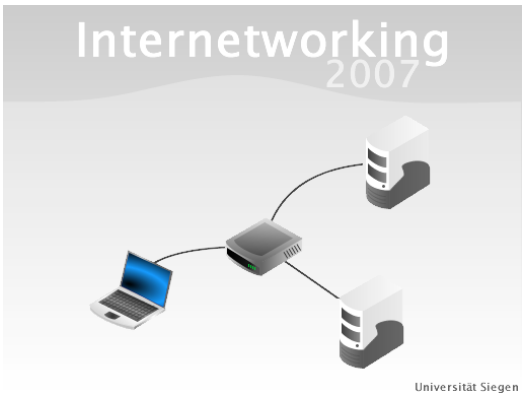
Abschlussbericht

Projektgruppe FILIUS



22. Oktober 2007

Eingereicht von:
André Asschoff, Johannes Bade, Carsten Dittich, Thomas Gerding, Nadja Haßler, Johannes Klebert, Michell Weyer

Betreuer:
Dipl.-Ing. Stefan Freischlad und Dipl.-Inf. Peer Stechert
Fachgruppe Didaktik der Informatik und E-Learning
Fachbereich 12, Universität Siegen

Inhaltsverzeichnis

1 Einleitung

1.1 Eine Lernsoftware zum Thema „Internetworking“ für die Sekundarstufe

In der heutigen Zeit wird im Berufsleben von jedermann erwartet gewisse oder zumindest grundlegende Kenntnisse im Umgang mit dem Internet zu haben. Es gibt kaum noch eine Firma und erst recht keine öffentlich Anstalt in der nicht Rechner miteinander vernetzt sind. Wenn sich ein junger Schulabgänger bei einer Firma um einen Ausbildungsplatz bewirbt, sollte er die Fähigkeit besitzen ein solches Informatiksystem zu bedienen. Dies erhöht seine Chancen die Stelle zu bekommen. Aber nicht nur im Beruf, sondern auch im Alltag kommt man nicht mehr an Netzwerktechniken vorbei. Angefangen beim verwalten des heimischen Telefones, bei dessen Installation und Wartung so ziemlich jeder Laie an seine Grenzen stößt, gibt es fast kein alltägliches Thema mehr bei dem nicht einmal ein Schlagwort wie „Internetseite“ oder „Download“ fällt. Daher ist es von enormer Wichtigkeit die Schüler schon früh an diese Themen heranzuführen und eventuell zu Fachkräften von morgen auszubilden!

Die vorliegende Dokumentation beschreibt detailliert die Funktionen und den Aufbau der explorativen Lernsoftware FILIUS[1]. Die Software ist unter anderem für den Einsatz im Unterricht an Allgemeinbildenden Schulen im Sekundarstufenbereich gedacht und soll dort für das Verständnis der Funktionsweise und Anwendungen des Internet eingesetzt werden. Dabei bietet sie die Möglichkeit Netzwerke aus verschiedenen Komponenten nachzubilden, zu konfigurieren und dann auch Anwendungen innerhalb des Netzwerks ausführen zu können. Zudem kann der Benutzer[2] auf verschiedenen Netzwerkschichten die dabei enstehenden Pakete und Nachrichten beobachten.

Aufgebaut ist FILIUS also so, dass man zwei Tätigkeiten unterscheidet: Das Konstruieren des Netzwerks, was auch die Konfiguration (z. B. manuelle Zuweisung der IP-Adresse) mit einschließt. Außerdem die Interaktion mit den simulierten Systemen. Hier kann auf den virtuellen Hosts des erstellten Netzwerks Software installiert und genutzt werden. Alle Vorgänge verhalten sich dabei „gesprächig“, so dass in einem separaten Fenster die Aktionen, aber auch der Datenaustausch auf den verschiedenen Netzschichten beobachtet werden kann.

[1] FILIUS bedeutet Freie Interaktive Lernumgebung für Internetworking der Universität Siegen.

[2] Soweit möglich, verwenden wir geschlechterneutrale Bezeichnungen. Wo dies nicht geeignet erscheint, haben wir die männliche Form verwendet

Als typische Komponenten eines Rechnernetzes stehen dem Benutzer dabei Rechner, Switches, Router und Kabel zur Verfügung. Diese können in beliebiger Anzahl im Entwurfsmodus eingefügt und miteinander verbunden werden. Die Software bietet dabei kaum Einschränkungen für den Benutzer. Beim Erstellen wird einzig darauf geachtet, dass nicht mehr Kabel an ein Gerät angeschlossen werden, als Anschlüsse am Gerät vorhanden sind. In der Grundkonfiguration der Software sind die Rechner auf einen Anschluss, Router auf vier und Switches auf acht Anschlüsse begrenzt. Diese Grenzen sind jedoch über Konfigurationsdateien änderbar.

Im Aktionsmodus steht dem Benutzer auf jedem Rechner ein virtueller Desktop zur Verfügung der, genau wie die Software, den heute üblichen Betriebssystemoberflächen nachempfunden ist. Auf diesem Desktop ist in jedem Falle ein Icon zum starten der Software-Installations-Anwendung vorhanden. Mit dieser Anwendung lassen sich auf jedem Host beliebige Kombinationen der zur Verfügung stehenden Software installieren. Jede so installierte Software legt ein Starticon auf dem Desktop ab. Wird eine Software gestartet, so öffnet sich die Benutzungsoberfläche der jeweiligen Software und kann vom Benutzer so bedient werden, wie er es von seinem Rechner gewohnt ist.

Die Software soll eine Unterstützung bei der Gestaltung des Informatikunterrichtes sein. Lehrer haben die Möglichkeit mit FILIUS gut verständliche Unterrichtseinheiten zu gestalten. Der explorative Charakter des Programms fördert das konstruktive Lernen der Schüler. Lernende können auf einer Arbeitsfläche symbolhafte Rechner mit Vermittlungsrechnern oder Switches zu einem Rechnernetz zusammenfügen. Die einzelnen Komponenten können dann auf verschiedene Weisen konfiguriert werden. Anschließend kann sich über Erfolg oder Misserfolg der Netzwerkkonfiguration in einem Aktionsmodus überzeugt werden.

FILIUS ist eine explorative Lernsoftware zum Thema Internetworking. Sie dient dem Erlernen von Grundwissen zum Internets und steht jedem frei zur Verfügung, der sich mit dem Thema Internet näher befassen möchte, ist aber insbesondere für den Einsatz in Schulen gedacht. Die Absicht der Entwickler dieser Software ist es, möglichst viele Nutzer zu gewinnen. Die Etablierung nicht nur an Schulen in Nordrhein-Westfalen, sondern in ganz Deutschland wäre der denkbar grösste Erfolg und Lohn für die Mühen, die das Entwickler-Team auf sich genommen hat. Kurz gesagt: Jeder ist herzlich eingeladen die Software herunter zu laden.

Die Dokumentation richtet sich in erster Linie an Lehrer und Entwickler. Für den Lehrer stellt sie eine Hilfestellung zur Planung und Durchführung des Unterrichts dar. Er kann sich zum Einen fachlich informieren wenn er

selbst nicht ganz sicher ist wie etwas bestimmtes im Zusammenhang mit Netzwerken funktioniert. Zum Anderen soll dieses Dokument dabei helfen, schneller eine Arbeitsanleitung und eventuell Arbeitsblätter zu erstellen. Dazu können in erster Linie die Grundlagen und die Lernszenarien eine Hilfe sein.

Exploratives Lernen heißt so viel wie entdeckendes, forschendes oder autonomes Lernen. „Ziel ist die Entwicklung von Bildungssoftware, die vollständig vom Schüler gesteuert wird. Ohne seine Aktivität laufen auch keine Systemprozesse ab. Der Schüler kann nicht ziellos durch multimediale Präsentationen navigieren. Es bedarf zumindest einer fachlich fundierten Hypothese zum Lerngegenstand, um die Interaktion mit dem Lernmaterial zu starten. Die Arbeit mit einem Explorationsmodul wird motiviert aus dem Wunsch des Schülers mit Elementen zu agieren, die auf Grund ihrer Abstraktion oder Komplexität mit anderen Lernmaterialien schwerer erfassbar oder schwerer darstellbar sind“[SS04, S. 142].

Im Informatikunterricht kann Lernsoftware gewinnbringend im Lehr-Lern-Prozess eingesetzt werden. Der Mehrwert von Lernsoftware liegt insbesondere in der Interaktivität. Die Interaktion muss fachdidaktisch begründet werden, da der Einsatz interaktiver Medien im Unterricht noch keine Interaktivität gewährleistet [Ke01].

„Explorative Lernaktivitäten sind üblicherweise nicht von Anderen (Lehrern, Vorgesetzten etc.) angeordnet, sondern gehen von der eigenen Person aus. Es stellt sich jedoch die Frage, ob es nicht möglich ist, dieses so mühelose explorative Lernen von außen zumindest zu fördern, wenn nicht sogar durch geeignete multimediale Lernangebote zu initiieren. Exploratives Lernen in solchen Lernumgebungen schließt aus, dass z. B. Ziele und/oder Wege des Lernens von außen vorgegeben werden. Es kann ein *Angebot* gemacht werden, das so zu gestalten ist, dass die Person motiviert ist, sich eigenständig mit einem Gegenstand auseinanderzusetzen“[Ke01, S. 217 f].

1.2 Organisation und Vorgehensweise der Projektgruppe

Nach Vorlage des Arbeitsauftrages durch die Betreuer, begann die Recherche zu Grundlagen und anschließende Planung der Software. Zur Vermittlung der Grundlagen hatte jeder Gruppenteilnehmer den Auftrag anhand eines Kurzvortrages ein Thema vorzustellen. Dasselbe wurde später mit den in dieser Ausarbeitung auftauchenden Lehr-Lern-Szenarien gemacht. Von Beginn an wurde ein fester, wöchentlicher Termin eingehalten, zu dem sich alle Teilnehmer versammelten und aktuelle Probleme besprochen und gelöst wurden.

Ein nächster Meilenstein war die Implementierung der anhand von UML-Diagrammen vorbereiteten Module zum World Wide Web, E-Mail und Internetbasierter Dateiaustausch. Als Ergebnis dieser Phase konnte die explorative Lernumgebung FILIUS präsentiert werden. Nach der Implementierung wurde eine ausführliche Dokumentation in Form dieser Arbeit geschrieben. Parallel dazu existiert das Benutzungshandbuch, bei dem die Benutzung durch den Anwender von FILIUS im Vordergrund steht. Eine klare Abgrenzung ist zwar nicht möglich, trotzdem lässt sich die Projektgruppe in drei Phasen unterteilen:

Erstellen eines Konzepts und Aufgabenverteilung: Zunächst haben wir uns in zwei Gruppen aufgeteilt. Eine Gruppe hat das, bereits vorgegebene, Lastenheft erweitert. Die zweite Gruppe hat ein Konzept für die grafische Benutzungsschnittstelle (GUI - Graphical User Interface) entwickelt. Es wurden mit Hilfe von Grafikanwendungen Bilddateien erstellt, die ein GUI darstellten ohne diese GUI tatsächlich zu implementieren. Dies sollte uns währen der Implementierung helfen eine Vorstellung dafür zu haben, wie das Programm später einmal aussehen wird. Außerdem haben wir mit Vorträgen unsere Kenntnisse gegenseitig über die verschiedenen geplanten Elemente des Projektes vertieft. Verwendete Hilfsmittel: Regelmäßige protokollierte Gruppensitzungen, Maillingliste, SVN und zum Teil Chatprogramme. Am Ende dieser Phase stand das Pflichtenheft.

Implementierung: Den Beginn der zweiten Phase des Projekts kann man mit der Aufgabenverteilung datieren. Während der Implementierung wurden natürlich immer wieder Aufgaben verteilt, da immer wieder neue Probleme auftraten, die es zu lösen galt. Insgesamt muss man unsere Aufteilung der Module und ihre Implementierung kritisch betrachten. Vieles, was eigentlich aufeinander aufbaut, wurde gleichzeitig entwickelt. Dies hatte zur Folge, dass Änderungen an grundlegenden Bereichen, vor Allem an der Protokollschicht, dazu führten, dass immer wieder Änderungen in den darauf aufbauenden Klassen nötig waren. Wie bereits eingangs erwähnt, gab und gibt es zur Zeit keine vergleichbare Software, so dass wir während der Implementierung immer wieder an die Grenzen unseres Konzepts gestoßen sind und dieses Erweitern mussten. Dies bedeutet natürlich einen kurzfristigen Rücksprung in Phase 1.

Dokumentation und Test: In der letzten Phase unseres Projektes lag es nahe, dass die Verteilung der Dokumentation der Verteilung der Imple-

mentierung entsprach. Logischerweise kennt sich der jeweilige Programmier eines Moduls selber am Besten mit dem Modul aus. Ein weiteres wichtiges Element dieser Phase ist das weitere Testen unserer Software. In Phase 3 haben wir auch das Konzept des Peer-Review angewendet. Jedes Projektmitglied hat die Ergebnisse der anderen Projektmitglieder getestet. Dies war sehr hilfreich, da so viele Fehler und Schwachstellen gefunden wurden, die man beim Testen der eigenen Klassen unbewusst vermeidet oder übersieht. Verwendete Hilfsmittel: Peer-Review und Bugtracker.

Gerade in Phase 2 wurde zum Teil sehr viel Zeit in Programmierleistungen investiert, die, rückblickend, als nutzlos betrachtet werden müssen. Eine top-down oder auch bottom-up Programmierweise, zum Beispiel am Internetschichtenmodell orientiert, hätte uns sicher viel unnütze Arbeit erspart. Dieser Ansatz ließ allerdings keine effektive Arbeitsteilung zu.

Nach der abgeschlossenen Implementierungsphase haben die Entwickler das Programm ausführlich getestet. Da erfahrungsgemäß trotzdem nicht alle Fehler gefunden werden, haben die Entwickler sich dazu entschieden das Testen durch Peer-Review fortzuführen. Als testende Personen wurden Menschen aus dem Bekanntschaftskreis der Teilnehmer gefragt. Dabei handelte es sich sowohl um Studenten der Informatik, also Fachleute, und um Laien auf dem Gebiet der Informatik.

Zum Testen wurde die Software BUGS als besonderes Werkzeug eingeführt. Ein internetbasierter Bugtracker (zu deutsch „Fehler-Verfolger“). In eine solche Anwendung werden aufgefundene Fehler erfasst und dokumentiert. Das System verwaltet diese für die spätere Bereinigung der Fehler. Sobald ein Fehler behoben wurde, wird sein Status als „*fixed*“ markiert.

2 Lehr-Lern-Szenarien

In diesem Abschnitt werden Lehr-Lern-Szenarien zu Themen aus dem Bereich Internetworking vorgestellt. In jedem Abschnitt gibt es die Punkte: Zielsetzung, Lernziele, Notwendiges Wissen und Tätigkeiten der Lernenden. Es werden folgende Themen behandelt: Client-Server-Modell, Dynamic Host Control Protocol (DHCP) und Domain Name System (DNS), Routing und Peer-to-Peer. Da technische Details in den einzelnen Lernszenarien nicht tief behandelt werden können, verweisen die Autoren auf den Anhang und das Buch von Peterson und Davie [PD03] und das Vorlesungsskript von Wismüller [Wi06].

2.1 Client-Server-Modell

Das Client-Server-Modell beschreibt in der elektronischen Datenverarbeitung eine softwaretechnische Zuordnung. Oftmals werden die Begriffe fälschlicherweise auch als eine Art der Netzwerkstruktur bezeichnet, was jedoch im eigentlichen Sinne falsch ist. Dies rührt daher, dass die Netzwerkstruktur, in der Client-Server-Anwendungen betrieben werden, oftmals eben genau einer Sterntopologie ähnelt, die dann als Client-Server-System bezeichnet wird. Zunächst ist bei dem Client-Server-Modell eine genaue Begriffsdefinition von Nöten, denn es handelt sich hierbei nicht, wie aus der falschen Verwendung des Begriffes geschlossen werden könnte, um eine Relation von zwei Rechnern. Dies geschieht nur indirekt. Beim Client-Server-Modell handelt es sich um eine Zuordnung zweier Prozesse: eines Client- und eines Server-Prozesses. Diese Prozesse wiederum laufen jeweils auf einem Client- und einem Server-Rechner. Der Server-Rechner stellt dabei in der Regel einen Zentralrechner dar, und die Client-Rechner sind einzelne Workstations, die über ein gemeinsames Netzwerk mit dem Server arbeiten. Der Client ist in diesem Szenario ein Programm, das auf einem Client-Rechner läuft, und den Dienst (E-Mail, HTTP) eines Servers in Anspruch nimmt, weil er ihn selbst nicht bereitstellen kann. Im Gegensatz dazu ist der Server ein Programm, das zentral Dienste bereitstellt. Er wartet an zentraler Stelle auf eine Anfrage eines Clients (beispielsweise ein Browser) und stellt den gewünschten Dienst, wenn möglich zur Verfügung.

2.1.1 Zielsetzung

Der Anwender von FILIUS bekommt die Möglichkeit simulierte Client-Server-Netzwerke zu bauen. Er hat die Gewissheit, dass er dabei im Gegensatz

zum realen Verbinden von Netzwerkkarten, nichts kaputt machen kann. FILIUS soll dabei helfen, ungeübte Menschen mit wenig oder gar keinem Vorwissen bezüglich Client-Server-Modellen einen Einblick in diese Materie zu ermöglichen.

2.1.2 Lernziele

Die Schülerinnen und Schüler

1. können die Komponenten eines Netzwerkes benennen, deren Funktion und dessen Topologie beschreiben,
2. erkennen, dass jeder Rechner im Internet eine eigene Internet-Adresse hat,
3. wissen, wie die Internetadresse aussieht, und dass sie sich aus einer Netz- und einer Rechneridentifikation aufbaut,
4. wissen, dass ein Domainname aus Bezeichnern für einen Rechner und eine Domainhierarchie besteht,
5. wissen, dass Datenaustausch in einem Rechnernetz nach dem Client-Server-Modell zwischen einem Server-Prozess und einem Client-Prozess abläuft.

2.1.3 Notwendiges Vorwissen

Die Grundlagen verzahnen, so dass es oftmals zu Überschneidungen kommen kann. Grundsätzlich benötigen die Schüler ein Grundverständnis über ihren Rechner, den sie bei der praktischen Arbeit im Netzwerk auch bedienen können müssen. Weiterhin müssen ihnen die elementaren Begriffsdefinitionen der Komponenten eines Netzwerkes bekannt sein, wie Host, Switch, Router oder ähnliches. Bei einer Beschränkung des Themas nur auf Client-Server-Modell, ist es obligat, dass die Schülerinnen und Schüler zumindest rudimentäre Kenntnisse besitzen über Internet-Adressierung, Domänen und Topologien.

2.1.4 Tätigkeiten der Lernenden

Die Schüler sollten sich im Vorhinein zum Beispiel durch eine Computerrecherche über einige Grundlegende Aspekte von Client-Server-Architekturen

informieren. Einfache Netzwerkbefehle sollten vorausgesetzt werden können. Bei einer Durchführung in der Schule könnten zum Beispiel durch die Bearbeitung von Arbeitsblättern der Umgang mit Internet- und Web-Adressen geschult werden. Speziell die Client-Server-Interaktion kann sehr gut durch Sequenz- oder Topologiediagramme aufgezeigt werden. FILIUS bietet speziell für diesen Aspekt die sogenannten Server- und Client-Bausteine. Mit diesen sollte der Anwender, ob nun auf einem oder mehreren Rechnern, Nachrichten verschicken. Im konkreten Bezug auf die Software sind Tätigkeiten wie

- Modellierung, Sowohl von Graphen als auch von Code im Sinne einer Anpassung, bzw. Erweiterung,
- Interaktion, im Sinne von Benutzung des Rechners allgemein, wie auch Nutzung der Software zur Entwicklung eines Verständnisses des Internets,
- Kommunikation, bei Absprache mit Mitschülern, sowie Gruppenarbeit.

zu nennen.

2.2 Dienste und Anwendungen

FILIUS simuliert das Internet, mit seinen Diensten und Anwendungen. Als Dienste wollen wir an dieser Stelle DHCP und DNS hervorheben. DHCP dient der automatischen Vergabe von IP-Adressen in einem Netzwerk. Das heißt, wenn ein neuer Rechner zum Netz hinzukommt, so bekommt er vom DHCP-Server ein IP-Adresse zugewiesen. DNS hingegen kümmert sich um die für den Menschen leichter lesbaren Domainnamen. Einer IP-Adresse wird eindeutig vom Benutzer ein gewünschter Name zugewiesen: Ein Domainname. Ein DNS-Server verwaltet alle diese Namen in einer Tabelle.

2.2.1 Zielsetzung

Bestimmte Dienste und einheitliche Schnittstellen haben die Kommunikation im Internet und die Bedienung dessen seit seiner Erfindung wesentlich vereinfacht und sind somit maßgeblich für den Erfolg des Internets mitverantwortlich. Im folgenden Unterrichts-Szenario beziehen wir uns auf die Dienste Domain Name System (DNS) und Dynamic Host Control Protocol (DHCP), sowie auf die Mensch-Maschine-Schnittstelle Webbrowser. Für Details in Bezug auf den Dienst DNS sei an dieser Stelle auf Kapitel 5.4.2 verwiesen. Der Schwerpunkt liegt hier auf dem Dienst DHCP.

2.2.2 Dynamic Host Configuration Protocol

Dieses Protokoll ermöglicht die automatische Einbindung eines Rechners in ein Netzwerk. Der neue Rechner sendet dabei ein Broadcast und wartet, dass ein DHCP-Server antwortet und ihm eine so genannte IP-Adresse zuweist. Dies ermöglicht eine relativ einfache Einbindung in das Netzwerk ohne das größere Einstellungen am Rechner vorgenommen werden müssen. Außerdem vermeidet dieses System die doppelte Vergabe derselben IP-Adresse. Um besagte doppelte Vergabe einer IP-Adresse zu vermeiden speichert der Server zu jeder IP-Adresse die er vergibt die MAC-Adresse , an die er die IP-Adresse vergeben hat.

Ein Webbrowser ist ein typisches Beispiel für Schüler, um einen geeigneten Zugang für das Prinzip von Internetanwendung zu schaffen. In der Praxis verarbeiten Webbrowser die im HTML-Format (Hypertext Markup Language) vorliegenden Daten zu einer grafischen Darstellung, die es dem Benutzer ermöglicht, die enthaltenen Informationen leichter zu erfassen.

2.2.3 Lernziele

Die Schülerinnen und Schüler

1. verstehen die Funktion des DNS in seinen Grundzügen,
2. kennen den Zusammenhang zwischen URL (Uniform Resource Locator) und IP-Adresse,
3. können für beliebige, real existierende, IP-Adressen oder URLs, die dazu gehörende URL oder IP-Adresse herausfinden,
4. verstehen die Funktion des DHCP in seinen Grundzügen.

2.2.4 Notwendiges Vorwissen

Im Umgang mit der Software werden keine besonderen Kenntnisse vorausgesetzt. Das Drag-and-Drop-Prinzip sollte bekannt sein. Kenntnisse bezüglich Rechnernetze werden allerdings vorausgesetzt. Der Lernende muss wissen, was eine URL und was eine IP-Adresse ist. Der Lernende sollte nach Möglichkeit bereits über Erfahrungen mit dem Internet verfügen. Es wäre für die Lernenden vorteilhaft, wenn auch nicht zwingend notwendig, wenn einige der anderen Szenarien, die in dieser Dokumentation angegeben sind, den Schülern bereits geläufig sind. Hier zu nennen wären die Szenarien Routing und vor allem Client-Server-Modell.

2.2.5 Tätigkeiten der Lernenden

Im Vorfeld sollten die Schüler eine Recherche anstellen. Darin sollten sie sich in Grundlegende Aspekte zu Internetdiensten und danach speziell zu DHCP und DNS aneignen. Dies kann im Unterricht auch durch geeignete, von der Lehrperson angefertigte, Arbeitsblätter geleistet werden. Die im Umgang mit dem Rechner geschulten Lernenden installieren einen DNS-Server in FILIUS und schauen sich die aktualisierte Tabelle an. In Unterrichtsgesprächen kann über schwierige Dinge gesprochen werden. Zum Beispiel sind Anfragen eines Rechners für den Entscheid des DHCP-Servers über die vergabe einer IP-Adresse nicht trivial. In FILIUS sollte der Anwender selbst einen DNS-Server konfigurieren und einen DHCP-Server integrieren.

2.3 World Wide Web

Mit diesem Thema können sich die Lernenden im vorhinein beschäftigen. Dieses Thema ist ungemein praxisrelevant.

2.3.1 Zielsetzung

Durch die Einarbeitung und exploratives Vorgehen in und mit FILIUS sollen die Schüler grundlegende Elemente des World Wide Web kennenlernen. Der Lernende wird anschließend nicht nur in der Lage sein Zusammenhänge zu erkennen, sondern auch eigene Webseiten mit einem selbstkonfigurierten Server ins Netzwerk einzubinden. Eine Sensibilisierung für Sicherheitsaspekte hilft dabei, im realen Leben nicht auf Fallen im Internet hereinzufallen. Es muss klar werden, dass sich hinter dem Domainnamen immer eine eindeutige IP-Adresse verbirgt. Sollten die im folgenden genannten Lernziele erreicht werden, ist dieses Kriterium voll erfüllt.

