespeichert.

Der Login Server ist die einzige zentrale Komponente im Skype-Netz.

Standardknoten Jeder Skype-Client stellt einen Standardknoten im Netz dar. Ein Standardknoten muss zur Anmeldung am Netz die Verbindung zu einem Supernode herstellen und sich am zentralen Skype Server authentifizieren.

Supernodes Jeder Standardknoten mit öffentlicher IP-Adresse, genug Bandbreite und Rechenkraft kann zu einem Supernode werden. Konnte dieses Verhalten im FastTrack-Netz noch deaktiviert werden, hat man bei Skype keinen direkten Einfluss mehr darauf.

[1] AES ... Advanced Encryption Standard [www-26]

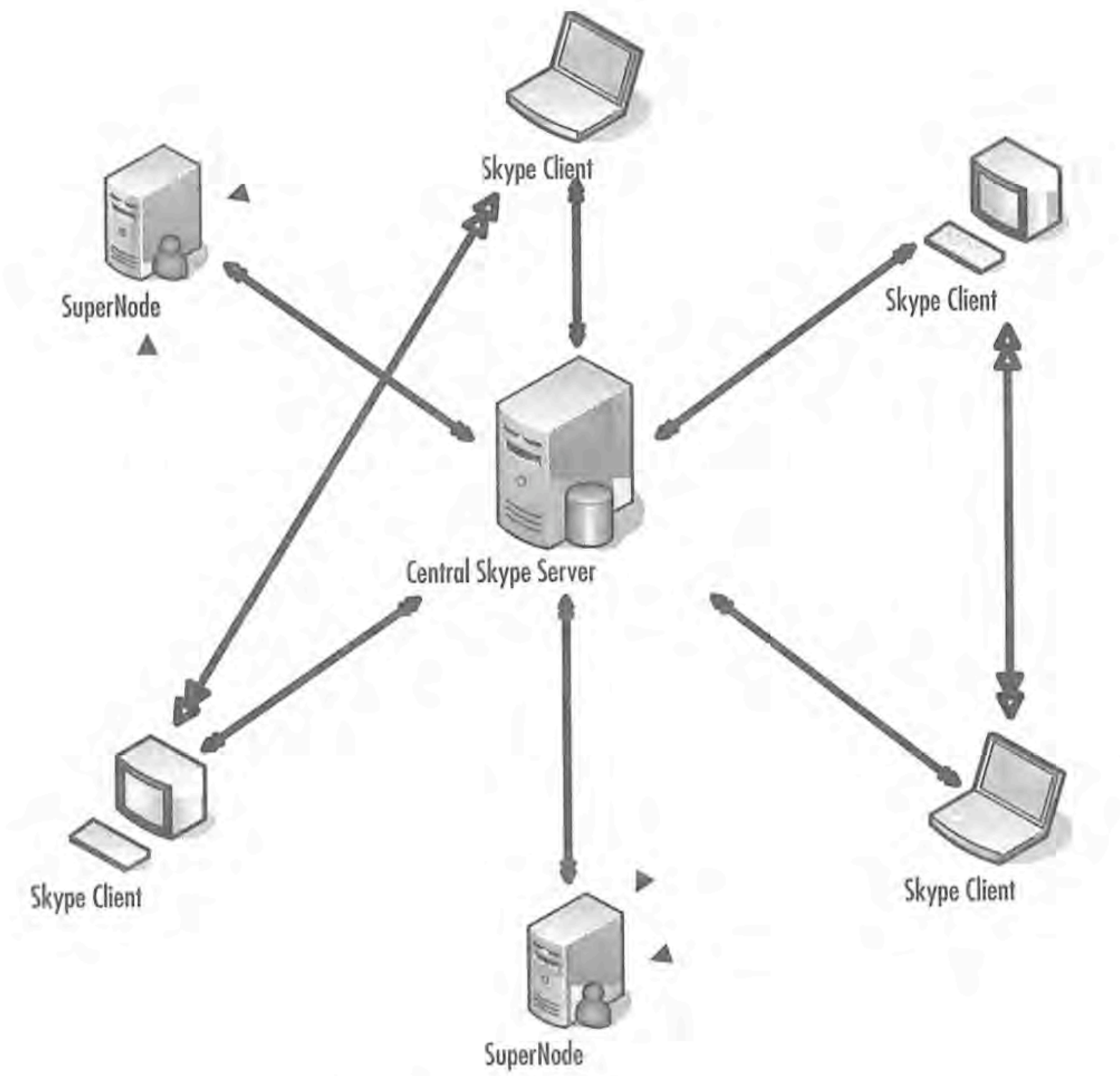


Abbildung 7.1: **Skype Netzarchitektur [PBE+05]**

Es existieren noch weitere Skype-Server, welche für die Dienste „Skype In“ und „Skype Out“ verwendet werden. Da diese jedoch für die Funktion des Peer-to-Peer Netzes keine grundlegende Bedeutung haben, wird hier nicht näher darauf eingegangen.

7.1.2 Verbindungsaufbau

Skype benötigt für die Verbindung zum Netz die IP-Adresse und den Port zumindest eines im Netz aktiven Supernodes. Der Skype Client hält eine lo-

kale Liste mit aktuellen Supernodes, welche ständig aktualisiert und erweitert wird, im sogenannten *Host Cache*.

Zuerst muss der Skype Client die Verbindung zu einem Supernode aufbauen. Der Ablauf des Verbindungsaufbaus ist in Abb. 7.2 dargestellt. Dieses Diagramm setzt voraus, dass im Host Cache nur ein Eintrag vorhanden ist. Sind mehrere Einträge vorhanden, so wird zuerst versucht, die Supernodes mittels UDP zu erreichen, bevor eine TCP-Verbindung aufgebaut wird [BS04].

Nachdem die Verbindung zu einem Supernode erfolgreich aufgebaut wurde, muss sich der Skype Client am *Login Server* authentifizieren.

7.1.3 Benutzersuche

Die Benutzersuche im Skype-Netz wird mit der von Skype auf der firmeneigenen Homepage [www-4] so bezeichneten „Global Index“ Technologie durchgeführt. Die Firma selbst gibt an, dass es sich um eine verteilte Suche handelt und dass ein Benutzer gefunden wird, wenn er in den letzten 72 Stunden vor der Suche online war.

Baset und Schulzrinne konnten bei ihrem Reverse-Engineering-Versuch des Skype Protokolls [BS04] keine genaue Aussage über die beteiligten Kommunikationspartner machen, da der Netzverkehr ab dem Supernode nicht nachvollziehbar ist.

Die Benutzersuche, soweit bekannt, läuft wie folgt ab [BS04]:

1. Der Skype Client kontaktiert seinen Supernode (TCP-Verbindung).

2. Der Supernode antwortet über die gleiche TCP-Verbindung.

3. Von hier an gibt es ein unterschiedliches Verhalten, je nachdem, ob der Client eine öffentliche IP-Adresse besitzt oder hinter einer Firewall oder mit NAT angeschlossen ist.

   (a) *Client hat eine öffentliche IP-Adresse:* Der Skype Client kontaktiert verschiedene (bisher unbekannte) Knoten über UDP.

(b) *Client mit NAT:* Der Client versucht, schon bekannte Knoten über UDP zu kontaktieren.

(c) *Client hinter Firewall:* Die Kommunikation findet ausschließlich mit dem Supernode statt.

Genaueres über die Ergebnisse konnte leider bisher noch nicht herausgefunden werden. In [BS04] wurde experimentell festgestellt, dass Suchanfragen sehr wahrscheinlich in Zwischenknoten gespeichert werden.

7.1.4 Verbindung zu anderen Benutzern (Sprachtelefonie)

Beim Aufbau der Verbindung und dem Datenaustausch während des Telefonats konnten [BS04] wieder ein unterschiedliches Verhalten feststellen.

Dabei wird hier davon ausgegangen, dass sich der Angerufene bereits in der Kontaktliste befindet. Ist dies nicht der Fall, dann führt Skype vor dem Aufbau der Telefonie-Verbindung eine Benutzersuche durch.

Man unterscheidet folgende drei Fälle [BS04]:

a) Benutzer-Clients auf öffentlicher IP-Adresse

Haben beide Benutzer eine öffentliche IP-Adresse und gibt es keine Einschränkungen auf erreichbare Ports, so wird eine TCP-Verbindung zum Angerufenen hergestellt. Auch der Anruf wird mittels TCP übertragen.

b) Anrufer über NAT angebunden, Angerufener hat öffentliche IP-Adresse

Beim Verbindungsaufbau kommuniziert der Anrufer mit verschiedenen Knoten und dem Supernode. Der Anruf wird mittels TCP über einen Proxy-Knoten (einen Supernode) übertragen.

c) Beide Benutzer hinter Firewall mit NAT

Auch hier findet wieder Kommunikation des Anrufers mit verschiedenen Knoten und dem Supernode statt. Der Anruf wird wiederum über einen Proxy-Knoten geleitet, diesmal allerdings mittels UDP.

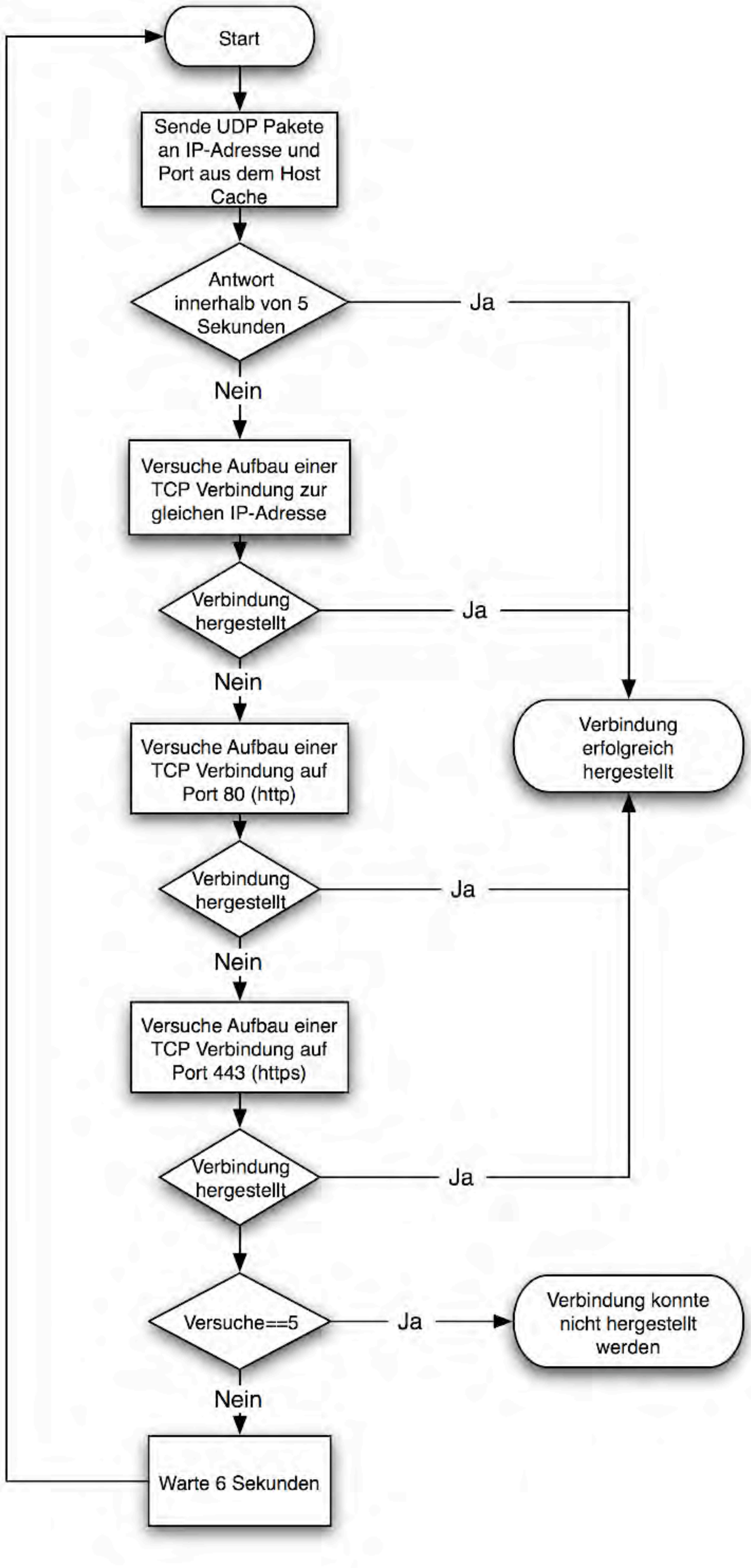


Abbildung 7.2: **Verbindungsaufbau des Skype-Clients zu einem Supernode [BS04]**

7.2 AOL Instant Messenger und ICQ

Als die Firma Mirabilis 1996 die Instant Messaging Software ICQ auf den Markt brachte und kostenlos zum Download anbot, hatten sich innerhalb eines halben Jahres über 850000 Benutzer angemeldet [PBE+05] (zieht man die damalige Grösse des Internets in Betracht, ein extrem hoher Wert). AOL hatte zu diesem Zeitpunkt bereits, allerdings exklusiv nur für seine Kunden, einen Instant Messenger, den AOL Instant Messenger, kurz AIM, veröffentlicht. Durch den grossen Erfolg von ICQ sah sich AOL genötigt, seinen Instant Messenger ebenfalls kostenlos zur Verfügung zu stellen [PBE+05].

Als AOL 1998 Mirabilis übernahm, liefen die beiden Instant Messenger bis zum Jahr 2002 nebeneinander, danach wurden die Clients von AOL verbunden. Die Clients verwenden heute die gleichen Login Server und Protokolle, der einzige Unterschied besteht darin, dass ICQ Nummern für die Kontaktliste verwendet, während AIM ausschließlich mit Benutzernamen arbeitet [PBE+05].

7.2.1 Netzarchitektur

ICQ und AIM verwenden eine **Client-Server Struktur**, es handelt sich also streng genommen nicht um Peer-to-Peer Protokolle, sondern mehr um „Person-to-Person“ Anwendungen. Alle Nachrichten werden an einen zentralen Server gesendet, welcher diese an den Empfänger weiterleitet (siehe Abb. 7.3.

Bedenklich erscheint hierbei, dass diese Nachrichten nicht verschlüsselt werden, es existiert also einerseits die Gefahr des Mithörens Dritter auf der Leitung und andererseits ist der zentrale Server ein nicht eliminierbarer „Man in the middle“, der die gesamte Kommunikation im Klartext mithören kann. Als Vorteil dieses Verfahrens kann gesehen werden, dass die IP-Adresse des Kommunikationspartners bei dieser Methode den Clients nicht bekannt ist.

Im Netz existieren zwei Arten von Servern [PBE+05]:

OSCAR Server Die OSCAR Server sind für die Authentifizierung der Clients und die Weitergabe der IP-Adresse eines zugeteilten BOS Servers zuständig.

BOS Server Über die BOS Server (Basic OSCAR Server) läuft die gesamte Kommunikation ab. Alle Nachrichten werden zuerst an einen BOS Server gesendet, dieser leitet sie an den richtigen Empfänger weiter. Ausserdem hält der BOS Server die Informationen über den On- und Offlinestatus der Benutzer aktuell und speichert Nachrichten für Benutzer, die derzeit nicht online sind, für die spätere Übermittlung.

