Themenspezifische Informationssuche im Internet mit Hilfe mobiler Programme

Von der Fakultät Informatik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr.rer.nat.) genehmigte Abhandlung

Vorgelegt von

Wolfgang Theilmann

aus Bielefeld

Hauptberichter:	Prof. Dr. rer. nat. Kurt Rothermel
Mitberichter:	Dr. ing. habil. Fritz Schmidt
Tag der mündlichen Prüfung:	25.Juli 2000

Institut für Parallele und Verteilte Höchstleistungsrechner (IPVR)
der Universität Stuttgart

2000

Danksagung

Mein besonderer Dank gilt meinem Doktorvater, Prof. Dr. Kurt Rothermel, der mich in einer optimalen Mischung aus helfender Unterstützung in schwierigen Situationen und der Gewährung großer Freiheit bei der Bearbeitung meines Forschungsthemas betreut hat.

Ebenso möchte ich meinem Zweitbetreuer, Dr. habil. Fritz Schmidt, herzlich danken für manch kritische Anmerkung und konstruktive Anregung, die er mir bei den Kolloquien des Graduiertenkollegs mit auf den Weg gab.

Großen Dank schulde ich der Agentengruppe aus Markus Straßer, Fritz Hohl und Joachim Baumann. Die drei haben mir mit großer Hilfsbereitschaft den Einstieg in die Forschungs- und Agentenwelt erheblich erleichtert und wurden nicht müde, irgendwelche langweiligen Artikel von mir Korrektur zu lesen.

Schließlich möchte ich mich auch bei meinen Kollegen aus der Abteilung Verteilte Systeme für die stets humorige Atmosphäre und die große Hilfsbereitschaft bedanken. Insbesondere ihr großes Wohlwollen gegenüber meinem Technikunwissen ist gar nicht lobend genug zu erwähnen.

Friedhelm Buchholz danke ich für seine höchst wertvollen Hilfestellungen bei Problemen der Graphentheorie.

Nicht zuletzt möchte ich mich bei der Deutschen Forschungsgemeinschaft (DFG) bedanken, die durch ihr großzügiges Stipendium diese Arbeit überhaupt erst möglich gemacht hat.

Stuttgart, den 17. September 2000

Wolfgang Theilmann

Kurzfassung

Das Forschungsgebiet der Informationssuche (Information Retrieval) wird durch das Aufkommen des Internets bzw. des World Wide Web mit völlig neuen Herausforderungen konfrontiert. Im Gegensatz zu herkömmlichen Datenbeständen zeichnet sich der Informationsraum Internet durch seinen immensen Umfang, die hohe Dynamik seiner Datenbestände, die Heterogenität der Inhalte und Formate sowie durch seine Verteilung über Rechner auf der ganzen Erde aus.

Um auch in diesem Informationsraum präzise, effizient und umfassend nach Informationen suchen zu können, wird in dieser Arbeit ein Konzept zur Spezialisierung von Suchmaschinen auf einzelne Themengebiete vorgeschlagen. Solche Suchmaschinen erkennen die für sie relevanten Dokumente anhand einer speziellen Filterfunktion und können ihren Benutzern eine spezifische, d.h. auf das jeweilige Themengebiet angepasste, Benutzungsschnittstelle und Suchfunktionalität bieten.

Um die für eine spezialisierte Suchmaschine relevanten Dokumente effizient zu lokalisieren, wird die Technologie der mobilen Programme eingesetzt. Anstatt alle zu untersuchenden Dokumente zur Suchmaschine zu übertragen, werden mobile Filterprogramme zu den Datenbeständen gesandt, untersuchen diese „vor Ort“ und liefern lediglich die relevanten Dokumente zurück. Es werden Verfahren vorgestellt, mit denen die Aussendung der mobilen Programme so koordiniert werden kann, dass die resultierenden Kommunikationskosten minimiert werden.

Da von diesen Aussendungsverfahren Kenntnisse über die netzwerktechnische Entfernung zwischen den beteiligten Rechnern benötigt werden, wird zudem ein Ansatz vorgestellt, der die Schätzung beliebiger Netzwerkdistanzen im Internet auf skalierbare und effiziente Weise ermöglicht.

Die Tragfähigkeit der Konzepte zur Schätzung von Netzwerkdistanzen und zur Aussendung mobiler Programme wird anhand umfangreicher Messungen evaluiert. Zudem wird in einer Fallstudie der Nutzen der themenspezifischen Suchmaschinen sowie der in ihrem Kontext erfolgende Einsatz mobiler Filterprogramme analysiert.

Domain specific Information Retrieval in the Internet with mobile Programs

English abstract of the dissertation

Research on information retrieval has been challenged in a new way by the increasing importance and usage of the Internet and especially the World Wide Web. In contrast to traditional data bases, the information space Internet is characterized by its enormous size, the dynamics of its content, the heterogeneity of the documents and formats and by the distribution of the available documents among hosts all over the world.

This thesis presents a new approach for search engines, which are specialized to single domains, thus enabling the precise, efficient and comprehensive retrieval of searched information. Such search engines use a domain specific filter function for recognizing relevant documents and are able to offer a user interface and retrieval functionality that are adapted to the specific domain.

For locating relevant documents in a fairly efficient way we rely on mobile program technology. Instead of downloading the documents to be analysed to the search engine, we send out mobile filter programs towards the relevant data sources. The mobile programs analyse the data locally and return only the relevant part of it. We present algorithms that allow to coordinate the program dissemination such that the resulting communication costs are minimized.

Since the dissemination algorithms need to know the network distance between the involved hosts we also present an approach for estimating the network distance between arbitrary hosts in the Internet in a scalable and efficient way.

We evaluate the methods for estimating network distances and for disseminating mobile filter programs on the basis of extensive measurements in the Internet. In addition, we present a case study in order to analyse the benefit of specialized search engines and especially of the employment of mobile filter programs in the context of such search engines.

The following sections present a short summary of the thesis chapters. They correspond to the three main subjects of the thesis, i.e. the approach of specialized search engines, the estimation of network distances and the dissemination of mobile programs. Three additional sections are devoted to the evaluation of each of the three approaches.

Domain experts for information retrieval in the Internet

Problem and Related Work

The traditional way of searching the Internet (resp. the World Wide Web) is either to rely on general purpose, robot-based search engines (such as AltaVista) or on manually generated directories (such as Yahoo). Robot-based search engines offer fairly comprehensive but imprecise results to user queries. In contrast, directories are precise but due to their manual generation extremely incomprehensive. Especially, they cannot keep up with the fast evolution of the Web.

Specialized automatic search tools seem to be a promising synthesis of the two approaches because they can achieve both, a comprehensive and within a certain domain precise index. However, they have to face three major problems:

1. The scope problem, i.e. how to determine whether an arbitrary document is relevant for the special scope of the search engine.
2. The location problem, i.e. how the relevant documents existing in the World Wide Web can be efficiently located.
3. The generality problem, i.e. whether it is possible to apply the techniques from one search engine to create a new one for another domain, and if yes, how much human effort is needed for this application.

This thesis deals with the location and the generality problem. The scope problem has been already extensively analysed by researchers in the area of artificial intelligence. We only address it in the context of a case study.

Approach

In the following, we describe our approach for specialized search engines which we call *domain experts*. A domain expert is supplied with domain specific knowledge (strategies, algorithms etc.), which enables it to decide on its own whether a document belongs to its domain or not.

Domain experts automatically build up a specialized index. In contrast to traditional indexes, this index comprises keywords that can be attributed in a domain specific way and meta descriptions of documents that are expressed implicitly, for instance in the document's structure or environment.

The construction of this index happens in the following way: At the beginning of its life cycle a domain expert has its domain specific knowledge but does not know any document that belongs to its domain. The expert expands its document knowledge by learning from a traditional search engine. In this phase the expert is already able to answer user queries. An incoming query is forwarded to a traditional search engine. The resulting links (URLs) are taken as a first hint pointing to interesting documents. Documents that are not yet registered in the expert's knowledge base are scanned with a domain specific filter function in order to decide which of them are relevant enough for being integrated into the knowledge base. Finally, the user query is answered on basis of the knowledge base.

Later on, as the document knowledge of the domain expert grows it becomes more and more *autonomous*, i.e. independent of traditional search engines. It is able to answer the majority of user queries directly by inspecting its document knowledge base. A search engine is just queried to check whether new documents have been published or whether old documents have been updated. Remote documents need to be examined only if new or modified documents have been found by the search engine. This is expected to happen only seldom in the autonomous phase.

Finally, a domain expert tries to discover new documents *proactively* without the context of a previous user query. This can be done in several ways:

- The environment of an interesting document (i.e. documents on the same Web server) can be examined for other possibly interesting documents.
- Relevant documents already found can be processed for further hints, such as URLs, a person's name, names of conferences etc. These hints can be exploited to find other interesting documents.
- External knowledge bases can be queried to acquire new or additional knowledge.

After acquiring enough hints the domain expert generates a proactive internal query which is then processed like an ordinary user query.

In order to reduce the network load induced by the examination of remote documents a domain expert exploits the technology of mobile programs. The filter function is designed as a mobile program that can be shipped towards those data sources that need to be examined. By this, the examination can be done almost locally and only the relevant documents need to be transferred over the network.