2.3.2 Lernziele

Die Schülerinnen und Schüler

1. wissen wie eine HTML-Seite aufgebaut ist und erkennen, dass eine HTTP-Anfrage über die Adresszeile im Browser geschickt wird,
2. kennen die Kommandos `GET` und `POST`, und können ihre Bedeutung erklären,

3. können Probleme beim Versenden einer HTTP-Anfage nachverfolgen,

4. erstellen eigene HTML-Seiten und binden diese über den Datei-Explorer (Importierer) ein.

2.3.3 Notwendiges Vorwissen

Um verstehen zu können, dass ein HTML-Quellcode via HTTP verschickt wird, ist es unumgänglich zumindest Grundkenntnisse über die Sprache HTML zu haben. Die Lernenden sollten zumindest wissen was <HEAD> und <BODY> bedeuten, und wie die Struktur einer HTML-Seite aufgebaut ist. Im Unterricht sollte über das Kommando GET gesprochen werden. An einigen Stellen sollten auch Sicherheitstechnische Relevanzen geklärt werden, da im Internet das Thema Sicherheit eine große Rolle spielt. Ein oder mehrere Standardbrowser sollten bekannt sein.

2.3.4 Tätigkeiten der Lernenden

Die Schüler sind in der Lage ein Netzwerk in FILIUS zu erstellen. Sie können HTTP-Server und ihre Klienten konfigurieren. Nun gilt es auch gültige HTTP-Anfragen zu stellen. In FILIUS wird dazu ein Webserver und Webbrowser benötigt. Idealerweise sollten diese auf unterschiedlichen Rechnern installiert werden. Nach erfolgreicher Installation schickt der Anwender HTTP-Anfragen über das Netz. Die IP-Adresse oder der DNS-Name müssen bekannt sein, um dies zu erreichen. Wenn Probleme auftauchen, muss der Anwender diesen auf den Grund gehen. Er muss prüfen, ob seine Eingabe falsch ist, oder ob es ein „technisches“ Problem ist. Der Fortgeschrittene Schüler erstellt eigene HTML-Seiten, und bindet diese über den Datei-Importierer (Datei-Explorer) ein.

2.4 Routing

Es ist wichtig sich mit dem Thema zu beschäftigen, weil die Lernenden verstehen sollen, wie es dazu kommt, dass das Internet so zuverlässig ist. Das liegt daran, dass es keine zentrale Komponente zum Dateiaustausch gibt. Es wird die aktuell beste Route über Vermittlungsrechner im Internet gesucht. Es kann daher sogar der Fall eintreten, dass eine E-Mail von Siegen nach Frankfurt gegebenenfalls über Madrid verschickt wird. Man kann nicht davon ausgehen, dass nur weil eine Mail inerhalb einer Stadt verschickt wird, auch nur Vermittlungsrechner in dieser Stadt benutzt werden. Es muss nur

ein wichtiger Punkt blockiert sein, und schon macht die Nachricht eine weite Reise.

2.4.1 Zielsetzung

Unter Routing versteht man im weitesten Sinne die Erstellung von Weiterleitungstabellen in Netzwerken, die dazu dienen, Pakete in einem Netzwerk möglichst kostengünstig zu ihrem Ziel zu befördern. Ihr kann man später entnehmen, welcher Weg zu welchem Ziel eingeschlagen werden muss und welche Kosten zur Erreichung des Ziels notwendig sind. Zum Routing gehören außerdem der Umgang mit IP-Adressen und Subnetting. Für detaillierte Information sei an dieser Stelle auf Peterson und Davie [PD03] verwiesen.

2.4.2 Lernziele

Die Schülerinnen und Schüler

1. kennen den Aufbau und die Funktionsweise von Switch und Router,
2. erklären, was man unter Routing, und Weiterleitung versteht,
3. wenden den Dijkstra-Algorithmus zur Bestimmung bester Wege auf einen Graphen an,
4. weisen in einem Internetwork (selbst erstellt oder vorgegeben) selbstständig IP-Adressen, Subnetzmasken und Subnetzadressen zu.

2.4.3 Notwendiges Vorwissen

Grundlagen der Graphentheorie sollten bekannt sein. Falls auch etwas mit IP-Adressen gemacht werden soll, sollten Komponenten einer IP-Adresse mit Rechner-ID, Netz-ID und Netzmaske bekannt sein. Der Umgang mit Rechnungen im Dualsystem sollte vertraut sein, um IP-Adressen gegebenfalls in Binärzahlen umzuwandeln. Der Aufbau des Internets aus mehreren Rechnernetzen sollte besprochen werden. Die Unterscheidung von Paketvermittlung und Leitungsvermittlung stellen Peterson und Davie dar [PD03]), der Aufbau von Datagrammen (IP-Paket mit Angabe der Sender- und Empfängeradresse) wird in Abschnitt E erklärt.

2.4.4 Tätigkeiten der Lernenden

Die Lernenden erstellen mit der Software Netzwerke oder bearbeiten Aufgaben an vorgegebenen Netzwerken. Wünschenswert wäre eine selbstständige Vergabe von IP-Adressen, Netzmasken und Netzadressen. Schülerinnen und Schüler haben darüber hinaus die Möglichkeit in der Theorie schrittweise optimierte Weiterleitungstabellen zu erstellen. Als Hilfestellung dienen beispielsweise die im Anhang zu Routing angegebenen Links zu Animationen. Speziell zum Routing kann FILIUS das Lernen durch die Möglichkeit der Beobachtung von Nachrichten im Nachrichtenfenster unterstützen.

2.5 Peer-to-Peer-Netzwerke

Peer-to-Peer Netzwerke sind heutzutage von großer Wichtigkeit. Im Folgenden seien drei Gründe genannt, für die Implementierung von Peer-to-Peer-Netzwerken in unserer Software genannt. Im Durchschnitt sind heutzutage zu jeder Tages- und Nachtzeit acht Millionen aktive Benutzer mit dieser Netztechnologie tätig. Dies sind wohl in den meisten Fällen Tauschbörsen für Musikdateien und Filme, die wohlgemerkt am Rande der Legalität arbeiten. Wir wollen hier jedoch nicht die rechtlichen Aspekte dieser Netze beleuchten, sondern uns ausschließlich mit deren Anwendung befassen. Fakt ist nun einmal, dass diese enorme Zahl von Personen Dateiaustausch betreiben, vermutlich wird davon ein Großteil aus der jugendlichen Bevölkerung stammen. Auch deshalb ist es sicherlich für Schüler interessant, diese Konzepte kennen zu lernen, da sie zu Hause real mit derartigen Programmen arbeiten. Als Beispiel seien dazu Gnutella, Kazaa, Napster, und Bitlord genannt. Ein positiver Nebeneffekt könnte für die Schüler sein, dass vielleicht ein intuitives Verständnis für eine illegale Handlung eintritt. Ein Transfervolumen von 10 Petabyte Daten zu jeder Zeit spricht für sich. Das entspricht etwa der Hälfte des gesamten Internet-Verkehrs. Wo so viele Daten ausgetauscht werden tummeln sich natürlich auch schwarze Schafe. Als Folge deren illegaler Bereitstellung von Musikdaten wurden bereits einige Tauschbörsen durch Gerichtsurteile stillgelegt. Des Weiteren kursieren tausende Einzelklagen gegen Nutzer von derartiger Software wegen Verletzung von Urheberrechten.

2.5.1 Zielsetzung

FILIUS bietet separate Funktionen zum Verbinden mit einem Peer-to-Peer-Netzwerk, danach zum Suchen einer bestimmten Datei und zum Übertragen. Zu guter letzt kann die erfolgreich geladene Datei mit dem Dateiexplorer

begutachtet werden. Durch die Funktionalitäten von FILIUS werden den Schülern explorative Werkzeuge zum Entdecken und Erforschen von Internetworking-Systemen an die Hand gegeben. Durch das explorative Lernen werden individuelle Kompetenzen im Bereich des wissenschaftlichen Arbeitens gefördert. Die Schülerinnen und Schüler bekommen ein Gefühl für die Herangehensweise an neue und unbekannte Probleme und Aufgabenstellungen. Diese Kompetenz wird dem Lernenden später im Berufsleben oder bei der Weiterbildung an Hochschulen zu Gute kommen. Des Weiteren wird die fachliche Kompetenz durch die Einführung in Peer-to-Peer gesteigert. Bei der Aufgabenstellung werden Grundlagen im Wissen um Peer-to-Peer vorausgesetzt. So müssen Schlagworte wie Pure-Peer-to-Peer, Hybrides-Peer-to-Peer und Super-Peer-to-Peer (siehe Abschnitt E) bekannt sein und vom Schüler in der Software umgesetzt werden können. Zusätzlich wird Wissen über eventuell noch unbekannte Hardware wie Hub, Switch, Router erlernt. Diese müssen ebenso wie alle Hosts konfiguriert werden. Je nach Intensität erlangen die Lernenden grundlegende, aber auch weiterführende Kenntnisse über Funktionalität und Konfiguration von Rechnernetzen. Durch die Möglichkeit Aufgaben auf verschiedenen Schwierigkeitsniveaus durchzuführen kann das Prinzip des Spiralcurriculum mit dieser Software sehr schön umgesetzt werden. Beginnend mit einer einfachen Aufgabe wie zum Beispiel das Erstellen und Konfigurieren eines Pure-Peer-to-Peer-Netzwerkes kann schrittweise erweitert und verkompliziert werden bis hin zum Super-Peer-to-Peer-Netzwerk.

2.5.2 Lernziele

Die Schülerinnen und Schüler

1. können ihr Grundwissen über Peer-to-Peer-Rechnernetze anwenden.
2. können Rechnernetze zum Austausch von Dateien konfiguriern.

2.5.3 Notwendiges Vorwissen

Die Schüler sollten in der Lage sein, per Drag-and-Drop Symbole in die Arbeitsfläche der Entwurfsansicht einzufügen. Ebenfalls müssen grundlegende Kenntnisse über Rechnernetze vorausgesetzt werden. Der Lernende sollte dazu zumindest wissen, mit welcher Hardware er es zu tun hat. Also Fachbegriffe wie Switch, Router, Host, WLAN, Server, Client, etc. sollten nicht gänzlich neu sein.

2.5.4 Tätigkeiten der Lernenden

Die Schüler müssen sich zunächst mit einigen theoretischen Aspekten dieser Netzwerke auseinandersetzen. Sie müssen verstehen, dass es sich um eine softwareseitige Realisierung handelt. Jedoch ist durch die Software ein völlig anderes Verhalten mit anderen Rechten, Vorgehensweisen beim Suchen, etc. gegeben. Wenn dies weitestgehend verstanden ist, sollte der Schüler einige Zeit bekommen, um sich mit dem Tool FILIUS zu beschäftigen. Er sollte jedoch schnell in der Lage sein dieses zu bedienen, da die Struktur nicht allzu kompliziert aufgebaut ist. Am Anfang sollte es sogar ausreichen wenn per Drag-and-Drop Symbole in die Arbeitsfläche gezogen und diese dann mit Hilfe des Kontextmenüs konfiguriert werden können. Danach sollten im Prinzip schon kleine Peer-to-Peer-Netzwerke unter Anleitung vom Lehrpersonal erstellt werden können. Einer Erhöhung des Niveaus für leistungsstarke Schüler steht ab diesem Zeitpunkt auch nichts im Wege.

3 Übersicht zur Lernumgebung

3.1 Sichten

Dieser Abschnitt soll einen kurzen Abriss über die Möglichkeiten, die FILIUS bietet geben.

3.1.1 Entwurfsansicht

In der Entwurfsansicht werden die Netzwerke durch Hardware beschrieben. Per Drag-and-Drop müssen Hardwarekomponenten in die Arbeitsfläche gezogen, und mit Kabeln zu einem funktionstüchtigen Netzwerk verbunden werden. Die Entwurfsansicht dient lediglich dem Zusammenstellen von Hardware. Hardware wird hier noch nicht mit Software bestückt. Dies geschieht später im Simulationsmodus (siehe 3.1.2). Ausnahmen bilden die Systemsoftware, also beim Host das Betriebssystem, welche automatisch bereitgestellt wird. Ansonsten wäre keine Konfiguration möglich. Außerdem wird in diesem Modus unter Umständen auch schon der DHCP Server installiert. Die Menüleiste ist im Prinzip so aufgebaut, wie man es von gängiger Software gewöhnt ist. Sie befindet sich horizontal am oberen Rand des Hauptfensters und verfügt über die Schaltflächen: *Neu*, *Öffnen*, *Speichern* (Grundfunktionen) und *Entwurf*, *Simulation* (Hautpfunktionen), sowie dem Softwareassistenten und eine Hilfefunktion. Zusätzlich gibt es einen Geschwindigkeitsregler zur Bestimmung des Tempos im Simulationsmodus.

Zum Hinzufügen von Hardware in die Arbeitsfläche, befindet sich links eine Palette mit Hardware-Symbolen. Per Drag-and-Drop werden die einzelnen Symbole mit der Maus nach rechts gezogen. Es stehen die Komponenten:

Kabel,

Switch,

Vermittlungsrechner (Router),

Personal Comuter,

Laptop und

Modem

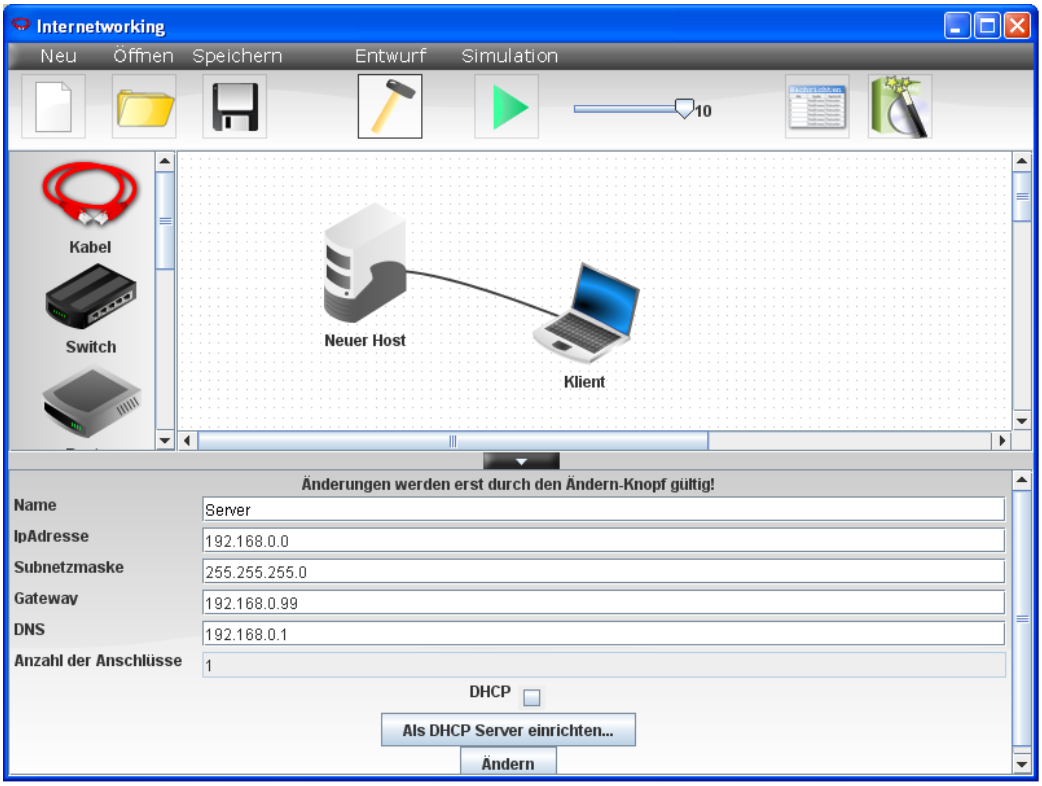


Abbildung 1: Entwurfsansicht

zur Verfügung.

Die Arbeitsfläche ist die Anzeigefläche für Hardwarekomponenten. Es ist nichts anderes als eine freie Fläche, auf der alle Hardwarekomponenten aufgesetzt werden. Es ist fester Bestandteil sowohl des Entwurfs- als auch des Simulationsmodus. Komponenten können nach belieben gesetzt, versetzt und miteinander verkabelt werden. Es ist auch möglich mehrere Hardware-Items mit einem Rahmen auszuwählen und gleichzeitig zu verschieben.

Am unteren Rand des Hauptfenster wird zu jeder aktuell gewählten Hardware die Konfigurationsparameter angezeigt. Dort können Daten verändert und Änderungen dauerhaft übernommen werden. Vorhandene Parameter können Name, IP-Adresse, Netzmaske, Gateway, DNS-Server, DHCP verwenden, DHCP-Server-Einstellungen sein. Das sind die Parameter, die im Abschnitt Konfigurationsdatei beschrieben wurden.

Server- und Client-Bausteine Server- und Client-Bausteine wurden implementiert, um den Lernenden ein Gefühl für den Datenaustausch zwischen Server und Client zu vermitteln. Von Wichtigkeit ist es zu Wissen, dass zunächst eine Verbindung aufgebaut werden muss. Nach dem erfolgreichen Verbinden, können dann Nachrichten an den Server geschickt werden (Abbildung 2). Zum Server soll deutlich gemacht werden, wie er mit ankommenden Nachrichten umgeht. Für ihn ist es lediglich eine Zeichenkette, die von einer bestimmten IP-Adresse kommt. Je nachdem um was für eine Anwendung es sich handelt, müsste er entscheiden was mit dieser Anfrage zu tun ist. In Abbildung 2 ist eine grafische Benutzungsschnittstelle einer einfachen Client-Server-Anwenundung dargestellt. Der *Software-Assistent* ist eine zusätzlich

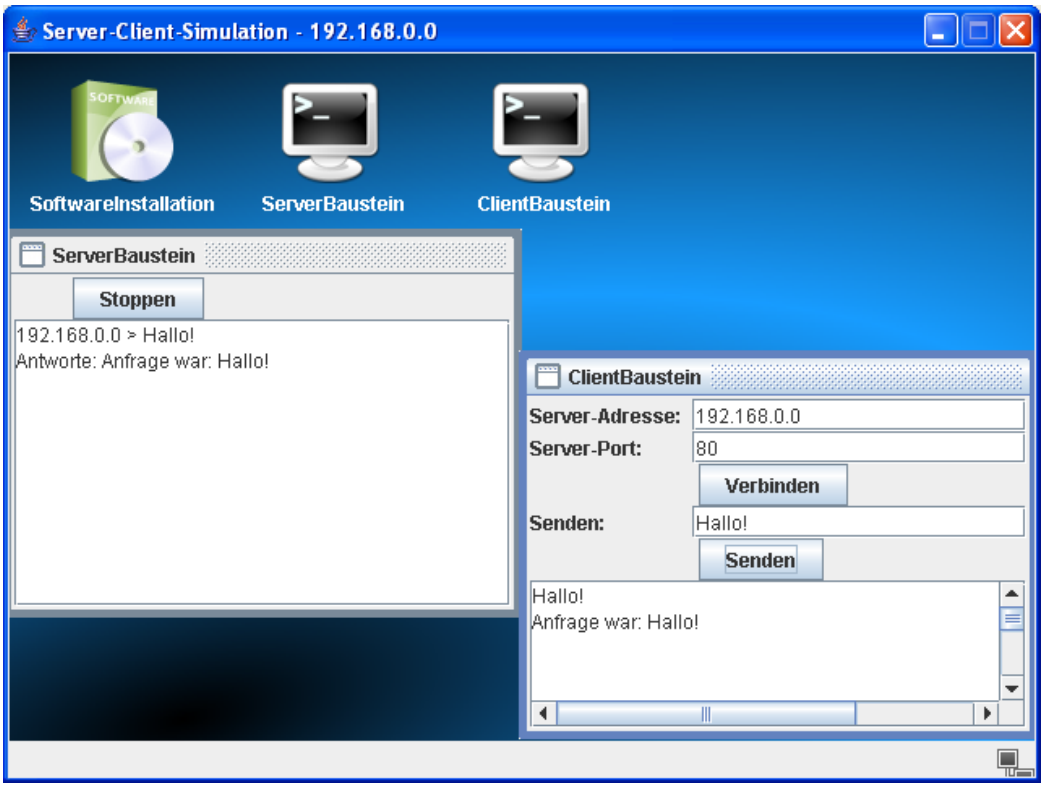


Abbildung 2: Beispiel für einen Server-Client-Baustein (GUI)

bereitgestellte Funktion von FILIUS. Der Assistent ermöglicht es dem Anwender, eigene Software zu entwickeln. Im ersten Schritt wird als Eingabe ein Klassenname verlangt. Es kann Server- oder Client-Klassen geben. Danach wird ein bespielhafter Quellcode aus vorgegebenen Attributen und Methodenrümpfen vorgegeben. Dieser Code kann aber nach belieben verändert

werden. Am Ende wird durch das *Kompilieren* eine neue Klasse erstellt.

3.1.2 Simulationsansicht

In der Simulationsansicht werden die im Entwurfsmodus entwickelten Netzwerkkomponenten mit Software bestückt. Sobald man von Entwurfsmodus in die Simulationsansicht wechselt verschwindet das Hardwarepanel und das Konfigurationsmenü. Man kann die Konfiguration ebenfalls über das Netzwerkstatusfenster im Simulationsmodus ändern. Der Simulationsmodus dient der Installation von Software und zum Benutzen der Hauptanwendungen.

Sobald eine Übertragung von Paketen über eine Verbindung (Kabel) läuft, fängt dieses an, grün zu blinken. Das kann nur im Simulationsmodus passieren. Programmintern hat jede Verbindung ein Attribut, welches besagt, ob es gerade aktiv ist (also benutzt wird). Sobald das vom Typ `Boolean` bestehende Attribut den Wert *true* annimmt, fängt das Kabel in der GUI an zu blinken.

Mit der rechten Maustaste ist es möglich das Kontextmenü der ausgewählten Komponente aufzurufen. Es enthält die Einträge *Neue eigene Software* und *Desktop anzeigen*. Darüber lässt sich ein neues Dialogfenster öffnen, indem wie der Anleitung in Abschnitt 3.1.2 beschrieben, eine oder mehrere Softwares installiert werden können. Durch Auswählen des Symbols *SoftwareInstallation* öffnet sich ein weiterer Dialog, in dem eine oder mehrere Programme auf dem Desktop installiert werden können. Dies geschieht einfach durch Auswahl und *Hinzufügen*. Der genaue Umgang mit den Programmen wird in den Abschnitten zu den Anwendungen 4 und Grundfunktionen 3.3 beschrieben.

Der *Geschwindigkeits-Regler* steuert die Ablaufgeschwindigkeit einer Simulation. Diese kann dynamisch zur Laufzeit verändert werden. Programmintern wird ein Faktor gesetzt, der als Multiplikator der Verzögerungskonstante beim Datenaustausch eingerechnet wird.

3.1.3 Nachrichtenkonzept

Damit die sehr detaillierte Realisierung vieler Bestandteile des Internets auch vom Benutzer beobachtet werden kann, wurde eine zusätzliche Funktion integriert, die es allen am Datenaustausch beteiligten Klassen, unabhängig von ihr jeweiligen Schicht, ermöglicht dem Benutzer mitzuteilen, was gerade geschieht. Dieses Konzept muss also sowohl für alle Klassen zugänglich sein, mit einer großen Menge von Nachrichten umgehen können und möglichst wenig Prozessorleistung benötigen, damit der Grad der Verfälschung auf Grund der

Beobachtung möglichst gering gehalten wird. Unser Ansatz hierfür ist eine Kombination der Entwurfsmuster *Singleton* und *Beobachter*.

Die Klasse `NachrichtenVerwaltung.java` ist ein *Singleton* und kann entsprechend von allen Klassen angesprochen werden, die eine Nachricht erzeugen möchten. Die Klassen müssen selbst entscheiden, ob es sich bei ihrer Nachricht um eine StatusNachricht, also eine Nachricht dir nur den Erzeuger selbst betrifft und auch nur entsprechende Informationen enthält, oder eine InteraktionNachricht, also eine Nachricht, die auf Grund des Datenaustauschs zweier Klassen erzeugt wird (zum Beispiel Webserver sendet Daten an den Webbrowser), handelt. Die Nachrichtenverwaltung nimmt diese Nachrichten entgegen, gibt jeder Nachricht eine eindeutige Nummer und speichert die Nachricht in einer Liste. Die GUI erzeugt nun Beobachter, für die `NachrichtenVerwaltung`. Diese Beobachter werden geweckt, falls eine neue Nachricht erzeugt wurde und prüfen nun, mit Hilfe einer Filterklasse, ob diese neue Nachricht für Sie von Bedeutung ist. Mit Hilfe dieser Filter kann der Benutzer nun ganz gezielt bestimmte Nachrichten gefiltert. Folgende Filterparameter wurden implementiert: *Absender* (z.B. Webserver), *Schicht* (nach Schichtemodell), *IP* des Absenders, *MAC-Adresse* des Absenders, *Hostname*.

Falls es sich bei der Nachricht um eine Interaktionsnachricht handelt, sind zudem folgende Filterparameter vorhanden: *Absender-Port, Empfänger, Empfänger-Port, Empfänger-IP-Adresse, Empfänger-MAC-Adresse*.

Diese Parameter können auch verknüpft werden, so unterstützt die Filterklasse die Operationen OR, AND und XOR.

Beispiel 1: Der Benutzer kann einen Beobachter so einstellen, dass er nur nach Nachrichten filtert, die eine bestimmte Sender-IP-Adresse und einen bestimmten Sender-Port hat (AND).