7.2.2 Kommunikation

Die Kommunikation läuft, wie in Abbildung 7.3 dargestellt, folgendermaßen ab [PBE+05]:

1. Der Client authentifiziert sich mit Benutzername und Passwort an einen der OSCAR Server.

   Ist die Authentifizierung erfolgreich, so bekommt der Client ein Ticket für die Anmeldung an einem BOS (Basic OSCAR) Server. Dieses Ticket wird ungültig, sobald sich der Client vom Netz abmeldet. OSCAR und BOS Server kommunizieren zu diesem Zweck direkt miteinander.

   Zusätzlich sendet der OSCAR Server dem Client seine Kontaktliste, welche am Server gespeichert ist.

2. Mit seinem Ticket meldet sich der Client am BOS Server an und erhält die Statusinformationen der Benutzer in seiner Kontaktliste.

   Sendet er nun eine Nachricht an einen Kontakt aus seiner Liste, so wird die Nachricht an den BOS Server gesendet.

3. Der BOS Server leitet die Nachricht an die richtige IP-Adresse weiter.

Das Protokoll für die Kommunikation ist ein geschlossenes Protokoll von AOL, das OSCAR-Protokoll. Es wurde nie vollständig veröffentlicht, da es je-

doch keinerlei Verschlüsselungsmethoden verwendet, konnte es mit Hilfe von Reverse-Engineering Methoden ziemlich schnell entschlüsselt werden. 1999 veröffentlichte AOL das TOC-Protokoll [PBE+05], ein Subset des OSCAR Protokolls, unter der GNU-Lizenz.

7.2.3 Verschlüsselung

Verschlüsselt wird beim OSCAR Protokoll lediglich das Passwort des Benutzers. Es wird eine XOR-Chiffrierung verwendet. Leider ist der Hash bereits bekannt und immer gleich, weshalb sich durch Mithören, beispielsweise auf unverschlüsselten WLANs, das chiffrierte Passwort aufzeichnen und problemlos entschlüsseln lässt.

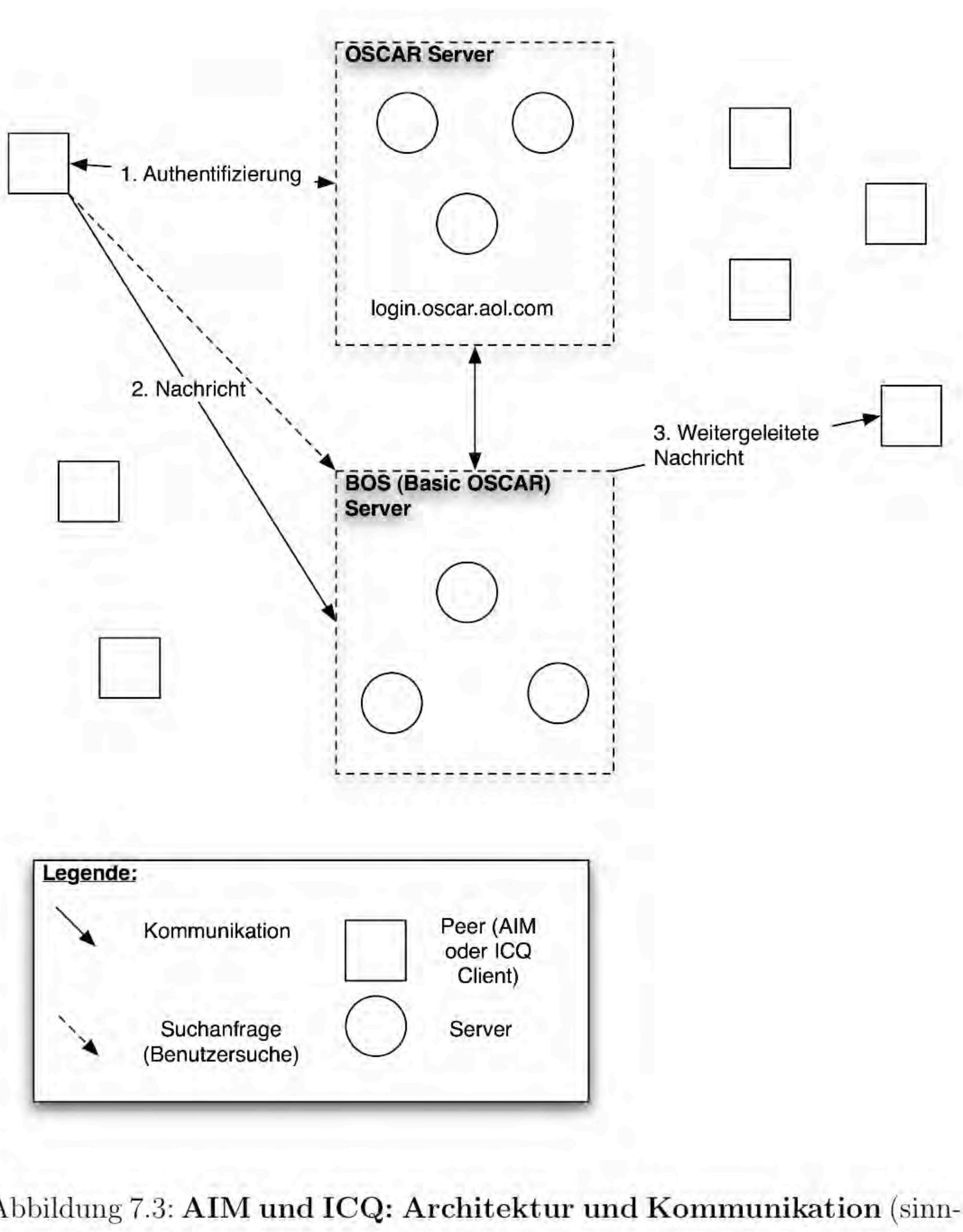


Abbildung 7.3: **AIM und ICQ: Architektur und Kommunikation** (sinngemäß: [PBE+05])

7.3 MSN Messenger

Der MSN Messenger wurde von Microsoft im Jahre 1999 veröffentlicht. Anfangs hatte der Microsoft-Client noch keine grosse Benutzergemeinde, was Microsoft durch die Kompatibilät zu AOLs AIM Instant Messenger Netz auszugleichen versuchte. AOL änderte daraufhin das Protokoll, Microsoft wiederum passte den Client an das neue Protokoll an. AOL wollte sich das nicht gefallen lassen, veränderte das Protokoll immer wieder und somit kam es im Jahr 1999 zu 21 veröffentlichten Versionen des Microsoft Messengers [www-27] [PBE+05].

Ab der Version 2.0, die noch Ende 1999 veröffentlicht wurde, ging Microsoft schließlich eigene Wege und verwendete ein eigenes Protokoll. Seit dieser Zeit hat sich sehr viel getan und so verfügt der Microsoft Messenger heute nicht nur über die grösste Benutzergemeinde aller IM-Anwendungen, sondern bietet auch vielfältige Zusatzfunktionen wie Video und Audio Chat oder Dateiaustausch an [PBE+05].

Die Kehrseite der weiten Verbreitung ist die grosse Anzahl an bekannten Sicherheitslücken, Viren und Würmern für diesen Client. Laut dem IM Logic Threat Center von Symantec hatten von allen im Jahr 2005 entdeckten Bedrohungen 57% den MSN Messenger zum Ziel (siehe auch Abb. 7.4) [Log06].

7.3.1 Netzarchitektur

Die relativ komplizierte Struktur des MSN-Messenger Netzes umfasst fünf verschiedene Servertypen [PBE+05] (siehe auch Abb. 7.5):

Dispatch Server Der Dispatch Server hält die Liste der aktiven Notification Server aktuell und gibt diese an die Clients weiter.

Notification Server Der Notification Server kennt die IP-Adressen und den Status (online, offline, etc.) der Benutzer. Außerdem ist er für den Aufbau der Verbindung zu den Switchboard Servern zuständig. Der

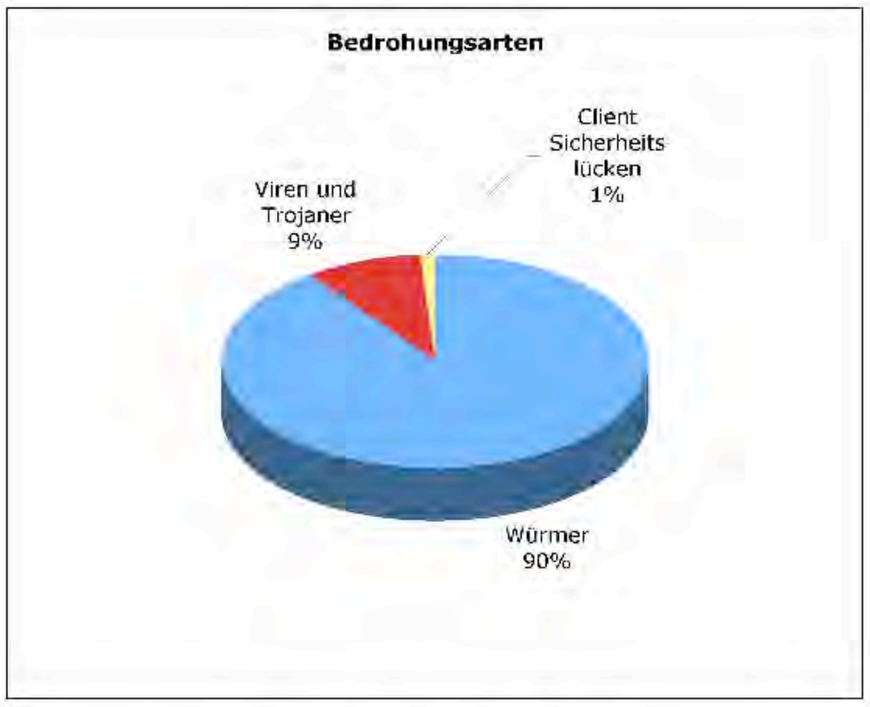


Abbildung 7.4: **Bedrohungen für Instant Messenger 2005** (Quelle: [Log06])

MSN-Messenger Client muss die Verbindung zum Notification Server während der gesamten Sitzung offen halten.

.NET Passport Login Server Der .NET Passport Login Server ist notwendig für die Authentifizierung am Notification Server.

Nexus Server Der Nexus Server ist ebenfalls für den Authentifizierungsprozess zuständig. Er versorgt den Client mit der URL eines Passport Login Servers.

Switchboard Server Switchboard Server stellen im MSN-Messenger Netz die Funktionen für Messaging und Dateitransfers zur Verfügung.

7.3.2 Verbindungsaufbau

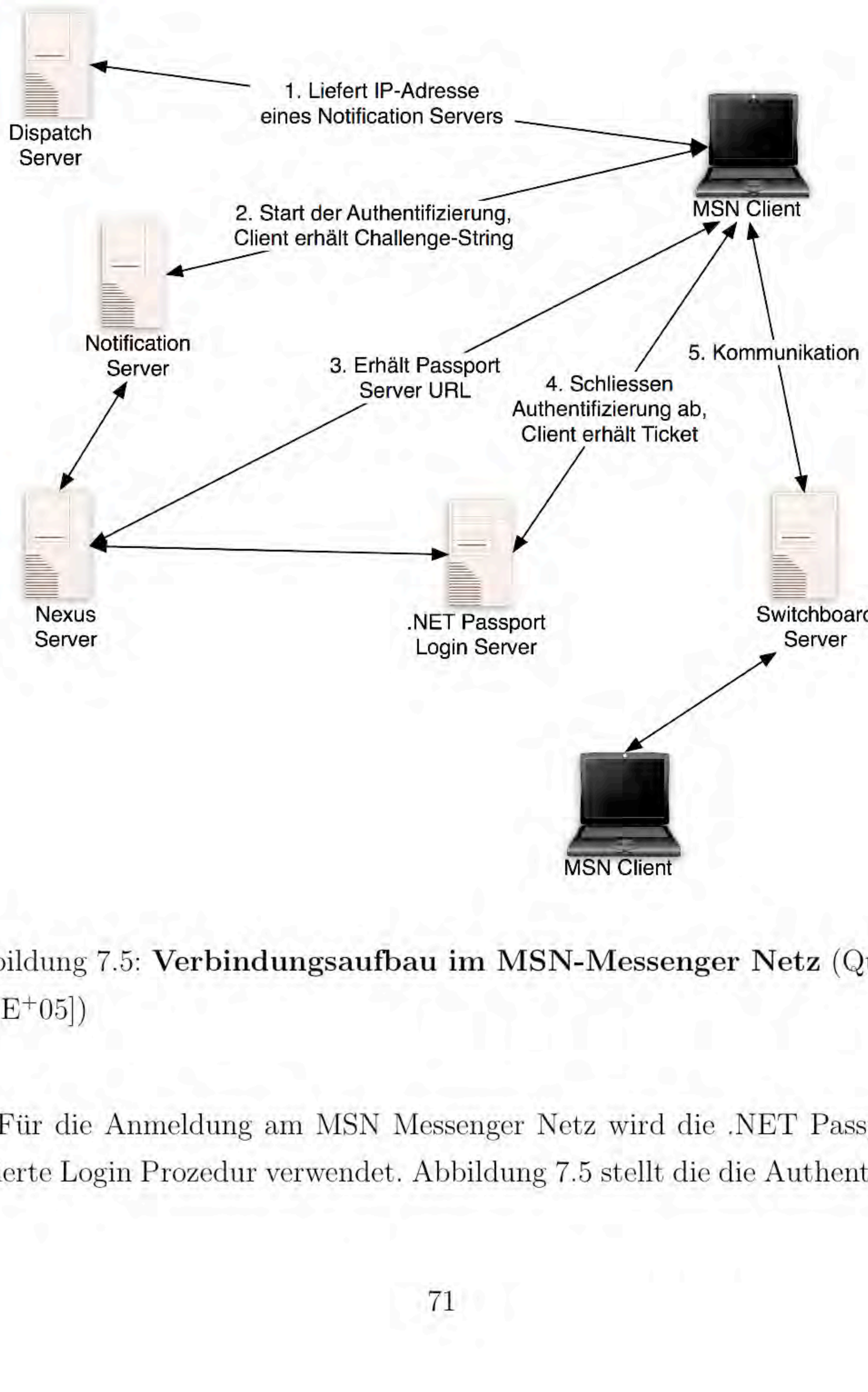


Abbildung 7.5: **Verbindungsaufbau im MSN-Messenger Netz** (Quelle: [PBE+05])

Für die Anmeldung am MSN Messenger Netz wird die .NET Passport-basierte Login Prozedur verwendet. Abbildung 7.5 stellt die die Authentifzie-

rung schematisch dar. Die einzelnen Schritte werden im folgenden detailliert beschrieben [PBE+05]:

1. Der Client verbindet sich zu einem Dispatch Server, welcher dem Client die IP-Adresse eines aktiven Notification Servers sendet. Hat ein Client einmal die IP-Adresse eines Notification Servers erhalten, so wird diese so lange verwendet, bis der Server nicht mehr aktiv ist. Sollte dies der Fall sein, verbindet sich der Client wieder zum Dispatch Server und erhält eine andere IP-Adresse.

2. Der Client stellt die Verbindung zum Notification Server her. Diese Verbindung muss während der gesamten Sitzung offen gehalten werden.

   Der Notification Server sendet dem Client den Challenge-String.

3. Der Client kontaktiert den Nexus Server, von welchem er die Passport Server URL erhält.

   Notification Server, Nexus Server und der .NET Passport Login Server tauschen während dieser Phase Authentifizierungsinformationen aus.