For easing the effort to create new domain experts we developed a framework that offers all the functionality that is needed by every domain experts. Most of the domain specific

aspects (such as specific attributes or a specific user interface) can be defined by a simple configuration file. The only aspect where manual implementation is still needed is the filter function.

The approach of domain experts together with an in depth review of related work has been published in [Theilmann & Rothermel, 1998] and [Theilmann & Rothermel, 1999c].

Dynamic distance maps of the Internet

Knowledge about network distances is an important prerequisite for many network applications that want to optimize their network communication. One such application is a domain expert that uses mobile programs for filtering remote data sources. Distance knowledge enables a domain expert to estimate the benefit of using a mobile filter program to perform a special filtering task.

Problem and Related Work

Facing the size of the Internet which consists of several million hosts it is very difficult to acquire distance knowledge in a scalable and efficient way. Obviously, it would not be a scalable solution to perform measurements for every pair of hosts.

A very efficient method to determine network distances would be to exploit the information available in the routing tables of Internet routers. However this has some problems too, because the access to routing tables is not public, tables only contain information about single subnets and within different subnets different distance measures are used.

Simple measurements from client hosts are not sufficient either because they only allow to estimate distances between the client and other hosts. Distances between two remote hosts cannot be obtained this way. I addition, there is a high overhead that comes an approach in which every single host is responsible for performing measurements. Consider, for example, two closely connected clients of a replicated service. Since they do not know of each other they have to do almost the same measurements.

Approach

Our approach to create a global view on the Internet, a so-called *network distance map*, assumes the availability of a set of measurement servers distributed all over the Internet and allowing to perform distance measurements to arbitrary hosts. The basic idea is to

estimate the distance between arbitrary hosts by first determining the two measurement servers that are most closely connected to them and then, measuring and taking the distance between the two servers as estimation for the inter-host distance. The scalability problem is addressed by partitioning the set of measurement servers into clusters of closely connected hosts. This partitioning process is recursively repeated, so we obtain a tree structure in which each node represents a cluster of hosts. The clusters are refined in a root-to-leaf direction. The tree's leaf nodes correspond to single measurement servers. Arbitrary hosts (which are not measurement servers) can be integrated into this data structure by assigning them to their closest measurement server. The cluster tree is extended by distance values between sibling nodes. The distance between two sibling clusters is an estimation for the average distance between arbitrary hosts within these clusters.

This representation of a computer network allows to estimate the network distance between arbitrary network hosts. Having two hosts we simply have to go up the tree hierarchy until we find two sibling ancestors of the hosts. The distance between these ancestors is then taken as estimation for the inter-host distance. The storage requirements for the representation of a distance map are linear in the number of hosts and measurement servers.

We developed a couple of algorithms for computing distance maps. First of all, we specified two cluster criteria according to which clustering is performed. The criteria either seek to minimize the distance between hosts sharing the same cluster or try to maximize the distance between hosts residing in different clusters. We also developed algorithms for integrating additional hosts into the cluster tree, a random algorithm and two tree insertion algorithms which follow a depth/breadth first search strategie. Finally, we present algorithms for adapting distance maps to changed network conditions.

Finally, we also developed protocols for coordinating the network measurements that need to be done and for distributing the resulting network distance map.

The approach of dynamic network distance maps together with an in depth review of related work has been published in [Theilmann & Rothermel, 2000a].

Dissemination of mobile programs for distributed information filtering

Problem and Related Work

In the context of domain experts we described a scenario in which data distributed among various remote data servers is to be examined with a mobile filter program. This scenario

can be exactly specified by the following parameters: (1) the data servers that are to be examined, (2) the amount of data to be examined on each platform, (3) the amount of data resulting from each examination, (4) the size of the mobile filter program and (5) the base platform, where the mobile filter is initially available and where all results must be delivered to.

A *dissemination schedule* describes how the dissemination of a mobile filter program is coordinated for filtering a given set of data servers. It consists of a sequence of program transfers and replications in form of a tree. Tree nodes represent code platforms to which the mobile filter is sent to, edges correspond to program transfers. Additionally, the schedule assigns data servers, which must be examined from this platform, to each visited code platform.

Assuming that network costs can be expressed as a function of the data amount and the network distance across which the data has to be shipped we can compute a cost function that describes the network costs related to an arbitrary dissemination schedule. The problem is to find a schedule that minimizes this cost function.

Unfortunately, we found out that the dissemination problem is NP-complete and not approximable with satisfying quality and efficiency. Algorithms for multicast-routing are not applicable since multicast groups are persistent for a large number of messages. Other methods developed in the context of replicate dissemination (which is an analogue problem) do not optimize the network costs or turn out to be too simplistic.

Approach

We developed two algorithms for computing efficient dissemination schedules. The *basic algorithm* follows a simple greedy heuristic. It starts with a trivial schedule, in which all data servers are examined from the base platform. Then it tries to extend the current solution step by step by adding that code platform to the schedule that leads to the largest cost savings when sending a mobile filter to it. This extension is done by first computing new schedules that are extended by one of the not yet used code platforms. For each such schedule, we can then compute an optimal assignment of the data servers which allows to compute the overall communication costs related to them. Finally, the schedule with the minimal costs is selected.

A more scalable, *hierarchical algorithm* follows a divide-and-conquer strategy. It exploits the data structure of cluster trees (see above) by traversing the cluster tree and applying the basic algorithm to groups of sibling clusters. The traversal starts from the base

platform, which is a leaf node of the tree. Then it goes up the tree applying the basic algorithm to each set of sibling nodes. If the basic algorithm decides that another tree node shall receive a copy of the filter the traversal opens another branch going down the tree from the receiving node. While the complexity of the basic algorithm is linear in the number of available code platforms the second one only has logarithmic effort.

The approach of disseminating mobile filter programs together with an in depth review of related work has been published in [Theilmann & Rothermel, 1999a] and [Theilmann & Rothermel, 2000c].

Evaluation of the distance map algorithms

Methodology

We performed an extensive experimental validation of the approach to compute network distance maps on the basis of data acquired from Internet measurements. The measurements were done for the two distance metrics hop-count and round trip time (rtt). We compiled a list of 119 measurement servers and 460 additional hosts, which are distributed among 5 continents and numerous autonomous systems. We measured the hop-count and the rtt distance from every measurement server to all other hosts (measurement servers and additional hosts).

The data analysis showed quite distinct characteristics for the two distance metrics. While hop-count distances are symmetrically distributed with a small deviation the distribution of round trip times turned out to be very asymmetric with a large deviation. In the sense of hop-count the average host has its closest measurement server at a distance of 7.1 hops, which is quite far away because the global average hop-count distance is only 16. In contrast the average rtt distance of hosts to their closest measurement server is only $27ms$ (global average is $255ms$). This shows that in the sense of rtt our set of 119 measurement servers covers the Internet quite well. For every host there is a closely connected measurement server. This property is fulfilled much worse for the hop-count distances.

Results

The main results of our experiments can be summarized in the following way: Clustering performs well for both cluster criteria and both distance metrics. However, the resulting estimation error and the efficiency of the cluster tree (which depends on how well the tree

is balanced) vary for the different scenarios. There is no natural clustering that has an optimal balance of estimation error and tree efficiency. The insertion of additional hosts into the cluster tree can be done most efficiently with the depth first search strategy.

The final estimation of distances between simple hosts is not yet satisfying. The order of magnitude is correctly estimated and this can be already very helpful for many applications. For example, it is sufficient for efficiently disseminating mobile filter programs. However, the remaining estimation error is still high. This has two reasons: As the data analysis has already shown for the hop-count distances the number of measurement servers is simply too small. For rtt distances the measurement method is not robust enough. Instead of relying on single measurements average values of a bundle of measurements should be used instead.

However, the successful experiments for two such different distance metrics promise that network distance maps can be built for arbitrary distance metrics.

Finally, the we conducted experiments that show that distance maps can be adopted to changed network conditions at low costs, i.e. with only some few measurements.

More details about the evaluation of network distance maps can be found in [Theilmann & Rothermel, 2000a].

Evaluation of the dissemination algorithms

Methodology

The evaluation of the dissemination algorithms was done with a trace-driven simulation which is based on the network measurements done in the context of network distance maps (see above). We assumed that each measurement server is a code platform, i.e offers to host mobile programs.

We analysed various test scenarios in which we made the following assumptions: The normalized size of the mobile filter is 1. Data amounts are always expressed with respect to this size. The amount of data is constant for all servers that are considered in a specific scenario. A single scenario is determined by the number of data servers and the amount of data to be examined. For each scenario we ran 100 tests with distinct sets of data servers and code platforms. Results are averaged over these 100 runs. The basic algorithm is executed on basis of the measured distances, the hierarchical algorithm on basis of estimations from the cluster tree.

For each scenario we computed the cost reduction on basis of the measured distances. This reduction is defined as 1 minus the ratio of the costs related to the execution of a dissemination schedule by the costs related to the traditional client server scenario in which all documents need to be transferred to the base platform.