Beispiel 2: Der Benutzer kann einen Beobachter so einstellen, dass er nur nach Nachrichten filtert, die entweder einen bestimmten Empfänger-Port oder eine bestimmten Sender-Port haben (XOR).

Die sinnvolle Verwendung der Filterklasse obliegt allerdings dem Benutzer.

Zur vereinfachten Anwendung innerhalb der Klassen und Senkung des Programmieraufwandes wurden die Klassen `StatusNachrichtHelfer` und `InteraktionsNachrichtHelfer` geschaffen. Bei diesen Klassen handelt es sich eigentlich nur um Container, in denen bestimmte Informationen gespeichert werden, die sich typischerweise selten ändern. Dies sind zum Beispiel *Hostname* oder *Schicht*. Eine bestimme Klasse ist in der Regel einer bestimmten Schicht zugeordnet. Diese Zuordnung ändert sich normalerweise nicht. Diese Helferklassen sollen die Anzahl der Parameter, die beim Erzeugen einer neuen Nachricht anfallen auf ein Minimum reduzieren.

Weitere wichtige Klassen werden im Folgenden erklärt:

`Filter:` Mit Hilfe dieser Klasse filtert ein Beobachter Nachrichten.

`InteraktionsNachrichtHelfer:` Klasse zum vereinfachten Erzeugen von InteraktionsNachrichten.

`Nachricht:` `InteraktionsNachricht` und `StatusNachricht` erben von dieser Klasse. Sie enthält lediglich das Attribut *typ*, an Hand dessen später intern die Nachrichten auseinander gehalten werden können.

`NachrichtenVerwaltung:` Kern des Nachrichtenverwaltungs-Konzepts. Verwaltet die Nachrichten und wird von den Beobachtern beobachtet. Sie ist somit Schnittstelle zwischen den Nachrichten erzeugenden Klassen und der GUI.

`StatusNachrichtHelfer:` Klasse zum vereinfachten Erzeugen von StatusNachrichten.

Im Nachrichtendialog werden in tabellarischer Form alle im Netzwerk ablaufenden Vorgänge dokumentiert. Es wird unterschieden zwischen Statusnachrichten und Interaktionsnachrichten (Siehe Abbildung 3). Die Nachrichtenverwaltung stellt die Schnittstelle zwischen den detaillierten Abläufen innerhalb unseres Programms und der GUI dar. Hierbei gilt, dass egal was tatsächlich intern passiert, wenn es nicht als Nachricht dargestellt wird, wird der Schüler nichts davon sehen.

Eine Statusnachricht sollte immer dann erzeugt werden, wenn etwas passiert, bei dem es nur einen Beteiligten gibt. Ein Beispiel dafür wäre die Installation eines Webbrowser mit der Anweisung: `StatusNachricht sn = new StatusNachricht(WebBrowser, bs.getIpAdresse(), 1, bs.getFirstMac(), bs.getName())`
Ein StatusNachrichtHelfer dient zur erleichterten Bedienung der Nachrichtenverwaltung. Er kann folgende Werte speichern: `String absender`, `String ip`, `int schicht`, `String mac`, `String hostname;`. Dies dient dazu, falls sehr oft eine Statusnachricht erzeugt wird, sich aber bis auf die Nachricht nichts ändert, muss weniger in die Anweisung eingegeben werden: `StatusNachrichtHelfer snh = new StatusNachrichtHelfer(WebBrowser, bs.getIpAdresse(),1, bs.getFirstMac, bs.getName())`
`StatusNachricht sn = new StatusNachricht(snh, Webbrowser installiert)`

Eine InteraktionsNachricht soll immer dann erzeugt werden, wenn der Fall eintritt, dass es für den Anlass der Nachricht zwei Beteiligte gibt. Dies gilt

natürlich nicht nur für die Realität sondern auch für Versuche! Also nicht nur A sendet Daten nach B sondern auch A versucht Daten nach B zu senden. Im Gegensatz zur StatusNachricht sind hierbei einige Information mehr von Bedeutung. InteraktionsNachricht
`in = new InteraktionsNachricht (String absender, String senderip, int senderport, int schicht, String nachricht, String empfaenger, String empfaengerip, int empfaengerport, String sendermac, String empfaengermac, String hostname)`
Genauso wie ein Statusnachricht-Helfer soll der Interaktionsnachricht-Helfer einem beim Erzeugen der Interaktionsnachricht helfen. Logischerweise müssen hier natürlich auch einige Informationen übergeben werden.
`in = new InteraktionsNachrichtHelfer(String absender, String senderip, int senderport, int schicht, String empfaenger, String empfaengerip, int empfaengerport, String senderMac, String empfaengermac, String hostname)`

Die korrekte Verwendung der Nachrichtenverwaltung wird im Folgenden kurz beschrieben. Zunächst werden entsprechende Variablen deklariert und eine Referenz auf die Nachrichtenverwaltung geholt, die ja bekanntlich ein Singelton ist: `private StatusNachricht sn;`; `private InteraktionsNachricht in;`, `private StatusNachrichtHelfer snh;` `private InteraktionsNachrichtHelfer inh;`; `NachrichtenVerwaltung nv = NachrichtenVerwaltung.getNachrichtenverwaltung()`.

Je nach Bedarf und Anwendung wird im Konstruktor bereits der oder die Helfer erzeugt. Dies kann wie folgt aussehen:

`snh = new StatusNachrichtHelfer(„DHCPClient\, bs.getIpAdresse(), 1,bs.getFirstMac(),bs.getName());`

Danach kann der eigentliche Nachrichtenverkehr gestartet werden:
`sn = new StatusNachricht(snh, DHCPClient gestartet);`
`nv.neueStatusNachricht(sn);`

Besonders wichtig ist, dass jedesmal eine neue Statusnachricht erzeugt wird. Es soll nicht so sein, dass einfach nur die Attribute der Nachricht neu gesetzt werden mit `setNachricht(String nachricht)`. Nur beim Erzeugen werden noch einige weitere Werte, wie z.B. eine ID und vor allem der Zeitpunkt des Erzeugens gesetzt. Wird also eine veraltete Nachricht wiederverwendet, hätte sie die falsche ID und vor allem einen falschen Erzeugungszeitpunkt. Dies sollten möglichst unterlassen werden. Die Schaltfläche *Nachrichtenfenster* öffnet den Dialog, der alle Status- und Interaktionsnachrichten des Systems verwaltet. Die beiden Nachrichtenarten werden im Abschnitt 3.1 näher erläutert.

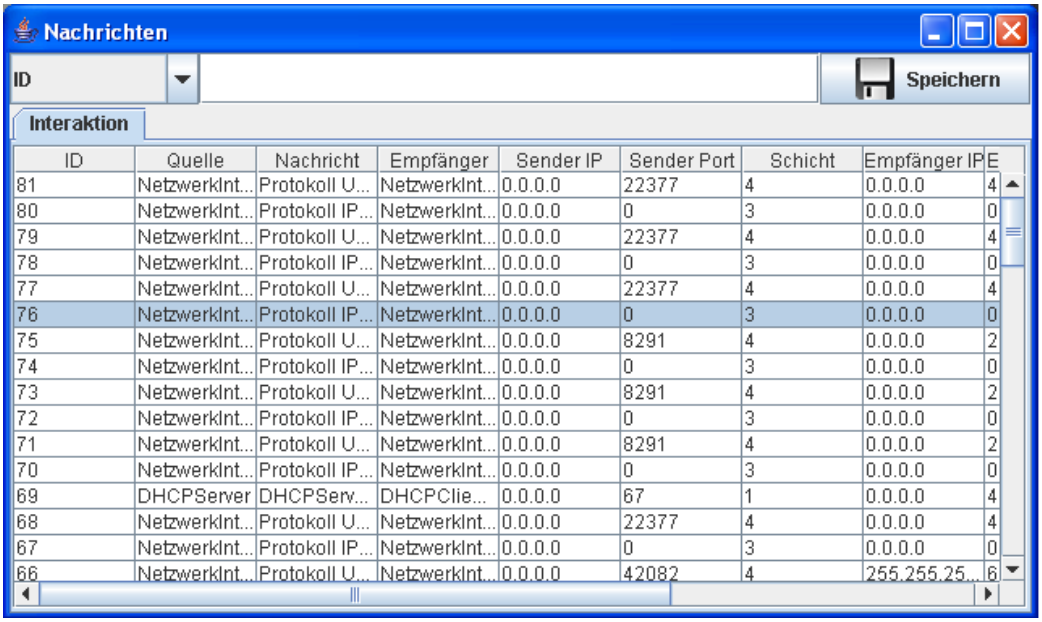


Abbildung 3: Nachrichten-Fenster

3.2 Virtualisierte Software

3.2.1 Internetanwendungen

Bisher sind drei Internetanwendungen implementiert worden:

- World Wide Web mit Hypertext Transfer Protocol
- E-Mail mit Simple Mail Transfer Protocol und Post Office Protocol Version 3
- Internetbasierter Dateiaustausch via Peer-to-Peer (Protokoll angelehnt an Gnutella)

Im Anwendungsbeispiel World Wide Web hat der Benutzer die Möglichkeit ein Netzwerk zu entwerfen, das wie sein reales Vorbild WWW funktioniert. Er kann zum Beispiel einen Rechner als Server deklarieren und darauf einen Webserver installieren (siehe Abschnitt 3.1.2). Der Webbrowser stellt zwar eine Standardseite zur Verfügung, aber wahlweise können auch eigene HTML-Dokumente erstellt oder mit Hilfe eines Datei-Importierers geladen werden. Als Klienten können dann ein oder mehrere Rechner oder Laptops jeweils mit

der Software Webbrowser ausgestattet werden. Diese können dann auf den Server zugreifen, und bekommen seine Webseite angezeigt. Voraussetzung ist natürlich richtig konfigurierte, und miteinander verbundene Hardware.

Das Anwendungsbeispiel E-Mail versetzt den Benutzer von FILIUS in die Lage, ein Netzwerk mit E-Mail-Server und E-Mail-Clients aufzusetzen. Dazu wird ein Netzwerk im Entwurfsmodus erstellt, und ein Rechner ausgewählt, der als Server fungieren soll. Dieser wird mit der Software E-Mail-Server ausgestattet (siehe Abschnitt 3.1.2). Ein Server hat Benutzerkonten, und stellt diese über seine Dienste für Clients zur Verfügung. Auf allen anderen Rechnern, werden E-Mail-Anwendungen installiert. Damit können E-Mails geschreiben, verschickt, und vom Server abgerufen werden, wie in einer realen E-Mail-Anwendung.

Das letzte, aber wohl gleichzeitig komplizierteste Anwendungsbeispiel, ist der internetbasierte Dateiaustausch über ein Peer-to-Peer-Netzwerk. Ein Vorbild aus dem Alltag ist Gnutella. Wie dort kann der Benutzer von FILIUS beliebig viele Rechner vernetzen und auf jedem eine Peer-to-Peer-Anwendung installieren (siehe Kapitel 3.1.2). Es können auf jedem Rechner virtuelle Dateien erstellt werden. Von den anderen, gleichgestellten Rechnern kann eine Suche nach diesen Dateien gestartet werden. Nach erfolgreicher Suche können die Dateien angefordert und angeschaut werden.

3.2.2 Das Betriebssystem

Um wichtige Systemfunktionen wie zum Beispiel Datei- und Netzwerkzugriff für einzelne Anwendungen bereitzustellen, wurde die Klasse `Betriebssystem` erstellt. Sie bündelt die gängigsten Aufrufe zum Erzeugen und Bearbeiten von Verzeichnissen und Dateien, stellt Sockets zur Verfügung und ermöglicht den Zugriff auf Dienste wie DNS und DHCP. Das Betriebssystem erbt von der Klasse Systemsoftware, da jede Hardware in FILIUS unterschiedliche Typen von Systemsoftware besitzen können. Das Betriebssystem ist dem Hardware-Typ `Host` vorbehalten, während zum Beispiel der Vermittlungsrechner (Router) eine Firmware besitzt. Alle Anwendungen werden über das Betriebssystem installiert, und auch gestartet. Die Arbeitsfläche auf der Anwendungen grafisch dargestellt werden können, wird ebenfalls hier verwaltet. Eine Übersicht über die enthaltenen Attribute ist Abbildung 4 zu entnehmen. Nutzdaten der simulierten Programme werden im jeweiligen simulierten Betriebssystem in einem Verzeichnisbaum gespeichert. Hierfür werden DefaultMutableTreeNodes verwendet. Gründe dafür sind folgende:

- Sie lassen sich leicht in einem Verzeichnisbaum darstellen, das ist zum Beispiel wichtig für den Datei-Explorer

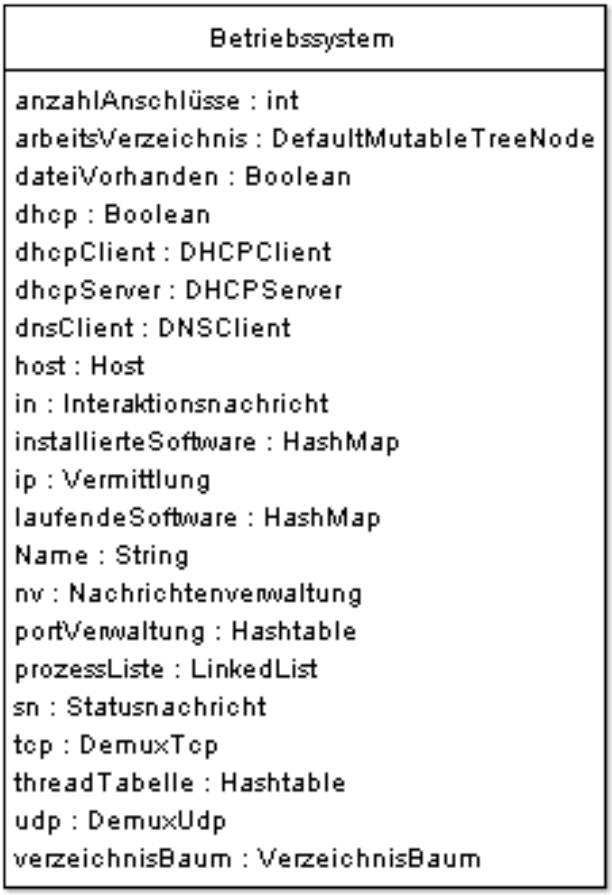


Abbildung 4: UML-Diagramm zu den Attributen des Betriebssystems

- Sie können mit *XMLEncoder* serialisiert werden

Zu Beginn der Planung dieses Projekts war es vorgesehen mit realen Dateien zu arbeiten. Im Laufe der Überlegungen haben sich dann aber folgende Vorteile einer virtuellen Dateistruktur abgezeichnet:

- keine „Dateileichen“ auf dem Rechner,
- keine Probleme mit anderen Programmen, die auf Dateien zugreifen,
- Speicherung vereinfacht,
- ermöglicht Speicherung aller (auch binärer) Dateien in den XML-Dateien durch Base64 Codierung.

Dem gegenüber steht lediglich der Nachteil, dass kein externer Zugriff stattfinden kann. Dieser Nachteil ist aber sogar noch eingeschränkt durch den Datei-Importierer.

Die Verzeichnisstruktur und die darin enthaltenen Dateien sind in Bäumen gespeichert, welche aus Knoten vom Typ `DefaultMutableTreeNode` bestehen. Der Zugriff erfolgt durch vom Betriebssystem von FILIUS bereitgestellte Funktionen `addOrdner()`, `addDatei()`, usw. Es ist jeweils möglich eine Pfadangabe in Form eines `String`, oder direkt einen Knoten als Stammverzeichnis anzugeben.

Es ist eine einfache Nutzung durch `JTree` aus dem Paket `swing` von Java möglich. Darüber hinaus wird das Dateisystem über einen Explorer verwaltet. Ein Datei-Importierer sorgt dafür, dass reelle Dateien importiert werden können. Anwendungen die auf Dateien zugreifen müssen, sollten sich jedoch ein Arbeitsverzeichnis anlegen. Das virtuelle Dateisystem bringt also zusammenfassend mehr Vorteile als ein reales Pendant. Der Zugriff erfolgt über einfache Funktionen des virtuellen Betriebssystems.

3.3 Weitere Grundfunktionen

Nach Auswählen des Symbols *Neu* wird der Benuter zunächst durch einen Dialog gefragt, ob er wirklich ein neues Projekt beginnen möchte. Die Funktion *Öffnen* wird wie von grafischen Betriebssystemen bekannt benutzt. Klicken Sie die Schaltfläche an, und wählen in dem erscheinenden Datei-Explorer ein zuvor erstelltes Netzwerk in Form einer XML-Datei. Wenn Sie eine ordnungsgemäß gespeicherte Datei gewählt haben, wird dieses Netzwerk sofort mit sämtlichen eingestellten Konfigurationen und installierter Software angezeigt. Wenn Sie ein Netzwerk aufgebaut, konfiguriert und einzelne Rechner mit Software versehen haben, können Sie es mit *Speichern* sichern. Es muss jedoch kein funktionstüchtiges Netzwerk sein. Es können auch Bruchstücke eines Netzes gespeichert werden, zum Beispiel wenn es später fertig gemacht werden soll.

Die Entwickler von FILIUS haben sich für die Speicherung in XML entschieden. Zur Entscheidung, die gegen die in Java integrierte Möglichkeit Objekte direkt durch Serialisierung zu speichern ausfiel, haben folgende Aspekte beigetragen:

1. Objektserialisierung ist versionsabhängig, dass heißt nach Änderungen am Programm sind alte Speicherstände nicht mehr verwendbar.

2. XML ist versionsunabhängig, hat eine für den Menschen relativ gut lesbare Struktur und kann daher auch mit anderen Werkzeugen bearbeitet werden.

Zum Speichern wurde der XML Encoder aus dem Java-Beans Paket genutzt. Um die einzelnen Einstellungen (IP-Adresse, Gateway, usw.) für die jeweiligen Hardware-Komponenten zentral speichern zu können, wurden diese in Klassen ausgelagert (z.B. `KonfigurationHost`), die von KonfigurationHardware erben. Nach dem Laden wird die Funktion `aktualisiereHardware()` in der Konfiguration aufgerufen, welche die Einstellungen auf die eigentliche Hardware überträgt.

Die Schaltflächen *Entwurf* und *Simulation* sind zum Hin- und Herwechseln zwischen den beiden Modi. Siehe Abschnitt Entwurfsansicht 3.1.1 und Simulationsmodus 3.1.2.

Die Oberfläche des Datei-Explorers ist ähnlich wie bekannte grafische Produkte gestaltet. In einer Baumstruktur können Verzeichnisse und Unterverzeichnisse angelegt werden. In diese Verzeichnisse können später zum Beispiel eigene Netzwerke gespeichert und selbsterstellte HTML-Seiten abgelegt werden. Über den Datei-Importierer, der in den Explorer integriert ist, können eigene HTML-Seiten in FILIUS eingebunden werden. Der Importierer funktioniert so, wie man es auch von einem Datei-Explorer aus grafischen Betriebsystemen gewöhnt ist. Man wählt die Datei einfach aus, und läd sie in FILIUS ein. Mit dem Datei-Explorer können reale Dateien in FILIUS importiert werden (siehe Abbildung 5). Die Realdateien werden als Binärstream mit einer Base64-Kodierung formatiert. Anschließend könnne sue in der internen Dateistruktur von FILIUS benutzt werden, und werden auch mit gespeichert wenn ein Projekt vom Anwender gesichert wird. Denkbar wäre das Importieren einer realen HTML-Seite *index.html* oder eines Liedes als *lied.mp3*. Der Terminal ähnelt der Eingabeaufforderung von bekannten grafischen Betriebssystemen. Im Terminal können durch Texteingabe einige Aktionen ausgeführt werden die sonst über die Benutzungsoberfläche getätigt werden würde. Es kann in Verzeichnisstrukturen navigiert und Programme können aufgerufen werden. Das Terminal ruft durch Eingabe einer Zeichenkette dieselben Funktionen auf, die sonst von der Benutzungsoberfläche aus geschehen würden.
.

Sinn und Zweck des Texteditors ist es hauptsächlich Dateien zu erstellen, die dann später in den Anwendungen benutzt werden können. Dies kann eine eigene HTML-Seite für den Webserver, oder eine Datei irgendeiner Endung zur Suche in der Peer-to-Peer-Anwendung sein.

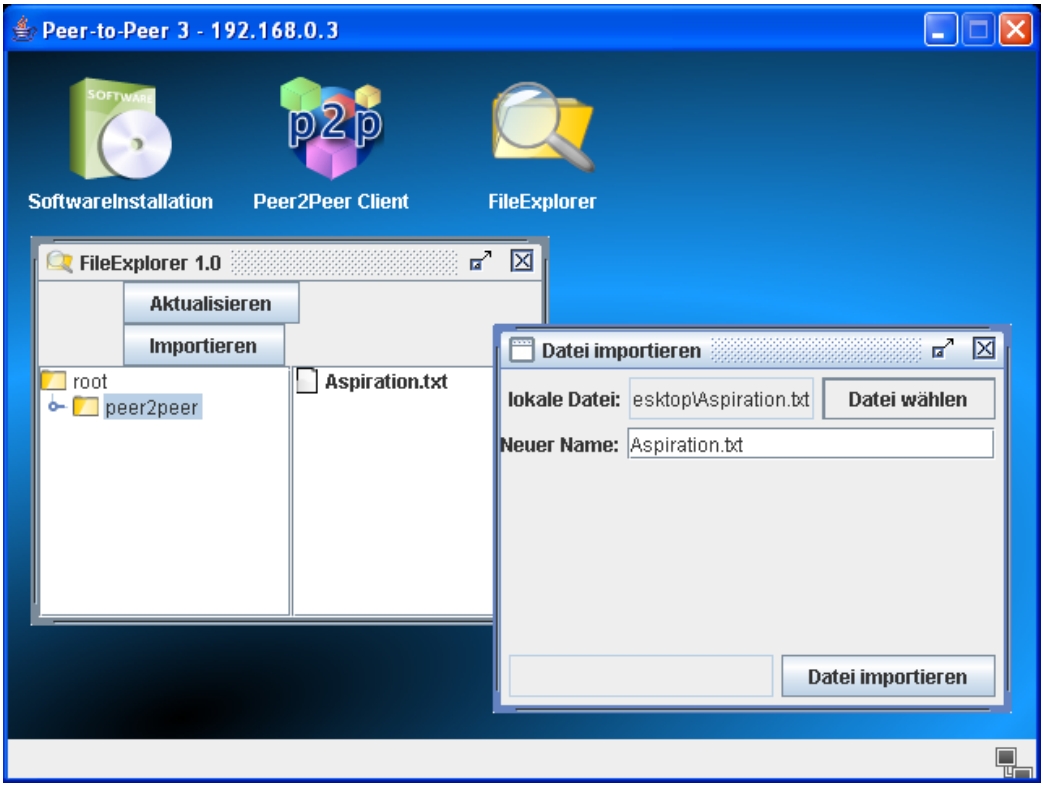


Abbildung 5: Der Datei-Explorer

Programmintern werden alle Attribute einer Hardwarekomponente nach folgendem Schema erstellt. Die erste Spalte gibt den Namen des Attributs an. Danach folgen Typ, Länge des Feldes in der GUI, interner Datentyp, Editierbarkeit und der aktuelle Wert.

```
IP-Adresse,ipAdresse,15,String,editable,000.000.000.000;
Name,name,20,String,editable,nn;
Anzahl der Anschlüsse,anzahlAnschluesse,2,int,neditable,4;
Gnutella,gnutella,1,boolean,editable,false;
```

Im ersten Beispiel der Aufzählung handelt es sich um die Konfigurationszeile eines Textfeldes für die Eingabe einer IP-Adresse. Dann folgt der Name der Variablen. Dieser ist wichtig, weil dieser String genutzt wird um die Attribute auszulesen und wieder zu setzen. Wenn man hier also ipAdresse

eingibt, dann wird `getipAdresse()` und `setipAdresse()` im Quellcode aufgerufen, weil diese Aufrufe dynamisch gestaltet sind. `15` steht für die Länge des erwarteten Eingabe-Strings, `editable` bedeutet, dass das Textfeld editierbar ist (also eine Benutzereingabe erfolgen kann), `000.000.000.000` legt die genaue Struktur fest, wie die Eingabe erfolgen muss (also je Block eine dreistellige Zahl). Dort steht in der Regel ein Vorgabewert. Wenn dort also 192.168.123.100 steht, dann wird im Eigenschaftsfenster dieser Wert als Standardkonfiguration bei jedem Item angelegt. Analog sind die drei anderen Beispiele für den Namen und die Anzahl der Anschlüsse einer Hardware (int steht für Integerwert), und die Benennung einer Anwendungs-Software (false ist ein Boolescher Wert).

4 Internetanwendungen

In den folgenden Abschnitten wird erläutert, wie die drei Internetanwendungen von FILIUS funktionieren und vor allem wie sie umgesetzt wurden. Es wird dazu speziell auf alle Komponenten der Anwendungen eingegangen.

4.1 E-Mail

Zur Konfiguration eines E-Mail-Servers ist die Eingabe eines Domainnamen erforderlich. Es können separat SMTP und POP gestartet und gestoppt werden. Zum Anlegen eines neuen Kontos sind Benutzername und Passwort notwendig.

Das schreiben einer E-Mail geschieht mit den gleichen Eingabeparametern wie bei einem realen E-Mail-Programm. Die Felder An:, CC:, und BCC erwarten die Eingabe einer E-Mail-Adresse. Der Text selbst wird in das große Textfeld geschrieben. Mit *Senden* wird der verfasste Text losgeschickt. In einer Auswahlbox werden die in den Einstellungen angelegten E-Mail-Konten zur Wiederverwendung gespeichert.