4. Der Client sendet den Challenge String, den Benutzernamen und das Passwort an die vom Nexus Server erhaltene URL. Sind die Informationen korrekt, so vergibt der Login Server ein *Ticket* an den Client, welches dieser an den Notification Server sendet, um den Authentifizierungsprozess abzuschliessen.

5. Nach der Anmeldung findet die Kommunikation mit dem Notification Server und den Switchboard Servern statt (mehr dazu in Kapitel 7.3.3).

7.3.3 Kommunikation

Wird ein Benutzer zu einem Chat eingeladen, so sendet der Notification Server eine Nachricht an den Client, woraufhin dieser sich sofort mit dem Switchboard Server verbindet [msn03]. Die Kommunikation mit den anderem

Benutzer findet über diesen Server statt, was bedeutet, dass theoretisch ein Abhören durch Microsoft ganz einfach möglich wäre.

Kapitel 8

Lösungen, Lösungsansätze

Der erste Schritt zur Lösung des „Problems“ Peer-to-Peer ist es, sich der Risiken und Gefahren, die davon ausgehen, bewusst zu werden. Kapitel 8.1 bietet hierfür Ansätze, wie das Peer-to-Peer in der Sicherheitspolitik eines Unternehmens thematisiert werden kann. Zusätzlich findet sich ein Lösungsansatz für die geplante Verwendung von Peer-to-Peer in einem Unternehmen am Beispiel Skype.

Kapitel 8.3 widmet sich aus technischer Sicht der Erkennung von Peer-to-Peer an den Unternehmensgrenzen. Abschließend werden Open Source Projekte vorgestellt, die für die Lösung des Problems verwendet werden können.

8.1 Einbeziehen von Peer-to-Peer in die Sicherheitspolitik des Unternehmens

Da die Entwickler von Peer-to-Peer Anwendungen sehr erfinderisch sind, was das Umgehen von Sicherheitsmaßnahmen wie Firewalls betrifft, wird es auf Dauer nicht ausreichen bzw. sehr aufwändig sein, technische Maßnahmen gegen Peer-to-Peer zu setzen. Es empfiehlt sich daher, Peer-to-Peer in die Sicherheitspolitik des Unternehmens zu integrieren.

Soll Peer-to-Peer nicht grundsätzlich verboten werden, so könnte eine Sicherheitspolitik folgende Punkte enthalten [Dam02]:

- Installation und Konfiguration von P2P-Anwendungen auf den Arbeitsplatzrechnern, die bereits mit integrierten Sicherheitsmaßnahmen konstruiert sind und für Unternehmen zusätzliche serverbasierte Sicherheitslösungen anbieten, die die Einhaltung von Sicherheitspolitiken an den Unternehmensgrenzen ermöglichen.
- Verbot und Verhinderung der Installation anderer P2P-Anwendungen.
- Ausbildung und Training der Mitarbeiter in den Gefahren von P2P-Anwendungen sowie deren sicherer Nutzung.

Der Einsatz von Peer-to-Peer Anwendungen in einem Unternehmen bedarf sorgfältiger Analysen und einer Risikoabschätzung. Das nachfolgende Beispiel soll zeigen, wie der Einsatz von Peer-to-Peer Anwendungen in einem Unternehmen realisierbar wäre.

8.2 Die Verwendung von Instant Messaging in Unternehmen am Beispiel Skype

Der Wunsch, Skype in einem Unternehmen einzusetzen, mag nahe liegen, stellt man die Möglichkeit kostenloser Telefonate der monatlichen Telefonrechnung gegenüber. Die einfache Handhabung und Benutzerfreundlichkeit von Skype im Vergleich zu vielen anderen Voice over IP-Anwendungen, gepaart mit der guten Sprachqualität, sprechen für diese Software.

Doch für den Einsatz in einem Unternehmen gilt es, das Programm nach möglichen Sicherheitsrisiken zu untersuchen, um das Risiko beim Einsatz dieser Software abschätzen zu können.

8.2.1 Sicherheitsaspekte von Skype

Da Skype mit einem geheimgehaltenen, verschlüsselten Protokoll arbeitet und auch der Quellcode der Software nicht frei verfügbar ist, fällt eine voll-

ständige Sicherheitsanalyse bei Skype nicht leicht. Zusätzlich zu den in Kapitel 4 angeführten allgemeinen Sicherheitsaspekten für Peer-to-Peer Software werden nachfolgend spezielle, für Skype interessante, Aspekte genauer betrachtet.

Geheimhaltung, Verschlüsselung

Skype verwendet, laut Angaben auf der firmeneigenen Homepage, 256 Bit AES[1] für die Verschlüsselung der Kommunikation und RSA [RSA78] mit Schlüssellänge 1024 Bit für den Austausch der Schlüssel. Diese Verschlüsselungsalgorithmen werden allgemein als sicher angesehen, Fehler können aber auch bei der Implementierung der Algorithmen passieren. Die Tatsache, dass Skype überhaupt Verschlüsselung verwendet, ist ein Fortschritt gegenüber den meisten heute existierenden VoIP-Anwendungen [Gar05].

In [Gar05] wird die Kommunikation zwischen zwei Skype-Clients als sicher bewertet, es wird jedoch darauf hingewiesen, dass es für einen Angreifer möglich ist, mit Hilfe der Analyse des Netzverkehrs festzustellen, welche Benutzer wann miteinander kommunizieren. In [BD06] werden mehrere mögliche Angriffe auf Skype und die verwendete Verschlüsselung beschrieben.

Integrität

Die Integrität der Software ist aus der Sicht eines Netzwerkadministrators ein wichtiger Punkt für die Gesamtsicherheit eines Systems und eines Netzwerks.

Wird ein Knoten zu einem *Supernode*, so werden die Ressourcen dieses Knotens stark beansprucht. Die Netzwerklast wird durch die zusätzliche Kommunikation, die nicht mit einem direkten Nutzen für den Benutzer verbunden ist, erhöht und auch Rechenleistung wird in Anspruch genommen [Gar05].

Der Hersteller der Skype-Software gibt an, dass Skype ohne Spyware oder Adware ausgeliefert wird, es ist jedoch nicht möglich, diese Gefahr gänzlich

[1] AES...Advanced Encryption Standard

auszuschliessen. Durch die automatische Aktualisierungsfunktion von Skype wird die Gefahr zusätzlich erhöht [Gar05].

Verfügbarkeit

Die Verfügbarkeit des öffentlichen Telefonnetzes beträgt in den meisten Gegenden 99.99905%, was einer Ausfallzeit von etwa 5 Minuten pro Jahr gleichkommt [Gar05]. Es ist schwierig, die Verfügbarkeit des Internets zu bewerten, es spielen jedoch noch andere Faktoren eine Rolle. Ein neuralgischer Punkt sind die zentralen Authentifzierungs-Server von Skype. Ein Skype Client muss sich an diesen Servern authentifizieren, ein Ausfall wäre gleichbedeutend mit dem Ausfall des Skype-Netzes [Gar05].

Abschließend kann gesagt werden, dass Skype prinzipiell mehr Sicherheit bietet als das analoge (bzw. ISDN-) Telefonnetz und die meisten Voice over IP-Anwendungen. Die Möglichkeit einer Kompromittierung des Sytems besteht jedoch, auch ist die Vertrauenswürdigkeit des Unternehmens, welches hinter Skype steht, kritisch zu betrachten [Gar05].

Unternehmen, welche mit dem Gedanken spielen, Skype im firmeneigenen Netzwerk und darüber hinaus einzusetzen, sollten, zusätzlich zu den in Kapitel 8.2.2 angeführten technischen Massnahmen, folgende Punkte beachten:

- Skype sollte nicht als Ersatz für bestehende Telefonleitungen dienen, da die Ausfallsicherheit nicht garantiert werden kann und die Herstellerfirma auch keine Garantien dafür übernimmt.

- Benutzer sollten angehalten werden, sichere Passwörter für Skype zu verwenden und diese regelmässig zu ändern. Außerdem darf das Skype-Passwort nicht identisch mit in der Firma verwendeten Passwörtern sein.

- Administratoren sollten die Aktivität von Skype auf den Arbeitsplatzrechnern überwachen, um möglicher Spyware oder Trojanern vorzubeu-

gen und um zu verhindern, dass die Ressourcen der Rechner zu stark beansprucht werden.

8.2.2 Technische Massnahmen [PBE+05]

Beim Einsatz von Skype im Unternehmen gilt es, wenn es die Sicherheitspolitik der Firewall des Unternehmens erlaubt, folgende Punkte zu beachten:

Verhindern, dass der Skype Client zum Supernode wird

Um festzustellen, ob der eigene Rechner als Skype Supernode fungiert, muss man die Verbindungen des Rechners über einen gewissen Zeitraum betrachten (beispielsweise mit dem Programm *netstat*). Baut der Rechner viele Verbindungen zu verschiedensten IP-Adressen auf, obwohl außer dem Skype-Client keine Netzwerkintensiven Anwendungen laufen, so lässt dies den Schluss zu, dass man die Rolle eines Supernodes eingenommen hat. Dies lässt sich durch den Einsatz eines NAT-Routers oder einer Firewall verhindern. Sobald Skype feststellt, dass es sich hinter einem solchen Gerät befindet, wird es den Client nicht zum Supernode machen [PBE+05].

Gespräche oder Dateitransfers sollen nicht über andere Supernodes relayed werden

Damit die Kommunikation der Clients nicht über andere Knoten relayed wird, muss die Firewall Peer-to-Peer Verbindungen erlauben. Konkret bedeutet dies, dass ein UDP/TCP-Port von außen erreichbar sein muss.

Skype legt bei der Installation einen **zufälligen Port** fest, auf welchem eingehende Verbindungen akzeptiert werden. Dieser Port muss auf der Firewall durchgeschaltet werden, um eine direkte Verbindung zu den Clients zu erlauben und somit Relaying zu verhindern.

Da der Port in der Datei `shared.xml` (diese befindet sich bei Windows Systemen im Skype-Ordner auf der lokalen Festplatte) abgespeichert wird, ist es möglich, diesen auf allen Clients automatisiert zu ändern. Innerhalb der

Datei wird der Port im Schlüssel `config/lib/Connection/ListeningPort` abgespeichert.

Gespräche innerhalb eines Firmennetzes werden, sofern innerhalb dieses Netzes Verbindungen zwischen den Arbeitsplatzrechnern erlaubt sind, direkt und ohne Relay-Knoten geführt.

Verbesserung der Sprachqualität

Die Sprachqualität hängt stark davon ab, ob Skype UDP verwenden kann oder nicht. Um optimale Bedingungen zu schaffen, sollten UDP Ports von innen nach außen geöffnet und UDP-Antworten erlaubt werden.

Von innen nach außen muss theoretisch nur Port 80 oder 443 geöffnet sein (was bei fast jedem Netzwerk der Fall sein wird). Aufpassen sollte man bei Firewalls, welche den Netzverkehr auf diesen Ports auf das HTTP(S)-Protokoll beschränken, da Skype diese Protokolle nicht verwendet.

Diagnosemöglichkeiten

Sollte es Probleme mit der Skype-Verbindung geben, empfiehlt der Hersteller selbst ein Werkzeug namens *NAT Check* [www-28]. Mit Hilfe dieses Werkzeuges kann die „Peer-to-Peer-Freundlichkeit“ einer Firewall überprüft werden.

Eine weitere Möglichkeit bietet Skype selbst durch das Anzeigen von erweiterten Informationen bei Anrufen. In der Datei `config.xml`, welche sich im Unterordner des jeweiligen Benutzers innerhalb des Skype Ordners befindet, kann der Wert des Schlüssels `config/UI/Messages/DisplayCallInfo` von „0“ auf „1“ geändert werden. Dadurch werden erweiterte Informationen über das Gespräch wie etwa Jitter, Paketverlust, das verwendete Protokoll oder die Anzahl der Relay-Knoten angezeigt.

8.3 Peer-to-Peer Netzverkehr an den Unternehmensgrenzen erkennen

Um Verbindungen von Peer-to-Peer Netzen über die Unternehmensgrenzen hinweg erkennen und kontrollieren zu können, gibt es mehrere Möglichkeiten, welche nachfolgend angeführt werden sollen.

Peer-to-Peer Netze der ersten Generation (wie z.B. Napster) konnten noch durch das Blockieren der Peer-to-Peer Server, sofern diese bekannt waren, kontrolliert werden. Die nächste Generation verwendet jedoch keine dedizierten Server mehr, bzw. wechseln diese sehr schnell und auch die Anzahl stieg auf ein kaum überschaubares Maß an.

8.3.1 Erkennen von Peer-to-Peer Protokollen anhand von TCP/UDP-Ports

Eine Standard-Firewall mit Paketfilter-Funktionen kann Peer-to-Peer Netzverkehr nur anhand der bei der Kommunikation verwendeten TCP bzw. UDP-Ports erkennen.

Eine Liste der von den Anwendungen verwendeten Standardports findet sich in Anhang A.1 ab Seite 102. Diese Erkennungsmethode ist für die aktuelle Generation an Peer-to-Peer Anwendungen nicht mehr ausreichend, da diese sich über jeden beliebigen Port mit dem Netz verbinden können und mehrere Techniken verwenden, um Firewalls übergehen zu können (siehe Kapitel 4.3.1, „Konzepte zur Überwindung von Firewalls“).

8.3.2 Erkennen von Peer-to-Peer Protokollen anhand der Analyse der Anwendungsschicht

Da die Erkennung über TCP/UDP-Ports bei aktuellen Peer-to-Peer Anwendungen nicht mehr möglich ist, versuchen neue Systeme, die Protokolle über die Analyse der Anwendungsschicht (Schicht 7 des ISO/OSI Referenzmodells [Zim80]) zu erkennen.

Zu diesem Zweck muss der Payload der übertragenen Pakete untersucht werden. Manchmal ist es auch notwendig, die Pakete einer Sitzung als Gesamtes zu betrachten, um feststellen zu können, um welches Protokoll es sich handelt. Es gibt verschiedene kommerzielle Anwendungen und einige Open Source Systeme, die sich dieser Aufgabe widmen und auf die später noch näher eingegangen wird. *Intrusion Detection Systeme* gehören prinzipiell auch zu diesen Anwendungen.

Leider haben auch diese Systeme ihre Grenzen, wie das aktuelle Peer-to-Peer Programm Skype im Test zeigt. Skype verschlüsselt die gesamte Kommunikation, was dazu führt, dass der Inhalt nicht mehr im Klartext analysiert werden kann. Um Skype zu blockieren, muss ein Proxy verwendet werden. Skype kann zwar über Proxies funktionieren, konfiguriert man den Proxy jedoch so, dass er die SSL Connect Methode beim Aufbau zu numerischen IP-Adressen blockiert, wird Skype am Verbindungsaufbau gehindert. In Umgebungen, welche den Internetzugriff nur über Proxy-Server zulassen, kann diese Methode zum Blockieren von Skype verwendet werden.

8.4 Open Source Projekte

Mit Hilfe der in den nächsten Kapiteln vorgestellten Open Source Software wurde versucht, Peer-to-Peer Netzverkehr an der Firewall zu erkennen. Die Ergebnisse sind in Tabelle 8.3 ablesbar, welche sich am Ende dieses Kapitels befindet. Außerdem wurden die Anwendungen auf ihre Performance untersucht, um in etwa eine Vorstellung zu bekommen, was Inhaltsanalyse auf der Anwendungsschicht an Rechenkraft kostet.