Results

The main results of our experiments can be summarized in the following way: Both algorithms (the basic and the hierarchical one) compute effective dissemination schedules that significantly reduce the communication costs for both distance metrics (hop-count and round trip time). The basic algorithm showed the following performance: Cost reduction for the round trip time (rtt) metric goes up to 90% and is already about 60% for only 10 data servers and a data amount equal to the filter size. Reductions in the sense of hop-count distances go up to 60% and are about 45% for 10 data servers and a data amount equal to the filter size. Of course, the basic algorithm achieves more effective schedules (at the cost of higher computational effort) than the hierarchical one. However the difference between both algorithms nearly vanishes for higher data amounts (4 times larger than the filter size) and larger sets of data servers (400). For the scenario with 10 data servers the hierarchical algorithm results to cost reductions of 35% (round trip time) and 13% (hop-count). We found out that the worse performance of the hierarchical algorithm is mainly caused by the large error of the distance estimations done with a cluster tree. Instead, the algorithmic abstraction of clusters does not have a significant effect.

More details about the evaluation of the dissemination algorithms can be found in [Theilmann & Rothermel, 1999a].

Case study for the domain of scientific articles

Finally, we performed a case study for a domain expert specialized to the domain of scientific articles. We implemented a simple filter function that is able to recognize whether a given HTML-document (written in English) is a scientific article or not. The filter function makes heavy use of the stereotypical structure of such documents and uses fuzzy logic for detecting typical structural elements in a given text.

The evaluation of this case study was done with some example queries, which are typical for the domain of scientific articles. The queries were sent to a traditional search engine

and the resulting links were analysed with the filter function. In addition, we simulated the case of performing the analysis with mobile filters, based on the settings already used for evaluating the dissemination algorithms (see above).

The experiments show that the use of a domain expert is highly beneficial for the user because it allows to locate relevant documents much faster (and at a much higher accuracy) than it is possible with a traditional search engine. Details of this assessment can be found in [Theilmann & Rothermel, 1999c].

The experiments also show that the use of mobile filter programs leads to a significant reduction of the communication costs. The reduction varies for the different scenarios (different distance metrics and algorithms) between 40% and 88%. More details about this evaluation are described in [Theilmann & Rothermel, 1999a].

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Motivation und Problemstellung 1

1.2 Überblick und Aufbau 3

2 Domänen–Experten zur Informationssuche im Internet **6**

2.1 Klassische Ansätze zur Informationssuche im Internet 8

2.1.1 Zentralisierte Ansätze 8

2.1.2 Hierarchisch verteilte Ansätze 9

2.1.3 Dezentrale Ansätze 12

2.1.4 Spezialisierte Ansätze 13

2.1.5 Diskussion 15

2.2 Domänen–Experten — Funktionsweise und Architektur 16

2.2.1 Bearbeitung von Benutzeranfragen 17

2.2.2 Verteilte Dokumentenfilterung 19

2.2.3 Wissensaktualisierung und -ausbau 21

2.3 Ein konfigurierbares Framework 22

2.4 Diskussion 27

3 Dynamische Distanzkarten vom Internet 29

3.1 Problemstellung 30

3.1.1 Motivation 30

3.1.2 Distanzmaße und Distanzmetriken 32

3.1.3 Grundlagen des Internet 33

3.2 Existierende Ansätze zur Distanzbestimmung 34

3.3 Ansatz der Netzwerk–Distanzkarten 37

3.3.1 Systemmodell 38

3.3.2 Anforderungen 39

3.3.3 Idee 40

3.3.4 Datenstruktur und Distanzableitung 42

3.4 Algorithmen zur Kartenerstellung und -anpassung 45

3.4.1 Ballungskriterien und -berechnug 45

3.4.2 Berechnung des Regionenbaumes 49

3.4.3 Integration einfacher Endsysteme 53

3.4.4 Aktualisierung von Distanzkarten 55

3.5 Koordination der Distanzmessungen und Kartenreplikation 58

3.5.1 Koordination der Regionenbaumberechnung 59

3.5.2 Koordination der Zuordnung von Endsystemen 61

3.6 Diskussion und Erweiterungen 64

3.6.1 Behandlung von Knoten- und Netzwerkfehlern 65

3.6.2 Flexible Anpassung von Distanzkarten 68

4 Aussendung mobiler Programme zur verteilten Informationsfilterung 69

4.1 Grundlagen mobiler Programme 71

4.2 Problemstellung 73

4.2.1 Erweitertes Systemmodell 73

4.2.2 Formalisierung des Aussendungsproblems 74

4.2.3 Komplexität des Aussendungsproblems 76

4.3 Verwandte Ansätze 77

4.3.1 Reduktion der Kommunikationskosten durch mobile Programme . . 77

4.3.2 Aussendungsverfahren 79

4.4 Allgemeines Aussendungsverfahren 81

4.5 Hierarchisches Aussendungsverfahren 83

4.6 Diskussion und Erweiterungen 89

4.6.1 Allgemeine und effiziente 1:n–Kommunikation 90

4.6.2 Zusammenspiel mit Caching–Verfahren 91

5 Experimente zur Kartenberechnung 93

5.1 Methodik 93

5.2 Initiale Erstellung von Karten 98

5.3 Anpassung von Karten an Netzwerkänderungen 107

5.4 Diskussion 110

6 Evaluierung der Aussendungsverfahren 112

6.1 Methodik 112

6.2 Basisalgorithmus mit gemessenen Distanzen 114

6.3 Basisalgorithmus mit geschätzten Distanzen 116

6.4 Hierarchischer Algorithmus 119

6.5 Diskussion 121

7 Fallstudie zur Domäne „wissenschaftliche Artikel“ 123

7.1 Filterverfahren 124

7.2 Effektivität des Filterverfahrens 127

7.3 Kostenreduktion durch mobile Filterprogramme 131

7.4 Diskussion 133

8 Resümee 135

8.1 Zusammenfassung 135

8.2 Bewertung und Ausblick 137

A Glossar und Abkürzungsverzeichnis 138

B Mathematische Bezeichner 144

Literaturverzeichnis 145

Abbildungsverzeichnis

2.1 Architektur eines Domänen–Experten; Anfragenbearbeitung 18

2.2 Gruppenweise Untersuchung von Dokumenten durch mobile Filter 20

2.3 Beispieldefinition eines Dokumentenmodells 23

2.4 Beispieldefinition einer Anfragemaske 24

2.5 Beispieldefinition einer Ergebnisdarstellung 25

3.1 Distanzschätzung mit Hilfe von Mess–Servern 41

3.2 Repräsentation einer Netzwerk–Distanzkarte 43

3.3 Algorithmus zur Berechnung einer *Max-k-Trennung* 48

3.4 Algorithmus zur Berechnung einer *Min-k-Ausdehnung* 49

3.5 Zwei extreme Ballungsergebnisse 49

3.6 Algorithmus zur Berechnung eines Regionenbaumes 50

3.7 Algorithmus zur Integration einfacher Endsysteme 54

3.8 Algorithmus zur Anpassung des Regionenbaumes 56

3.9 Algorithmus zur Anpassung des Zuweisung von Endsystemen 57

3.10 Steuerung einer Ballung durch einen Mess–Koordinator 60

3.11 Steuerung der Integration von Endsystemen 62

3.12 Steuerung der Anpassung von Endsystemen durch einen Mess–Server . . . 64

4.1 Beispiel zur Aussendung eines mobilen Filterprogrammes 75

4.2 Basisalgorithmus zur Aussendungsplanberechnung 82
4.3 Transformation eines Netzwerkbaums 85
4.4 Hierarchischer Algorithmus zur Aussendungsplanberechnung 87

5.1 Histogramme der gemessenen Distanzen aus Datenerhebung $D1$ 96
5.2 Divergenzverteilung zwischen den Datenerhebungen $D1$ und $D2$ 98
5.3 Optimale Ballung: Schätzfehler und Regionenbaumtiefe 99
5.4 Gemischte Ballung (Knotenzahl): Schätzfehler und Messaufwand 102
5.5 Gemischte Ballung (Umlaufzeit): Schätzfehler und Ordnungsfehler 103
5.6 Gemischte Ballung (Umlaufzeit): Messaufwand 104
5.7 Integration einfacher Endsysteme: Approximationsratio 105
5.8 Fehler der Distanzschätzung zwischen einfachen Endsysteme 106
5.9 Anpassung einfacher Endsysteme: Approximationsratio vs. Messaufwand . 110

6.1 Basisalgorithmus (Knotenzahl): Kostenreduktion und Filtertransfers 114
6.2 Basisalgorithmus (Umlaufzeit): Kostenreduktion und Filtertransfers 115
6.3 Basisalgorithmus (Knotenzahl) basierend auf Regionenbäumen 117
6.4 Vergleich Basisalgorithmus (Knotenzahl) mit zwei Ballungskriterien 117
6.5 Basisalgorithmus (Umlaufzeit) basierend auf Regionenbäumen 118
6.6 Vergleich Basisalgorithmus (Umlaufzeit) mit zwei Ballungskriterien 118
6.7 Hierarchisches Verfahren (Knotenzahl) 119
6.8 Verfahrensvergleiche (Knotenzahl) 120
6.9 Hierarchisches Verfahren (Umlaufzeit) 121
6.10 Verfahrensvergleiche (Umlaufzeit) 121