Zur Erstellung eines neuen Kontos sind folgende Angaben notwendig: E-Mail-Adresse, POP-Server, POP-Port, SMTP-Server, SMTP-Port, Benutzername und Passwort. Wenn alle Daten richtig eingegeben wurden, kann mit *Erstellen* das Konto angelegt werden. Es wird sofort in der Liste aller Konten erscheinen. In Abbildung 6 ist der Konfigurationsdialog dargestellt.

4.1.1 E-Mail-Server

Der E-Mail-Server dient dazu, Anfragen von einem E-Mail-Client, der später noch eingehender erläutert wird, zu verarbeiten. Dabei hat er grundsätzlich verschiedene Aufgaben zu übernehmen, was zusätzlich in seiner Implementierungsstruktur erkennbar wird. Der E-Mail-Server implementiert zwei weitere Unterklassen, namentlich `SMTPServer` und `POPServer`. Über diese Unterklassen findet der weitere Datenaustausch statt. Per SMTP, und damit über den SMTP-Server ist es für einen E-Mail-Client möglich, E-Mails zu versenden. Mittels POP, kann ein Nutzer durch einen Client die Nachrichten, in seinem E-Mail-Postfach, das er auf dem E-Mail-Server zuvor angelegt hat, abrufen. So ist es möglich, sich Nachrichten im Überblick anzuschauen, einzelne Mails abzurufen, E-Mails zu löschen, usw. Der dazu erforderliche Datenaustausch läuft dabei folgendermaßen ab: ein E-Mail-Client sendet zunächst eine Verbindungsanfrage an den E-Mail-Server. In dem Programm ist dies

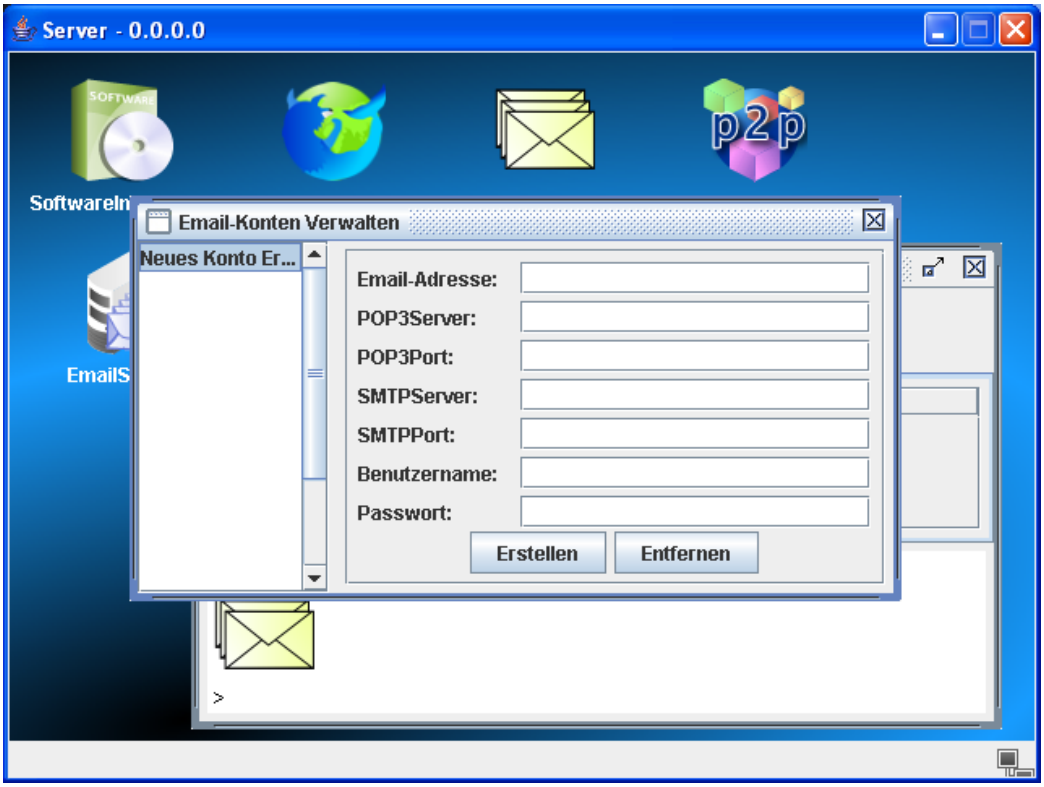


Abbildung 6: Konfiguration eines E-Mail-Kontos

derart realisiert, dass der E-Mail-Client eine definierte Verbindungsanfrage entweder an den SMTP-Server oder den POP-Server, die Unterklassen des E-Mail-Servers, sendet. Beide Klassen haben dieselbe IP-Adresse, unterscheiden aber die Ports, auf denen sie auf eine Verbindungsanfrage warten. Der SMTP-Server hört Port 25 ab, der POP-Server Port 110. Erkennt nun entweder der SMTP-Server oder der POP-Server eine Verbindungsanfrage eines E-Mail-Clients, so erstellt dieser sich für den Verlauf der weiteren Kommunikation mit genau diesem Client einen Mitarbeiter, der die Interaktion mit dem E-Mail-Client durchführt. Auf diesem Wege wird ein neuer Port vereinbart, über den der folgende Datenaustausch stattfinden wird und der ursprüngliche Port wieder für neue Verbindungsanfragen frei ist. Das wird dadurch realisiert, dass den Anwendungen E-Mail-Client und dem jeweiligen E-Mail-Server, an den die Verbindungsanfrage gestellt wurde, Sockets übergeben werden, die die Schnittstelle zu den Transportprotokollen darstellen, und so den Datenaustausch zwischen den Prozessen realisieren. Der Socket, der dem Mitarbeiter des Servers übergeben wurde, handelt mit dem

Client diesen oben genannten Port aus. Wurde eine neue Verbindung auf diesem Wege hergestellt, kann der eigentliche Datenaustausch mit dem Anwendungsprotokoll beginnen.

Wie bereits angeklungen , ist der E-Mail-Server die Oberklasse von SMTP- und POP-Server. Er ist weiterhin im eigentlichen Sinne eine Software, und erbt daher sämtliche Funktionalitäten der Klasse Software. Der E-Mail-Server organisiert das starten und stoppen seiner Unterklassen und verwaltet alle Benutzerkonten. Dazu bietet er verschiedene Methoden an, die es gestatten, beispielsweise neue Benutzerkonten bei einem E-Mail-Server hinzuzufügen, und zu löschen. Daneben ist er auch für die persistente Speicherung und das anschließende Laden der Konten verantwortlich, also für ihre Verfügbarkeit auch über das Programmende hinweg. Zusätzlich erlaubt er seinen Unterklassen durch sich den Zugriff auf den Host oder das Betriebssystem.

Der E-Mail-Server dient dem Versand und der Weiterleitung von E-Mails. Die hier eingesetzte Technik unterscheidet sich jedoch geringfügig von der in der Realität bereitgestellten Funktionalität. Während in der Realität beim SMTP-Server noch zwischen Mail Submission Agent (MSA) und Mail Transfer Agent (MTA) unterschieden wird, die zudem auch auf unterschiedlichen Ports auf Verbindungsanfragen warten (MSA auf Port 587, MTA auf 25), übernimmt der in diesem Projekt implementierte SMTP-Server beide Funktionalitäten und überwacht zudem nur den Port 25. Dies entspricht der ursprünglichen Implementierung von SMTP, bei dem ein Server die Aufgaben von MSA und MTA übernahm. Der hier implementierte SMTP-Server nutzt im Wesentlichen die Funktionen, wie sie die Request for Comments [RFC 2821] beschrieben sind. Es ist zu beachten, dass keine neuen Implementierungen wie beispielsweise Authentifizierung realisiert wurden. SMTP stützt sich, wie auch POP auf menschenlesbare Nachrichten gegenüber dem Austausch von binär codierter Befehle, die genutzt werden, um mit einem Server zu kommunizieren. Diese Befehle, die eigentlich von einem Mail User Agent (MUA) unsichtbar für einen Benutzer genutzt werden, können auch von einem Anwender in einem Befehlszeilenfenster mittels beispielsweise Telnet per Hand eingegeben werden. Folgende Befehle des Simple Mail Transfer Protocols wurden implementiert:

HELO: Dient der Identifizierung des Klienten beim Server.

MAIL FROM: Dieser Befehl wird genutzt, um die Übertragung einer Mail zu initiieren.

RCPT TO: Mit diesem Befehl wird der Empfänger einer E-Mail identifiziert. Entgegen der eigentlichen Strategie, diesen Befehl bei mehr als

einem Empfänger mehr als einmal aufzurufen, wird in diesem Projekt der Befehl nur einmal aufgerufen, jedoch enthält der übertragene String alle Empfänger-Adressen.

DATA: Nach diesem Befehl folgt der Datenteil der E-Mail ohne weitere Benachrichtigung. Der Empfänger antwortet auf diesen Befehl normalerweise mit „354“ und behandelt alle nachfolgenden Zeilen als reinen Text, bis die Email mit „<CRLF>.<CRLF>“geschlossen wird.

QUIT: Zwingt des Empfänger der Daten ein „OK“ zu senden und den Transportkanal zu schließen.

Dies sind zugleich die nötigen Befehle für eine einfache Kommunikation zwischen Client und SMTP-Server und zum Versand einer E-Mail. Multimedia-Erweiterungen fanden keinen Eingang in die Realisierung der E-Mail-Funktionalität. Ist eine E-Mail von einem E-Mail-Client an den E-Mail-Server, bei dem der Anwender ein E-Mail-Konto besitzt, übermittelt worden, beginnt die Verarbeitung der E-Mail. Der SMTP-Server überprüft die E-Mail zunächst. Dabei kann es verschiedene Möglichkeiten geben, nach denen Verfahren werden muss. Die einfachste Möglichkeit ist dass die E-Mail, die ihn erreicht hat, an ihn gerichtet ist, und keine weiteren Empfänger hat. Eine weitere Möglichkeit ist, dass die E-Mail an ihn adressiert ist, aber zusätzlich an weitere Empfänger weitergeleitet wird. Oder es kann der Fall eintreten, dass die E-Mail nicht an ihn adressiert ist, und weitergeleitet werden muss. Um herauszufinden, wie der SMTP-Server verfahren muss, wird der angestrebte Empfänger untersucht. Findet er eine Adresse, die mit einem eigenen Konto übereinstimmt, so gehört die E-Mail in das Postfach dieses Nutzers. Eine Kopie dieser E-Mail wird in seiner Nachrichtenliste gespeichert, zugleich der Name aus der Liste der Empfänger, an die die Nachricht weitergeleitet werden muss, gelöscht. In allen Fällen, in denen eine angekommene E-Mail weitergeleitet werden muss erfolgt anschließend ein erneuter Verbindungsaufbau dieses SMTP-Servers mit dem nächsten SMTP-Server, um die E-Mail der richtigen Destination zuzuführen. Damit er weiß, wohin die E-Mail weiterzuleiten ist, wird der Domainname der jeweiligen Empfänger-Adresse mittels Domin Name Service (DNS) aufgelöst. So ist im ersten Fall die E-Mail nicht weiterzuleiten. Eine Kopie der E-Mail wird gespeichert, und da die Liste der Empfänger, keine weiteren Elemente enthält, ist die Verarbeitung der E-Mail abgeschlossen. Im zweiten und dritten Fall ist die E-Mail entsprechend weiterzuleiten nach dem vorgegebenen Schema. Das Post Office Protocol Version 3 (POP3) ist ein Protokoll, mit dem ein E-Mail-Client interagieren kann, um seine dort gespeicherten E-Mails zu administrieren.

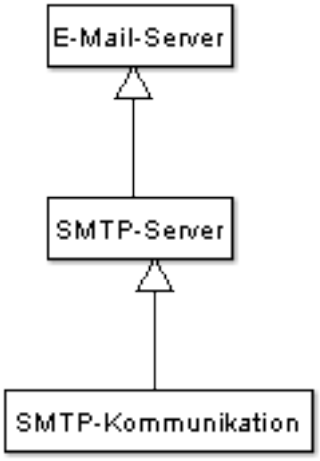


Abbildung 7: SMTP-Server

Dazu wird das Übertragungsprotokoll POP3 verwendet. In Analogie zum SMTP-Server lauscht auch der POP-Server auf einem Port, allerdings auf 110. Die Realisierung der Programmstruktur des POP-Servers gleicht der des SMTP-Servers. Der POP-Server lauscht auf Port 110, um eingehende Anfragen auf diesem Port entgegenzunehmen. Anschließend wird der Datenaustausch, und somit die Programmlogik auf den Thread `POPKommunikation` ausgelagert, der dann u. a. einen Socket mit einem neuen Port für den folgenden Datenaustausch mit nur dem Client, der die Anfrage gestellt hat, mit übergeben bekommt. Der POP-Server empfängt seine Anweisungen, ebenso wie es beim SMTP-Server der Fall war, in Form von Strings, die in der Request for Comments [RFC 1939] festgelegt sind. Im Rahmen dieses Projektes wurden die folgenden Befehle realisiert:

USER: Der Befehl gefolgt von einem Attribut dient der Eingabe des Benutzernamens, damit sich der Benutzer anmelden kann.

PASS: Wie USER auch, dient dazu sich mit dem darauf folgenden Attribut anzumelden, indem man das Passwort (seines Benutzerzugangs auf dem entsprechenden E-Mail-Server) eingibt.

STAT: ohne weitere Attribute liefert den Status der Mailbox, d. h. Anzahl der E-Mails, sowie die Größe der Nachrichten in Bytes.

LIST: implementiert die Funktion, sich wahlweise eine oder alle E-Mails, die gespeichert wurden, anzuzeigen zu lassen. Dabei kann wahlweise ein Integer-Wert als Attribut mitgegeben werden, der dann den Index

der E-Mail kennzeichnet, die aufgelistet werden soll. Bei dem Auflisten der E-Mails wird zuvor noch einmal der Status der Mailbox, wie er bei STAT einzusehen ist, angezeigt, bevor die eigentliche(n) E-Mail(s) aufgelistet werden. Zwar ist es möglich, sich ohne ein vorheriges ausführen von STAT bestimmte E-Mails anzeigen zu lassen, jedoch kann man dann nicht wissen, welche E-Mail welchen Index besitzt.

RETR: Kennzeichnet die Funktion sich eine E-Mail mit dem Index, der als Attribut mit übergeben wird, abzurufen.

DELE: Wird dieser Befehl ausgeführt, wird eine E-Mail, die dem Index, der als Attribut mitgeliefert wird, entspricht, als zu Löschen markiert.

RSET: Mit diesem Befehl werden die vorhergehenden Befehle, wie etwa zuvor mittels DELE als zu Löschen markierte E-Mails, rückgängig gemacht.

NOOP: Dieser Befehl provoziert eine Antwort vom Server. Damit ist zu erkennen, ob die Sitzung noch aktiv ist.

QUIT: Beendet die Sitzung. Die E-Mails, die markiert wurden, werden nun gelöscht.

Mit diesen soeben dargestellten Befehlen realisiert der POP-Server seine Funktionalität. Diese Befehle müssen von einem E-Mail-Client gesendet werden, um anschließend vom POP-Server interpretiert zu werden. Der POP-Server hat dabei verschiedene Möglichkeiten zu antworten. Im Rahmen dieses Projektes wurde die Konvention übernommen, dass als positive Antwort ein `+OK` gesendet wird, vorausgesetzt, es wurde keine Anfrage auf Datenübermittlung (wie etwa bei `RETR`, wo die Gegenstelle die E-Mail übermittelt haben möchte) gesendet. Im Falle eines ungültigen oder unbekannten Befehls wird dahingegen ein `-ERR` gesendet, das gegebenenfalls um eine Nachricht wie beispielsweise `unknown command` erweitert werden kann.

4.1.2 E-Mail-Anwendung

Diese Klasse beinhaltet die Programmlogik sowohl für den Datenaustausch zwischen ihm und einem SMTP-Server, als auch zwischen ihm und einem POP-Server. Dies gestaltet sich derart, dass für den Datenaustausch mit dem SMTP-Server, der den Versand von E-Mails sicher stellt, lediglich eine Funktion von zentraler Bedeutung ist, während die übrigen, diesem Teil

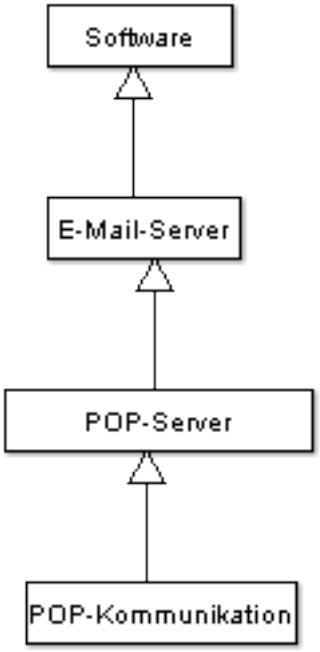


Abbildung 8: UML E-Mail-Server und POP

des Programms zugehörigen Methoden lediglich unterstützende Funktionalität bereitstellen und so auch Anwendung in dieser zentralen Methode für die Kommunikation mit dem SMTP-Server finden. Diese Klasse hat die Syntax `versendeEmail(Email email)`. Dieser Methode wird ein Parameter vom Typ `EMail` übergeben, das die E-Mail an sich ist, die versendet werden soll. Da der Versand einer E-Mail mit dem dazugehörigen Datenaustausch zwischen E-Mail-Client und SMTP-Server nach einem festen Schema abläuft, beinhaltet diese Methode alle nötigen Schritte zum Versand einer E-Mail in der richtigen Reihenfolge. Diese kann nur vollständig ablaufen, wenn alle einzelnen Zwischenschritte ohne Fehler abgearbeitet werden können. Da diese Funktionalitäten in einzelne, eigenständige Methoden ausgelagert wurden, ist es möglich, dass der Versand einer E-Mail auch schrittweise und eigenhändig von einem Anwender ausgeführt werden kann, ohne dass das Programm dies automatisieren würde. Dann muss der Anwender aber auch selbst Sorge dafür tragen, dass der Ablauf des Datenaustauschs in der richtigen Reihenfolge abläuft. Dieser sieht wie folgt aus: Zunächst wird der Absender übermittelt, anschließend der, bzw. die Empfänger, die sich als ein langer String, getrennt durch Kommata dargestellt werden. Daran schließt sich der Datenteil an, dessen Ende dadurch gekennzeichnet ist, dass ihm ein `<CRLF>.<CRLF>` nachfolgt. Zuletzt wird ein QUIT gesendet. Das bedeutet, dass die Nachricht

vollständig übermittelt wurde. Es ist darauf hinzuweisen, dass sich im Rahmen dieser Implementierung die Angaben von MAIL FROM: sowie RCPT TO: aus den Attributen Absender und Empfänger der E-Mail ableiten. Des Weiteren bietet die Klasse der E-Mail-Anwendung die nötigen Operationen zum Datenaustausch mit dem POP-Server. So kann er sich in geeigneter Weise bei diesem Server anmelden und authentifizieren, und im weiteren Verlauf sein Postfach administrieren. Hier gibt es keinerlei Besonderheiten. Er analog zur Funktionalität des POP-Servers aufgebaut. Die Funktionen, die dort angeboten werden, können durch Methoden in der E-Mail-Anwendung, angesprochen werden. Den Schluss bilden die Methoden `kontaktHinzufuegen()`, sowie `kontaktLoeschen()`, die es erlauben, dem in der E-Mail-Anwendung integrierten Adressbuch neue Kontakte hinzuzufügen, beziehungsweise alte zu löschen.

4.2 World Wide Web

Wechselt man nach Erstellung eines Netzwerks im Entwurfsmodus in den Simulationsmodus, so ist es möglich für jeden Rechner eine Arbeitsfläche zu öffnen. Im sich öffnenden Dialog bekommt man durch Auswahl des Software-Symbols eine Liste aller installierbarer Software angezeigt. Es ist nun ein Webserver auf dem Host installiert. Durch Auswahl dieses Symbols und anschließendem *Starten* wird der Server zum laufen gebracht. Ab diesem Zeitpunkt können Anfragen von anderen Host die einen Webbrowser installiert haben gestellt werden.

Webserver und Webbrowser können in einigen Details konfiguriert werden. Im Simulationsmodus (Abschnitt 3.1.2) kann der Benutzer dann mit dem Webbrowser eine Anfrage nach einer bestimmte Webseite tätigen. Diese Eingabe kann direkt per IP-Adresse geschehen. Das ist aber in der Realität eher selten, da die IP-Adresse einer Webseite meist nicht bekannt ist, beziehungsweise der Anwender sich eine URL besser merken kann. In diesem Fall, also Eingabe einer URL mit Domainname braucht es natürlich den Dienst DNS (Domain Name Service, Abschnitt 5.4.2). Der Webserver bekommt nun also eine Anfrage und prüft ob er für diese zuständig ist. Wenn ja, bearbeitet er die Anfrage und schickt die Daten per HTTP-String an den anfragenden Browser zurück. Der Browser, kann den Quellcode in für den Menschen lesbarer Weise anzeigen.

Im Browserfenster ist erst einmal die Standard-Seite des Browsers und eine Eingabeleiste wie bei einem realen Browser zu sehen. Wenn dort eine URL eingegeben wird, hängt das Ergebnis dieser Suche davon ab, ob auf einem

Zielrechner ein Webserver installiert ist. Ist tatsächlich auf dem Zielrechner ein Webserver installiert, so wird die Seite angezeigt. Die Anfrage kann in verschiedenen Formen eingegeben werden. Bei der ersten Möglichkeit erfolgt die Eingabe über die IP-Adresse. Dazu muss natürlich auch die IP-Adresse des gewünschten Servers bekannt sein. Die andere Möglichkeit ist die Eingabe über den Domainnamen. Dafür muss allerdings ein DNS-Server installiert sein.

4.2.1 Webserver

Auch der Webserver ist, wie die beiden anderen Anwendungsbeispiele, seinen realen Vorbildern nachempfunden. Ein bekannter Vertreter ist Apache[3]. Anfragen werden entgegengenommen, überprüft, und eine Antwort an den Clients zurückgeschickt. Es können verschiedene Fehler auftreten. Wenn alles gut geht, und die vom Klienten gewünschte Seite vorhanden ist, wird sie umgehend abgeschickt. Wie eine Anfrage und ihre Antwort aufgebaut sein müssen, sehen Sie im nächsten Abschnitt zur Klasse `WebServerMitarbeiter`. Im Folgenden werden ausgewählte Methoden der Klasse `WebServer` näher erläutert: `erzeugeHtmlSeiten()` legt bei Initialisierung eines Webservers Dateien für alle Fälle an, die auftreten können. Das sind speziell die Datei *index.html*, und Fehlermeldungen 400 (Falsche Anfrage), 404 (Seite nicht gefunden), und 500 (Interner Server Fehler). Falls später ein Fehler auftreten sollte, wird zum Beispiel der Inhalt der Datei `webserver 400.txt` geschickt. Diese .txt Dateien sind verlgeichbar mit den Konfigurationsdateien von Hard- und Software. Sie werden bei der Erstellung des Objekts zu HTML-Dokumenten im virtuellen Dateisystem des jeweiligen Hosts „übersetzt“. Die Methode `starteWebServer()` erzeugt einen neuen ServerSocket, gibt eine Statusnachricht aus, dass der Server gestartet ist, erzeugt einen `WebServerThread` und gibt die Nachricht „Webserver gestartet“ an das Logfeld in der GUI. Mit `dateiEinlesen(String datei)` werden Konfigurationsdateien eingelesen (Siehe Abbildung 9). `beenden()` beendet den Webserver-Thread und alle laufenden Mitarbeiter.