8.4.1 IPP2P

Das IPP2P-Projekt [www-20] ist eine Erweiterung für Netfilter/IPTables [www-19], das Paketfilter-Framework für den Linux-Kernel. IPP2P erweitert

das Framework als Modul um Funktionen für das Erkennen von Paketen, welche von Peer-to-Peer Tauschbörsen stammen.

IPP2P kann keine Instant Messaging Protokolle erkennen, benötigt man solche Funktionen, so sei hier auf das ebenfalls in Kapitel 8.4.2 besprochene *Layer7-filter* Modul für Netfilter, auf Snort oder auf kommerzielle Lösungen verwiesen. IPP2P wird von der Firma *ipoque* [www-29] unterstützt, welche Hardware-Geräte zum Filtern von Peer-to-Peer und Instant Messaging Protokollen produziert.

Die Firewall-Lösung *Gibraltar* [www-16] der oberösterreichischen Firma eSys [www-30] verwendet IPP2P als Modul, eingebettet in die komfortable Benutzeroberfläche der Firewall (siehe Abb. 8.1).

Anwendung

Module werden bei Netfilter mithilfe des `-m` Schalters eingebunden. IPP2P lässt sich als Modul mit ein paar Zusatzschaltern einfach einbinden:

```
iptables -A FORWARD -m ipp2p --ipp2p -j DROP
```

Mit diesem Befehl werden alle von IPP2P erkannten Pakete verworfen, der Zusatzschalter `--ipp2p` bewirkt das Aktivieren aller Erkennungsfunktionen. Die Methoden können jedoch auch für die verschiedenen Protokolle einzeln aktiviert werden:

```
iptables -A FORWARD -m ipp2p --edk --kazaa --gnu --bit -j DROP
```

Dieser Befehl weist IPP2P an, die in dieser Arbeit besprochenen Tauschbörsen eDonkey, FastTrack (Kazaa), Gnutella und BitTorrent zu erkennen, diese Pakete werden anschließend von Netfilter verworfen. Tabelle 8.1 zeigt die von IPP2P, aktuell in der Version 0.8.1, erkennbaren Peer-to-Peer Protokolle und die dazugehörigen Schalter.

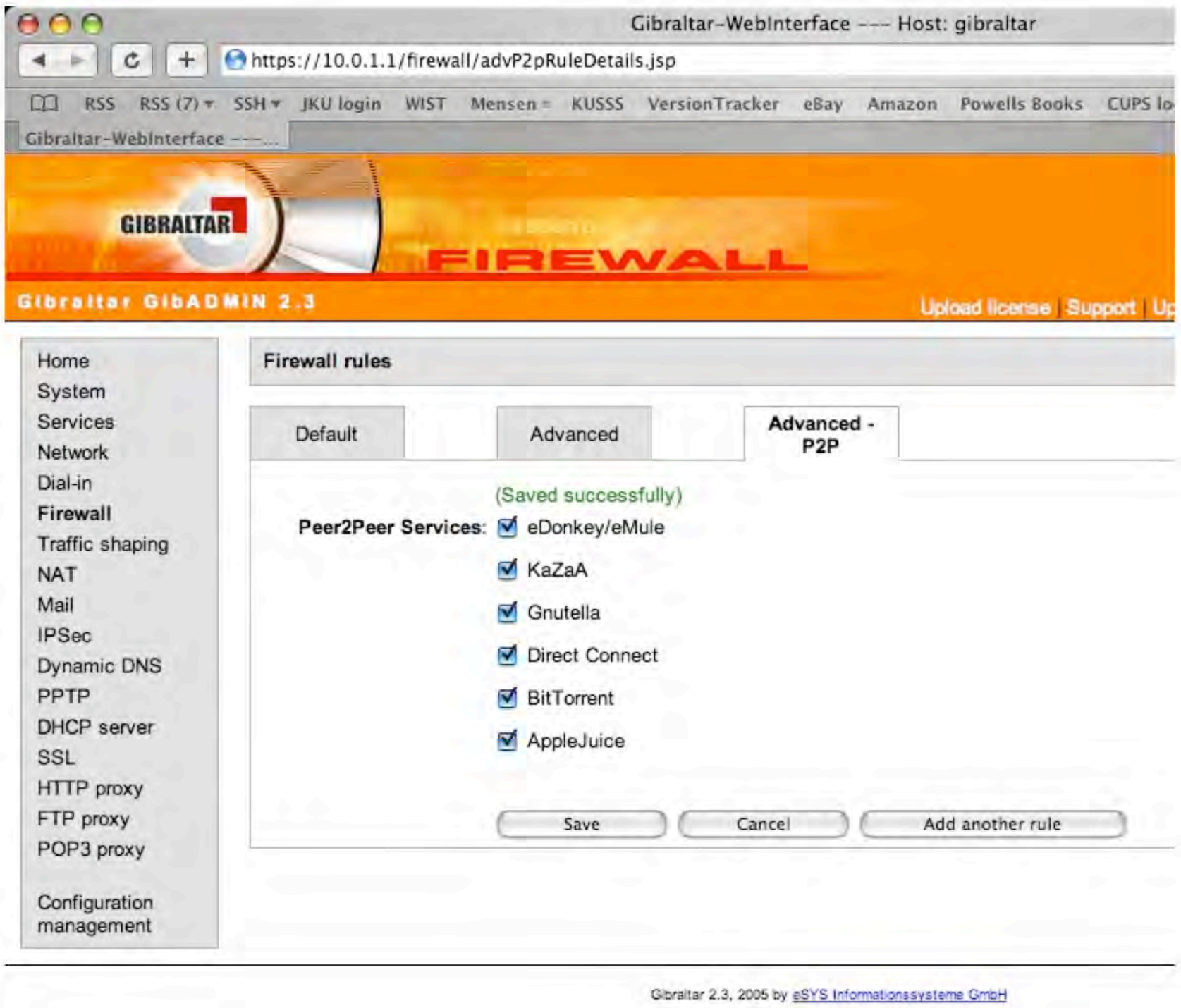


Abbildung 8.1: **Gibraltar Firewall: Verwendung des IPP2P-Moduls**

Funktionsweise

IPP2P ist darauf ausgelegt, die ersten Pakete einer Peer-to-Peer Verbindung zu erkennen. Soll Peer-to-Peer Netzverkehr einfach verboten werden, so ist das kein Problem: Die Pakete werden verworfen und die Verbindung kann nicht hergestellt werden.

In Bezug auf **Traffic Shaping** sieht das Ganze etwas anders aus: Hier ist es wichtig, dass bei einem Auftreten des Pakets die gesamte Verbindung markiert wird. In IPTables kann dies beispielsweise durch die Verwendung des CONNMARK Moduls herbeigeführt werden.

Tabelle 8.1: **IPP2P Version 0.8.1: Erkennbare Peer-to-Peer Netze mit Schaltern** [www-20]

P2P Netz	**Protokoll**	**Schalter**
Gnutella	TCP, UDP	`--gnu`
eDonkey, eMule	TCP, UDP	`--edk`
FastTrack (Kazaa)	TCP, UDP	`--kazaa`
BitTorrent	TCP, UDP	`--bit`
Direct Connect	TCP	`--dc`
AppleJuice	TCP	`--apple`
WinMX	TCP	`--winmx`
SoulSeek	TCP	`--soul`
Ares	TCP	`--ares`

Wie erkennt IPP2P nun, ob ein Paket von einer Peer-to-Peer Anwendung stammt? Die Methoden zur Erkennung wurden alle als C-Methoden im Quellcode implementiert, was es für einen Administrator umständlich gestaltet, diese Methoden abzuändern. Der Vorteil ist, dass die Software dadurch sehr schnell arbeitet.

Listing 8.1 zeigt als Beispiel eine Methode von IPP2P und zwar die Erkennungsmethode für Gnutella GET.

Listing 8.1: **IPP2P: Gnutella GET Erkennungsmethode**

```
1 /*check for gnutella get command*/
2 int
3 search_gnu (const unsigned char *payload, const u16 plen)
4 {
5     if ((payload[plen-2] == 0x0d) && (payload[plen-1] == 0
      x0a))
6     {
7         if (memcmp(payload, "GET /get/", 9) == 0)
              return ((IPP2P_DATA_GNU * 100) + 1);
```

```
8           if (memcmp(payload, "GET /uri-res/", 13) == 0)
                return ((IPP2P_DATA_GNU * 100) + 2);
9       }
10      return 0;
11  }
```

Die Tauschbörsen Gnutella, eDonkey, FastTrack, BitTorrent und Direct-Connect, mit welchen Tests durchgeführt wurden, wurden alle problemlos erkannt und konnten blockiert werden.

8.4.2 Layer7-filter

Auch bei Layer7-filter [www-31] handelt es sich, wie bei IPP2P, um ein Modul für Linux Netfilter. Mit Hilfe von Layer7-filter lässt sich die Anwendungsschicht jedes Paketes untersuchen und nach dem Protokoll klassifizieren. Unabhängig vom TCP- oder UDP-Port funktioniert dies auch über mehrere Pakete hinweg.

Anwendung

Da Layer7-filter ebenfalls ein Modul für Netfilter ist, lässt es sich wie jedes andere Modul mit dem `-m` Schalter in einem IPTables-Befehl verwenden:

```
iptables -A FORWARD -m layer7 --l7proto http -j DROP
```

Das obige Beispiel würde Pakete, welche das HTTP-Protokoll verwenden, erkennen und verwerfen. Hier wird deutlich, dass Layer7-filter nicht nur auf das Erkennen von Peer-to-Peer Protokollen ausgelegt ist, sondern dass theoretisch jedes Protokoll erkannt werden kann. Zwar ist es nicht Ziel von Layer7-filter, Dateien zu filtern, doch es ist theoretisch auch möglich, beispielsweise nach Windows .exe-Dateien oder JPEGs zu suchen.

Eine aktuelle Liste der erkennbaren Protokolle ist auf der Homepage des Projekts ersichtlich [www-32]. In Hinblick auf Peer-to-Peer und Instant Messaging sei auf Tabelle 8.2 verwiesen, in welche nur Protokolle aufgenommen wurden, die gut erkannt werden können.

Tabelle 8.2: **Layer7-filter: aktuell erkennbare Protokolle [www-32]**

Protokoll	Pattern-Name
Peer-to-Peer Filesharing Protokolle	
Gnutella	`gnutella`
BitTorrent	`bittorrent`
Direct Connect	`directconnect`
eDonkey 2000, eMule	`edonkey`
FastTrack (Kazaa)	`fasttrack`
AppleJuice	`applejuice`
Ares	`ares`
GnucleusLAN	`gnucleuslan`
Napster	`napster`
OpenFT	`openft`
Soribada	`soribada`
Soulseek	`soulseek`
Xunlei	`xunlei`
Instant Messaging Protokolle	
AIM - AOL Instant Messenger	`aim`
AIM Web Content (Werbung)	`aimwebcontent`
IRC (Internet Relay Chat)	`irc`
Jabber	`jabber`
MSN Messenger	`msnmessenger`
MSN Messenger - Dateitransfer	`msn-filetransfer`
Tencent QQ	`qq`
Yahoo! Messenger	`yahoo`

Funktionsweise

Layer-7 Filter benötigt für die Funktion sog. *Pattern* Dateien. Auf der Homepage können jederzeit die aktuellen offiziellen Patterns heruntergeladen werden. Diese sind in einem Ordner abgelegt (normalerweise im Verzeichnis `/etc/l7-protocols`) und können mit der oben beschriebenen Methode aufgerufen werden.

In den Patterns wird das Protokoll mit Hilfe von regulären Ausdrücken beschrieben, Layer-7 Filter verwendet sog. „V8 regexps“, die genaue Grammatik kann bei [www-33] nachgelesen werden. Das folgende Beispiel stammt aus der Pattern-Datei für Gnutella. Die Suche bezieht sich auf die erste Client- bzw. Servernachricht (wie diese Nachrichten aussehen, wird in Kapitel B.1 genau beschrieben).

```
^(gnutella connect/[012]\.[0-9]\x0d\x0a|get /uri-res/n2r\?urn
:sha1:|get /.*user-agent: (gtk-gnutella|bearshare|mactella|
gnucleus|gnotella|limewire|imesh)|get /.*content-type:
application/x-gnutella-packets|giv [0-9]*:[0-9a-f]*/|queue
[0-9a-f]*[1-9][0-9]?[0-9]?\.[1-9][0-9]?[0-9]?\.[1-9][0-9]?[0-9]
?\.[1-9][0-9]?[0-9]?:[1-9][0-9]?[0-9]?[0-9]?|gnutella.*
content-type: application/x-gnutella)
```

Layer-7 Filter konnte im Test bis auf Skype alle Peer-to-Peer und Instant Messaging Netze erkennen. Das Skype-Pattern musste deaktiviert werden, da es bei fast allen Paketen Alarm schlug.

8.4.3 Snort

Das Open Source Intrusion Detection System Snort [www-34] ist der de-facto Standard unter den Intrusion Detection Systemen. Um mit Hilfe von Snort Netzverkehr zu erkennen, welcher über die Firewall läuft, muss Snort entweder an einem Mirror Port, der zur Firewall geht, angeschlossen sein oder direkt auf der Firewall laufen.

Anwendung

Snort verwendet eine eigene Regelsyntax [Sno05] und hat schon sehr viele vorgefertigte Regeln standardmässig mit an Bord. Schlägt eine der Regeln an, so muss der Administrator definieren, was geschehen soll. Normalerweise wird ein Alarm ausgelöst, die Meldung in eine Logdatei geschrieben, eine E-mail versendet oder Ähnliches.

Snort läuft als eigener Prozess und bekommt die Pakete quasi in Kopie, ein gänzlich anderer Ansatz als bei den beiden vorigen Projekten. Eine Snort Regel für Gnutella würde folgende Form haben:

```
alert tcp $INTERNAL_NET 1024:65535 -> $EXTERNAL_NET any
(content:"GNUTELLA CONNECT/"; nocase; offset:0; flow:from
client; classtype:bad-p2p; msg:"Gnutella Connect to Server";)
```

Snort kann vom Administrator verwendet werden, um Netzverkehr zu erkennen. Es können standardmässig keine Pakete blockiert werden.

Der verwendete Regelsatz stammt von [www-18], welche sehr viele und gute Regeln für Snort anbietet. Die verwendeten Regeln wurden modifiziert und können in Anhang C ab Seite 119 nachvollzogen werden. Im Test erkannte Snort auch alle Pakete, für Skype existieren leider keine Regeln, da das Protokoll verschlüsselt ist. Allerdings kann Snort die Installation eines Skype-Clients und das Starten erkennen (siehe Regeln Seite 119).

8.4.4 Zusammenfassung

Die nachfolgende Tabelle 8.3 gibt Aufschluss darüber, welche Peer-to-Peer Netze von der getesteten Software zuverlässig erkannt wurde (in der Tabelle durch + gekennzeichnet) oder nicht erkannt wurde (durch – gekennzeichnet).

Tabelle 8.3: **Vergleich: Open Source Software zur Erkennung von P2P**

	IPP2P	**L7-filter**	**Snort**
Peer-to-Peer Tauschbörsen			
Gnutella	+	+	+
eDonkey/eMule	+	+	+
FastTrack (Kazaa)	+	+	+
BitTorrent	+	+	+
Direct Connect	+	+	+
Peer-to-Peer Tauschbörsen			
AIM/ICQ	–	+	+
MSN Messenger	–	+	+
Yahoo! Messenger	–	+	+
Skype	–	–	–

Kapitel 9

Fazit

Abschließend lässt sich sagen, dass die Sicherheitsprobleme und -risiken, welche durch die Peer-to-Peer Software verursacht werden, sowohl von den Benutzern dieser Software als auch von Sicherheitsverantwortlichen in Unternehmen oder Universitätsnetzen ernst genommen werden müssen.