7.1 Definition von Strukturattributen durch unscharfe Mengen 126
7.2 Erkennungsqualität für die Anfragen (1) und (2) 129
7.3 Erkennungsqualität für Anfrage (3) 130
7.4 Präzision–Abruf–Diagramme für die Anfragen (1) und (3) 131
7.5 Präzision–Abruf–Diagramm für Anfrage (2) 132

Tabellenverzeichnis

3.1 Optimierung einiger heuristischer Ballungskriterien 47

5.1 Eigenschaften von Regionenbäumen auf den Ebenen 0–2 100
5.2 Anpassung von Regionenbäumen auf Ebene 1 und 2 108

7.1 Drei Beispielanfragen 128
7.2 Resultate zu den Beispielanfragen aus Tabelle 7.1 128
7.3 Reduktion der Kommunikationskosten bei drei Beispielanfragen 133

Kapitel 1

Einleitung

1.1 Motivation und Problemstellung

In den vergangenen Jahren hat sich das Internet zu einem Publikationsforum von überragender Bedeutung entwickelt. Über eine Reihe von Dienste können Benutzer des Internets beliebige Dokumente der Öffentlichkeit zur Verfügung stellen. Unter diesen Diensten hat das *World Wide Web*[↑1] (kurz Web) die mit Abstand größte Bedeutung erlangt. Ein paar Zahlen zum Umfang und Wachstum der im Web verfügbaren Dokumente mögen dies verdeutlichen. So waren Ende 1997 bereits 320 Millionen Web–Seiten öffentlich zugänglich [Lawrence & Giles, 1998]. Bis Februar 1999 stieg diese Zahl weiter auf 800 Millionen Seiten, die zirka 6 Terabyte an Daten umfassten und auf 3 Millionen Servern verteilt waren [Lawrence & Giles, 1999]. Zieht man in Betracht, dass Web–Server neben einfachen Web–Seiten noch eine Fülle von Dokumenten anderen Typs zur Verfügung stellen (z.B. Postscript–Dateien, Sprach- und Klangdateien, Programmcode), so muss der wahre Umfang des Webs noch wesentlich größer geschätzt werden. Schließlich gibt es im Internet neben dem Web noch eine Reihe von anderen, allerdings weniger populären Informationsdiensten, so zum Beispiel *Usenet–News*[↑], *FTP*[↑] oder Digitale Bibliotheken.

Damit Benutzer von dem ausuferndem Informationsraum Internet auch wirklich profitieren können, müssen ihnen Werkzeuge zu dessen Beherrschung gegeben werden. Insbesondere braucht ein Benutzer Werkzeuge, um genau die für ihn relevanten Informationen mit möglichst geringem Aufwand zu finden. In diesem Zusammenhang werden

[1]Der Pfeil [↑] verweist darauf, dass der vorangehende Begriff im Glossar (Anhang A) aufgeführt ist.

zwei grundsätzliche Aufgabenstellungen unterschieden [Wondergem et al., 1997]: Das Gebiet des *Information Retrieval*$^{\uparrow}$ (IR) versucht, dynamische Informationsbedürfnisse, die spontan auftreten, zu befriedigen. Ein Beispiel für ein spontanes Informationsbedürfnisse ist die Suchanfrage nach Abhandlungen über Mozart's Kindheit. Dagegen behandelt das Gebiet des *Information Filtering*$^{\uparrow}$ (IF) Szenarien, in denen Informationsbedürfnisse über einen längeren Zeitraum hinweg existieren, und wo die jeweils relevanten Informationen aus einem Datenstrom herauszufiltern sind. Ein Beispiel hierzu ist die Anfrage, fortlaufend mit allen Informationen zur wirtschaftlichen Entwicklung eines bestimmten Unternehmens versorgt zu werden. Im Gegensatz zum Daten–Retrieval in herkömmlichen Datenbanken besteht eine wichtige Charakteristik des IR/IF in der Vagheit der Anfragen, zu denen eine Antwort a priori nicht eindeutig definiert ist, sowie in der Unschärfe des im Informationsraum verfügbaren Wissens, dessen Semantik in aller Regel nur sehr begrenzt repräsentiert werden kann (nach [Rijsbergen, 1979, Kapitel 1]). Um in dem obigen Beispiel zu bleiben, so ist es nur vage definiert bis zu welchem Alter die Kindheit andauert und es ist ebenso unklar, ob auch Texte, die lediglich einen einzigen Satz zu Mozart's Kindheit beinhalten, zurückgeliefert werden sollen.

Diese Arbeit behandelt Probleme des Information Retrieval im Internet, der Einfachheit halber wird auch oft von *Informationssuche* im Internet gesprochen. Im Vergleich zu traditionellen Datenbanken weist das Internet einige Spezifika auf, die an die IR–Forschung fundamental neue Herausforderungen stellen (siehe auch [Kosmynin, 1997]). Diese Spezifika sind:

- Umfang: Der Umfang der zugänglichen Daten ist um Größenordnungen größer als in herkömmlichen Datenbanken. Dieser Aspekt stellt enorme Anforderungen an die Skalierbarkeit von IR–Systemen.
- Verteilung: Die Daten sind über die ganze Welt (das ganze Internet) auf Millionen von Servern verteilt. Hierdurch ist der Zugriff auf einzelne Daten wesentlich zeitaufwendiger als in einer lokal zugreifbaren Datenbank und belastet zudem das zugrundeliegende Netzwerk.
- Dynamik: Der Inhalt der im Internet publizierten Informationen ist permanenten Änderungen unterworfen. Jeden Tag werden tausende von Dokumenten neu erstellt oder geändert. Diese Tatsache macht es für IR–Systeme sehr schwierig, ihr in der Vergangenheit erworbenes Wissen aktuell zu halten.
- Fehlende zentrale Steuerung: Es gibt keine zentrale Autorität, die das Publizieren neuer Inhalte steuert. Jedermann, der Zugang zu einem Informations–Server

hat, kann dort beliebige Dokumente veröffentlichen. Dadurch entsteht ein völlig unorganisierter Informationsraum, in dem keinerlei Garantie über die Qualität der verfügbaren Informationen gegeben werden kann.

- Heterogenität: Die verfügbaren Dokumente haben sehr heterogene Formate, von simplen Textdateien, über Programme bis hin zu Bildern und Multi–Media–Dokumenten. Zudem können täglich neue Dokumentformate hinzukommen.

Die Diskussion vorhandener Ansätze zur Informationssuche im nächsten Kapitel wird zeigen, dass bisher keiner dieser Ansätze mit allen genannten Spezifika in befriedigender Weise umgehen kann.

1.2 Überblick und Aufbau

Diese Arbeit stellt ein neues Verfahren zur Informationssuche im Internet vor. Anstatt wie bisher üblich ein einziges Suchwerkzeug für beliebige Informationsbedürfnisse anzubieten, wurden Suchwerkzeuge entwickelt, die jeweils auf ein Themengebiet spezialisiert sind, sogenannte *Domänen–Experten*. Durch diese Spezialisierung reduziert sich der für ein einzelnes Werkzeug relevante Informationsraum und wird somit besser beherrschbar. Zudem ermöglicht sie eine präzisere Formulierung und Beantwortung von Benutzeranfragen.

Jeder Domänen–Experte verfügt über eine spezielle Filterfunktion, mit deren Hilfe er für ein beliebiges Dokument entscheiden kann, ob es relevant im Sinne seiner Domänen (seines Themengebietes) ist. Um also ein im Internet verfügbares Dokument bezüglich seiner Relevanz zu überprüfen, muss es zunächst vom Domänen–Experten geladen und dann mit dem Filterverfahren untersucht werden. Stellt sich das Dokument als relevant heraus, wird es in der Wissensbasis des Experten abgespeichert, andernfalls verworfen. Dieses Vorgehen birgt die Ineffizienz, dass zahllose Dokumente über das Internet übertragen werden, nur um kurz danach verworfen zu werden. Um dies zu umgehen, wird in dieser Arbeit der Einsatz sogenannter *mobiler Programme*[↑] vorgeschlagen. Hierbei handelt es sich um Programme, die auf entfernte Rechner übertragen werden können und dann dort zur Ausführung kommen. Ein Domänen–Experte realisiert seine Filterfunktion als mobiles Programm, das er zu den über das Internet verteilten, zu untersuchenden Dokumenten sendet. Das Programm analysiert die Dokumente dann „vor Ort“ und überträgt letztlich nur die relevanten Dokumente zum Domänen–Experten. Auf diese Weise lässt sich in vielen Situation die Menge der über das Netzwerk zu übertragenden Daten und damit die

Belastung für das Netzwerk erheblich reduzieren. Um diese theoretisch mögliche Reduktion der Netzwerkbelastung auch in realen Szenarien zu erreichen, werden in dieser Arbeit Algorithmen entwickelt, die die Aussendung mobiler Filterprogramme optimal koordinieren. Die Evaluation dieser Verfahren wird zeigen, dass in vielen Fällen Reduktionen bis zu 90% erreicht werden (im Vergleich zur herkömmlichen Klient–Server–Lösung).