Der Klasse `WebServerMitarbeiter` kommt in der Implementierung eine zentrale Bedeutung zu. Grundlegende Methoden wie `httpAnfrageBeantworten(String anfrage)`, `beenden()`, und die `run()`-Methode für Threads werden hier gesteuert. Auf die wichtigsten Methoden sei auch hier wieder eingegangen: Die Routine `run()` startet einen Thread der ständig im Hintergrund läuft, und auf den Empfang eines Anfrage-Strings wartet. Sobald dies geschieht, ruft er

[3]Siehe www.apache.org

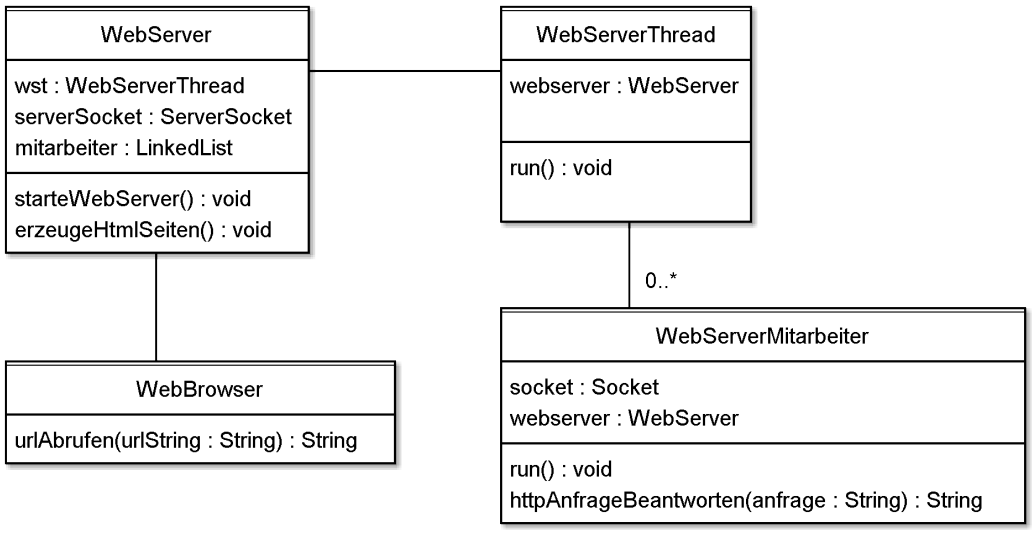


Abbildung 9: Klassendiagramm zum Webserver

`httpAnfrageBeantworten(result)` auf und schickt die Antwort als String an den Client zurück. Die Methode `httpAnfrageBeantworten(String -anfrage)` macht folgendes: Als erstes setzt sie den Text der Anfrage in das Logfeld der GUI. Danach zerstückelt sie die Anfrage und extrahiert die URI. Wenn keine spezielle Anfrage nach einer Seite gestellt wurde, wird die Datei *index.html* genommen. Ansonsten wird nach dem Vorhandensein einer entsprechenden Datei gesucht. Egal, welche Art von Anfrage gestellt wurde, es wird immer ein Code für Erfolg oder Misserfolg mitgeschickt. Eine Anfrage muss immer nach folgender Form erfolgen:

```
GET index.html HTTP/1.1
Host: www.die.informatik.uni-siegen.de
```

Eine Antwort muss ebenfalls genau so strukturiert sein:

```
HTTP/1.1 301 Moved Permanently
Location: http://www.die.informatik.uni-siegen.de/index.html
```

Diese Konventionen wurden aus dem Vorlesungsskript von Wismüller entnommen [Wi06].

4.2.2 Webbrowser

Die Software Webbrowser wurde wie alle anderen Anwendungen von FILIUS seinen reellen Vorbildern nachempfunden. Eine Webseite kann durch Eingabe der IP-Adresse des Servers erfolgen, oder über einen Domainnamen. Eingabemöglichkeiten werden in Abschnitt 3.1 beschrieben. In diesem Abschnitt soll es darum gehen, wie der Webbrowser implementiert wurde. Eine Übersicht in Form eines UML-Diagramms ist in Abbildung 10 gegeben. Mit der Methode `starten()` wird eine Statusnachricht über den erfolgreichen Start ausgegeben. Durch Interaktion des Programmnutzers über die GUI wird durch den Aufruf von `URLAbrufen(String URLString)` die eigentliche Kommunikation gestartet. Für URLs wird das gleichnamige Paket aus der Java-Bibliothek verwendet - dort können vorgefertigte Methoden wie `getHost()` verwendet werden. Im Verlauf der Abarbeitung der Methode werden noch diverse Meldungen an das Nachrichtenfenster, bzw. die Nachrichtenverwaltung gegeben. `URLAnfrage(URL url)` baut die Anfrage nach der oben genannten Konvention zusammen.

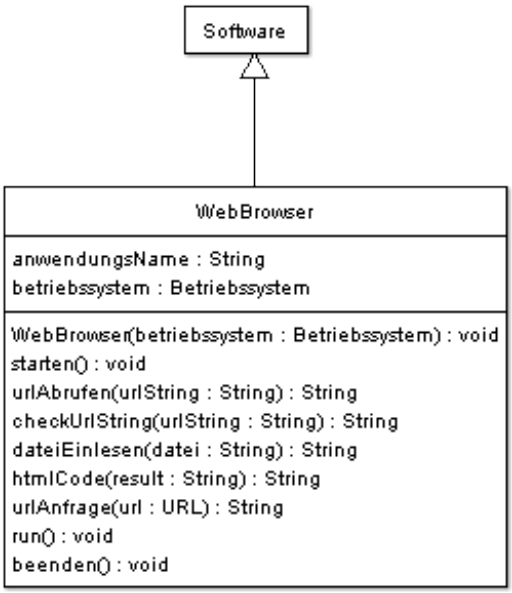


Abbildung 10: Klassendiagramm des Webbrowsers

4.3 Internetbasierter Dateiaustausch

Die Ansicht des Fensters zur Peer-to-Peer-Anwendung ist folgendermaßen aufgebaut: Es gibt je einen Reiter für *Netzwerk*, *Suche* und *Dateien*. Auf der Karte *Netzwerk* fügt man durch die Angabe der IP-Adresse einen anderen Rechner hinzu, von dem man weiß, dass dort ebenfalls eine Peer-to-Peer-Anwendung installiert ist und diese gestartet wurde. Der weitere Aufbau des Netzwerks geschieht dann automatisch. Wechselt man auf den Reiter *Suche*, so wird man aufgefordert über die genaue Eingabe eines Dateinamens nach einer Datei zu suchen. Auch das herunterladen ist bei erfolgreicher Suche hier möglich. Unter *Dateien* können alle heruntergeladenen Dateien auf einen Blick angeschaut werden. Beim hier dargestellten internetbasierten

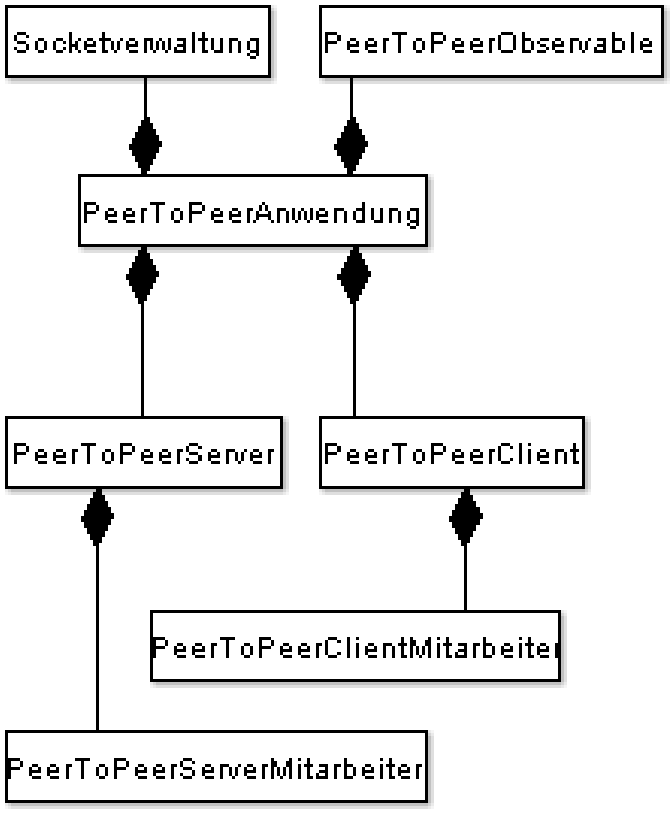


Abbildung 11: Peer-to-Peer Klassendiagramm ohne Pakete

Dateiaustausch handelt es sich um eine dezentrale Peer-to-Peer-Simulation. Das heißt, ein Teilnehmer des Peer-to-Peer-Netzes arbeitet gleichzeitig als Server und als Client, einen dedizierten Server gibt es nicht. Aus diesem Grund wird ein Teilnehmer nicht als Server oder Client, sondern meistens

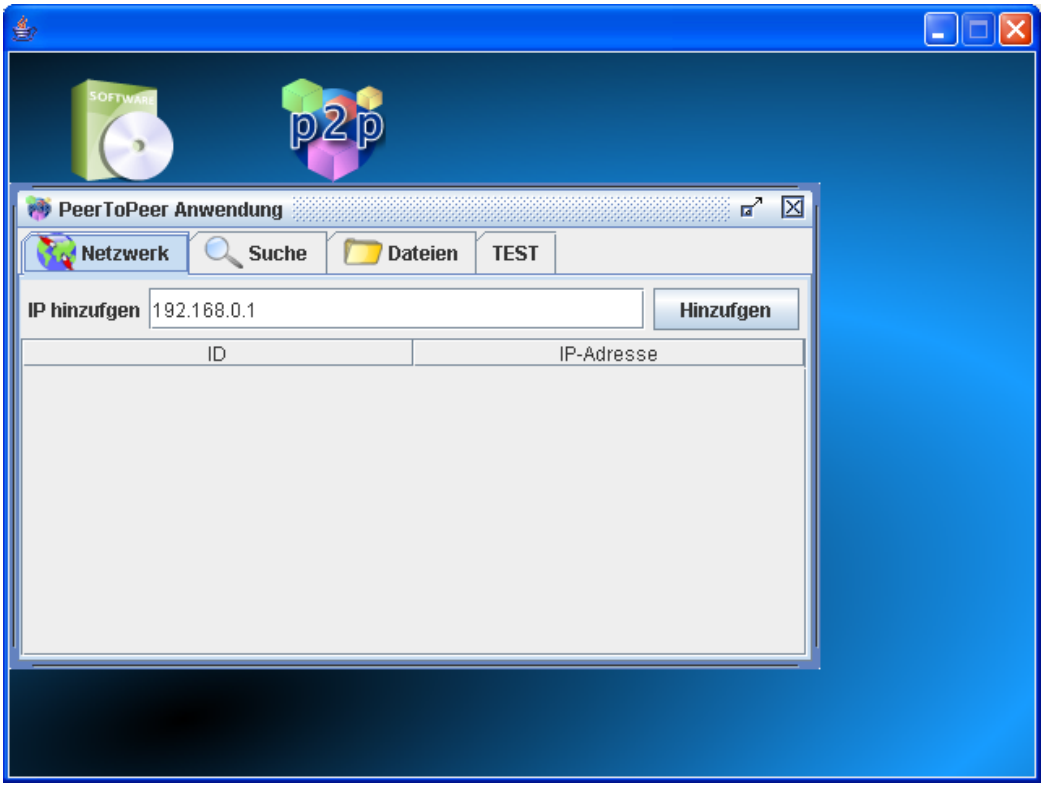


Abbildung 12: Peer-to-Peer

als Peer bezeichnet. Die realisierte Anwendung funktioniert ähnlich wie das bekannte Gnutella. An einigen Stellen mussten jedoch Abstriche gemacht, Dinge verändert oder weggelassen werden. Die Anwendung wird deshalb mit dem allgemeinen Begriff Internetbasierter Dateiaustausch bezeichnet. Die angegebenen Aktivitätsdiagramme sollen die Funktionsweise verdeutlichen.
Die Anwendung besteht aus den drei Hauptklassen: `PeerToPeerAnwendung`, `PeerToPeerClient` und `PeerToPeerServer`. Darüber hinaus gibt es zwei Klassen für den Datenaustausch: `PeerToPeerClientMitarbeiter` und `PeerToPeerServerMitarbeiter` und fünf Nachrichtenklassen. Aufgrund des bei Peer-to-Peer enorm hohen Verkehrsaufkommens wird eine Klasse Socketverwaltung benötigt, welche die zahlreichen Sockets verwaltet. Eine Speicherung (mit Ausnahme der Speicherung von Dateien) ist in der gesamten Anwendung nicht nötig.

Die Nachrichten enthalten fast die selben Merkmale wie das Gnutella-Protokoll. Auf Push-Nachrichten wurde verzichtet, da die Anwendung keine Benutzung von Firewalls unterstützt. Jede der vier Nachrichten Ping, Pong, Query

und Query-Hit enthält grundsätzlich folgende Attribute (Vgl. [Me02], [Re02], [SW]):

GUID: eindeutige Identifikationsnummer.

PAYLOAD: Art des Pakets.

TTL: „Time-To-Live“, ist die TTL abgelaufen, so wird das Paket verworfen. Ein unendliches Herumschwirren von Paketen wird somit verhindert.

HOPS: Anzahl der Peers, die ein Paket schon durchlaufen hat

PAYLOADLENGTH: Gesamtlänge inklusive Body.

BODY: Im Body enthalten sind weitere Attribute. Der Body wird im Quelltext nicht explizit erwähnt.

Ping	***Pong***	***Query***	***Query-Hit***
Keine weiteren Informationen	IP-Adresse	Minimale Geschwindigkeit	Anzahl Hits
	Port	Suchkriterien	Port
	Anzahl zu Verfügung stehender Dateien		IP-Adresse
			Geschwindigkeit
	Größe zu Verfügung stehender KBs		Ergebnis
			Server-Identifizierung

Abbildung 13: Tabelle zum Internetbasierten Dateiaustausch

Normalerweise gibt es in der Ping-Nachricht des Gnutella-Protokolls keine weiteren Attribute. Hier enthält eine solche Nachricht die IP-Adresse des Senders, damit andere Peers den Sender als Nachbar hinzufügen können. Das Ergebnis bei den Query-Hit-Paketen müsste eigentlich eine Liste sein, hier wird für jedes Ergebnis eine einzelne Antwort-Nachricht verschickt.

Nach der Installation kann die Peer-to-Peer-Anwendung gestartet werden. Zunächst werden intern automatisch der Server und der Client der Anwendung gestartet. Der Server geht sofort in den Warte-Modus (siehe Abbildung 14). Der Nachrichtenempfang ist nun möglich. Darüber hinaus wird beim Start eine Zufallszahl m zwischen drei und fünf erzeugt, die angibt, wie viele Einträge von Nachbarn in die Liste von bekannten Teilnehmern eingetragen

werden dürfen. In der Realität liegt diese maximale Anzahl bei Gnutella oft bei vier. Um bei wenigen Hosts eine Bildung von vielen kleinen separaten Netzen zu vermeiden (z. B. wenn fünf Rechner im Netz sind, kennt jeder jeden; für einen sechsten Rechner ist es jedoch nicht möglich, jemanden kennen zu lernen), wurde diese Regelung hier eingeführt. Die Zahl der maximal bekannten Teilnehmer kann später vom Benutzer bearbeitet werden.

Abbildung 14: Aktivitätsdiagramm zum Start der Anwendung

Sobald eine Peer-to-Peer-Anwendung gestartet wurde, kann sich der Peer mit einem anderen Peer verbinden. Dazu gibt er die IP-Adresse des gewünschten Teilnehmers an. Der Aufbau einer Verbindung zum Versand von Nachrichten ist jedoch nur möglich, wenn beide Teilnehmer die Peer-to-Peer-Anwendung gestartet haben. Durch erneutes Verbinden kann die Liste der bekannten Teilnehmer neu erstellt werden. Die alte Liste wird in diesem Fall gelöscht (Siehe Abbildung 15).

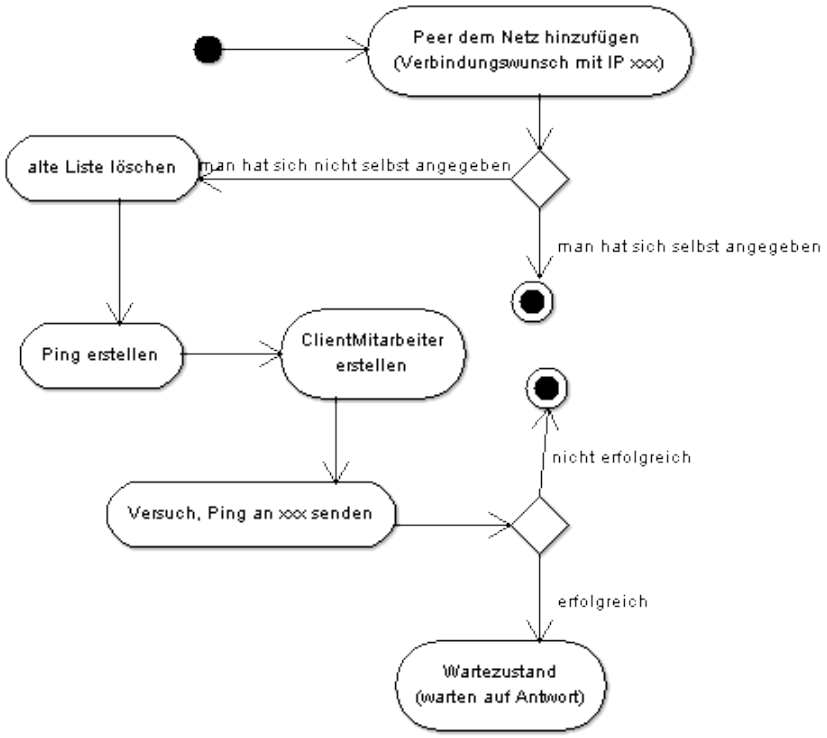


Abbildung 15: Aktivitätsdiagramm einer Verbindung zum Peer-To-Peer-Netzwerk

Angenommen, man selbst sei Host X. Gibt es einen zulässigen Peer Y im Netz, mit dem man sich verbinden möchte, so wird eine Ping-Nachricht erstellt und an die IP-Adresse von Y weitergeleitet. Hat Y noch Platz in seiner Liste, so trägt er X in die Liste ein und antwortet mit einer Pong-Nachricht. Empfängt X das Pong Paket von Y, so trägt auch er die IP-Adresse von X in seine Liste ein. X und Y kennen sich nun. Neben diesen Tätigkeiten inkrementiert Y die Hops des Pings, dekrementiert die TTL und sendet den Ping (falls die TTL noch gültig ist) weiter an all seine bekannten Nachbarn (außer X). Diese verarbeiten den Ping auf dem selben Weg und senden ggf. Pong-Pakete in Richtung von X, sodass X später bis zu m Teilnehmer kennt (Siehe Abbildung 16).

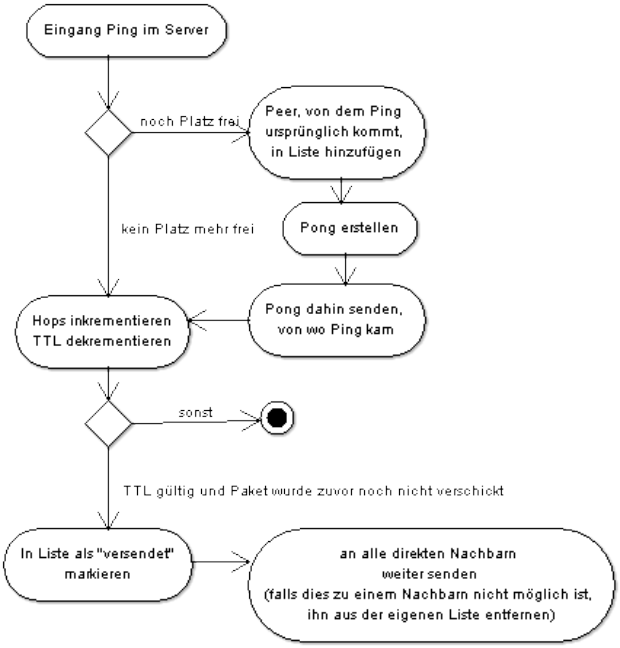


Abbildung 16: Aktivitätsdiagramm zur Verarbeitung einer Ping-Nachricht

Hat man sich erfolgreich mit dem Peer-to-Peer-Netzwerk verbunden und kennt mindestens einen anderen Peer, so kann man eine Anfrage starten. Auch hier wird eine eindeutige Identifikationsnummer erzeugt und abgespeichert, um mögliche Antworten zuordnen zu können. Diese Anfrage wird an jeden bekannten Nachbarn weitergeleitet. Dazu wird pro Nachbar ein Client-Mitarbeiter erzeugt, der die Verbindung zum jeweiligen Teilnehmer verwaltet. Anschließend wartet man auf Eingänge von Antwortnachrichten in den Client-Mitarbeiter.

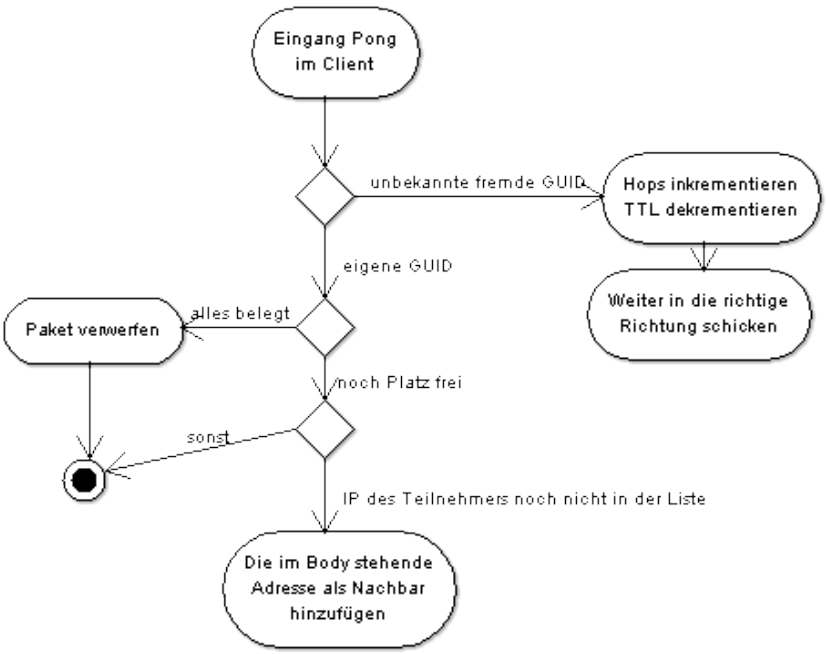


Abbildung 17: Aktivitätsdiagramm zum Pong-Empfang

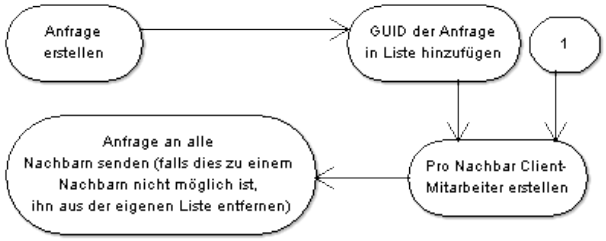


Abbildung 18: Aktivitätsdiagramm zu einer Anfrage

Empfängt ein Peer solch eine Anfrage-Nachricht (Query), so prüft er zunächst, ob die GUID des eingegangenen Pakets bekannt ist. Falls ja, wird die Nachricht verworfen, um Schleifen zu vermeiden. Ist die GUID nicht bekannt, so wird geprüft ob noch weniger als 15 Einträge in seiner Liste von Anfragen gespeichert sind. Ist dem nicht so, wird die älteste Nachricht gelöscht und die GUID zusammen mit dem Absender des neuen Pakets hinzugefügt; diese Liste arbeitet also wie eine Schlange. Anschließend wird nach Dateien gesucht, die die Suchkriterien erfüllen. Dabei werden die Dateinamen auf Teilstrings untersucht. Groß- und Kleinschreibung spielen dabei keine Rolle. Eine Suche nach dem String „gar“ würde so z.B. sowohl die Dateien `Kindergarten.txt` und `Tiefgarage.doc` als auch die Datei `Gartentor.txt` finden. Für jede gefundene Datei wird ein Antwortpaket erstellt und auf dem selben Weg wieder zurück versendet bis es an seinem Ziel angekommen ist. Treffen die Suchkriterien auf keine der Dateien zu, so werden TTL dekrementiert, die Zahl der Hops inkrementiert und das Paket wiederum, falls die TTL noch gültig ist, an alle bekannten Nachbarn weiter geleitet. Das Netz wird also geflutet.