Firmen sollten rechtzeitig reagieren und sich der Gefahren bewusst werden, bevor das Problem nur noch schwer kontrollierbar ist und, etwa durch das gesteigerte übertragene Datenvolumen, zusätzliche Kosten entstehen oder wichtige Ressourcen belegt werden. Ein wichtiger Schritt ist die Thematisierung des Problems in der Sicherheitspolitik des Unternehmens, ein weiterer die Durchsetzung dieser Sicherheitspolitik.

Die in dieser Arbeit getesteten, frei erhältlichen Softwareprodukte, die sich diesem Problem widmen, können zu einer Lösung beitragen. Man kann sagen, dass diese, zumindest für die Peer-to-Peer Netze der sog. „zweiten Generation“[1], durchwegs gute Ergebnisse lieferten, wobei die beanspruchte Rechenleistung im vertretbaren Rahmen blieb.

Leider versagten alle getesteten Produkte bei der sog. „dritten Generation“, deren einziger bisheriger Vertreter die Sprachtelefonie-Software Skype ist. Diese Generation von Peer-to-Peer Software verwendet sowohl für

[1] Als Peer-to-Peer Netze der zweiten Generation werden die Netze bezeichnet, die keine zentralen Server mehr verwenden.

die Daten als auch für die Kommunikation im Netz starke Verschlüsselung. Zusätzlich werden verschiedene Maßnahmen in Kombination verwendet, um über NAT-Geräte und Firewalls hinweg problemlos zu funktionieren. Es bleibt abzuwarten, welche Art von Software der dritten Generation noch veröffentlicht wird und wie die Reaktion der Hersteller von Sicherheitssoftware auf diese veränderte Situation ausfällt.

Berücksichtigen muss man auf alle Fälle auch die Möglichkeit des gewollten und produktiven Einsatzes von Peer-to-Peer Software in einem Unternehmensnetz. Abgesehen von den Lösungen, die speziell für diese Fälle entwickelt wurden, könnte die Verwendung von Software wie BitTorrent oder Skype für ein Unternehmen sehr interessant sein. Auf den Fall Skype wurde in dieser Arbeit eingegangen und es hat sich gezeigt, dass es möglich ist, Peer-to-Peer Software effizient und doch möglichst sicher zu nutzen.

Literaturverzeichnis

[Bar01] Barkai, David: *Peer-to-Peer Computing. Technologies for Sharing and Collaboration on the Net.* Intel Press, 2001.

[BD06] Biondi, Philippe and Fabrice Desclaux: *Silver needle in the skype.* Technical report, EADS Corporate Research Center – DCR/STI/C, IT sec Lab, Suresnes, France, March 2006.

[Ber01] Berg, Al: *P2p, or not p2p?* Information Security, February 2001.

[BS04] Baset, Salman A. and Henning G. Schulzrinne: *An analysis of the skype peer-to-peer internet telephony protocol.* Technical report, Department of Computer Science, Columbia University, New York, New York, December 2004.

[Cla01] Clark, David: *Face-to-face with peer-to-peer networking.* IEEE Computer, 34(1):18–21, January 2001.

[Cou02] Couch, William: *Peer-to-peer file-sharing networks: Security risks.* Technical report, The SANS Institute, 2002.

[Dam02] Damker, Herbert: *Sicherheitsaspekte von P2P Anwendungen in Unternehmen*, Seiten 209–228. In: Schoder, Detlef et al. [SFT02], 2002.

[Fra02] Frase, Dan: *The instant messaging menace: Security problems in the enterprise and some solutions.* Technical report, The SANS Institute, 2002.

[HAS02] Hess, Thomas, Markus Anding und Matthias Schreiber: *Napster in der Videobranche? Erste Überlegungen zu Peer-to-Peer Anwendungen für Videoinhalte*, Seiten 25–40. In: Schoder, Detlef et al. [SFT02], 2002.

[Hes04] Hess, Thomas: *Musiktauschbörsen.* Informatik-Spektrum, 27(3):273–275, Juni 2004.

[Hon01] Hong, Theodore: *Performance*, chapter 14, pages 203–241. In Oram, Andy [Ora01], 2001.

[KB05] Kulbak, Yoram and Danny Bickson: *The emule protocol specification.* Technical report, DANSS Lab, The Hebrew University of Jerusalem, Jerusalem, January 2005.

[Lai04] Lai, Wayne: *Managing peer-to-peer applications in dormitory networks.* Technical report, The SANS Institute, 2004.

[Log06] Logic, IM: *2005 real-time communication security: The year in review.* Technical report, IM Logic Threat Center, 2006.

[Mil01] Miller, Michael: *Discovering P2P.* Sybex, San Francisco, 2001.

[MRI02] Matei Ripeanu, Ian foster and Adriana Iamnitchi: *Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design.* IEEE Internet Computing, 6, 2002.

[MS02] Mitnick, Kevin and William Simon: *The Art of Deception: Controlling the Human Element of Security.* John Wiley & Sons, 2002.

[NLW03] Nathaniel Leibowitz, Matei Ripeanu and Adam Wierzbicki: *Deconstructing the kazaa network.* In *3rd IEEE Workshop on Internet Applications (WIAPP'03)*, San Jose CA, June 2003.

[Ora01] Oram, Andy (editor): *Peer-to-Peer: Harnessing the Benefits of a Disruptive Technology.* O'Reilly, 2001.

[PBE+05] Piccard, Paul L., Brian Baskin, Craig Edwards, George Spillman, and Marcus H. Sachs: *Securing IM and P2P Applications for the Enterprise.* Syngress Publishing, 2005.

[Rip01] Ripeanu, Matei: *Peer-to-peer architecture case study: Gnutella network.* In *1st IEEE International Conference on Peer-to-peer Computing (P2P2001)*, Linkoping Schweden, August 2001.

[RSA78] Rivest, Ronald, Adi Shamir, and Len Adleman: *A method for obtaining digital signatures and public-key cryptosystems.* Communications of the ACM, 21(2):120–126, 1978.

[Rö03] Röttgers, Janko: *Mix, Burn & R.I.P. – Das Ende der Musikindustrie.* Verlag Heinz Heise, Hannover, 2003.

[SF02] Schoder, Detlef und Kai Fischbach: *Peer-to-Peer. Anwendungsbereiche und Herausforderungen*, Seiten 3–21. In: Schoder, Detlef et al. [SFT02], 2002.

[SFT02] Schoder, Detlef, Kai Fischbach und René Teichmann (Herausgeber): *Peer-to-Peer. Ökonomische, technologische und juristische Perspektiven.* Springer-Verlag, Berlin Heidelberg New York, 2002.

[SGG02] Saroiu, Stefan, P. Krishna Gummadi, and Steven D Gribble: *A measurement study of peer-to-peer file sharing systems.* In *Proceedings of Multimedia Computing and Networking*, 2002.

[Sno05] The Snort Project: *Snort Users Manual 2.4.0*, August 2005.

[SW02] Sen, Subhabrata and Jia Wang: *Analyzing peer-to-peer traffic across large networks.* In *Second Annual ACM Internet Measurement Workshop*, November 2002.

[SW04] Steinmetz, Ralf und Klaus Wehrle: *Peer-to-Peer-Networking & -Computing.* Informatik-Spektrum, 27(1):51–54, Februar 2004.

[Zim80] Zimmermann, Hubert: *Osi reference model - the iso model of architecture for open systems interconnection.* IEEE Transactions, 28(4):425–432, April 1980.

Internetquellen

[fas] *FastTrack.* URL: http://de.wikipedia.org/wiki/Fasttrack . [Abruf: 6.8.2006].

[Fis06] Fisk, Adam A.: *Dynamic query protocol.* URL: http://www.the-gdf.org/index.php?title=Dynamic_Querying , March 2006. [Abruf: 8.8.2006].

[Gar05] Garfinkel, Simon L.: *Voip and skype security.* URL: www.simson.net/ref/2005/OSI_Skype6.pdf , January 2005. [Abruf: 20.8.2006].

[Gas03] Gasior, Geoff: *Icarus blocks p2p on campus network.* URL: http://techreport.com/onearticle.x/5727 , October 2003. [Abruf: 8.5.2006].

[gnu] *Gnutella.* URL: http://de.wikipedia.org/wiki/Gnutella . [Abruf: 6.8.2006].

[Har04] Hargreaves, Tom: *The fasttrack protocol.* URL: http://cvs.berlios.de/cgi-bin/viewcvs.cgi/*checkout*/gift-fasttrack/giFT-FastTrack/PROTOCOL , July 2004. [Abruf: 8.8.2006].

[Hei06] Heidrich, Joerg: *Rechtliche Konsequenzen der eDonkey-Razzia.* URL: http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/73525 , Mai 2006. [Abruf: 8.8.2006].

[Hur02] Hurwicz, Michael: *Emerging technology: Peer-to-peer networking security.* URL: http://www.itarchitectmag.com/shared/article/showArticle.jhtml=articleId=8703302 , June 2002. [Abruf: 7.5.2006].

[JK03] Jessup, Troy and Pete Kruckenberg: *The kazaa problem.* URL: http://www.abaju.com.br/files/the_kazaa_problem.ppt , September 2003. [Abruf: 21.4.2006].

[Jon01] Jones, Paul: *Rfc 3174: Us secure hash algorithm 1 (sha1).* URL: http://www.ietf.org/rfc/rfc3174.txt , September 2001. [Abruf: 8.8.2006].

[Kir03] Kirk, Patrick: *Rfc-gnutella 0.6.* URL: http://rfc-gnutella.sourceforge.net/developer/testing/index.html , July 2003. [Abruf: 8.4.2006].

[Kle06] Kleinz, Torsten: *Ermittler hatten Zugriff auf eDonkey-Server.* URL: http://www.heise.de/newsticker/result.xhtml?url=/newsticker/meldung/73446 , Mai 2006. [Abruf: 8.8.2006].

[Met] MetaMachine: *About mftp.* URL: http://www.edonkey2000.com/documentation/mftp.html . [Abruf: 8.8. 2006].

[msn03] *Msn messenger protocol.* URL: http://hypothetic.org/docs/msn/ , March 2003. [Abruf: 10.8.2006].

[Ora00] Oram, Andy: *Peer-to-peer makes the internet interesting again.* URL: http://www.linuxdevcenter.com/pub/a/linux/2000/09/22/p2psummit.html , September 2000. [Abruf: 7.3.2006].

[Riv92] Rivest, Ronald: *Rft 1320: The md4 message-digest algorithm.* URL: http://www.ietf.org/rfc/rfc1320.txt , April 1992. [Abruf:8.8.2006].

[Sch01] Schneider, Jeff: *Convergence of peer and web services.* URL: http://www.openp2p.com/pub/a/p2p/2001/07/20/convergence.html , July 2001. [Abruf: 7.3.2006].

WWW-Verweise

[www-1] Napster-Homepage (Musik-Downloads): http://www.napster.com

[www-2] Homepage des SETI@Home Projekts der Universität Berkeley: http://setiathome.berkeley.edu/

[www-3] ICQ-Homepage (Instant Messenger): http://www.icq.com

[www-4] Skype-Homepage (P2P Sprachtelefonie): http://skype.com

[www-5] Gnutella-Homepage (P2P Filesharing): http://www.gnutella.com

[www-6] Kazaa-Homepage (P2P Fileharing): http://www.kazaa.com

[www-7] BitTorrent-Homepage (P2P Filesharing): http://www.bittorrent.com

[www-8] Freenet-Homepage (P2P Filesharing und Publishing): http://freenet.sourceforge.net

[www-9] Boinc - Verteiltes Rechnen: http://www.boinc.de/

[www-10] Groove Networks (P2P Groupware): http://www.groove.net

[www-11] Microsoft Netmeeting Homepage (P2P Groupware): http://www.microsoft.com/windows/netmeeting

[www-12] Security Focus Vulnerability Datenbank: http://www.securityfocus.com/vulnerabilities

[www-13] Limewire Homepage (P2P Filesharing): http://www.limewire.com

[www-14] Limewire Frequently Asked Questions: http://limewire.org/wiki/index.php?title=Frequently_Asked_Questions

[www-15] Gerichtsprozess RIAA gegen Napster auf furia.com: http://www.furia.com/page.cgi?type=misc&id=Napster

[www-16] Gerichtsprozess MGM Studios gegen Grokster: http://www.eff.org/IP/P2P/MGM_v_Grokster/04-480.pdf

[www-17] Gibraltar Firewall: http://www.gibraltar.at/

[www-18] Multi Router Traffic Grapher: http://mrtg.hdl.com

[www-19] Bleeding Edge Threats: http://www.bleedingsnort.com

[www-20] Netfilter Homepage: http://www.netfilter.org

[www-21] IPP2P-Modul Homepage: http://www.ipp2p.org

[www-22] Netfilter Connmark Modul Documentation: http://home.regit.org/?page_id=7

[www-23] Slyck File Sharing News & Info Site: http://www.slyck.com

[www-24] Overnet Homepage (P2P Filesharing): http://www.overnet.org/

[www-25] Redhat Linux Homepage: http://www.redhat.com

[www-26] Blizzard Entertainment: http://www.blizzard.com

[www-27] NIST AES Information: http://csrc.nist.gov/CryptoToolkit/aes/

[www-28] Old Versions IM Page: http://www.meetingbywire.com/OldVersions.htm

[www-29] NAT Check Utility: http://midcom-p2p.sourceforge.net

[www-30] ipoque GmbH Homepage: http://www.ipoque.com

[www-31] eSys Informationssysteme GmbH Homepage: http://www.esys.at

[www-32] Layer 7 Filter Modul: http://l7-filter.sourceforge.net/

[www-33] Layer 7 Filter Modul - unterstützte Protokolle: http://l7-filter.sourceforge.net/protocols

[www-34] Syntax für Reguläre Ausdrücke (Regular Expressions V8): http://www.hmug.org/man/3/regsub.html

[www-35] Snort IDS Homepage: http://www.snorg.org

[www-36] BitTorrent Protocol (Documentation for Developers): http://www.bittorrent.org/protocol.html

Alle Weblinks wurden am 10.10.2006 auf Korrektheit überprüft.

Anhang A

Standardports von Peer-to-Peer Tauschbörsen und Instant Messengern

Um Administratoren und auch Anwendern von Peer-to-Peer Software die Identifikation der Peer-to-Peer Protokolle etwas zu erleichtern, wurden in der nachfolgenden Tabelle alle Ports, die diese Anwendungen standardmässig verwenden, nach der Portnummer sortiert zusammengetragen.

Hinweis: Es handelt sich hierbei um die von den Anwendungen verwendeten **Standardports**! Fast alle Peer-to-Peer Anwendungen funktionieren über jeden beliebigen Port (z.B. 80, 443)!