Diese Algorithmen benötigen jedoch Informationen über die Charakteristiken des zugrundeliegenden Netzwerkes. Insbesondere muss man wissen, wie weit zwei beliebige Rechner netzwerktechnisch voneinander entfernt liegen. Nur so kann bestimmt werden, ob ein mobiles Filterprogramm „nahe genug“ an ein zu untersuchendes Dokument heran gebracht werden kann, so dass sich mit der Programmaussendung die Belastung des Netzwerkes auch wirklich reduzieren lässt. Zur Bestimmung und Bereitstellung solcher Entfernungsinformationen wird in dieser Arbeit der neue Ansatz der sogenannten *Netzwerk–Distanzkarten* vorgestellt. Diese stellen bildlich gesprochen eine Karte vom Internet dar, derart dass beliebige Netzwerkentfernungen daraus abgeleitet werden können. Experimente werden zeigen, dass bereits mit geringem Aufwand sehr nützliche Entfernungsschätzungen möglich sind.

So sind aus der ursprünglichen Problemstellung der Informationssuche zwei weitere, prinzipiell ganz eigenständige Probleme erwachsen. Dementsprechend hat diese Arbeit drei Schwerpunkte: Erstens den Ansatz der Domänen–Experten. Hierbei wird im Wesentlichen die prinzipielle Architektur eines solchen Suchwerkzeugs diskutiert. Die Entwicklung effizienter Filterverfahren ist dagegen nicht Gegenstand dieser Arbeit. Zweitens werden Algorithmen und Protokolle zur Erstellung von Netzwerk–Distanzkarten entwickelt und bewertet. Und drittens wird das Problem der optimalen Aussendung mobiler Filterprogramme theoretisch analysiert und es werden Verfahren zu seiner Lösung entwickelt und bewertet. Als ein schöner Nebeneffekt wird sich herausstellen, dass die entwickelten Lösungen auch unabhängig voneinander bedeutsam sind und in verschiedenen Kontexten eingesetzt werden können.

Im Einzelnen ist die Arbeit wie folgt gegliedert: Kapitel 2 diskutiert bisherige Ansätze zur Informationssuche im Internet und motiviert daraus das Konzept der Domänen–Experten, welches daraufhin detailliert beschrieben wird. In Kapitel 3 wird der Ansatz der Netzwerk–Distanzkarten mit den dazugehörigen Algorithmen und Protokollen vorgestellt. Kapitel 4 beschreibt zwei Algorithmen, mit denen die Aussendung mobiler Filterprogramme optimal koordiniert werden kann. Die dann folgenden Kapitel sind der Evaluierung der vorgestellten Ansätze gewidmet. Kapitel 5 beschreibt, wie die erfolgreiche Distanzschätzung durch Netzwerk–Distanzkarten auf Basis realer Internet–Messungen

nachgewiesen wurde. In Kapitel 6 wird anhand der vorangegangenen Messungen simulativ bestimmt, wie groß der mögliche Effizienzgewinn ist, der sich mit dem koordinierten Einsatz mobiler Filterprogramme erzielen lässt. Schließlich wird in Kapitel 7 eine Fallstudie für einen konkreten Domänen–Experten vorgestellt, der auf das Gebiet der wissenschaftlichen Publikationen spezialisiert ist.

Kapitel 2

Domänen–Experten zur Informationssuche im Internet

In diesem Kapitel wird das Konzept der Domänen–Experten vorgestellt, welche die präzise, umfassende und effiziente Informationssuche innerhalb einzelner Themenbereiche realisieren. Zur Motivation dieses Konzeptes werden in Abschnitt 2.1 zunächst herkömmliche und verwandte Ansätze zur Informationssuche beschrieben und diskutiert. Aus der zusammenfassenden Diskussion dieser Ansätze wird schließlich der Bedarf für spezialisierte Suchwerkzeuge abgeleitet und es werden die wichtigsten der von ihnen zu lösenden Herausforderungen bestimmt. Die Abschnitte 2.2 und 2.3 stellen dann das Konzept der Domänen–Experten im Detail vor und zeigen, wie die zuvor identifizierten Herausforderungen an spezialisierte Suchwerkzeuge gelöst werden.

Systemmodell

Zur Vereinfachung der Diskussion benutzt dieses Kapitel das folgende Systemmodell: Der Informationsraum Internet besteht hiernach einfach aus einer Menge von *Daten–Servern*$^{\uparrow}$. Auf diesen können Autoren mit den entsprechenden Zugangsrechten beliebige Dokumente publizieren. Es werden nur solche Dokumente betrachtet, die global zugreifbar sind, z.B. über einen *Uniform Resource Locator*$^{\uparrow}$ (URL). Dieses Modell umfasst das Web, FTP und Usenet–News. Es abstrahiert allerdings von spezifischen Eigenschaften wie der Hypertext–Struktur von Web–Seiten oder der Kategorisierung von *News*–Artikeln durch die globale Hierarchie von *News*–Gruppen. Digitale Bibliotheken, deren Dokumente nicht direkt zugreifbar sind, werden in dieser Arbeit nicht betrachtet. Die Diskussion

beschränkt sich auf textbasiertes Information Retrieval, bei dem Dokumente und Anfragen durch einen Text, also eine Folge von Wörtern, beschrieben sind. Abweichungen von diesem Systemmodell werden explizit hervorgehoben.

Bewertung von IR–Systemen

Um verschiedene IR–Systeme miteinander vergleichen zu können, müssen diese bewertbar sein. Hierzu werden in dieser Arbeit die beiden Standardmaße zur Bewertung von IR–Systemen, *Effektivität* und *Effizienz*, verwendet [Rijsbergen, 1979, Kapitel 7]. Die Effektivität beschreibt das Aufwand–Nutzen–Verhältnis für den Anwender eines IR–Systems. Während der Aufwand sich durch Zeit, Nerven u.ä. spezifizieren lässt, wird der Nutzen üblicherweise durch die Maße *Präzision*$^{\uparrow}$ (engl. precision) und *Abruf*$^{\uparrow}$ (engl. recall) beschrieben.

Definition 2.1 (Präzision): Die Präzision ist definiert als Anteil der auf eine Anfrage gelieferten relevanten Dokumente gegenüber der Gesamtheit aller gelieferten Dokumente.

Definition 2.2 (Abruf): Der Abruf gibt den Anteil der auf eine Anfrage gelieferten relevanten Dokumente über allen (im jeweils betrachteten Dokumentenraum) existierenden relevanten Dokumenten wieder.

Die Präzision beschreibt also die Relevanz, der Abruf die Vollständigkeit der zurückgelieferten Ergebnismenge. Eine wichtige Voraussetzung für hohe Abrufwerte ist natürlich eine große *Abdeckungsrate*, also eine möglichst vollständige Erfassung des Dokumentenraumes durch das IR–System. Aussagen über den mit einem IR–System erreichten Abruf sind nur in Verbindung mit der gleichzeitig erreichten Präzision sinnvoll, da ein perfekter Abruf von 100% auch dadurch erreicht werden kann, dass zu jeder Anfrage alle verfügbaren Dokumente zurückgeliefert werden.

Die Effizienz eines IR–Systems kann über die benötigten Systemressourcen wie Speicherbedarf, Netzwerk– und CPU–Belastung beschrieben werden. Ein IR–System ist umso effizienter, je weniger Ressourcen es bei gleich bleibender Effektivität verbraucht.

Das ultimative Ziel von IR–Systemen ist die Erreichung hoher Präzisions- und Abrufwerte bei gleichzeitiger Minimierung der Antwortzeit und der verbrauchten Systemressourcen. In realen Systemen wird man allerdings immer einen Kompromiss zwischen diesen Zielen finden müssen [Dreilinger & Howe, 1997], da die einseitige Verbesserung in einer Kategorie fast zwangsläufig Verschlechterungen in anderen Kategorien mit sich bringt. Ein weiterer Aspekt, der von obigen Bewertungsmaßen nicht erfasst wird, ist der Aufwand, der zur Etablierung eines IR–System im Internet erforderlich ist. So ist die praktische

Umsetzung von Ansätzen, die große Änderungen an der bestehenden Infrastruktur des Internets erfordern, äußerst unwahrscheinlich.

2.1 Klassische Ansätze zur Informationssuche im Internet

Der folgende Überblick ist nach den möglichen Architekturen für IR–Systeme gegliedert, insbesondere nach den Verteilungsaspekten.

2.1.1 Zentralisierte Ansätze

Die ersten und bis heute erfolgreichsten Werkzeuge zur Informationssuche im Internet folgen einem zentralisiertem Ansatz. Zwei Hauptrichtungen können unterschieden werden.