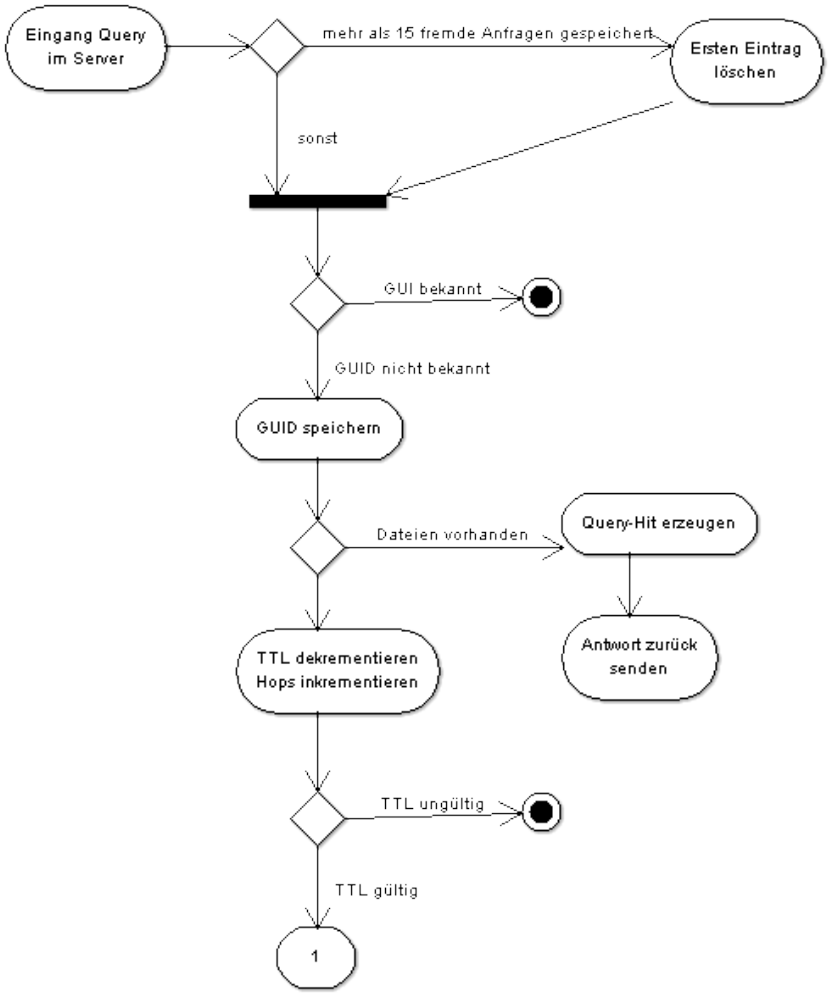


Abbildung 19: Aktivitätsdiagramm zum Query-Empfang

Empfängt ein Peer eine Antwortnachricht (Query-Hit), so prüft er, ob er selbst auf diese Antwort wartet. Ist dies der Fall, wird das Ergebnis in die Ergebnisliste mit aufgenommen. Andernfalls wird die GUID des Pakets aus der Anfrageliste fremder Anfragen heraus gesucht und an den letzten Absender weitergeleitet. Ist die GUID nicht vorhanden, wird die Nachricht verworfen. Hat der ursprüngliche Peer Antwortnachrichten auf seine Anfrage

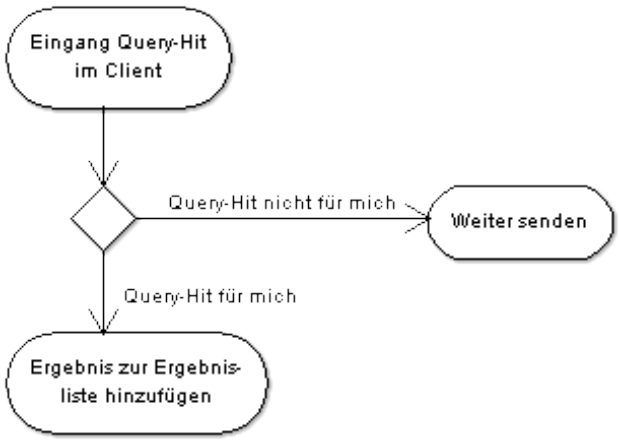


Abbildung 20: Aktivitätsdiagramm zum Query-Hit-Empfang

erhalten, so kann er diese direkt beim angegebenen Teilnehmer herunterladen. Dazu stellt er eine direkte Verbindung zum jeweiligen Peer her, erstellt eine HTTP-Anfrage und verschickt sie auf direktem Weg. Die HTTP-Anfrage hat die folgende Form:

```
GET Kindergarten.txt HTTP 1.1
```

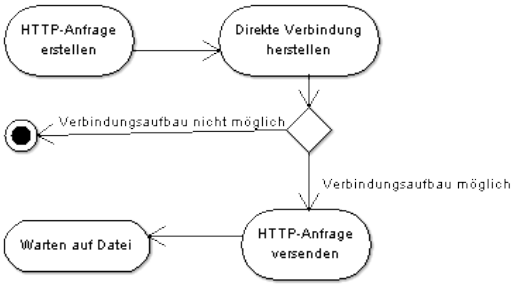


Abbildung 21: Aktivitätsdiagramm zum Abruf einer Datei

Nach dem Eingang der HTTP-Anfrage im bereitstellenden Host, sucht dieser die Datei heraus und verschickt sie direkt an den suchenden Peer. Die Dateien werden im Verzeichnis *root/peer2peer* gespeichert. Alle Dateien dieses

Verzeichnisses werden anderen Peers zur Verfügung gestellt. Beim Beenden der Anwendung werden alle offenen Sockets geschlossen, alle Listen geleert und sowohl Server, als auch Client der Anwendung gelöscht.

Schlägt an einer beliebigen Stelle der Verbindungsaufbau zu einem bekannten Teilnehmer fehl, so wird dieser aus der eigenen Liste der bekannten Teilnehmer gelöscht.

5 Netzwerksimulation

In diesem Abschnitt werden die Protokollschichten beschrieben, wie sie implementiert wurden. Technische Details der Implementierung werden dazu beschrieben und erklärt.

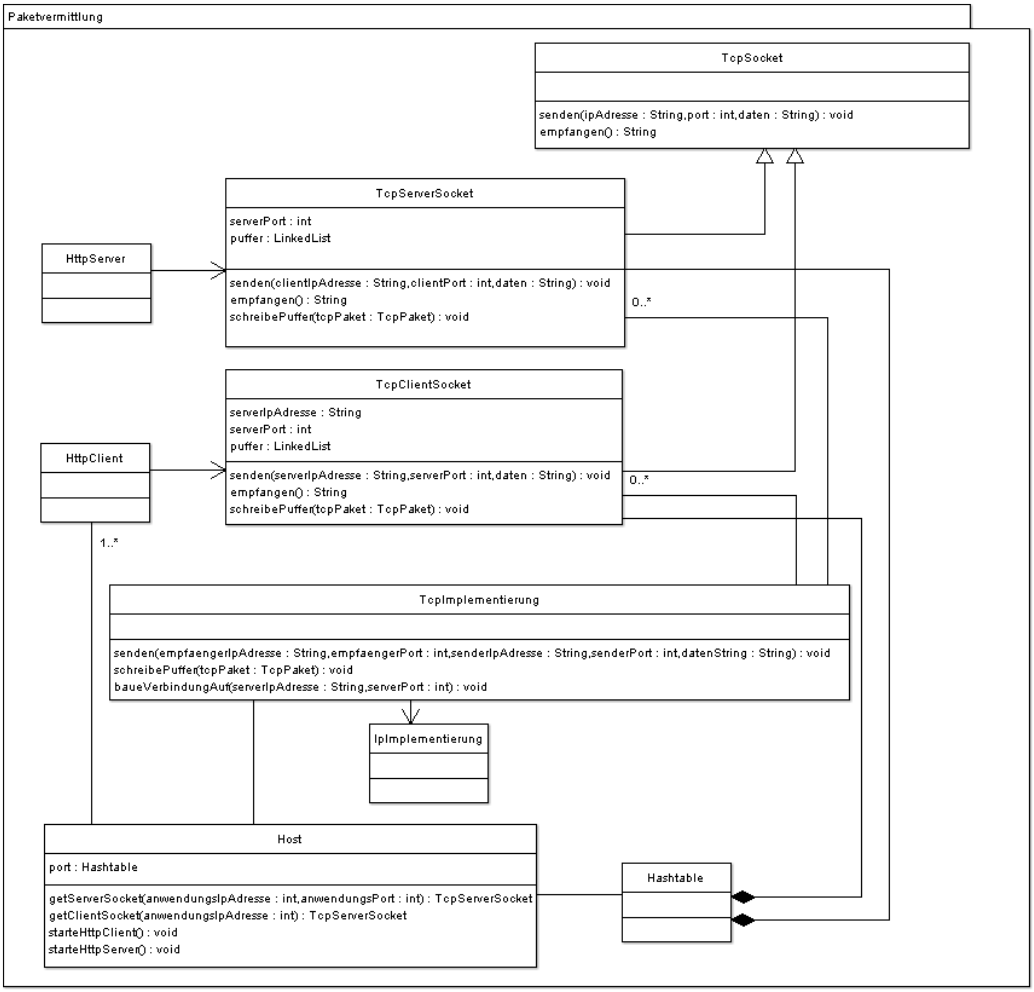


Abbildung 22: Klassendiagramm der Schnittstellen-Anwendung

5.1 Netzwerkschicht

Die Netzwerkschicht sorgt für die Vermittlung der Dateneinheiten über einen 48-bit-Schlüssel, die MAC-Adressen. Diese soll sicherstellen, dass alle Sy-

steme in einem lokalen Rechnernetz unterschiedliche Adressen haben. Das Ethernet überträgt die Daten auf dem Übertragungsmedium. Ein Ethernetrahmen, adressiert an die MAC-Adresse gilt als Broadcast. Diese Rahmen werden an alle weiteren Pakete in ihrem Subnetz versendet. Aufgrund dieser Rundruf-Nachrichten ist es möglich, einen gesuchten Zielrechner zu ermitteln oder gar sämtliche Rechner im lokalen Netz mit gewissen Informationen zu versorgen.

Im folgenden wird eine Liste mit Beschreibungen von Operationen und Daten der Ethernetschicht aufgeführt: Das Versenden der Pakete geschieht über die Operation `senden(Object daten, String zielMac, String typ)`. `Object daten` ist der zu versendende Datenanteil. Dieser wird dann in der Operation in einem Ethernetrahmen in den Datenteil verpackt. `String zielMac` ist die MacAdresse des Zielrechners, der den Datenstrom empfangen soll. `String typ` gibt den Typ der Daten wieder. Dabei gilt die 0x0800 als IP-Paket und 0x0806 als ARP-Paket. Weitere Kodierungen werden in unserem System nicht genutzt. Die einzelnen Daten werden zu einem neuen Datenpaket zusammengesetzt - das Ethernetpaket. In der Realität ist dieses Datenpaket ein kontinuierlicher Datenstrom, der sich nach einem gewissen Schema zusammen setzt. Viele der enthaltenden Informationen sind für unser System noch nicht zu gebrauchen und werden somit auch nicht genutzt. Ein Ethernetpaket wird mit folgenden Informationen erstellt. `EthernetPaket(String zielMacAdresse, String quellMacAdresse, String typ, Object daten)`. Das physikalische Versenden wird in dem Projekt folgendermaßen realisiert.

Das erstellte Ethernetrahmen wird nun in den Ausgangspuffer des Anschlusses der Netzwerkkarte gelegt. An diesen Anschluss ist eine Verbindung (Kabel) angekoppelt, die ständig diesen Puffer überwacht. Sobald dort Daten anliegen, wird dieser Rahmen aus dem Puffer entfernt und in den Eingangpuffer des Anschlusses auf der anderen Seite des Kabels gelegt. In der Verbindung selber wird eine variable Verzögerung realisiert.

5.2 Vermittlungsschicht

Die Vermittlungsschicht sorgt für die Weitervermittlung von Datenpaketen. Die Datenübertragung geht über das gesamte Rechnernetz hinweg und schließt die Wegesuche zwischen den Netzknoten mit ein. Da nicht immer ein direkter Datenaustausch zwischen Absender und Ziel möglich ist, müssen Pakete von Knoten, die auf dem Weg liegen, weitergeleitet werden. Weitervermittelte Pakete gelangen nicht in die höheren Schichten, sondern werden mit einem neuen Zwischenziel versehen und an den nächsten Knoten gesendet.

Das IP-Protokoll sorgt für die Vermittlung im Netz aufgrund der IP-Adressen. Die Methode zum versenden von Daten ist `senden(String quellIp, String zielIp, int protokoll, Object paket)`
Dabei sind die einzelnen Angaben recht klar, bis auf das Protokoll. In unserem Projekt werden dabei folgende Werte genutzt : 6 für TCP- und 17 für UDP-Segmente. Mehr Protokolle sind zur Zeit nicht vorgesehen, sind aber jederzeit erweiterbar.

Das Versenden eines IP-Paketes verläuft folgendermaßen:

1. Die Operation `senden` wird mit entsprechenden Daten aufgerufen.

2. Die Weiterleitungstabelle wird nach den zu nutzendem Gateway und der Schnittstelle bei gesuchter Ziel-IP-Adresse befragt.

3. Anschließend wird aufgrund des Gateways und der Schnittstelle die Ziel-MAC-Adresse mit Hilfe des ARP ermittelt.

4. Nun wird ein IP-Paket mit den erhaltenden Daten zusammengestellt.

5. Zuletzt wird das IP-Paket auf die erhaltene Schnittstelle an die Ethernetschicht mit entsprechenden Parametern(Ziel-MAC-Adresse, Protokoll und Daten) zum Versand übergeben.

6. Sollte eine gesuchte IP-Adresse nicht auffindbar sein, so wird dieses durch eine Exception gemeldet.

   Das Address Resolution Protocol (ARP) ist ein Netzwerkprotokoll, das die Zuordnung von Netzwerkadressen zu Hardwareadressen möglich macht. In unserem Programm wird das ARP in den einzelnen Subnetzten dazu genutzt, noch nicht identifizierte Rechner aufzuspüren. Dazu sendet der QuellRechner einen Broadcast im Subnetz mit der gesuchten IP-Adresse aus. Dieser Broadcast wird von sämtlichen Rechnern im Subnetz empfangen und ausgewertet. Besitzt einer dieser Rechner die gesuchte IP-Adresse, sendet er ein entsprechendes Antwort-ARP-Paket mit den benötigten Daten (MAC-Adresse) zurück. Dieser zu Beginn eines Netzaufbaus durchgeführter Datentransfer wird von den entsprechenden Vermittlungsknoten (Switch) dazu genutzt, ihre eingenen Routingtabellen aufzufüllen. Dadurch wird eine spätere saubere und schnelle Weitervermittlung durch den Knoten erreicht. Erhält der Quellrechner die ARP-Antwort vom gesuchten ZielRechner, wird die nun erhaltende MAC-Adresse für die weitere Kommunikation genutzt.

5.3 Transportschicht

Die Aufgabe der Transportschicht ist es, Nachrichten zu segmentieren und entsprechenden Anwendungen Ende-zu-Ende-Verbindung zur Verfügung zu stellen. Dieses realisieren die beiden Protokolle UDP und TCP. Der große und wichtige Unterschied dieser beiden Protokolle liegt in ihrem Dienstmodell. TCP fordert eine fehler- und verlustfreie Datenübertragung und erreicht dies über Kontrollmechanismen. UDP hingegen versendet nur die Daten und nimmt dabei Verluste in Kauf.

Der Datenaustausch mit TCP durchläuft drei wesentliche Schritte, die im Programm soweit wie möglich realistisch umgesetzt worden sind: Verbindungsaufbau, Datenaustausch und Verbindungsabbau. Die Serveranwendung, die TCP verwendet, muss dazu einen Server-Socket starten, der auf einem gewählten Port lauscht. Sobald Anfragen für diesen Port kommen, wird kontrolliert, ob zu dieser Anfrage ein Socket existiert. Dies wird mithilfe des Ports und der Quell-IP-Adresse des Paketes durchgeführt. Wird ein entsprechender Socket gefunden, bekommt dieser das Paket übergeben, andernfalls ein neuer Socket erstellt, das Paket übergeben und dann in die SocketListe eingetragen. Diese Aufgabe übernimmt die Operation `hinzufuegen(String zielIp, int zielPort, Object paket)` der Klasse `ServerSocket`.

Eine Serveranwendung hat nun die Möglichkeit über die Operation `ServerSocket.lauschen()` auf jegliches Eintreffen einer neuen Anfrage zu reagieren. Diese überwacht die `SocketListe` und holt den erzeugten Socket aus der Liste heraus und gibt ihn der anfordernden Anwendung. Clientseitig verläuft der Verbindungsaufbau über Sockets ein wenig anders. Die Clientanwendung erzeugt sich einen Socket mit den Informationen der Ziel-IP-Adresse und des Ziel-Ports. Der Socket versucht nun nach dem bekannten Muster des TCP-Protokolles die Verbindung aufzubauen. In der Operation `Socket.baueVerbindungAufClient()` wird diese geregelte Interaktion durchgeführt. Der Rückgabewert dieser Methode ist ein `Boolean` und gibt Aufschluss darüber, ob eine Verbindung zustande gekommen ist oder nicht. Kann eine Verbindung nicht aufgebaut werden, wirft der Konstruktor des Sockets eine `VerbindungsException`, die dann entsprechend von der Anwendung gehandhabt werden muss. Ist der Verbindungsaufbau zwischen Client und Server mit entsprechenden Informationsaustausch von Sequenznummern und Flags abgeschlossen, kann der Datenaustausch mit der Methode `Socket.sendenTcp(String daten)` erfolgen. Dabei werden die lange Daten in einzelne Datensegmente zerteilt und in einzelne TCP-Pakete verpackt. Das Zerteilen der Pakete übernimmt `Socket.erstelleTcpSegmente(String daten)`. Nun werden die einzelnen Pakete versendet. Der hier implementier-

te Algorithmus geht dabei so vor, dass immer nur ein Segment abgesendet wird und anschließend auf die Antwort, bzw. die Bestätigung gewartet wird. Erst wenn der Empfang des versendeten Segments quittiert wurde, wird das nächste Segment auf die Reise geschickt. Konnte ein Segment nicht mehr zugestellt werden, so wird die Methode mit einer `TimeOutException` abgebrochen.

Die Serveranwendung erhält durch den `ServerSocket` die Mitteilung, dass eine neue Anforderung eingegangen ist und erzeugt ihrerseits nun einen Mitarbeiter, der über den erzeugten Socket mit dem Client kommunizieren kann. Dieser Mitarbeiter ruft die Operation `Socket.empfangen()` und wartet nun auf den ihm zugesandten Befehl. Der Verbindungsabbau kann von beiden Parteien eingeleitet werden. Dazu steht die Operation `Socket.beenden()` zu Verfügung. Mit ihr wird der typische Verbindungsabbau des TCP-Protokolles eingeleitet. Sobald der Abbau vollzogen ist, beenden sich die Sockets und tragen sich aus den entsprechenden Listen aus, so dass nachfolgenden Sockets der genutzte Port wieder zur Verfügung steht. Die beiden Klassen `Socket` und `ServerSocket` wurden neu programmiert und sind nicht die von Java mitgelieferten. Dabei bieten die Klassen nun alle nötigen Operationen für den Datenaustausch zwischen einer Client-Anwendung und einem Server-Prozess über TCP und UDP.

Das Protokoll UDP arbeitet aufgrund der nicht vorhandenden Sicherung der Datenübertragnung um ein vielfaches einfacher. Dadurch erhöht sich natürlich die Menge der zu übertragenden Daten, da nur ein geringer Teil der Leitungskapazität benötigt wird. Beim Erzeugen des Sockets wird allein die Ziel-IP-Adresse getestet, ob sie ansprechbar ist. Ist diese gültig, so ist der Verbindungsaufbau vollzogen. Der Datenaustausch verläuft ähnlich unkompliziert. Die UDP-Segmente werden segmentiert und dann nacheinander versendet. Beim Server werden die eingehenden Pakete in einer Liste verwaltet und nacheinander abgearbeitet. Die Möglichkeit, dass die UDP-Pakete in einer unterschiedlichen Reihenfolge ankommen, ist in dem Programm nicht gegeben. Die Verbindung wird schließlich mit `Socket.beenden()` geschlossen.

5.4 Anwendungsschicht

5.4.1 Dynamic Host Control Protocol

Obwohl es sich bei Dynamic Host Control Protocol rein technisch betrachtet um eine Software handelt, die auch von der Klasse Software erbt, nimmt der

DHCP-Server in unserem Projekt einem gesonderte Stellung ein. Als einigste Software wird der DHCP-Server nicht erst im Aktionsmodus installiert, sondern wird bereits im Entwurfsmodus installiert und eingerichtet. Im Gegensatz zum DNS handelt es sich beim DHCP um einen nicht grundlegenden Dienst. Benutzer der Software können IP-Adressen auch manuell vergeben.

Die Verwendung des Dienstes DHCP ist optional. Der Benutzer kann selbst entscheiden, ob ein Host seine IP-Adresse von einem DHCP-Server, falls vorhanden, beziehen soll, oder ob er die IP-Adresse manuell festlegt. Zum Beziehen einer IP-Adresse von einem DHCP-Server muss natürlich ein Solcher im Netzwerk vorhanden sein. Daher wird im Folgenden auf zwei Möglichkeiten, die sich direkt auf DHCP beziehen, eingegangen, die der Benutzer beim Konfigurieren eines Host hat. Zum Einen hat der Benutzer die Möglichkeit dem Host zu sagen, er soll seine IP-Adresse von einem DHCP-Server beziehen. In diesem Fall muss der Benutzer an der entsprechenden Stelle ein Häkchen setzen. Diese Einstellung muss mit *Ändern* bestätigt werden. Wählt der Benutzer die Schaltfläche *Als DHCP Server einrichten*, so wird dieser Host als DHCP-Server eingerichtet. Nun muss der Benutzer angeben, welchen Adress-Raum der DHCP-Server verwaltet, in dem er Ober- und Untergrenze angibt. Soll der DHCP-Server einem anfragenden Client auch DNS-Server und das Gateway mitteilen, so kann der Benutzer dies nun auch angeben. Die DHCP-Kommunikation erfolgt über UDP-Broadcasts. Diese wurden in diesem Projekt ausschließlich für den Gebrauch von DHCP realisiert. Der Ablauf der DHCP-Pakete orientiert sich ebenfalls an der Realität, wobei Letztere um einige, für dieses Projekt nicht relevante, Informationen gekürzt wurden. Grundsätzlich gilt: Jeder Host darf als DHCP-Server fungieren und ein Client kann nur dann eine IP-Adresse beziehen, wenn ein DHCP-Server im Netzwerk vorhanden ist.

Die Klasse `DHCPServer` erbt von Software und erzeugt einen DHCPServerThread. Sie nimmt Anfragen auf Port 67 entgegen. Die Klasse `DHCPServerThread` dient zum Starten und Beenden ausgelagerter Threads. Sie erzeugt die `DHCPServerMitarbeiter`. `DHCPServerMitarbeiter` regelt den Datenaustausch mit dem DHCP-Client eines Hosts. `DHCPClient` ist fest im Betriebssystem integriert und kann nur über dieses angesprochen werden. Datenaustausch ist über Zustände realisiert, damit die Kommunikation nicht bei wiederholten Wechsel in den Aktionsmodus erneut abläuft. `DHCPClientThread` dient dem Starten und Beenden ausgelagerter Threads.

5.4.2 Domain Name System

Die in diesem Projekt abgebildete Funktionsweise des Domain Name System lässt sich gut an Hand der beiden Hauptkomponenten erklären. Dem DNS-Server, der die verschiedenen Einträge verwaltet und dem DNS-Client, der die jeweiligen Anfragen an den DNS-Server stellt. Bei dem DNS-Server handelt es sich um eine Software, die installiert und gestartet werden muss.

Um den Schülern einen leichteren Zugang zu gewähren, haben wir das von uns abgebildete Modell des DNS leicht vereinfacht. Während ein DNS-Server in der Realität eine Vielzahl verschiedener Einträge kennt, haben wir und auf so genannte a- und mx-Records beschränkt. Bei a-Records handelt es sich um einfache Zuweisung einer IP-Adresse zu einer URL, zum Beispiel *www.gmx.de* und *217.72.204.254*. Bei mx-Records handelt es sich um spezielle Einträge, die von einer E-Mail-Anwendung verwendet werden. Hierbei legt das Programm den passenden a-Record gleich mit an.

Ohne einen installierten und gestarteten DNS-Server, der über entsprechende Einträge verfügt, kann ein Webbrowser keine Webseite via URL aufrufen. Auch das Versenden von E-Mails zwischen verschiedenen E-Mail-Domains ist ohne DNS nicht möglich. Die Arbeitsweise dieses Dienstes ist recht einfach. Der DNS-Server lauscht auf Port 53 auf Anfragen eines oder mehrerer Clienten. Bei einer Anfrage wird eine so genannte Mitarbeiterklasse gestartet, die die Anfrage bearbeitet und eine entsprechende Antwort verschickt. Der DNS-Client hingegen ist fest im Betriebssystem integriert und muss auch nicht extra installiert werden. Er wird von den verschiedenen Anwendungen auch nur über das Betriebssystem angesprochen. Die jeweiligen Anwendungen erstellen sich also keinen eigenen DNS-Client. Die in Abbildung 23 zu sehenden Klassen werden im folgenden Abschnitt erklärt.

`DNSServer` verwaltet die verschiedenen Einträge. Dies sind mx- und a-Records. Das Anlegen und Löschen von Einträgen erfolgt über diese Klasse. Sie erzeugt einen Serversocket, der auf Port 53 lauscht. Bei `DNSServerThread` handelt es sich um einen ausgelagerten Thread, der vom DNS-Server verwendet wird. Bei einer neuen Anfrage erzeugt diese Klasse einen neuen DNS-Server-Mitarbeiter. `DNSServerMitarbeiter` bearbeitet die Anfragen eines DNS-Clients. `DNSClient` ist fest im Betriebssystem integriert und über dieses angesprochen. Sie bietet die Funktionen `urlaufloesen()`, `ipaufloesen()` und `maildomainaufloesen()`.

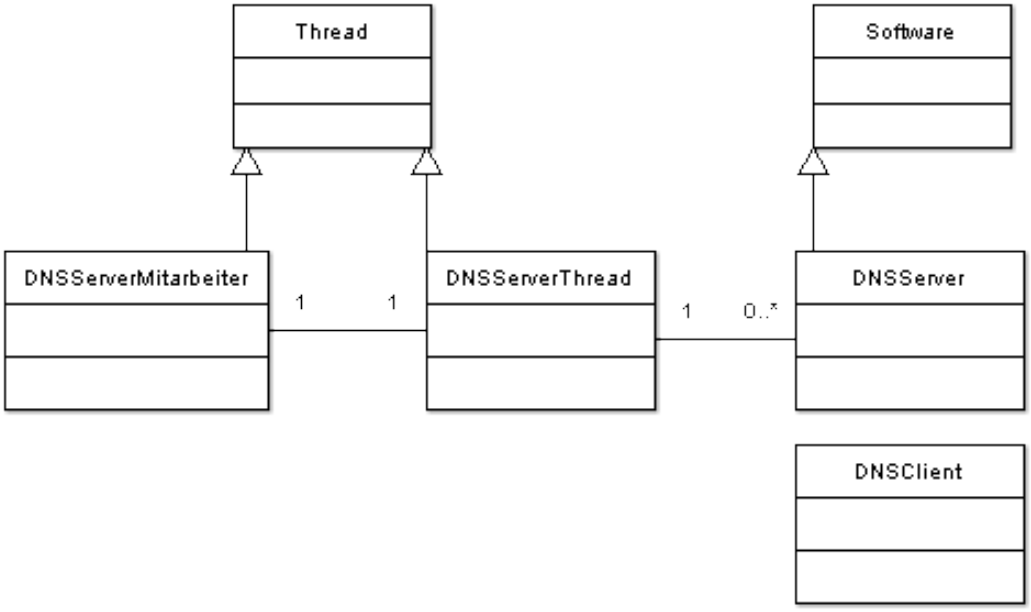


Abbildung 23: Das Domain Name System

6 Anwendungsszenario

Im folgenden Abschnitt wird die Erstellung eines Szenarios beschrieben. Wir möchten zeigen, wie es Schritt für Schritt möglich ist, ein Netzwerk aus verschiedenen Hardwarkomponenten zu erzeugen, diese mit Software zu versehen, und eine Simulation ablaugen zu lassen. Wir haben uns für die Anwendung WWW entschieden, da diese am einfachsten ist.