Tabelle A.1: **Standardports von Peer-to-Peer Protokollen**

Protokoll		**Port**
FastTrack (Kazaa)	TCP,UDP	1214
Skype	TCP,UDP	1387
Direct Connect	TCP,UDP	1412
MSN Messenger Application Sharing		1503
Fortsetzung auf nächster Seite		

Tabelle A.1 – Fortsetzung von voriger Seite

Protokoll		Port
MSN Messenger Game Audio		1838
MSN Messenger Sign-In		1863
BitTorrent alternative Ports	TCP	3881:3889
eDonkey 2000, eMule	TCP	4662
eMule	UDP	4662
MSN Messenger Video/Webcam		5004:65535
MSN Messenger Audio Chat		5004:65535
Yahoo! Messenger	TCP,UDP	5050
AIM / ICQ	TCP,UDP	5190
Gnutella	TCP,UDP	6346:6347
BitTorrent Dateitransfer	TCP	6881:6889
MSN Messenger Dateitransfer	TCP,UDP	6891:6900
MSN Messenger VoIP	TCP,UDP	6901
BitTorrent Tracker	TCP	6969
eDonkey 2000	UDP	12980

Anhang B

Protokoll-Analyse

B.1 Gnutella

Die nachfolgende Analyse des Gnutella-Protokolls stammt einerseits aus [Kir03] und wurde teilweise durch Mithören des Netzverkehrs herausgefiltert.

B.1.1 Handshake [Kir03]

1. Der Client baut eine TCP-Verbindung mit dem Server auf.
2. Der Client sendet `GNUTELLA CONNECT/0.6<cr><lf>`.
3. Der Client sendet Protokoll-spezifische Header, durch `<cr><lf>` getrennt, mit einem extra `<cr><lf>` am Ende.
4. Der Server antwortet mit `GNUTELLA/0.6 200 OK<cr><lf>`.
5. Der Server sendet seine Header, gleiches Format wie unter 3.
6. Der Client sendet `GNUTELLA/0.6 200 OK<cr><lf>` wenn er nach dem Parsen der Server-Header noch eine Verbindung aufbauen möchte, wenn nicht, antwortet er mit einem Fehlercode und schließt die Verbindung.
7. Der Client sendet die Hersteller-spezifischen Header, gleiches Format wie unter 3.

8. Client und Server können nun Binärnachrichten mit den aus 3. und 5. gewonnen Informationen austauschen.

B.1.2 Suchanfragen (Queries) [PBE+05]

Für alle administrativen Funktionen im Gnutella-Netz werden sogenannte *Descriptor Pakete* verwendet. Das Protokoll ist UDP. Der Header der Pakete ist 23 Bytes lang und setzt sich wie folgt zusammen:

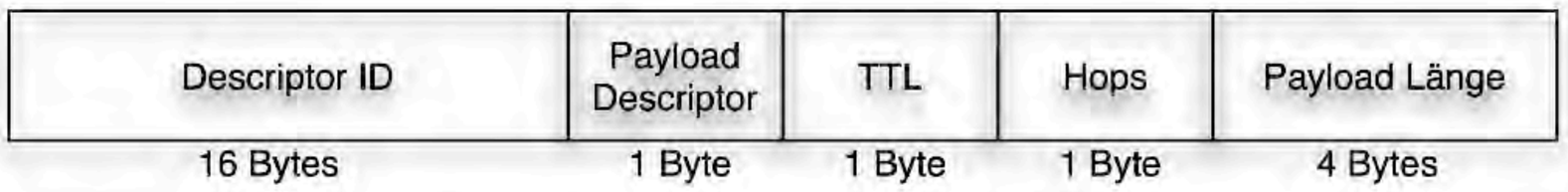


Abbildung B.1: **Descriptor Paket Header (23 Bytes)**

- Die *Descriptor ID* dient der eindeutigen Identifizierung des Pakets im Netz.
- Der *Payload Descriptor*, gibt den Pakettyp des Payloads an.
- Die *TTL* wird bei jedem Hop über einen Servent verringert, bei TTL=0 wird das Paket verworfen.
- Die *Hops* geben die Anzahl der Rechner an, die das Paket bereits relayed haben.

Der Payload Descriptor kann folgende (Hex-)Werte annehmen:

00 Ping, dient der Überprüfung, ob ein Rechner online ist

01 Pong, Antwort auf Ping-Paket

80 Query

81 QueryHits

40 Push, wird an Peers hinter Firewalls gesendet, um die Dateien trotz Firewall übertragen zu können

Query Pakete

Ein Query-Paket enthält ein Byte im Payload, welches die **minimale Verbindungsgeschwindigkeit** (in Kilobit pro Sekunde) eines Peers, von dem heruntergeladen werden soll, angibt. Weiters ist der **Suchstring** mit NULL (00) beginnend und endend, enthalten. Optional können noch **zusätzliche Query-Daten** angegeben werden, beispielsweise ein Hash-String zur Identifikation.

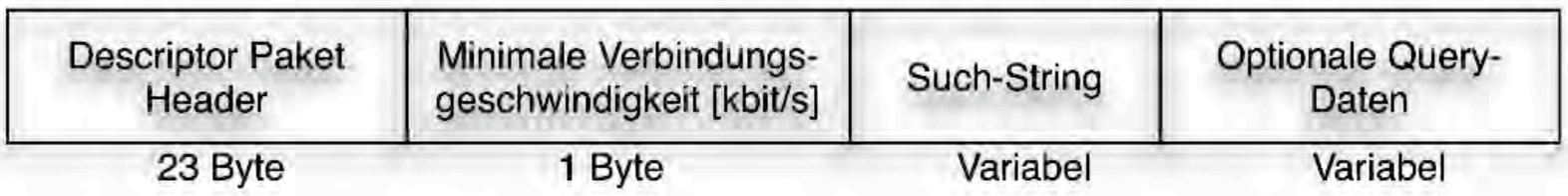


Abbildung B.3: **Query Descriptor Paket**

QueryHits Pakete

QueryHits Pakete werden als Antwort auf Query-Pakete gesendet, wenn der Servent einen oder mehrere Treffer liefern kann.

Abbildung B.3: **QueryHits Descriptor Paket**

Im Payload ist enthalten, **wie viele Treffer** der Servent liefert, welche **IP-Adresse** er besitzt und auf welchem **Port** er Anfragen erwartet.

Ausserdem wird die **Verbindungsgeschwindigkeit** angegeben. Die **Treffermenge** wird unten beschrieben. Zusätzlich wird noch eine 16 Byte lange **Servent-ID** geliefert, welche den Servent im Gnutella-Netz eindeutig identifiziert. Die **optionalen erweiterten Daten** dienen für eventuelle spätere Protokollerweiterungen.

Die Treffermenge (das QueryHits Result Set) hat folgendes Format:

Datei Index-Nummer	Dateigrösse [Byte]	Dateiname	Optionale Daten
4 Byte	4 Byte	Variabel	Variabel

Abbildung B.4: **QueryHits Results Set Format (Treffermenge)**

Der Dateiname ist ein mit NULL terminierter String, genauso die optionalen Daten, bei denen es sich meist um Hashwerte handelt. Enthält ein Paket mehrere Treffer, so sind mehrere Einträge der gleichen Art vorhanden.

B.1.3 Dateitransfer [Kir03]

Dateitransfers werden bei Gnutella über HTTP abgewickelt, die IP-Adresse und den Port des jeweiligen Servents kennt der Client bereits aus dem QueryHits Paket, genauso wie den Dateiindex und den Dateinamen.

Dateitransfer

Kennt der Client mehrere Quellen mit der gleichen Datei und möchte die Datei gleichzeitig von verschiedenen Servents laden, dann funktioniert dies über eine Aufteilung des *Range*-Feldes. Die Anforderung für einen Dateitransfer hat die nachstehende Form.

```
GET /get/<File Index>/<File Name> HTTP/1.1<cr><lf>
User-Agent: Gnutella<cr><lf>
```

```
Host: a.b.c.d:6346<cr><lf>
Connection: Keep-Alive<cr><lf>
Range: bytes=0-<cr><lf>
<cr><lf>
```

Die Antwort auf eine Anfrage hat folgende Form:

```
HTTP/1.1 200 OK<cr><lf>
    Server: Gnutella<cr><lf>
    Content-type: application/binary<cr><lf>
    Content-length: 1234567890<cr><lf>
    <cr><lf>
```

Hernach folgen die zu übertragenden Daten.

B.2 Bit Torrent

Die nachfolgende Analyse des BitTorrent-Protokolls und die Auflistung der verwendeten Nachrichten findet sich bei [www-35].

B.2.1 Bencoding

Bit Torrent verwendet für die Codierung der Metadaten ein Format namens Bencode, die Codierung wird wie folgt vorgenommen:

- **Strings** werden mit der Länge (zur Basis 10) im Präfix dargestellt. Bsp: `3:fim`

- **Integer** werden durch das Zeichen 'i', gefolgt von der Zahl (zur Basis 10) und dem Zeichen 'e' dargestellt. Bsp: `i99e` entspricht der Zahl 99, `i-3e` entspricht -3.

  `i-0e` ist ungültig, Voranstellungen der 0 sind ebenfalls verboten (i0e ist gültig). Es gibt für Integer keine Größenbeschränkung.

- **Listen** werden durch das Zeichen 'l', gefolgt von den durch ':' getrennten Listenelementen (in Bencode) dargestellt und mit 'e' abgeschlossen. Bsp: `15:apple:4peare` beschreibt die Liste ['apple','pear']

- **Dictionaries** werden durch 'd' gefolgt von den Schlüsseln und deren Werten dargestellt und mit 'e' abgeschlossen.

  Bsp: `d5:apple3:pie4:pear6:brandye` entspricht {'apple':'pie', 'pear':'brandy'}

B.2.2 Dictionary für Torrent-Dateien

Die nachfolgenden Schlüssel können in einer Torrent-Datei verwendet werden.

announce Tracker-URL

info Dieses Dictionary enthält folgende Schlüssel:

name String mit dem vorgeschlagenen Dateinamen bzw. Verzeichnis.

piece length Gibt die Größe der einzelnen Dateiteile in Byte an.

pieces Jeweils 20 Zeichen dieses Strings bilden den SHA1-Hash eines Dateiteiles.

length Gibt die Länge der gesamten Datei in Bytes an (wenn nur eine Datei vorhanden ist).

files Sind mehrere Dateien vorhanden, so findet sich in diesem String für jede Datei die Schlüssel `length` (Dateigröße in Byte) und `path` (Verzeichnis und Dateiname).

B.2.3 GET Anfragen an den Tracker

GET Anfragen an den Tracker verwenden folgende Schlüssel:

info_hash Der 20 Byte Hash des Teils der Datei (in der Torrent-Datei im Schlüssel `pieces` codiert).

peer_id Zufälliger String der Länge 20, welche der Downloader als Identifikation verwendet.

ip IP-Adresse oder DNS-Name des Downloaders. Optional.

port TCP Port-Nummer, auf welchem der Peer auf Anfragen wartet (Standard: 6881).

uploaded Bisher sbsolute Upload-Menge (Ascii, Basis 10).

downloaded Bisher absolute Download-Menge (Ascii, Basis 10).

left Anzahl der Bytes, die vom Peer noch heruntergeladen werden muss (Ascii, Basis 10).

event Wurde der Download gerade gestartet, dann ist `event` auf `started` gesetzt, ist der Download beendet, dann ist der Wert `completed`. Beim Abbruch wird der Wert auf `stopped` gesetzt.

B.2.4 Tracker-Antworten

Die Antworten des Trackers sind ebenfalls Dictionaries in Bencode.

failure reason Der Schlüssel `failure reason` enthält einen String mit einer Fehlermeldung. Ist dieser Schlüssel vorhanden, so müssen keine anderen Schlüssel vorhanden sein. Andernfalls müssen die nachfolgenden beiden Schlüssel vorhanden sein.

interval Gibt die Anzahl der Sekunden, die zwischen den Anfragen gewartet werden muss, an.

peers Eine Liste von Dictionaries, die folgende Einträge enthalten:

peer id Selbstgewählte ID des Peers.

ip IP-Adresse oder DNS-Name des Peers.

port Port-Nummer, auf der der Peer die Antwort erwartet.

B.2.5 Verbindungsaufbau

Das Protokoll arbeitet über TCP. Der Handshake läuft wie folgt ab:

- Die Nachricht beginnt mit '19' gefolgt von 'BitTorrent protocol'.
- Nach diesem Header kommen 8 reservierte Bytes (alle auf 0 gesetzt), gefolgt vom 20 Byte SHA1-Hash aus der Metainfo Datei des Trackers (der `info_hash`-Schlüssel). Stimmt dieser Wert nicht auf beiden Seiten überein, so wird die Verbindung getrennt.
- Nach dem Download-Hash kommt die 20 Byte lange Peer-ID.

B.2.6 Nachrichtenaustausch

Die Nachrichten können von beiden Seiten gleichberechtigt ausgetauscht werden. Alle Nachrichten, welche auf den Handshake folgen, enthalten als Präfix die Länge der Nachricht, gefolgt von der Nachricht selbst. Nachrichten mit Länge 0 sind Keep-Alive Nachrichten, welche normalerweise alle 2 Minuten gesendet.

Alle anderen Nachrichten beginnen mit einem Byte, welches den Typ der Nachricht bestimmt:

- 0 - **choke**
- 1 - **unchoke**

  Die `choke` und `unchoke` Nachrichten haben keinen Payload und werden verwendet, um einen Upload zu drosseln (weil z.B. die Anzahl der maximal erlaubten Verbindungen erreicht ist).

- 2 - **interested**
- 3 - **not interested**

  `interested` und `not interested` haben ebenfalls keinen Payload und werden vom Downloader verwendet, um den Download zu drosseln.

- 4 - **have**

  Enthält eine Nummer, und zwar den Index des Teils, welcher gerade heruntergeladen wurde (und dessen Hashcode überprüft wurde).

- 5 - **bitfield**

  Die `bitfield` Nachricht wird als erste Nachricht gesendet und enthält einen Array von Bits, welche die Teile der Datei indizieren, welche der Downloader bereits besitzt ('1' → Teil schon heruntergeladen, '0' → Teil noch nicht heruntergeladen).

- 6 - **request**

  Gibt den Index des Startteils und die Anzahl der nachfolgenden Dateiteile, welche heruntergeladen werden sollen, an.

- 7 - **piece**

  `piece`-Nachrichten enthalten die Teile der Datei. Die Nachricht enthält einen Index und die Daten.

- 8 - **cancel**

  `cancel`-Nachrichten haben den gleichen Payload wie `request`-Nachrichten und geben an, dass ein Request abgebrochen werden soll (weil z.B. der Teil schon von einem anderen Peer heruntergeladen wurde).

B.3 FastTrack

Das Fast Track Protokoll ist ein geschlossenes, verschlüsseltes Protokoll. [Har04] konnte Teile des Protokolls mittels Reverse-Engineering Techniken entschlüsseln, ein grosser Teil ist jedoch noch immer weitgehend unbekannt. Die hier angegebenen Pakete entstammen seiner Dokumentation unter [Har04] und [PBE+05]. In dieser Beschreibung wurden lediglich bereits entschlüsselte Pakettypen, welche für die Verbindung und die Dateisuche und -übertragung wichtig sind.

B.3.1 UDP-Pakete

Node ping

Art der Nachricht (0x27)	Art der Verschlüsselung	unbekannt (0x80)	Netzwerk-Name
1 Byte	4 Byte	1 Byte	Variabel

Abbildung B.5: **FastTrack Paket: Node Ping**

Die Art der Nachricht ist bei einem *Node ping* Paket immer `0x27`. Der Netzwerkname wird mit dem NULL-String (0x00) terminiert.

Node ping Pakete werden von Hosts vor der Herstellung einer Verbindung zum Supernode gesendet.