Suchmaschinen$^{\uparrow}$ wie z.B. AltaVista[1] durchsuchen mit sogenannten *Robotern*$^{\uparrow}$ automatisch den gesamten Informationsraum nach Dokumenten und versuchen, so viele Dokumente als möglich zu erfassen und im sogenannten *Index*$^{\uparrow}$ abzulegen. Der Index speichert zu jedem Wort, wie oft und evtl. an welchen Positionen es in welchen Dokumenten vorkommt. Die genaue Auslegung dieses Index kann von Suchmaschine zu Suchmaschine variieren. Manche verzichten z.B. auf die Speicherung sogenannter Stoppwörter, das sind alltägliche Wörter wie Artikel u.ä. Andere reduzieren alle Wörter auf ihre Stammform. Benutzeranfragen, die durch Stichwörter beschrieben werden müssen, können anhand des Index beantwortet werden. Obwohl Suchmaschinen vollständig automatisch arbeiten, können sie nur einen begrenzten (und rückläufigen) Anteil des Internets erfassen. Konnte im Dezember 1997 die größte Suchmaschine noch 30% aller im Web verfügbaren Dokumente erfassen [Lawrence & Giles, 1998], so sank dieser Anteil auf 16% im Februar 1999 [Lawrence & Giles, 1999]. Ein weiteres Problem von Suchmaschinen ist die mangelhafte Aktualität ihres Index. Laut [Lawrence & Giles, 1999] sind 5% aller von einer durchschnittlichen Suchmaschine erfassten Dokumente nicht mehr vorhanden. Daneben weisen Dokumente selbst bei ihrer Erfassung im Schnitt bereits ein Alter von mehreren Monaten auf. Die erreichbare Präzision hängt vom Typ der Benutzeranfrage ab. Sehr spezielle Anfragen, die z.B. seltene Akronyme oder lange Phrasen beinhalten, können oft sehr

[1] AltaVista: http://www.altavista.com/

genau beantwortet werden. Hingegen führen allgemeinere Anfragen oft zu einer immensen Anzahl von zurückgelieferten Dokumenten, von denen ein Großteil für den Benutzer irrelevant ist.

Die Präzision von Suchmaschinen ließe sich um einiges steigern, wenn Autoren ihre Dokumente um Meta–Informationen, z.B. Angaben zum Thema oder Zielpublikum, anreichern würden. Damit aber Suchmaschinen hiervon profitieren können, muss ein weltweiter Standard zur Meta–Beschreibung geschaffen und benutzt (!) werden. Existierende Standards wie z.B. die *Dublin Core Metadata* Initiative[2] oder das *Resource Description Framework*[3] haben letzteres bisher nicht erreichen können, und es ist fragwürdig, ob es ihnen in Zukunft gelingen wird. Hinzu kommt, dass jeder derartige Standard zwangsläufig eine begrenzte semantische Ausdrucksmächtigkeit hat, die nicht alle möglichen oder wünschenswerten Meta–Beschreibungen abdeckt. Daher kann auf diesem Wege die Präzisionsproblematik allenfalls gelindert, jedoch nicht gelöst werden.

Kataloge wie z.B. Yahoo[4] ordnen ihre indizierten Dokumente in einer vordefinierten Themenhierarchie an. Benutzer können dann in beliebigen Teilbereichen dieser Hierarchie eine Suche starten. Neue Dokumente müssen vom Katalogbetreiber per Hand kategorisiert werden. Daher ist der Abdeckungsgrad von Katalogen zwangsläufig wesentlich geringer als bei Suchmaschinen. Dafür ist aufgrund der Themenkategorisierung die Präzision höher. Der Abruf hingegen kann je nach Anfrage stark variieren. Anfragen zu sehr populären Themengebieten, in deren Erfassung der Katalogbetreiber viel Arbeit investiert hat, können recht vollständig beantwortet werden. Hingegen kann es bei spezifischeren Anfragen auch vorkommen, dass überhaupt kein relevantes Dokument im Katalog vorhanden ist.

2.1.2 Hierarchisch verteilte Ansätze

Um die Skalierungsprobleme zentralisierter Suchwerkzeuge zu lösen, wurden einige verteilte Systeme vorgeschlagen, die auf einer Hierarchie von sogenannten *Maklern* (engl. broker) beruhen. Unter den homogenen Systemen, in denen alle Makler unter der Kontrolle eines einzigen Betreibers stehen, lassen sich zwei Hauptansätze ausmachen.

Makler mit Server–Beschreibungen sammeln ihr Wissen immer über den Inhalt kompletter Daten–Server. Dieses aggregierte Wissen können sie allerdings nicht mehr auf den

[2] *Dublin Core Metadata* Initiative: URL: http://purl.oclc.org/dc/
[3] *Resource Description Framework:* URL: http://www.w3.org/RDF
[4] Yahoo!: http://www.yahoo.com/

Inhalt einzelner Dokumente herunterbrechen. Die Systeme *WHOIS++* [Weider et al., 1996] und *Discover* [Sheldon et al., 1995] fassen den Inhalt jedes Daten–Servers zusammen und senden diese Zusammenfassung an den jeweils zugeordneten Makler, der wiederum eine Zusammenfassung aller erhaltenen Server–Beschreibungen erstellt und an den übergeordneten Makler weiterreicht. Dieser Prozess wird rekursiv bis zu einem Hauptmakler fortgesetzt. Anfragen müssen immer an den Hauptmakler gerichtet werden und werden von diesem dann an passend erscheinende Untermakler weitergereicht, bis sie von den Daten–Servern beantwortet werden können. Zwei prinzipielle Probleme lassen sich in derartigen Ansätzen ausmachen. Zum einen können irreführende Beschreibungen entstehen: Wenn z.B. ein Daten–Server Dokumente über Bildverarbeitungssoftware und über mobile Programme enthält, beschreibt er dieses durch die Stichwörter *Software*, *Bildverarbeitung*, *mobile* und *Programme*. Offensichtlich ist der Schluss, dass auf diesem Server auch Software für mobile Programme zu finden ist, falsch. Das zweite, gravierendere Problem besteht darin, dass Anfragen oft zu sehr vielen Daten–Servern weitergeleitet werden müssen. So ist der Hauptmakler bei sehr speziellen Anfragen nicht in der Lage die passenden Untermakler herauszufinden und muss die Anfrage daher an (fast) alle Untermakler weiterreichen. Für eher allgemeine Anfragen kann hingegen eine große Zahl von Untermaklern relevantes Wissen umfassen. Abermals muss die Anfrage also an sehr viele Makler weitergeleitet werden.

Das System *HyPursuit* [Weiss et al., 1996] löst das Problem irreführender Server–Beschreibungen, indem es ähnliche Dokumente in Gruppen zusammenfasst, so dass eine Server–Beschreibung aus einer Menge solcher Gruppen besteht. Diese werden auf höherer Ebene dann weiter zusammengefasst. Im oben genannten Beispiel würde der Inhalt des Daten–Servers also über zwei Gruppen, eine über Bildverarbeitungssoftware und die andere über mobile Programme, beschrieben. Das Problem der Weiterleitung von Anfragen wird hiermit allerdings nicht gelöst.

Makler mit spezialisiertem Index versuchen sich auf einen bestimmten Themenbereich zu konzentrieren. Hierdurch können Anfragen in einer Makler–Hierarchie wesentlich effizienter und zielgerichteter weitergeleitet werden. Das System *Harvest* [Bowman et al., 1995] benutzt sogenannte *Sammler* (engl. gatherer), um Index–Informationen von einzelnen Daten–Servern zu sammeln. Makler bauen dann einen spezialisierten Index auf, indem sie nur das Wissen von manuell ausgewählten Sammlern und anderen Maklern integrieren. Aufgrund der manuellen Auswahl können Makler neue, für ihren Themenbereich relevante Dokumente, nicht automatisch auffinden. Zudem ist der Aspekt, welche der von einem Sammler/Makler gelieferten Dokumente letztlich indiziert werden, nicht

spezifiziert.

Das Suchwerkzeug *Pharos* [Dolin et al., 1997] erlaubt die automatische Kategorisierung von Dokumenten und Dokumentmengen anhand von drei Taxonomien (über Thema, Ort und Zeit). Entsprechend der Taxonomien wird eine Makler–Hierarchie aufgebaut und die (automatisch kategorisierten) Dokumente werden dem jeweils speziellsten passenden Makler zugeordnet. Jeder Makler kennt am Ende eine Reihe von Dokumenten sowie die Kategorisierung der ihm untergeordneten Makler. Schließlich kann eine Anfrage, die von Hand kategorisiert wurde, ausgehend von einem Hauptmakler an den jeweils speziellsten zuständigen Makler weitergereicht und von diesem beantwortet werden. Mit diesem Ansatz wird erstmals ein voll automatisches und skalierbares Werkzeug zur Informationssuche erreicht, das zudem aufgrund der Kategorisierung präzisere Ergebnisse als klassische Suchmaschinen liefern kann. Nachteil des Ansatzes ist allerdings die statische Definition der Taxonomien, die den Aufbau von spezifischen Indizes erheblich einschränkt. Es können eben nur solche Indizes erstellt werden, die sich innerhalb der Taxonomie ausdrücken lassen.

Meta–Suchmaschinen sind spezielle Makler, die den Zugang zu mehreren heterogenen Suchwerkzeugen zusammenfassen sollen. Typischerweise werden Meta–Suchmaschinen und die von ihnen benutzten Suchwerkzeuge von verschiedenen Parteien betrieben. Angesichts der Heterogenität des Internets ist ein solcher Ansatz wesentlich realistischer als die Annahme einer globalen und homogenen Makler–Hierarchie.