6.1 Erstellen eines Netzwerks

Wenn der Start von FILIUS ordnungsgemäß funktioniert hat, ist das Hauptfenster des Programms mit einer leeren Arbeitsfläche, einer Palette für Hardware und einer Menüleiste am oberen Rand zu sehen. Das Progamm befindet sich im Entwurfsmodus. Im Hintergrund wurde zusätzlich jeweils ein Dialogfenster *Nachrichten* und *Pakete* geöffnet. Im Entwurfsmodus wird das für das Szenarion zu Grunde liegende Netzwerk zusammengestellt. Wir wollen für dieses Szenario ein nicht zu kompliziertes Netz zusammenbauen. Beginnen wollen wir damit, einen Switch in der Palette am linken Rand auszuwählen und ihn mit der Maus auf der Arbeitsfläche postieren. Am unteren Rand des Hauptfensters errscheint ein Eingabefeld mit der Anzahl der Anschlüsse des Switches. Als nächstes fügen wir einen Vermittlungsrechner in analoger Vorgehensweise hinzu. Wir verändern den Namen auf „Router-Box“indem wir einfach das Textfeld auswählen und den Namen editieren. Der Vermittlungsrechner besitzt zwei Anschlüsse und wie zu sehen ist, gibt es dafür je einen Reiter zur Konfiguration von NIC 0 und NIC 1. Wir wählen NIC 0 und geben als IP-Adresse `192.168.0.5` und als Subnetz `255.255.255.0` ein Mit `Ändern` werden alle Eingaben bestätigt. Weiterhin brauchen wir einen Rechner und einen Laptop, die wir beliebig neben den Vermittlungsrechner setzen. Um eine konsistente Einheit zu bekommen, verbinden wir die beiden Rechner und den Vermitllungsrechner mit dem Switch mit einem Kabel aus der Palette. Der Switch ist also nun der Mittelpunkt und somiit zentrale Schaltstelle unseres Systems. Miteinander kommunizieren können die beiden Rechner aber noch nicht, da wir sie nicht konfiguriert haben. Zuerst wollen wir den Rechner, nennen wir ihn ab jetzt Server, einstellen. Durch Auswahl erscheinen, genauso wie beim Vermittlungsrechner, im unteren Bereich Konfigurationdaten. Dort ändern wir den Namen auf *Server*, alle anderen Optionen lassen wir so wie voreingestellt. Dem Laptop geben wir den Namen *Laptop*. Bei IP-Adresse und Subnetzmaske müssen wir nun aufpassen, um ein funktionsfähiges Netzwerk zu konfigurieren. Für IP-Adresse wählen wir `192.168.0.1`, das Subnetz bleibt auf `255.255.255.0`. Der Gateway hat in diesem Netz keinen Einfluss,

daher lassen wir es wie es voreingestellt ist. Auch DNS wollen wir zunächst unbeachtet lassen. Nachdem wir nun ein fertiges, und warscheinlich logisch funktionierendes, kleines Netzwerk zusammen gestellt haben, müssen wir zur Installation von Software, und dem Erzeugen einer Simulation in den Simulationmodus wechseln. Dort werden die Auswahlkoompnenten verschwinden, da sie dort nicht mehr gebraucht werden. Es werden im Simulationsmodus keine weiteren Hardwarekomponenten hinzugefügt.

6.2 Installation von Software

Aufgabenstellung war in diesem Szenario einen HTTP-Datenaustausch durchzuführen. Dazu brauchen wir folglich einen Webserver und einen Webbrowser. Um diese hinzufügen zu können, wechseln wir wie bereits erwähnt, in den Simulationsmodus. Über die mit einem grünen Dreieck dekorierte Schaltfläche *Simulation*. Es ist nur noch die Arbeisfläche mit den Hardwarekomponenten zu sehen. Den Webserver installieren wir auf dem Rechner und Browser auf dem Laptop. Zuerst widmen wir uns dem Server. Im Kontextmenü, das durch die rechte Maustaste geöffnet wird, wählen wir *Desktop anzeigen* aus. Dadurch öffnet sich ein neues Fenster. Unter dem Symbol *Software-Installation* verbirgt sich eine Liste mit der in FILIUS verfügbaren Software. Wir fügen einen Webserver hinzu und bestätigen die Wahl mit *Ändern*. Nach der erfolgreichen Installation muss der Server noch gestartet werden. Durch Auswahl des neu hinzugefügten WWW-Symbols, erscheint ein Fenster für Konfigurationsdaten, mit dem der Server letzlich gestartet werden kann. Im darin befindlichen Logfeld erscheint die Nachricht *Webserver gestartet*. Das Logfeld zeigt alle Anfragen die von anderen Clients and den Server gestellt wurden an. Es ist vergleichbar mit einer Log-Datei eines realen Webservers. Die Arbeitsfläche des Servers kann nun wieder geschlossen werden. Analog zur Vorgehensweise beim Rechner, installieren wir einen Browser auf dem Laptop und starten diesen anschließend. In der Eingabeleiste wird das Eintragen einer gültigen URL erwartet. Wir können den Server nur über die IP-Adresse ansprechen, da wir noch keinen DNS-Server im Netzwerk integriert haben. Die richtige Eingabe muss lauten: `http://192.168.0.0`. Mit der Schaltfläche *Go* geht es los. Es öffnet sich ein neuer Reiter mit der Standardseite des Servers. Die Anfrage war erfolgreich. Initial haben alle Server in FILIUS dieselbe Standardseite. Wie bereits erwähnt besteht aber die Möglichkeit eigenen HTML-Seiten in FILIUS einzubinden, oder direkt selbst mit Hilfe des Texteditors zu erstellen. Später in diesem Abschnitt wird erklärt wie eine eigene Seite einbeogen werden kann Zur Zeit ist es nur möglich HTTP-Anfragen via IP-Adresse zu stellen. Im nächsten Abschnitt wollen wir die Ergänzung mit

DNS-Server beschreiben.

6.3 Hinzufügen eines DNS-Servers

Der DNS-Server soll sich auf einem separaten Rechner befinden. Einen zusätzlichen Rechner können wir im Simulationsmodus jedoch nicht erstellen. Also gehen wir über das mit einem Hammer dekorierte Schaltfläche in der Menüleiste zurück in den Entwurfsmodus. Dort wählen wir einen zweiten Rechner, fügen ihn in der Arbeitsfläche ein und schließen ihn mit einem Kabel an den Switch an. Anschließend wird er *DNS-Server* genannt und ihm die IP-Adresse `192.168.0.2` gegeben. An dieser Stelle wollen wir die Änderungen übernehmen und in den Simulationsmodus wechseln. Dort installieren wir einen DNS-Server, der auch noch konfiguriert werden muss. Wir fügen den Eintrag `www.server.de` für die IP-Adresse `192.168.0.0` hinzu und starten den DNS-Server. Anschließend können wir die Arbeitsfläche verlassen. Um dem Server und Client beizubringen, dass nun ein DNS-Server im Netzwerk ist, müssen wir dort in Konfigurationsfeld (Entwurfsmodus) die IP-Adresse des DNS-Servers eintragen. Im nächsten Abschnitt testen wir ob alles ordnungsgemäß konfiguriert wurde.

6.4 Testen und Erweiterung des Szenarios

Zur Ausführung des eben erstellten HTTP-Szenarios wechseln wir vom Entwurfs- in den Simulationsmodus. Anschließend wird der Browser auf dem Laptop gestartet und die Anfrage *http://www.server.de* eingegeben. Es sollte nach einer kurzen Verbindung, die grafisch durch blinkende Kabel deutlich wird, wieder die Standardseite des Webservers angezeigt werden.

An dieser Stelle wollen wir nun eine eigene HTML-Seite erstellen, um nicht immer nur die Standardseite des Webservers betrachten zu müssen. Dazu wird der Text-Editor benötigt. Wir installieren ihn auf dem Server, da dort auch der Webserver installiert ist. Der Editor wird gestartet und der HTML-Quelltext eingetragen.

```
<html>
    <head>
        <title>Eigene Seite 1<title>
    </head>
```

```
    <body>
        <p>Willkommen auf meiner eigenen Seite</p>
    </body>
</html>
```

Anschließend wird über das Menü des Texteditors *Datei* und *Speichern unter* der Speicherdialog geöffnet und *eigeneSeite.html* gespeichert. Dazu wird das Verzeichnis *Webserver* ausgewählt (dort liegen bereits vier Dateien) und die Datei mit *Speichern* bestätigt. Die Seite ist nun angelegt, sodaß via HTTP darauf zugegriffen werden kann. Zum Test wird zum Client gewechselt und die Datei im Webbrowser über folgende Anfrage aufgerufen: `http://www.server.de/eigeneSeite.html`.

Ergänzend könnten Anfragen auf einen explizit anderen Port gelegt werden. Dazu muss beim Webserver in der Software *Webserver* explizit als Port der Wert 80 entfernt werden. Dazu ist das Beenden notwendig. Stattdessen wird 50 an dieser Stelle eingetragen und der Webserver wieder gestartet. Die Anfrage von Seiten des Webbrowsers müsste nun folgendermaßen aussehen: `http://www.server.de:50/eigeneSeite.html`. Es wird hinter den Domainnamen eine 50 für den Port hinzugefügt.

Als nächstes wollen wir kurz auf die Funktion DHCP eingehen. Dazu verwenden wir das bereits erstellte Netzwerk, und schauen uns die Auswirkung eines DHCP-Servers auf neu hinzu kommende Rechner an. DHCP weist neu hinzukommenden Rechnern automatisch eine IP-Adresse zu . Zunächst erstellen wir einen DHCP-Server. Dazu nehmen wir einen neuen Rechner und nennen ihn DHCP-Server. Über *DHCP-Server einrichten* öffnet sich ein Dialogfenster. Dort geben wir als Obergrenze `192.168.0.30` und als Untergrenze `192.168.0.20` ein. Das hat zur Folge, dass nun im Netzwerk maximal elf Rechner hinzugefügt werden können. Sie bekommen eine Adresse zwischen 20 und 30 zugewiesen. Die Felder Gateway und DNS lassen wir frei und bestätigen unsere Wahl. Bei allen Rechnern, die nun automatisch eine IP-Adresse zugewiesen bekommen sollen, muss *DHCP* aktiviert werden. Achtung: Auch beim DHCP-Server selbst muss *DHCP* aktiviert werden. Anschließend wechseln wir in den Simulationsmodus. Dort werden sofort die Kabel beginnen für kurze Zeit zu blinken. Die IP-Adressen werden gesetzt. Wir wollen uns das Ergebnis einmal genau anschauen: Mit dem Zurückwechseln in den Entwurfsmodus sehen wir, dass die IP-Adressen der gewünschten Rechner auf einen Wert zwischen 20 und 30 geändert wurden.

6.5 Didaktische Betrachtung des Szenarios

Generell ist zunächst zu sagen, dass explorative Lehr-Lern-Software das konstruktive Lernen der Schüler unterstützt. Beim explorativen, also entdeckenden, forschenden Lernen sollen sie eigentlich ohne Anleitung und Hilfe in FILIUS ausprobieren und Wege erforschen, wie sie bei einer gegebenen Aufgabenstellung zum Ziel gelangen können. Nimmt man diese Anleitung zum Erstellen eines HTTP-Szenarios, so hat das nichts mit Exploration gemein. Man sollte es daher von einer anderen Sichtweise betrachten. Die Anwender bekommen hier, indem sie einmal an die Hand genommen werden, einen schnellen Einblick in FILIUS. Sie lernen Werkzeuge und Funktionen kennen und bekommen Mittel und Wege gezeigt, mit FILIUS zu arbeiten. Danach sollten sie in der Lage sein, selbst mit der Softwar zurecht zu kommen. Dann kann man den Lernenden gezielte Aufgabenstellungen geben, die sie selbstständig bearbeiten und lösen sollen.

7 Zusammenfassung und Ausblick

Mit FILIUS ist es gelungen drei Internetanwendungen getreu dem Vorbild des realen Internets nachzubilden. Es wurden Protokolle wie ARP, TCP/IP, DHCP, DNS, HTTP, POP implementiert. Dabei wurden nicht genau alle Funktionsweisen umgesetzt. Dann hätte die Projektgruppe das Internet neu programmiert. Aber im nach außen sichtbaren Verhalten sind sie den realen Vorlagen sehr ähnlich. Teilweise, gerade bei TCP und IP wurden die Protokolle und Anwendungen auch sehr detailliert umgesetzt.

Die Anwendung Peer-to-Peer ist wohl am schwierigsten umsetzbar gewesen. Hier musste aufgrund diverser Probleme eine Mischung aus Gnutella und anderen Tauschbörsen entwickelt werden. Weitestgehend ist aber Gnutella das Vorbild zur Realisierung gewesen. Die Entwickler haben sich darauf geeinigt, dass sich der erste Teilnehmer selbst hinzufügen muss. Bei Gnutella wäre das direkt ein Nachbar. Zu besonderen Schwierigkeiten bei der Implementierung der Software zum internetbasierten Dateiaustausch kam es vor allem aufgrund der enormen Anzahl an Nachrichten, die während der Nutzung der Peer-to-Peer-Anwendung verschickt werden. Zahlreiche Sockets müssen zeitgleich erstellt werden und mehrere Verbindungen pro Rechner aufrecht erhalten werden. Eine Suchanfrage kann theoretisch bis zu $4^8 = 65536$ Nachrichtensendungen auslösen. Schon allein beim Aufbau eines Peer-to-Peer-Netzes mit nur acht Rechnern werden über 100 Nachrichten verschickt. Aufgrund dieser Besonderheiten wurden Client- und Servermitarbeiter und eine Socketverwaltung eingeführt, die die Informationsmengen verwalten. Außerdem mussten die Schichten zur Kommunikation mehrfach abgeändert und angepasst werden. Zwar wurde versucht, sich möglichst nah an der Realisierung des Gnutella-Protokolls zu orientieren, jedoch mussten zahlreiche Abstriche gemacht werden. Dadurch ergeben sich hier natürlich weitere Ausbaumöglichkeiten der Software, z.B. Realisierung von Timeouts, Einführung von Push-Nachrichten, und so weiter.

Die E-Mail-Anwendung leistet im Prinzip alles was man von ihr erwartet. Abrufen, Versenden, Beantworten von Nachrichten. Benutzerkonten waren schwierig, da das Team komplett auch die von Java bereitgestellten Methoden zur Client-Server-Kommunikation verzichten wollte. Daher musste dies alles über Threads nachempfunden werden. Als Ergebnis kann man aber sagen, dass sowohl SMTP als auch POP ihre Dienste tun. World Wide Web ist von der Komplexität her, mit dem geringsten Entwicklungsaufwand verbunden gewesen. Trotzdem ist sie gerade für Einsteiger, nach Meinung der Autoren die wichtigste Anwendung zum Verständnis der Vorgänge in einem Rechnernetz.

A Routing

Die Routingtabelle des Knotens B in der Abbildung 25 würde initial aussehen, wie in Tabelle 1 beschrieben. Nach einigen Schritten würde man die fertige Tabelle wie in Tabelle 2 erhalten. Die Beschreibung des genauen Algorithmus würde hier zu weit gehen. Eine anschauliche, detaillierte Animation kann jedoch unter [RVS] aufgerufen werden.

Knoten	Kosten	Next Hop
A	1	A
C	#	-
D	1	E
E	#	-
F	#	-
G	#	#

Tabelle 1: Distance Vector Routing - Schritt 1

Knoten	Kosten	Next Hop
A	1	A
C	2	A
D	2	A
E	1	E
F	3	A
G	2	E

Tabelle 2: Distance Vector Routing - Schritt 2

Bei den drei Tabellen handelt es sich um Distanzvektoren von A. Die Verbindung zwischen A und B wird unterbrochen. B bemerkt dies, erfährt aber, dass C einen Weg zu A kennt, der zwei Schritte lang ist. B weiß nicht, dass C auch B als Zwischenschritt benutzt, also übernimmt B den Weg über C und erhöht ihn um eins. Da C den Punkt B benutzt, um A zu erreichen, erhöht er seinen Weg um eins (weil B den Weg zu A erhöht hat. Nun erhöht wiederum B seinen Weg zu A, da er C benutzt und dieser seinen Weg um eins erhöht hat. B und C schaukeln sich so gegenseitig hoch, bis die Entfernung zum Knoten A gegen unendlich geht.
Das Prinzip des Link State Routings unterscheidet sich stark vom oben beschriebenen Konzept, ist aber leicht erklärt: Jeder Knoten kennt zunächst nur

seine direkten Nachbarn und misst seine Kosten zu ihnen. Nun schickt jeder Knoten Link State Pakete an alle anderen Router (Fluten), diese leiten die Pakete nach bestimmten Kriterien weiter, bis jeder über die direkten Nachbarn aller anderen Knoten Bescheid weiß. Schließlich werden kürzeste Wege mit Hilfe der erhaltenen Informationen durch den Dijkstra-Algorithmus (siehe [LI06]) in den jeweiligen Knoten berechnet. Auch zum Link State Routing findet man unter dem oben angegebenen Link eine Animation.

Aufgabenbeispiele

1. Erstellen Sie ein Netzwerk, in welchem sich drei Router, sieben Hosts und ein Switch befinden. Legen Sie bei jeder Hardwarekomponente fest, in welcher Art Netz sie sich befindet. Warum ist es sinnvoll, zwischen A-, B- und C-Netzen zu unterscheiden?

2. Ordnen Sie jeder der Komponenten eine entsprechende IP-Adresse zu!

3. In einem Netzwerk befinden sich 450 Hardwarekomponenten, welche eine IP-Adresse benötigen. Um was für ein Netz handelt es sich vermutlich?

4. Zu welcher Netzklasse gehört der Host mit der IP-Adresse *200.1.1.127*?

5. Wo liegt der Unterschied zwischen Routing und Forwarding?

6. Gegeben sei folgender Graph, welcher ein Netzwerk darstellt (siehe Abbildung 25):
   Bestimmen Sie mit Hilfe des Distance Vector Routings schrittweise die Distanzvektoren der einzelnen Knotenpunkte und die Routingtabellen von A, D und E! Wenn Sie sich nicht mehr genau an den Algorithmus erinnern können, schauen Sie sich die Animation auf der Seite [RVS] an.

7. Gegeben sind die folgenden Routingtabellen:

   Versuchen Sie ein Netzwerk mit den Knoten A-F aufzubauen, aus denen diese beiden Routingtabellen für A und B entnommen sein könnten!

8. Erstellen Sie ein Netzwerk wie in Abbildung 25.
   Führen Sie den Distance Vector Routingalgorithmus durch, um Routingtabellen für alle Knoten zu erstellen. Löschen Sie anschließend die Verbindung zwischen D und F und führe den Algorithmus erneut durch. Was passiert? Warum? Wie könnte man das Problem verhindern?

A:			B:		
Ziel	Entfernung	Next Hop	Ziel	Entfernung	Next Hop
B	3	B	A	3	A
C	3	D	C	5	D
D	2	D	D	4	D
E	8	E	E	5	E
F	14	B	F	6	E

Tabelle 3: Routingtabellen

9. Vergleichen Sie Aufgabe 6 mit Link State Routing.

10. Wann wendet man besser das Distance Vector Routing an? Wann eignet sich wiederum besser das Link State Routing?

11. Gegeben ist ein Netzwerk wie in Abbildung 24: Bestimmen Sie mit

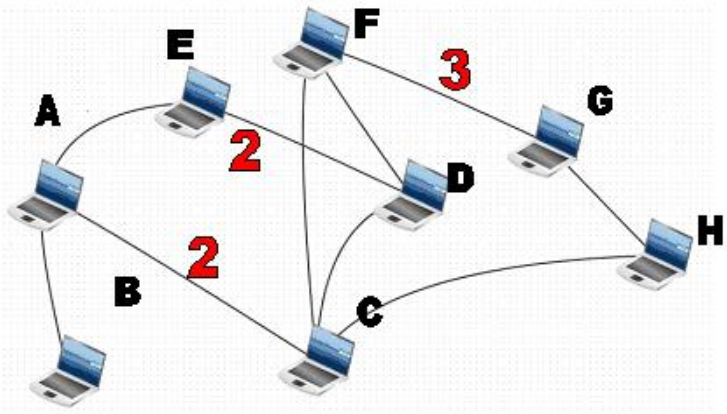


Abbildung 24: Darstellung eines Netzwerks

Hilfe des Dijkstra-Algorithmus die kürzesten Entfernungen von E zu allen anderen Knoten! (Alle Kanten, an denen keine Kosten stehen, haben die Kosten 1)

12. Die Firma EAT verfügt über ein Klasse C Netz. Die Netzadresse lautet 199.10.2.0 In der Firma gibt es vier Abteilungen, für jede soll ein Subnetz eingerichtet werden. In der Produktion befinden sich 11 Rechner, in der Qualitätssicherung 22, im Versand 33 und im Verkauf 88. Wie könnten Subnetzmasken und Subnetzadressen vergeben werden?

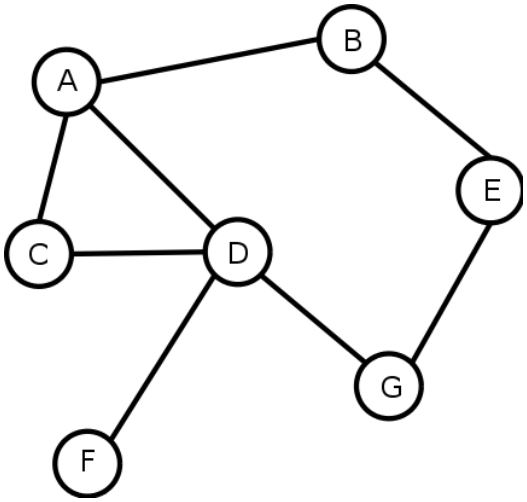


Abbildung 25: Beispiel für das Distance Vector Routing

B Client-Server-Modelle

Aufgabenbeispiele

1. Die Schülerinnen und Schüler könnten ein, bzw. das Schulrechnernetz in seiner Topologie skizzieren, nachdem eine vom Lehrer geführte Schulbesichtigung, in der sich die Schülerinnen und Schüler über das Intranet informieren konnten, durchgeführt worden ist. Der Lehrer steht dabei für Fragen zur Verfügung, gibt Ratschläge und Tipps während der Führung, und ist weiterhin erster Ansprechpartner bei der anschließenden Bearbeitung der Aufgaben.

2. Mit den Systembefehlen `nslookup`, bzw. `ipconfig` kann das Netzwerktopologie-Diagramm nun um Internet-Adressen erweitert werden. Die dafür nötigen Kenntnisse werden hierbei vorausgesetzt. U.U. können sich 2er, bzw. 3er Gruppen zur gemeinsamen Bearbeitung zusammenfinden.

3. Auf einem Arbeitsblatt kann man Sätze vorgeben, die Client-Rechner, Client, Server-Rechner und Server charakterisieren, bzw. solche, die dies nur augenscheinlich tun. Die Aufgabe der Schülerinnen und Schüler ist dabei, herauszufinden, welche der Aussagen zutreffen, alternativ, die Aussagen den richtigen Begriffen zuordnen.

4. Von den Schülerinnen und Schülern kann ein Sequenzdiagramm erstellt werden über die Kommunikation zwischen Client und Server. Dies kann an einer Beispielanwendung wie Browser oder E-Mail geschehen. Weitere, auch von Schülern angeregte Szenarien sind möglich. In allen Fällen sind, wie bei Arbeitsblättern üblich, Sicherungen der erbrachten Leistungen an der Tafel o.ä. durchzuführen, damit auch die Schülerinnen und Schüler eine Kontrolle über ihren Leistungsstand haben, und Lernfortschritte überhaupt erzielen können.

C Dienste und Schnittstellen

Aufgabenbeispiele

1. Die Schüler könnten eine Recherche im Internet zur Begriffsklärung von DNS und DHCP durchführen.

2. Die Schüler könnten auf der website www.dnsstuff.com mit der *DNS-Lookup*-Funktion die IP-Adresse zu einer Ihnen bekannten Website herausfinden, und anschließend statt der Ihnen bekannten URL, diese IP-Adresse in den Browser eingeben.

3. Die Schüler könnten, in Form eines Lücken-Textes und unter der Angabe von geeigneten Recherche Möglichkeiten, sich die Grundlagen von DNS und DHCP erarbeiten.

4. Mit den Schülern und Schülerinnen könnte im Unterrichtsgespräch ein Sequenzdiagramm erstellt werden, welches den Ablauf der Vorgänge skizzierend darstellt, die geschehen, wenn ein neuer Rechner an ein Netzwerk mit DHCP-Server angeschlossen wird.