Node pong

Art der Nachricht (0x28)	Art der Ver-schlüsselung	unbekannt (0x00)	unbekannt	Load	unbekannt	Netzwerk-Name
1 Byte	4 Byte	1 Byte	3 Byte	1 Byte	1 Byte	Variabel

Abbildung B.6: **FastTrack Paket: Node Pong**

Die meisten Werte des Node pong-Pakets sind unbekannt, Load gibt die Last des Supernodes in Prozent an. [Har04] geht davon aus, dass es sich um die freie Kapazität des Supernodes handelt. Der Netzwerkname wird wieder mit 0x00 terminiert.

Der Supernode versendet das Node pong Paket als Antwort auf das Node ping eines Clients.

B.3.2 TCP-Pakete

Supernode list

Art der Nachricht (0x00)	IPv4 address	Port	last seen	Load
1 Byte	4 Byte	2 Byte	1 Byte	1 Byte

Abbildung B.7: **FastTrack Paket: Supernode List**

Diese Struktur wird innerhalb einer Nachricht mehrmals wiederholt, es werden damit aktive Supernodes mit ihren IP-Adressen und Portnummern vom Supernode zum Client übertragen.

User information

Art der Nachricht (0x02)	IPv4 address	Port	Bandbreite	unbekannt	Benutzer-name
1 Byte	4 Byte	2 Byte	1 Byte	1 Byte	Variabel

Abbildung B.8: **FastTrack Paket: User information**

Dieses Paket wird vom Client zum Supernode übertragen und enthält die IP-Adresse, den Port und den Benutzernamen. Die Bandbreite wird logarithmisch übertragen, wobei `0xd1` unendlich (momentan 1680 kbps) bedeutet. Der Wert wird folgendermassen berechnet: $14 * log_2(x) + 59$, wobei x der Übertragungsrate in kbps entspricht.

Unshare file

Enthält den Hashwert, ID-Nummer und Grösse einer Datei. Wird zum Supernode gesendet um eine Datei aus dem Suchindex zu löschen.

Art der Nachricht (0x05)	Hashwert der Datei	Datei ID (Integer)	Dateigrösse (Integer)	Anzahl der Tags (Integer)
1 Byte	20 Byte	Variabel	Variabel	Variabel

Abbildung B.9: **FastTrack Paket: Unshare file**

Search query

Art der Nachricht (0x06)	unbekannt	max. Trefferanzahl	Such Id	unbekannt (0x01)	realm	Anzahl Suchtherme
1 Byte	2 Byte	2 Byte	2 Byte	1 Byte	1 Byte	1 Byte

Abbildung B.10: **FastTrack Paket: Search query**

Die Such Id ist eine eindeutige Nummer und dient der Identifikation der Anfrage. Realm definiert den gesuchten Dateitypen und kann folgende Werte annehmen:

`0xA1`	Audio
`0xA2`	Video
`0xA3`	Images
`0xA4`	Documents
`0xA5`	Software
`0xBF`	Everything

Die Suchtherme werden wie folgt kodiert:

Die Art des Vergleichs kann folgende Werte annehmen:

Vergleichs-art	unbekannt	Suchlänge (Integer)	Suche
1 Byte	Variabel	Variabel	Variabel

Abbildung B.11: **FastTrack Paket: Search term**

0x00	genau
0x02	höchstens
0x03	suche „ungefähr“
0x04	suche zumindest
0x05	suche Teilstring
0x06	unbekannt

Search reply

Art der Nachricht (0x07)	Supernode IPv4 Adresse	Supernode Port	Such Id	Anzahl Treffer
1 Byte	4 Byte	2 Byte	2 Byte	2 Byte

Abbildung B.12: **FastTrack Paket: Search reply**

Dieses Paket enthält die Suchergebnisse eines einzelnen Supernodes. Die einzelnen Treffer werden wie folgt kodiert:

IPv4 Adresse	Port	Bandbreite	Benutzer und Netzwerkname	Dateihash	Datei Id	Datei-grösse	Anzahl Tags
4 Byte	2 Byte	1 Byte	Variabel	20 Byte	Variabel	Variabel	Variabel

Abbildung B.13: **FastTrack Paket: Search result**

B.3.3 Dateitransfer

Dateitransfers können bei Fast Track auf zwei Arten angefordert werden [Har04];

Hash-basiert `GET /.hash=<20 Byte Hash> HTTP/1.1`

Dateinamen-basiert `GET /13609/filename.ext HTTP/1.1`

Für die Übertragung der Daten wird das http-Protokoll verwendet, der angefragte Client kann entweder mit einer `503 Service Unavailable` Nachricht antworten, nach der abgebrochen wird, oder mit einer `HTTP/1.1 200 OK` Nachricht, welche die Daten enthält.

Auch bei FastTrack ist es möglich, eine Datei von mehreren Clients gleichzeitig herunterzuladen. Ein Header würde in diesem Fall in etwa so aussehen [PBE+05]:

```
HTTP/1.1 206 Partial Content
Content-Range: bytes 2475125-2593614/3062673
Content-Length: 118490
Accept-Ranges: bytes
Date: Fri, 07 Oct 2005 22:51:15 GMT
...
X-Kazaa-Username: anonymous_user
X-Kazaa-IP: 123.123.123.123:1234
X-KazaaTag: 5=192
X-KazaaTag: 4=Title of Song
```

Im http-Header werden mit speziellen, sog. **X-KazaaTags** Informationen über die Datei und den Benutzer mitgesendet. Diese enthalten Metadaten über die Datei wie Sprache, Genre oder Qualität für Musikdateien. Suchroutinen von Firewalls verlassen sich manchmal auf diese Tags und klassifizieren nach ihnen. In Tabelle B.1 sind die wichtigsten Tags aufgelistet.

Tabelle B.1: **X-KazaaTags** (Quelle: [PBE+05])

Tag ID	Datentyp	Beschreibung
1	4-byte Integer	Jahr der Veröffentlichung
3	Hexadecimal	Hashwert
4	ASCII String	Audio Titel
5	4-byte Integer	Dauer in Sekunden
6	ASCII String	Name des Künstlers
8	ASCII String	Name des Albums
10	ASCII String	Sprache
12	ASCII String	Schlüsselwörter
13	2 4-byte Integer	Auflösung (für Videodateien)
14	ASCII String	Genre
16	ASCII String	Betriebssystem (für Software)
17	4-byte Integer	Bit-Tiefe
18	ASCII String	Dateityp
21	4-byte Integer	Qualität
24	ASCII String	Version
25	4-byte Integer	unbekannt
26	ASCII String	Dateibeschreibung
28	ASCII String	Codec
29	4-byte Integer	Bewertung
33	4-byte Integer	Grösse
37	unbekannt	unbekannt
45	unbekannt	unbekannt
53	unbekannt	unbekannt
65539	ASCII String	Hashwert der Datei

Anhang C

Snort Regeln

Die verwendeten Regeldateien stammen von der Homepage [www-18] und wurden für die Tests angepasst, d.h. es wurden nur die Regeln für die in dieser Diplomarbeit behandelten Instant Messaging und Tauschbösen-Netzwerke übernommen.

C.1 Peer-to-Peer Filesharing

```
1 #
2 # $Id: bleeding-p2p.rules,v 1.976 2006/08/25 12:38:04
    jonkman Exp $
3 # Bleeding Snort P2P rules.
4 #
5 # SID's are 2000000+ to avoid conflicts
6 #
7 # Only basic testing has been done. At this point all we
    guarantee is that they won't crash a recent snort
    release.
8 #
9 # More information available at www.bleedingsnort.com
10 #
11 # Please submit any custom rules or ideas to
    bleeding@bleedingsnort.com or the snort-sigs mailing
    list
12 #
```

```
13 #
   **********************************************************************

14 #
15 #  Copyright (c) 2006, Bleedingsnort.com
16 #  All rights reserved.
17 #
18 #  Redistribution and use in source and binary forms, with
   or without modification, are permitted provided that the
19 #  following conditions are met:
20 #
21 #  * Redistributions of source code must retain the above
   copyright notice, this list of conditions and the
   following
22 #    disclaimer.
23 #  * Redistributions in binary form must reproduce the
   above copyright notice, this list of conditions and the
24 #    following disclaimer in the documentation and/or other
    materials provided with the distribution.
25 #  * Neither the name of the nor the names of its
   contributors may be used to endorse or promote products
   derived
26 #    from this software without specific prior written
   permission.
27 #
28 #  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
   CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES
   ,
29 #  INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
    MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
   ARE
30 #  DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
   CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
   INCIDENTAL,
31 #  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
    BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
32 #  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
   INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
   LIABILITY,
33 #  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (
   INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
   OUT OF THE
```

```
34 #   USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
       OF SUCH DAMAGE.
35 #
36 #
37
38
39 # By Chich Thierry
10 alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
     BLEEDING-EDGE P2P BitTorrent peer sync"; flow:
     established; content:"|0000000d0600|"; offset: 0; depth:
      6; reference:url,bitconjurer.org/BitTorrent/protocol.
     html; classtype: policy-violation; sid: 2000334; rev:7;
     )
11 alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
     BLEEDING-EDGE P2P BitTorrent Traffic"; flow: established
     ; content:"|0004009070000000|"; offset: 0; depth: 8;
     reference:url,bitconjurer.org/BitTorrent/protocol.html;
     classtype: policy-violation; sid: 2000357; rev:3; )
12 alert tcp $HOME_NET any -> $EXTERNAL_NET 6969 (msg: "
     BLEEDING-EDGE P2P BitTorrent Announce"; flow: to_server,
     established; content:"/announce"; reference:url,
     bitconjurer.org/BitTorrent/protocol.html; classtype:
     policy-violation; sid: 2000369; rev:4; )
13
14 #by markmc
15 alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"BLEEDING
     -EDGE P2P Direct Connect Traffic (client-server)"; flow:
     from_client,established; content:"$MyINFO"; offset:0;
     depth:7; classtype:policy-violation; reference:url,en.
     wikipedia.org/wiki/Direct_connect_file-
     sharing_application; sid:2002814; rev:1;)
16
17 # By Chich Thierry
18 alert tcp any any -> any 4660:4799 (msg: "BLEEDING-EDGE P2P
     ed2k connection to server"; flow: to_server,established
     ; content:"|e3|"; offset: 0; depth: 1; content:"
     |00000001|"; offset: 2; depth: 4; reference:url,www.giac
     .org/practical/GCIH/Ian_Gosling_GCIH.pdf; classtype:
     policy-violation; sid: 2000330; rev:5; )
19 alert tcp any any -> any 4660:4799 (msg: "BLEEDING-EDGE P2P
     ed2k file search"; flow: to_server,established; content
     :"|e3|"; offset: 0; depth: 1; content:"|00000016|";
     offset: 2; depth: 4; reference:url,www.giac.org/
```

```
    practical/GCIH/Ian_Gosling_GCIH.pdf; classtype: policy-
    violation; sid: 2000331; rev:5; )
50 alert tcp any any -> any 4660:4799 (msg: "BLEEDING-EDGE P2P
    ed2k request part"; flow: to_server,established;
    content:"|e3|"; offset: 0; depth: 1; content:"|00000047|
    "; offset: 2; depth: 4; reference:url,www.giac.org/
    practical/GCIH/Ian_Gosling_GCIH.pdf; classtype: policy-
    violation; sid: 2000332; rev:5; )
51 alert tcp any any -> any 4660:4799 (msg: "BLEEDING-EDGE P2P
    ed2k file request answer"; flow: to_server,established;
     content:"|e3|"; offset: 0; depth: 1; content:"
    |00000059|"; offset: 2; depth: 4; reference:url,www.giac
    .org/practical/GCIH/Ian_Gosling_GCIH.pdf; classtype:
    policy-violation; sid: 2000333; rev:5; )
52 alert udp $EXTERNAL_NET any -> $HOME_NET any (msg: "
    BLEEDING-EDGE P2P Kaaza Media desktop p2pnetworking.exe
    Activity"; content:"|e30cb0|"; offset: 0; depth: 6;
    threshold: type limit, track by_dst, count 1 , seconds
    600; reference:url,www.giac.org/practical/GCIH/
    Ian_Gosling_GCIH.pdf; classtype: policy-violation; sid:
    2000340; rev:5; )
53
54 #Submitted by Sam Evans
55 alert tcp $HOME_NET any -> $EXTERNAL_NET 4660:4799 (msg: "
    BLEEDING-EDGE P2P eDonkey File Status"; flow: to_server,
    established; content:"|e3 14|"; offset: 0; depth: 2;
    classtype: policy-violation; reference:url,www.edonkey.
    com; sid: 2001296; rev:4; )
56 alert tcp $HOME_NET any -> $EXTERNAL_NET 4660:4799 (msg: "
    BLEEDING-EDGE P2P eDonkey File Status Request"; flow:
    to_server,established; content:"|e3 11|"; offset: 0;
    depth: 2; classtype: policy-violation; reference:url,www
    .edonkey.com; sid: 2001297; rev:5; )
57 alert udp $HOME_NET any -> $EXTERNAL_NET 4660:4799 (msg: "
    BLEEDING-EDGE P2P eDonkey Server Status Request";
    content:"|e3 96|"; offset: 0; depth: 2; rawbytes;
    classtype: policy-violation; reference:url,www.edonkey.
    com; sid: 2001298; rev:3; )
58 alert udp $HOME_NET 4660:4799 -> $EXTERNAL_NET any (msg: "
    BLEEDING-EDGE P2P eDonkey Server Status"; content:"|e3
    97|"; offset: 0; depth: 2; rawbytes;classtype: policy-
    violation; reference:url,www.edonkey.com; sid: 2001299;
    rev:3; )
```

```
59 alert udp $HOME_NET any -> $EXTERNAL_NET 4660:4799 (msg: "
     BLEEDING-EDGE P2P eDonkey Search"; content:"|e3 0e|";
     offset: 0; depth: 2; rawbytes;classtype: policy-
     violation; reference:url,www.edonkey.com; sid: 2001305;
     rev:3; )
60 #alert tcp $HOME_NET any -> $EXTERNAL_NET 4660:4799 (msg: "
     BLEEDING-EDGE P2P eDonkey Hello Request"; flow:
     to_server,established; content:"|e3|"; content:"|01|";
     offset: 0; depth: 7; classtype: policy-violation;
     reference:url,www.edonkey.com; sid: 2001300; rev:4; )
61
62
63 #From Cooljay
64 alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
     BLEEDING-EDGE P2P Gnutella Connect"; flow: established,
     to_server; content:"GNUTELLA CONNECT/"; nocase; offset:
     0; depth: 17; classtype: policy-violation; reference:url
     ,www.gnutella.com; sid: 2001664; rev:3; )
65
66 #by Jeff Kell
67 # Looking for Gnucleus/GnucDNA UDP Ultrapeer handshakes
68 alert udp $HOME_NET any -> $EXTERNAL_NET any (msg: "
     BLEEDING-EDGE P2P GnucDNA UDP Ultrapeer Traffic";
     content:"SCP@|83|DNA@"; threshold: type both,track
     by_src,count 10,seconds 600; classtype: policy-violation
     ; sid:2002760; rev:1;)
69 # Looking for Gnucleus/GnucDNA running Ultrapeer
70 alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
     BLEEDING-EDGE P2P Gnutella TCP Ultrapeer Traffic"; flow:
      established,to_server; content:"GNUTELLA"; offset:0;
     depth:8; content:"X-Ultrapeer\: True"; nocase;
     threshold: type both,track by_src,count 5,seconds 3600;
     classtype: policy-violation; sid:2002761; rev:1;)
71
72
73 #By Bob Grabowsky
74 alert udp $HOME_NET 1024:65535 -> $EXTERNAL_NET 1024:65535
     (msg: "BLEEDING-EDGE P2P kazaa over UDP"; content:"KaZaA
     "; nocase; threshold: type threshold, track by_src,count
      10, seconds 60; classtype: policy-violation; reference:
     url,www.kazaa.com/us/index.htm; sid: 2001796; rev:3; )
75 alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg: "
     BLEEDING-EDGE KazaaClient P2P Traffic"; flow:
```

```
        established; content:"Agent\:␣KazaaClient"; nocase;
        classtype: policy-violation; reference:url,www.kazaa.com
        /us/index.htm; sid: 2001812; rev:3; )
76
77 #By Bob Grabowsky
78 alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
        BLEEDING-EDGE␣P2P␣LimeWire␣P2P␣Traffic"; flow:
        established; content:"User-Agent\:␣LimeWire"; nocase;
        classtype: policy-violation; reference:url,www.limewire.
        com; sid: 2001808; rev:3; )
79 alert udp $HOME_NET 6346 -> $EXTERNAL_NET 6346:6700 (msg: "
        BLEEDING-EDGE␣P2P␣Limewire␣P2P␣UDP␣Traffic"; content:"
        |49␣50␣40␣83␣53␣43␣50␣41|"; threshold: type threshold,
        track by_src,count 10, seconds 60; classtype: policy-
        violation; reference:url,www.limewire.com; sid: 2001809;
         rev:3; )
80 alert udp $HOME_NET 6345:6349 -> $EXTERNAL_NET 6345:6349 (
        msg: "BLEEDING-EDGE␣P2P␣UDP␣traffic␣-␣Likely␣Limewire";
        threshold: type threshold,track by_src,count 40, seconds
         300; classtype: policy-violation; reference:url,www.
        limewire.com; sid: 2001841; rev:5; )
81
82 #Submitted by Matt Jonkman
83 alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
        "BLEEDING-EDGE␣P2P␣Morpheus␣Install"; flow: to_server,
        established; uricontent:"/morpheus/morpheus.exe"; nocase
        ; reference:url,www.morpheus.com; classtype: policy-
        violation; sid: 2001035; rev:5; )
84 alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
        "BLEEDING-EDGE␣P2P␣Morpheus␣Install␣ini␣Download"; flow:
         to_server,established; uricontent:"/morpheus/
        morpheus_sm.ini"; nocase; reference:url,www.morpheus.com
        ; classtype: policy-violation; sid: 2001036; rev:5; )
85 alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
        "BLEEDING-EDGE␣P2P␣Morpheus␣Update␣Request"; flow:
        to_server,established; uricontent:"/gwebcache/gcache.asg
        ?hostfile="; nocase; reference:url,www.morpheus.com;
        classtype: policy-violation; sid: 2001037; rev:5; )
```