Eine der ersten Meta–Suchmaschinen war *MetaCrawler* [Selberg & Etzioni, 1995]. Benutzeranfragen werden von dieser parallel an mehrere Suchwerkzeuge weitergeleitet. Die Dokumente der zurückgelieferten Referenzen werden dann von der Meta–Suchmaschine geladen und genauer analysiert. Dadurch wird die Aktualität der Ergebnisse sowie ein einheitliches Bewertungsschema erreicht. Der Preis hierfür ist allerdings eine sehr große Belastung des Netzwerkes, da für jede einzelne Anfrage sehr viele Unteranfragen und Dokumente über das Netzwerk transferiert werden müssen.

Ein netzwerktechnisch effizienterer Ansatz wird von *SavySearch* verfolgt [Dreilinger & Howe, 1997]. Durch Analyse der letztlich vom Benutzer verfolgten Dokumentreferenzen (der *angeklickten* URLs) wird ein Metaindex aufgebaut, der Suchanfragen mit den Suchwerkzeugen korreliert, die hierfür relevante Ergebnisse geliefert haben. Dieser Metaindex wird dann benutzt, um zukünftige Anfragen wesentlich zielgerichteter nur an die vielversprechenden Suchwerkzeuge weiterzuleiten.

Mit Meta–Suchmaschinen konnte die Abdeckung des Internets wesentlich verbessert werden. So kann allein die Kombination der Ergebnisse der 6 größten Suchmaschinen die

Abdeckung des Webs von 16% auf 42% steigern [Lawrence & Giles, 1999]. Die mit Meta–Suchmaschinen erreichbare Präzision hängt von zwei Faktoren ab: Zum einen von dem internen Auswahlmechanismus, der festlegt, an welche Suchwerkzeuge eine Anfrage weitergeleitet wird. Zum anderen aber auch, und damit außerhalb der Kontrolle der Meta–Suchmaschine, von der Präzision der zugrundeliegenden Suchwerkzeuge. Somit wird das Problem ungenügender Präzision nicht von Meta–Suchmaschinen gelöst.

2.1.3 Dezentrale Ansätze

In einigen Projekten wurden komplett dezentrale Architekturen zur Informationssuche entwickelt. Im System *Ingrid* [Francis et al., 1995] erfolgt eine dezentrale Verteilung auf Basis einzelner Dokumente. Dokumente werden durch 15 Schlüsselwörter beschrieben sowie durch Verweise auf andere Dokumente, mit denen sie möglichst viele Schlüsselwörter gemeinsam haben. Dabei wird sichergestellt, dass zwischen Dokumenten mit gemeinsamen Schlüsselwörtern immer eine Verweiskette existiert. Benutzeranfragen werden dann ausgehend von dem ersten gefundenen relevanten Dokument entlang dieser Verweisketten weitergeleitet. Abgesehen von der sehr ausdrucksschwachen Dokumentenbeschreibung durch lediglich 15 Wörter scheint dieser Ansatz im Sinne des erzeugten Netzwerkverkehres überhaupt nicht zu skalieren, da eine einzelne Anfrage u.U. an eine sehr große Zahl von Rechnern geleitet werden muss.

Eine Verteilung auf Basis von Dokumentengruppen, die zu einzelnen Themen erstellt werden, wird von [Kosmynin, 1997] vorgeschlagen. Ausgehend von einer vordefinierten Themenliste erweitert er die Funktionalität von Web–Proxies zu sogenannten Proxy–Suchmaschinen. Eine solche Suchmaschine beobachtet das Verhalten ihrer Benutzer im Web und leitet daraus Präferenzen ab. Zu den bevorzugten Themengebieten baut sie eine lokale Dokument–Wissensbasis auf. Zudem lernt sie von anderen Proxy–Suchmaschinen deren Spezialisierungsgebiete. Auf diese Weise können Benutzeranfragen häufig anhand der lokalen Wissensbasis beantwortet werden oder an eine Proxy–Suchmaschine weitergeleitet werden, die auf das entsprechende Themengebiet spezialisiert ist. Schlägt auch letzteres fehl, so wird auf eine traditionelle Suchmaschine zurückgegriffen. Nimmt man eine erfolgreiche Themenerkennung von Dokumenten und Anfragen an, so scheint dieser Ansatz eine effektive Verteilung von Indexdaten und Anfragen und somit eine effiziente, skalierbare Informationssuche zu ermöglichen. Nachteil des Verfahrens ist allerdings, dass jeder Web–Proxy zu einer Proxy–Suchmaschine ausgebaut werden muss, was eines globalen Standards bedarf. Die mit diesem Ansatz erzielbare Präzision ist genauso hoch wie bei herkömmlichen Suchmaschinen.

2.1.4 Spezialisierte Ansätze

Ein offensichtlicher Ansatz zur Verbesserung der Präzision herkömmlicher Suchwerkzeuge besteht in der Spezialisierung von Werkzeugen auf einzelne Themengebiete, im Folgenden auch *Domänen* genannt. Solche Werkzeuge können dann zwar nur Anfragen zu ihrer Domäne sinnvoll bearbeiten und beantworten — dies aber in der Regel wesentlich präziser. Als Nebeneffekt verkleinert sich damit auch der Raum der für das Suchwerkzeug und seinen Index relevanten Dokumente und wird somit besser handhabbar. Wie ein Benutzer bzw. eine Anfrage zu einem adäquaten, spezialisierten Suchwerkzeug findet, ist nicht Gegenstand dieser Arbeit. Der interessierte Leser sei auf die Arbeiten von [Manber & Bigot, 1997], [Dolin et al., 1997] und [Dreilinger & Howe, 1997] sowie auf die kurze Diskussion dieser Problematik in Abschnitt 2.4 verwiesen.

Das Internet verfügt mittlerweile über tausende von spezialisierten Suchdiensten. Die meisten basieren auf dem in Abschnitt 2.1.1 vorgestellten Katalogansatz, in dem Dokumente von Hand aufgenommen werden müssen. Es gibt aber auch einige Suchmaschinen, die Dokumente ihrer Domäne automatisch auffinden.

Der prominenteste Vertreter dieser Gattung von Suchwerkzeugen ist das System *Ahoy!*, welches die Suche nach persönlichen Web–Seiten (*Homepages*) ermöglicht [Shakes et al., 1997]. Anfragen werden von *Ahoy!* zunächst an eine klassische Suchmaschine weitergeleitet und deren Ergebnisse anhand eines Dienstes für E-Mail–Adressen sowie einer Datenbank über Institutionen nachgefiltert. In einer internen Datenbank werden erfolgreich gefundene *Homepages* gespeichert. Zusätzlich verfügt *Ahoy!* über eine Komponente, die typische URL–Muster von Web–Seiten von Institutionen und Personen lernt. Scheitern die einfachen Suchmechanismen, so wird diese Komponente zum Raten der URL einer gesuchten *Homepage* genutzt. *Ahoy!* hat sich als effektives und effizientes Werkzeug für seine Domäne erwiesen. Es ist jedoch kaum möglich, die verwendeten Konzepte auch auf andere Bereiche zu übertragen, da in diesen keine externen spezialisierten Dienste (wie z.B. ein Dienst für E-Mail–Adressen) vorausgesetzt werden können und das Raten von URL–Mustern auch weniger erfolgversprechend erscheint als im (eher stereotypen) Bereich persönlicher Web–Seiten.

[Seacord et al., 1998] haben ein recht simples Werkzeug zur Suche nach Java–Applets entwickelt, das auf einer herkömmlichen Suchmaschine aufbaut und ein spezielles, dort verfügbares Meta–Attribut nutzt. Der Fokus dieser Forschungsarbeit liegt in der domänenspezifischen Repräsentation der Ergebnisse.

Der *Nordic Web Index* [Ardö & Lundberg, 1998] ist ein Suchdienst, der sich auf die Region der skandinavischen Länder beschränkt. Er benutzt die *toplevel domain* des *Domain Name System*↑ (DNS) zur Filterung. Dokumente aus nicht ländergebundenen *toplevel domains* werden nicht integriert.

CiteSeer ist eine Suchmaschine für wissenschaftliche Artikel [Bollacker et al., 1998]. Benutzeranfragen werden von ihr (erweitert um ein paar für das Themengebiet typische Stichwörter) zunächst zu einer konventionellen Suchmaschine gesandt. Die von den resultierenden Verweisen referenzierten Dokumente werden dann geladen, genauer auf ihre Relevanz untersucht und ggf. in einer internen Datenbasis abgelegt. Der Fokus von *CiteSeer* liegt in der Bereitstellung von vielfältigen Suchmöglichkeiten in dieser Datenbasis, z.B. nach referenzierenden, referenzierten oder ähnlichen Artikeln.

Der *ShopBot* von [Doorenbos et al., 1997] analysiert automatisch die Seiten von gegebenen Produktanbietern im Web und lernt, wie er an diese Anfragen stellen und aus den Ergebnisseiten die relevanten Produktinformationen extrahiern kann. Als Domänenmodell muss ein ShopBot lediglich einige typische Produkte sowie die möglichen Produktattribute (und deren Synonyme) kennen. Der Ansatz nutzt eine Reihe von Uniformitäten, die typisch für Produktanbieter im Web sind.