5. Auf einem Arbeitsblatt Schnittstellen könnten die Schülerinnen und Schüler, angefangen über Mensch-Mensch-Kommunikation, an die Thematik der Schnittstellen innerhalb der Informatik, sowohl Mensch-Maschine, als auch Maschine-Maschine, herangeführt werden.

6. Die Schüler könnten eine vom Lehrenden gestaltet Website besuchen um sich anschließend den Quelltext anzusehen. Es ist empfehlenswert, dass selbst eine Website gestaltet wird, da aktuelle Websites verstärkt auf erweiterte Browserfunktionen wie z.B. php, flash etc. zurückgreifen und der HTML-Aspekt in den Hintergrund getreten ist.

D Peer-to-Peer-Netzwerke

Die Schülerinnen und Schüler lernen Charakteristika dieser Form von Peer-to-Peer-Netzwerken kennen: Jede beliebige Einheit kann aus dem Netz entfernt werden, ohne dass das Netz Schaden nimmt. Beispiele aus der Praxis sind Gnutella, Overnet, Jabber. Vorteile davon sind: Zensur-Resistenz, schwere Angreifbarkeit, Kosten sind unabhängig von der Benutzerzahl. Nachteile sind: Performance der einzelnen Workstations schlechter, technische Umsetzung gestaltet sich schwieriger als beim Client-Server-Modell.

Die Schülerinnen und Schüler lernen Charakteristika von hybriden Systemen kennen: Eine oder mehrere zentrale Instanzen übernehmen Teile der Funktionalität, bei File- Sharing ist dies typischerweise Meta-Daten-Verwaltung. Dies ist bei mehreren Instanzen in der Kommunikationsarchitektur nötig. Beispiele aus der Praxis dazu sind Napster und eDonkey2000. Vorteile davon sind: Effiziente Suche, Kontrolle für den Betreiber. Nachteile sind: Angriffsziel ist Server, Serverkapazität limitiert Performance, Indexaktualisierung ist problematisch.

Die Schülerinnen und Schüler bekommen Einblicke in die Charakteristika der meist real angewendeten Super-Peer-to-Peer-Rechnernetze: Es handelt sich um eine Mischung aus hybriden und Pure-Peer-to-Peer-Systemen. Leistungsfähige Nodes fungieren als Server für Cluster von Clients, dadurch ergibt sich eine erhöhte Zuverlässigkeit durch mehrerer Server in einem Cluster. Ein Beispiel aus der Praxis ist KaZaA. Problem: Der gesamte Ressourcenverbrauch ist höher als bei alternativen Lösungen, hohe Anzahl von Loads für Super-Nodes - dazu erklärt sich in der Regel nicht jeder gerne bereit!

Aufgabenbeispiele

1. Entwerfen Sie ein Pure-Peer-to-Peer-Netzwerk. Ziehen Sie dazu vier Hosts und einen Switch in die Arbeitsfläche und verbinden Sie alle Hosts via Kabelverbindung mit dem Switch. Konfigurieren Sie alle Hosts mit entsprechenden IP- Adressen, sodass jeder Host mit jedem anderen kommunizieren kann.

2. Starten Sie die Anwendung Gnutella und suchen Sie nach der Musik - Datei „Madonna“. Sollte die Suche erfolgreich sein, laden Sie die Datei herunter und beobachten Sie, was alles im Log-Fenster passiert. Versuchen Sie die Log-Einträge nachzuvollziehen! Anschließend sollten Sie alle neu gewonnenen Erkenntnisse überdenken und überlegen, was Sie neues gelernt haben.

3. Entwerfen Sie ein Super Peer-to-Peer Netzwerk. Dabei ist darauf zu achten, eine klare Clusterbildung zu realisieren. Es sollten circa drei Cluster entworfen werden, jedoch nicht weniger als zwei und nicht mehr als fünf. Pro Cluster sollten Idealerweise zwei Server vorhanden sein. Siehe als Anlehnung an die Augabenbeispiele Abbildung 26.

Peer A ist Suchender einer Webseite, Peer B der Anbieter der Musikdatei.Der MessageReceiver von B hört auf Port 6346 und bekommt schließlich irgendwann eine Nachricht von A. Er schaut, welcher Port darin steht. Falls es sich um Port 6346 handelt, weiss er, dass die Anfrage für ihn ist und er somit der richtige Anbieter für den Klienten ist. Er stellt dann die angefragte Datei zur Verfügung und schickt sie an den Klienten. Dort wird die gesamte Nachricht zunächst im Puffer der Internet-Modell-Schichten gepuffert. Die Adresse des Empfängers ist natürlich in der Nachricht enthalten. Jetzt bemerkt der horchende Thread von Peer A, dass er auf dem Port 6346 eine Antwort bekommen hat. Er läd die Nachricht aus dem Puffer und erhält so die gesuchte Datei „Madonna“.

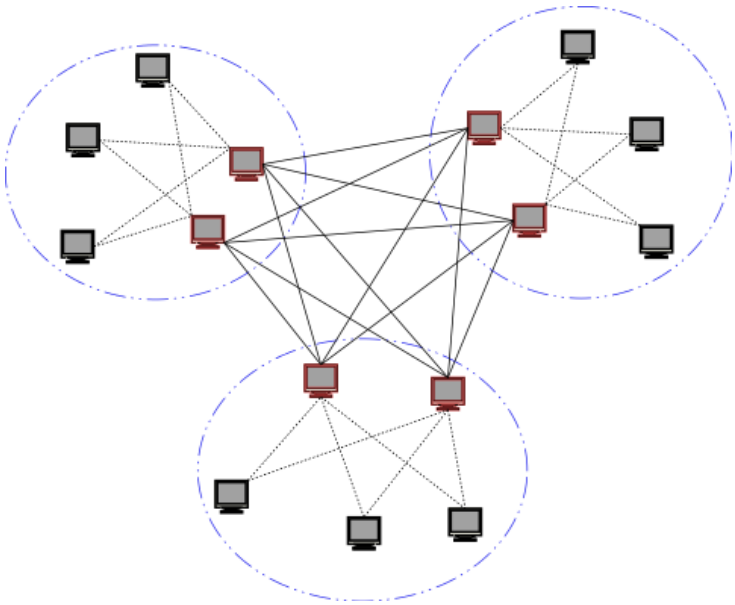


Abbildung 26: Darstellung eines Super-Peer-to-Peer Netzwerks

Hierzu sein Zustandsübergangsdiagramm (beispielhaft - Suchanfrage nach dem „Napster - Prinzip“ an die Musik- Datei „Madonna“) gegeben. Siehe dazu die folgende Abbildung 27:

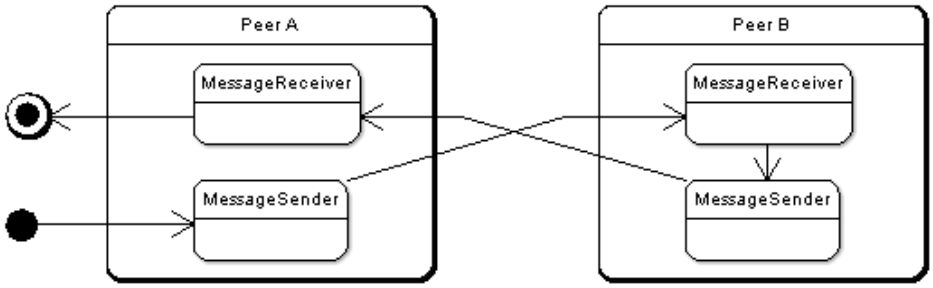


Abbildung 27: Suchanfrage als Zustandsübergangsdiagramm

E Grundlagen

Der Abschnitt Grundlagen beschäftigt sich mit den in FILIUS angewandten Protokollen. Dazu zählen UDP, TCP, IP, SMTP, POP, ARP, HTTP. Die einzelnen Protokolle sind nach dem Internetschichtenmodell geordnet. Diese sind Vermittlung, Transport, Anwendung und sonstige. Die unterste Schicht des Modells behandelt Techniken zur Datenübertragung wie Ethernet, Token Ring und FDDI und PPP (Punkt-zu-Punkt-Protokoll). Die Vermittlungsschicht befasst sich mit Weiterleitung von Paketen und Umsetzung von Routing-Algorithmen. Ein empfangenes Paket wird auf sein Ziel hin untersucht, und dann an dem bestmöglichen nächsten Haltepunkt weitergeleitet. Kern der Schicht ist das Internetprotokoll.

Das Internet-Protokoll (IP) gehört zur TCP/IP Protokollfamilie und ist das heute wichtigste Werkzeug, um skalierbare, heterogene Internet-Netzwerke (das wohl bekannteste davon ist das Internet) zusammenzustellen. IP läuft auf allen Knoten jedes physikalischen Netzwerks und definiert die Infrastruktur, die es diesen Knoten und Netzwerken erlaubt, wie ein einziges logisches Internetwork zu funktionieren. Stichwort: „run over everything“. IP bietet nur das, was mit jeder Netztechnologie realisiert werden kann. IP hat maßgeblich die Aufgabe, Datenpakete zu adressieren und im Netzwerk zu vermitteln (Routing). IP ist in seiner Eigenschaft verbindungslos und paketorientiert, was bedeutet, dass es keine Sorge dafür trägt, ob ein Datenpaket auch das Ziel erreicht. Die Daten werden in Pakete zerlegt und versendet. Daher ist IP unsicher, und die Fehlerüberprüfung muss durch ein anderes Protokoll oder Anwendungsprogramm übernommen werden. Dieses Aufspalten in Pakete nennt man Fragmentierung oder auch Reassembly. Dabei wird eine Datei oder ähnliches in Pakete zerlegt und einzeln verschickt. Das Problem dabei ist, dass jedes Netzwerk seine eigene maximale Framegröße hat.

Ein IP-Paket besteht immer aus einem Header und einem Datenrumpf. Dieses sieht wie in Abbildung 28 aus Wismüller [Wi06] aus: Erklärung der einzelnen Inhalte:

HLen: Länge des Headers in 32-Bit Worten

Length: Länge des Datenteils in Bytes (Maximale Länge 64 Kilobyte)

Ident/ Flags/ Offset: für Fragmentierung/ Reassembly

TTL: Zur Erkennung endlos kreisender Pakete- wird von jedem Router heruntergezählt, bei 0 wird das Paket verworfen

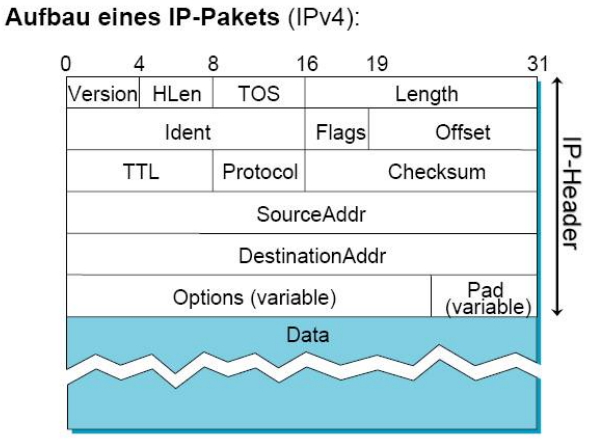


Abbildung 28: Aufbau eines IP-Headers aus Wismüller [Wi06]

Protocol: Demultiplex-Schlüssel kennzeichnet das über IP sitzende Protokoll

Checksum: Prüfsumme für den Header

Mit dieser Struktur ist es dem IP möglich, ein Adressierungsschema im Netzwerk zu erstellen und eine Datagramm- Zustellung zu ermöglichen. Die Adressierung erfolgt durch eine Hierarchisierung und Klassifizierung von Netzen unterschiedlicher Größe. Dabei wird eine 32-Bit-Adresse angegeben, unterteilt in vier 8-Bit-Adressbereiche für die Unterteilung in Netzadresse und Hostadresse, gemäß dem Schema das aus der folgenden Abbildung 29 zu erkennen ist:

Diese binäre Adressierung kann auch als Dezimalzahlen, nach allen 8-Bit mit einem Punkt getrennt, notiert werden.
Bsp.: 11000000.10101000.01111111.11111110 = 192 .168 .127 .254.
Diese Adressierung dient der Skalierbarkeit. Die IP-Adresse wird hierbei unterteilt in eine Netzadresse und eine Hostadresse. Zur weiteren Unterscheidung kann zusätzlich auch noch eine Subnetzmaske über die IP-Adressierung gelegt werden, um besonders große Adressräume weiter zu unterteilen, um so zu einer besseren Ausnutzung der vorhandenen Adressen (Adressräume) zu gelangen. Unter Datagramm- Zustellung versteht man, wie bereits oben

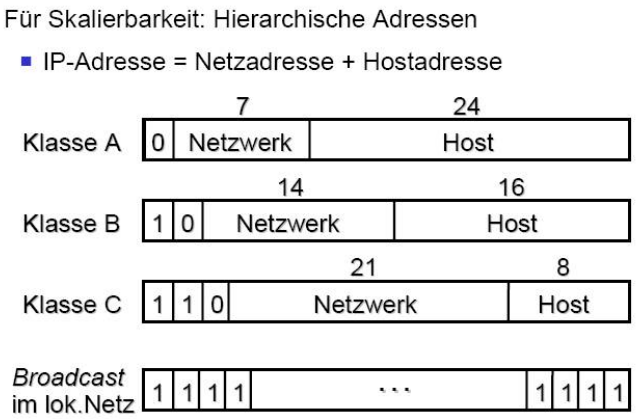


Abbildung 29: Aufbau von IP-Adressen aus Wismüller [Wi06]

angesprochen, dass ein Datagramm eine Paketart ist, die auf verbindungslose Weise über ein Netzwerk versendet wird. Dieser Versand wird nach der „Best-Effort-Methode“ durchgeführt, was bedeutet, dass lediglich versucht wird, das Paket ans Ziel zu bringen, jedoch nichts unternommen wird, wenn das Paket bspw. verloren geht, verfälscht oder falsch zugestellt wird. Daher bezeichnet man IP auch als unzuverlässigen Dienst. In der nächst unteren Schicht des OSI-Referenzmodells findet dann eine Umsetzung der IP-Adressierung eines Pakets, bzw. Frames in eine MAC-Adresse statt, um somit das IP-Paket zu einem eindeutigen Rechner schicken zu können.

Die Transportschicht hat im Internetschichtenmodell die Aufgabe Informationen in Pakete zu verpacken. Das dafür benutzte Protokol heißt TCP. Im Schichtenmodell handelt es sich um die Schicht drei. Sie kennt die darüberliegende Schicht nicht. Es spielt also keine Rolle, von welcher Anwendung TCP im Einzelfall benutzt wird. Die Darunterliegende Schicht eins ist aber bekannt. In der Internetschicht werden die TCP-Pakete wiederum mit IP verpackt (siehe E).

Das Transmission Control Protokoll (TCP) ordnen wir im Internet-Schichtenmodell in die Schicht Nummer drei ein. Es ist verbindungsorientiert. Ein praktisches Beispiel, um sich dies zu merken, ist: Ein Telefonat ist verbindungsorientiert, im Gegensatz zu einer SMS die verbindungslos übertragen wird. Außerdem zählt es zu den zuverlässigen Protokollen im Gegensatz zum

User Datagram Protocol (UDP). Da nach jedem gesendeten Segment ein Acknowlegdement (ACK) vom Empfänger zurückgeschickt wird, kann der Sender sicher sein das alle Daten ohne Verlust angekommen sind. UDP ist lediglich eine Erweiterung vom Internet Protocol (IP) mit den zusätzlichen Eigenschaften der Möglichkeit von Kommunikation zwischen Prozessen und Checksum über die Daten. Hierbei handelt es sich jedoch um ein verbindungsloses und unzuverlässiges Protokoll. Anwendung findet es dort wo nicht zwingend Daten zuverlässig übertragen werden müssen, also zum Beispiel zum Download von Video-Streams. Wenn dort einige Bits verloren gehen bemerkt das der Anwender meist gar nicht.

TCP kennt die darüber liegendende Schicht Anwendung (SMTP/POP, -HTTP) nicht. Es kennt nur die in der Hierarchie niedrigere Internetschicht auf der das Internet Protocol läuft. TCP selbst verpackt die Daten der Anwendungsschicht in einem Paket welches mit dem TCP -Header versehen ist. Das heißt, das fertige TCP -Paket wird in der IP -Schicht noch einmal verpackt (siehe dazu Abschnitt E zum Internet Protocol).

Verbindungsaufbau über Three-way Handshake: Ein Client stellt die Anfrage SYN mit zufällig gewählter Sequenznummer x an Server, Server antwortet mit ACK x+1 und eigener Sequenznummer y, Client antwortet wieder mit ACK y+1. Dieser Vorgang ist notwendig weil TCP einen Verbindungsaufbau benötigt, denn wie oben angesprochen ist es verbindungsorientiert. Der Datenfluss wird dann mit Hilfe des Sliding-Window-Algorithmus durchgeführt. Dabei handelt es sich um eine Erweiterung des Stop-and-Wait-Algorithmus bei dem die Leitung nie ausgelastet wird. Durch das Sliding-Window können mehrere Frames gleichzeitig gesendet werden. Dadurch wird die Bandbreite der Leitung besser ausgenutzt. Erst wenn alle links angeordneten Frames (siehe Abbildung 30) übertragen sind schiebt sich das Fenster weiter. Die Zuverlässigkeit der Übertragung von Byte-Strömen wird durch ständigen Austausch von Acknowledgements (ACK) sichergestellt. Sobald die Bestätigung zurückkommt kann der Sender sicher sein dass ein Frame auch wirklich übertragen wurde. Das Senden erfolgt in Segmenten. Bei TCP sprechen wir von Segmenten im Gegensatz zu IP/UDP bei denen die Datenblöcke Frames genannt werden.

Stop-and-Wait: Bei dieser Idee wird immer ein einzelner Frame gesendet. Der Empfänger antwortet mit einem ACK. Dadurch weiss der Sender dass der Frame angekommen ist und sendet nun den nächsten. Dieser Algorithmus kommt mit einer 1 Bit Sequenznummer aus. Da die Reihenfolge der Frames bekannt ist, reicht es einfach abwechselnd 0 bzw. 1 zu vergeben. Der Empfänger prüft damit ob ein Frame fehlt oder doppelt gesendet wurde. Der große Nachteil von Stop-and-Wait ist jedoch die schlechte Auslastung der

Bandbreite.

Sliding Window: Eine Verbesserung gegenüber Stop-and-Wait ist der Sliding-Window-Algorithmus bei dem ein Schiebe-Fenster eingeführt wird. Es werden mehrere Frames gleichzeitig losgeschickt. Das Fenster wird nach rechts verschoben, wenn alle linken Frames erfolgreich versendet wurden. Logische Folgerung ist eine Sequenznummer von Größe 32 Bit. Großer Vorteil ist die optimale Auslastung der Leitung. Die Fenstergröße ist zunächst statisch (SWS = Sliding Window Size). Dies kann zu Überlast im Netzwerk führen wenn ein oder mehrere Sender gleichzeitig auf einen Empfänger senden. Siehe dazu Abbildung 30. Advertised Window: Der Empfänger gibt dem Sender die

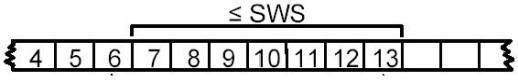


Abbildung 30: Schema des Sliding Windows aus Wismüller [Wi06]

Fenstergröße an. Diese ist also dynamisch und realisiert somit eine Flusskontrolle. Nun kann es nur noch zu Überlastsituationen kommen wenn mehrere Host gleichzeitig senden.

Flusskontrolle: Ein Sender pumpt mehr Daten in die Leitung als der Empfänger aufnehmen kann. Lösung dieses Problem erreicht man durch den Advertised Window. Der Empfänger teilt dem Sender mit wie groß die Fenster-Größe des Sliding Windows höchstens sein darf.

Überlastkontrolle mit Congestion-Window: Die dahinter stehende Idee ist dass jeder Sender selbst bestimmt für wie viele Pakete Platz im Netzwerk ist. Wenn das Netz gefüllt ist bleibt die Ankunft von Acknowledgements (ACKs) aus. Das deutet darauf hin, dass mehrere Sender Daten ins Netzwerk pumpen. Die Daten von allen Sendern gehen dabei auf ein und denselben Router. Dieser hat einen Puffer der überläuft, weil er die Menge an Daten nicht temporär speichern kann. Jeder Host versucht über die Menge der verlorenen Pakete zu bemerken, dass eine Überlastsituation aufgetreten ist. Er reagiert damit, dass er weniger Daten sendet. Anlehnend an die bisher genannten Verfahren des Transmission Control Protocol sei in der folgenden Abbildung 31 der Aufbau eines TCP-Headers gezeigt und anschließend erläutert:

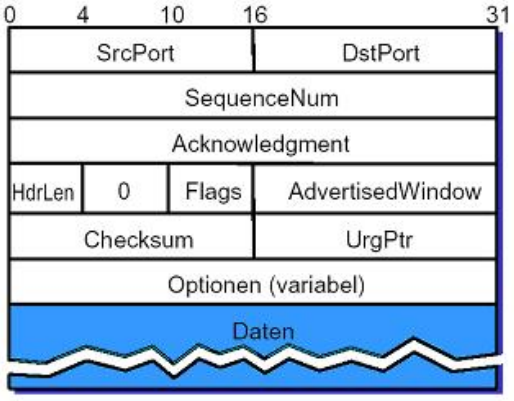


Abbildung 31: Aufbau eines TCP-Headers aus Wismüller [Wi06]

Erläuterung der einzelnen Felder des TCP-Headers:

SrcPort: Source-Port, beinhaltet die Daten die vom Sender ins Netzwerk entlassen werden.

DstPort: Empfänger Port, gibt den Port an auf dem der Empfänger den Daten Einlass gewährt.

SequenceNum: Sequenznummer, beinhaltet eine 32 Bit große Nummer, die zufällig erzeugt wird.

Acknowledgment: Bestätigung, Abk.: ACK, berechnet die Antwort auf die Sequenznummer welche als Antwort zurückgeschickt wird.

HeaderLength: bestimmt die maximale Größe des Headers.

Flags: gibt eine Menge von Standardbefehlen an. In dieser Menge sind folgende Befehle enthalten: SYN = Verbindungsaufbau, FIN = Verbindungsabbau, ACK, URG = dringende Daten, PUSH, RESET = Verbindungsabbruch nach Fehler.

AdvertisedWindow: ist ein 16 Bit großes Feld für Fenstergröße des AdvertisedWindow.

Checksum: enthält die Prüfsumme über die enthaltenen Daten.

Datagramm: enthält die tatsächlichen Daten die im Anhang mitgesendet werden sollen.

Die Anwendungsschicht umfasst alle Protokolle, die mit Anwendungensprogrammen zu tun haben. Anwendungsprotokolle sind in FILIUS: HTTP, POP, SMTP, DNS und DHCP.

Literatur

[Pl76] K.H. Platte und A. Kappen - *Wirtschaftslehre im Unterricht 1.* Ravensburg 1976.

[PD03] Larry L. Peterson, Bruce S. Davie - *Computernetze - Ein modernes Lehrbuch.* 1. Auflage, 2000

[Pe03] Larry L. Peterson, Bruce S. Davie - *Computernetze - Eine systemorientierte Einfuehrung.* dpinkt.verlag GmbH 2003.

[Wi06] Skript von Roland Wismueller: *Rechnernetze 1*, Universität Siegen, URL: http://www.bs.informatik.uni-siegen.de/www/lehre/ss06/rn1. Stand SS 2006.

[Lu06] Count to Infinity Problem: http://wiki.uni.lu/secan-lab/Count-To-Infinity+Problem.html. 2004-2006 University of Luxembourg, SECAN-Lab

[Si06] Simba-RvS: Routing-Prinzipien, http://rvs.die.informatik.uni-siegen.de/Wilus?document= Routing_Animation&kapitel=Weitverkehrsdatennetze. Simba, 2003

[Ke01] Kerres: *Multimediale und telemediale Lernumgebungen.* 2001

[CS01] A. Schwill und V.Claus:*Duden der Informatik.* 2001

[SS04] Sigrid Schubert, Andreas Schwill: *Didaktik der Informatik.* 2004

[Bl00] Herwig Blankertz: *Theorien und Modelle der Didaktik.* 2000

[Gu03] Herbert Gudjons: *Pädagogisches Grundwissen.* 2003

[LI06] http://www.log-in-verlag.de/news-archiv/newsletter_02_2006.htm

[RVS] http://rvs.die.informatik.uni-siegen.de/Wilus?document=Routing_Animation&kapitel=Weitverkehrsdatennetze

[Me02] Merzky, A.: *Einführung in P2P-Netzwerke. Das Gnutella Netz.* Limewire Gnutella Servent. http://archiv.tu-chemnitz.de/pub/2002/0047/data/html_sem.html (zuletzt gesichtet am 16. August 2007)

[Re02] Reiser, H.-P.: *Vorlesung Verteilte Algorithmen.* 11. Peer-to-Peer-Netzwerke. http://www4.informatik.uni-erlangen.de/Lehre/WS02/V_VA/Skript/VA-11a_2p.pdf (zuletzt gesichtet am 16. August 2007)

[SW] SarWiki: *Gnutella 0.4.* http://sarwiki.informatik.hu-berlin.de/Gnutellaß_0.4 (zuletzt gesichtet am 16. August 2007)