C.2 Instant Messaging

```
1 #
```

```
2 # $Id: bleeding-policy.rules $
3 # Bleeding Snort Policy rules.
4 #
5 # SID's are 2000000+ to avoid conflicts
6 #
7 # Only basic testing has been done. At this point all we
      guarantee is that they won't crash a recent snort
      release.
8 #
9 # More information available at www.bleedingsnort.com
10 #
11 # Please submit any custom rules or ideas to
      bleeding@bleedingsnort.com or the snort-sigs mailing
      list
12 #
13 #
      ********************************************************************

14 #
15 #  Copyright (c) 2006, Bleedingsnort.com
16 #  All rights reserved.
17 #
18 #  Redistribution and use in source and binary forms, with
      or without modification, are permitted provided that the
19 #  following conditions are met:
20 #
21 #  * Redistributions of source code must retain the above
      copyright notice, this list of conditions and the
      following
22 #    disclaimer.
23 #  * Redistributions in binary form must reproduce the
      above copyright notice, this list of conditions and the
24 #    following disclaimer in the documentation and/or other
       materials provided with the distribution.
25 #  * Neither the name of the nor the names of its
      contributors may be used to endorse or promote products
      derived
26 #    from this software without specific prior written
      permission.
27 #
28 #  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
      CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES
      ,
```

```
29 #  INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
       MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
      ARE
30 #  DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
      CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
      INCIDENTAL,
31 #  SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
       BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
32 #  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
      INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
      LIABILITY,
33 #  WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (
      INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
      OUT OF THE
34 #  USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
       OF SUCH DAMAGE.
35 #
36 #
37
38 #Submitted by Joel Esler
39 alert tcp $HOME_NET any <> $EXTERNAL_NET any (msg: "
      BLEEDING-EDGE CHAT MSN file transfer request"; flow:
      established; content:"MSG "; depth: 4; content:"Content-
      Type|3A|"; nocase; distance: 0; content:"text/x-
      msmsgsinvite"; nocase; distance: 0; content:"Application
      -Name|3A|"; content:"File Transfer"; nocase; distance:
      0; classtype: policy-violation; sid: 2001241; rev:3; )
10 alert tcp $HOME_NET any <> $EXTERNAL_NET any (msg: "
      BLEEDING-EDGE CHAT MSN file transfer accept"; flow:
      established; content:"MSG "; depth: 4; content:"Content-
      Type|3A|"; nocase; content:"text/x-msmsgsinvite";
      distance: 0; content:"Invitation-Command|3A|"; content:"
      ACCEPT"; distance: 1; classtype: policy-violation; sid:
      2001242; rev:3; )
11 alert tcp $HOME_NET any <> $EXTERNAL_NET any (msg: "
      BLEEDING-EDGE CHAT MSN file transfer reject"; flow:
      established; content:"MSG "; depth: 4; content:"Content-
      Type|3A|"; nocase; content:"text/x-msmsgsinvite";
      distance: 0; content:"Invitation-Command|3A|"; content:"
      CANCEL"; distance: 0; content:"Cancel-Code|3A|"; nocase;
       content:"REJECT"; nocase; distance: 0; classtype:
      policy-violation; sid: 2001243; rev:3; )
12 #Matt Jonkman, more msn
```

```
13 alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
       "BLEEDING-EDGE Policy MSN IM Poll via HTTP"; flow:
       established,to_server; uricontent:"/gateway/gateway.dll?
       Action=poll&SessionID="; nocase; threshold: type limit,
       track by_src, count 10, seconds 3600; classtype: policy-
       violation; sid: 2001682; rev:5; )
14
15 #Submitted by Scott Melnick
16 alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
       BLEEDING-EDGE POLICY MSN status change"; flow:
       established,to_server; content:"CHG "; depth:55;
       classtype:policy-violation; sid:2002192; rev:2;)
17 alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
       BLEEDING-EDGE POLICY MSN Game Loading"; flow:established
       ,to_server; content:"|6D 73 6E 67 61 6D 65 2E 61 73 70
       78|"; within:90; classtype:policy-violation; sid
       :2002312; rev:1;)
18
19 #Submitted by Joel Esler
50 alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg: "
       BLEEDING-EDGE CHAT Yahoo IM successful logon"; flow:
       from_server,established; content:"YMSG"; nocase; depth:
       4; content:"|00 01|"; offset: 10; depth: 2; classtype:
       policy-violation; sid: 2001253; rev:3; )
51 alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg: "
       BLEEDING-EDGE CHAT Yahoo IM voicechat"; flow:
       from_server,established; content:"YMSG"; nocase; depth:
       4; content:"|00|J"; offset: 10; depth: 2; classtype:
       policy-violation; sid: 2001254; rev:3; )
52 #Commenting out, duplicated in Snort.org set
53 #alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
       BLEEDING-EDGE CHAT Yahoo IM ping"; flow: to_server,
       established; content:"YMSG"; nocase; depth: 4; content
       :"|00 12|"; offset: 10; depth: 2; classtype: policy-
       violation; sid: 2001255; rev:4; )
54 alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg: "
       BLEEDING-EDGE CHAT Yahoo IM conference invitation"; flow
       : from_server,established; content:"YMSG"; nocase; depth
       : 4; content:"|00 18|"; offset: 10; depth: 2; classtype:
        policy-violation; sid: 2001256; rev:3; )
55 alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg: "
       BLEEDING-EDGE CHAT Yahoo IM conference logon success";
       flow: from_server,established; content:"YMSG"; nocase;
```

```
depth: 4; content:"|00 19|"; offset: 10; depth: 2;
classtype: policy-violation; sid: 2001257; rev:3; )
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
BLEEDING-EDGE CHAT Yahoo IM conference message"; flow:
to_server,established; content:"YMSG"; nocase; depth: 4;
content:"|00 1D|"; offset: 10; depth: 2; classtype:
policy-violation; sid: 2001258; rev:3; )
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
BLEEDING-EDGE CHAT Yahoo IM Unavailable Status"; flow:
to_server,established; content:"|59 47 00 0b 00 00 00 00
00 12 00 00 00 00|"; depth: 55; classtype: policy-
violation; sid: 2001427; rev:3; )
alert tcp $HOME_NET any <> $EXTERNAL_NET any (msg: "
BLEEDING-EDGE CHAT Yahoo IM file transfer request"; flow
: established; content:"YMSG"; nocase; depth: 4; content
:"|00|M"; offset: 10; depth: 2; classtype: policy-
violation; sid: 2001259; rev:4; )
#alert tcp $HOME_NET any <> $EXTERNAL_NET any (msg: "
BLEEDING-EDGE CHAT Yahoo IM message"; flow: established;
content:"YMSG"; depth: 4; classtype: policy-violation;
sid: 2001260; rev:4; )
alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg: "
BLEEDING-EDGE CHAT Yahoo IM successful chat join"; flow:
from_server,established; content:"YMSG"; nocase; depth:
4; content:"|00 98|"; offset: 10; depth: 2; classtype:
policy-violation; sid: 2001261; rev:3; )
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
BLEEDING-EDGE CHAT Yahoo IM conference offer invitation"
; flow: to_server,established; content:"YMSG"; nocase;
depth: 4; content:"|00|P"; offset: 10; depth: 2;
classtype: policy-violation; sid: 2001262; rev:3; )
alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg: "
BLEEDING-EDGE CHAT Yahoo IM conference request"; flow:
to_server,established; content:"<R"; depth: 2; pcre:"/^\
x3c(REQIMG|RVWCFG)\x3e/ism"; classtype: policy-violation
; sid: 2001263; rev:3; )
#alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg: "
BLEEDING-EDGE CHAT Yahoo IM conference watch"; flow:
from_server,established; content:"|0D 00 05 00|"; depth:
4; classtype: policy-violation; sid: 2001264; rev:3; )

#Matt Jonkman
```

```
66 alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
    "BLEEDING-EDGE CHAT Yahoo IM Client Install"; flow:
    to_server,established; uricontent:"/ycontent/stats.php?
    version="; nocase; uricontent:"EVENT=InstallBegin";
    nocase; classtype: policy-violation; sid: 2002659; rev
    :1; )
67 #By Merphie from the forums
68 alert tcp $HOME_NET any -> $EXTERNAL_NET 5190 (msg: "
    BLEEDING-EDGE POLICY ICQ Status Invisible"; flow:
    from_client,established; content:"|2A02|"; depth: 2;
    content:"|001900130005|"; offset: 4; depth: 6; classtype
    : policy-violation; sid: 2001801; rev:3; )
69 alert tcp $HOME_NET any -> $EXTERNAL_NET 5190 (msg: "
    BLEEDING-EDGE POLICY ICQ Status Change (1)"; flow:
    from_client,established; content:"|2A02|"; depth: 2;
    content:"|000E00010011|"; offset: 4; depth: 6; classtype
    : policy-violation; sid: 2001802; rev:4; )
70 alert tcp $HOME_NET any -> $EXTERNAL_NET 5190 (msg: "
    BLEEDING-EDGE POLICY ICQ Status Change (2)"; flow:
    from_client,established; content:"|2A02|"; depth: 2;
    content:"|00120001001E|"; offset: 4; depth: 6; classtype
    : policy-violation; sid: 2001803; rev:4; )
71 alert tcp $HOME_NET any -> $EXTERNAL_NET 5190 (msg: "
    BLEEDING-EDGE POLICY ICQ Login"; flow: from_client,
    established; content:"|2A01|"; depth: 2; content:"
    |00010001|"; offset: 8; depth: 4; classtype: policy-
    violation; sid: 2001804; rev:3; )
72 alert tcp $HOME_NET any <> $EXTERNAL_NET any (msg: "
    BLEEDING-EDGE POLICY ICQ Message"; flow: established;
    content:"|2A02|"; depth: 2; content:"|000400060000|";
    offset: 6; depth: 6; classtype: policy-violation; sid:
    2001805; rev:3; )
73
74 #By Chich Thierry
75 alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
    "BLEEDING-EDGE Policy Skype VOIP Checking Version (
    Startup)"; flow: to_server,established; uricontent:"/ui/
    "; nocase; uricontent:"/getlatestversion?ver="; nocase;
    reference:url,www1.cs.columbia.edu/~library/TR-
    repository/reports/reports-2004/cucs-039-04.pdf;
    classtype: policy-violation; sid: 2001595; rev:6; )
76 alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:
    "BLEEDING-EDGE Policy Skype VOIP Reporting Install";
```

```
flow: to_server,established; uricontent:"/ui/"; nocase;
uricontent:"/installed"; nocase; reference:url,www1.cs.
columbia.edu/~library/TR-repository/reports/reports
-2004/cucs-039-04.pdf; classtype: policy-violation; sid:
 2001596; rev:6; )
```

Lebenslauf

Georg Linhard

Kontakt: e-mail georg.linhard@mac.com

Präsenzdienst

Einjährig-Freiwilligenausbildung in Hörsching und Langenlebarn von Oktober 1998 bis September 1999

Ausbildung

seit 2000: Johannes Kepler Universität Linz, Diplomstudium Informatik

1999-2000: Technische Univ. Wien, Studium Informatik

1998: Matura am Bundes-Oberstufen-Realgymnasium Grieskirchen

1994-1998: Bundes-Oberstufen-Realgymnasium Grieskirchen, Schulform Oberrealgymnasium mit erg. Unterricht in Biologie, Physik und Chemie

1992-1994: BG/BRG Dr. Schauerstrasse Wels, Schulform mit besonderer Berücksichtigung der Informatik

1990-1992: Öff. Hauptschule Bad Schallerbach

1986-1990: Öff. Volksschule Langholzfeld

Berufliche Erfahrung

VOEST Alpine Informationstechnologie GmbH, Abteilung TSS; Linz — seit 09/2006

Security Management

VOEST Alpine Informationstechnologie GmbH, Abteilung TAH; Linz — 2004-2006

Helpdesk

Netzwerkadministrator WIST-Oberösterreich; Linz — 2003-2005

Wartung der Server und Netzwerkgeräte (u.a. Cisco-Router), Hardwarebeschaffung und -installation.

Im Rahmen dieser Tätigkeit absolvierte ich das Projektpraktikum „Netzwerktechnische Absicherung eines Studentenheims“ am Institut für Informationsverarbeitung und Mikroprozessortechnik der Johannes Kepler Universität Linz.

Verwaltung Netzwerk Pasching; Pasching — 2000-2002

Wartung der internen Server, Hardwarebeschaffung und -installation, Webdesign.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplom- bzw. Magisterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Des weiteren versichere ich, dass ich diese Diplom- bzw. Magisterarbeit weder im In- noch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Linz, 30.8.2006

.....

Georg Linhard