Domänenspezifische Suchwerkzeuge haben sich als sehr nützlich erwiesen, da sie wesentlich präzisere Ergebnisse liefern können als dies mit universellen Ansätzen möglich ist. Bei automatisierter Entdeckung von Dokumenten können sie prinzipiell eine vollständige Abdeckung des Informationsraumes und somit hohe Abrufwerte erreichen. Es bleibt allerdings die Frage, inwieweit die einzelnen Konzepte zur Dokumentenentdeckung für beliebige Domänen anwendbar sind. Schließlich ist sicherlich auch der Aufwand zur Erstellung einer Vielzahl solcher Werkzeuge ein kritischer Faktor.

Frameworks[5] für spezialisierte Suchmaschinen sollen die Konstruktion neuer, spezialisierter Suchmaschinen möglichst weit vereinheitlichen und somit vereinfachen.

Von den oben vorgestellten Ansätzen ist lediglich das Konzept von *CiteSeer* auf beliebige Domänen übertragbar. Allerdings erzeugt dieser Ansatz eine enorme Netzwerkbelastung, da sehr viele und insbesondere auch viele irrelevante Dokumente über das Netzwerk transferiert werden.

[5] Der Begriff *Framework* bezeichnet ein unvollständiges Programm, welches als eine Art Gerüst für eine ganze Familie von Programmen dienen soll. Einzelne Programme aus dieser Familie lassen sich mit Hilfe des Frameworks und einiger weniger Codepartien, die es individuell zu programmieren gilt, relativ einfach erstellen. Zum Begriff des *Frameworks* gibt es bisher keine anerkannte deutsche Übersetzung.

Das Framework *Sphinx* ermöglicht die Konstruktion von Suchmaschinen mit spezialisierten Web–Robotern [Miller & Bharat, 1998]. Um deren Indizierungsverhalten festzulegen, müssen lediglich die zwei Funktionen $visit(Page\ p)$ und $shouldVisit(Link\ l)$ definiert werden. $visit$ beschreibt die Aktionen, die für eine gegebene Seite ausgeführt werden sollen, $shouldVisit$ entscheidet, ob ein gegebener *Hyperlink* weiter verfolgt werden soll. Das System sieht auch die dynamische Verlagerung von Robotern auf entfernte Server vor, allerdings werden keine Verfahren angegeben, die festlegen, wann und wohin eine solche Verlagerung stattzufinden hat. Es erscheint zumindest fragwürdig, inwieweit die Funktion $shouldVisit$ den Bereich des spezialisierten Roboters angemessen begrenzen kann, da ihr Argument lediglich den Text und die URL eines *Hyperlinks* umfasst.

[Chakrabarti et al., 1999] stellen einen spezialisierten Web–Roboter vor, der bevorzugt *Hyperlinks* von als relevant erkannten Seiten oder von Seiten, von denen aus relevante Seiten gefunden wurden, verfolgt. Eine initiale Liste von einigen relevanten Web–Dokumenten muss gegeben sein. In Experimenten wird für einige Domänen nachgewiesen, dass 40% aller durch den Roboter untersuchten Dokumente auch relevant sind. Das Konvergenzverhalten des Ansatzes wird hingegen nicht analysiert. Somit bleibt unklar, wie viel Suchzeit erforderlich ist, um hohe Abrufwerte zu erreichen.

Ansätze mit spezialisierten Robotern funktionieren schlecht in Domänen mit einer geringen Anzahl von Querverweisen (*Hyperlinks*) und überhaupt nicht in Informationsräumen, die gar keine Querverweise enthalten, wie zum Beispiel FTP.

2.1.5 Diskussion

Um den vorangehenden Überblick zusammenzufassen, lassen sich folgende Charakteristiken feststellen. Herkömmliche Suchmaschinen können sehr hilfreich sein, leiden aber oft an einer mangelhaften Präzision. Zudem wird der Abruf mit dem weiter anwachsenden Internet immer schlechter. Makler mit Server–Beschreibungen skalieren in dieser Hinsicht wesentlich besser, verursachen aber durch die ineffiziente Weiterleitung von Anfragen einen immensen Netzverkehr. Dieses Problem wird von Maklern mit spezialisierten Indizes gelöst, allerdings bleibt unklar, wie eine solche Spezialisierung automatisch und flexibel erreicht werden kann. Außerdem erfordern Makler–basierte Ansätze den Aufbau einer großen und neuen Infrastruktur. Meta–Suchmaschinen verfolgen im Gegensatz dazu einen realistischeren Ansatz, indem sie auf vorhandenen Suchwerkzeugen aufsetzen. Mit dezentralen IR–Architekturen wurde kein signifikanter Fortschritt erzielt.

Spezialisierte Suchmaschinen versprechen (und ermöglichen zum Teil) erstmals auf skalierbare Art und Weise hohe Präzision bei gleichzeitig hohem Abruf. Allerdings sind die verwendeten Konzepte oft nicht auf mehrere oder sogar beliebige Domänen anwendbar, womit der Aufwand zur Entwicklung einer Vielzahl solcher spezialisierter Suchmaschinen sehr groß wird. Daher braucht man Frameworks, die diese Entwicklung vereinfachen. Obwohl dieser Bedarf schon früh erkannt wurde [Bowman et al., 1994], gibt es bisher kaum Ansätze hierzu.

Drei wesentliche Probleme müssen von Frameworks für spezialisierte Suchmaschinen gelöst werden:

1. Das Relevanzproblem, d.h. die Frage, wie man erkennt, ob ein gegebenes Dokument für eine bestimmte Domäne relevant ist. Die existierenden Ansätze in diesem Bereich greifen meist auf Verfahren aus der sogenannten Künstlichen Intelligenz zurück.
2. Das Entdeckungsproblem, d.h. die Frage, wie man die relevanten Dokumente innerhalb des riesigen Internets auf effiziente Art und Weise findet. Existierenden Frameworks gelingt die Synthese von hoher Abdeckung und hoher Effizienz bisher nicht.
3. Das Aufwandsproblem, d.h. die Frage, wie man den menschlichen Aufwand zur Entwicklung neuer, spezialisierter Suchmaschinen aus einem Framework heraus minimiert. Dieser Aspekt wird von den aktuellen Ansätzen kaum beachtet.

Das Relevanzproblem wird im Rahmen dieser Arbeit nicht weiter behandelt. Der interessierte Leser sei auf die umfangreichen Arbeiten im Bereich der Wissensaquisition (siehe z.B. [KAW, 1999]) und der Themenkategorisierung verwiesen. Einen Überblick über verschiedene Verfahren mitsamt Effektivitätsanalyse geben [Yang & Liu, 1999]. [Dolin et al., 1998] stellen Ergebnisse zur Klassifikation nach der *Library of Congress Classification* vor

Nachfolgend wird nun das Konzept der Domänen–Experten vorgestellt, wobei insbesondere darauf eingegangen wird, wie diese das Entdeckungs- und das Aufwandsproblem lösen.

2.2 Domänen–Experten — Funktionsweise und Architektur

Die Beschreibung des Ansatzes der Domänen–Experten, welcher erstmals in [Theilmann & Rothermel, 1998] vorgestellt wurde, erfolgt in drei Schritten. Zunächst wird deren

Architektur erläutert und es wird erklärt, wie Domänen–Experten die für sie relevanten Dokumente auffinden und Benutzeranfragen bearbeiten. Im Abschnitt 2.3 wird dann gezeigt, wie ein Framework für solche Experten den Aufwand zur Erstellung eines neuen Experten minimieren kann. Der letzte Abschnitt dieses Kapitels schließlich diskutiert, wie Domänen–Experten in eine allgemeine Infrastruktur von IR–Systemen integriert werden können.

Kernkomponente eines Domänen–Experten ist seine domänenspezifische Filterfunktion, mit der er entscheiden kann, ob ein beliebiges Dokument zu seiner Domäne gehört oder nicht. Daneben kann er (aber muss nicht) noch mit zusätzlichem Wissen ausgestattet sein, das die Bestimmung von domänenspezifischen Informationen in einem Dokument ermöglicht.

Mit Hilfe dieses Wissens kann ein Domänen–Experte vollständig automatisch eine Wissensbasis über Dokumente seiner Domäne aufbauen. Diese Wissensbasis umfasst einen Index, der mit domänenspezifischen Attributen versehen sein kann, sowie weitere Meta–Beschreibungen, die aus dem jeweiligen Dokument oder seiner Umgebung bestimmt wurden. Meta–Beschreibungen können z.B. die Sprache eines Dokumentes, sein Erstellungsdatum oder die Namen seiner Autoren umfassen.

2.2.1 Bearbeitung von Benutzeranfragen

Zu Beginn seines Lebenszyklus kennt ein Domänen–Experte kein einziges für seine Domäne relevantes Dokument. Trotzdem ist er schon in der Lage, Benutzeranfragen zu bearbeiten. Eine Anfrage wird dabei zunächst an eine klassische Suchmaschine weitergeleitet. Die zurückgelieferten Dokumentenverweise (URLs) werden dann mit Hilfe der domänenspezifischen Filterfunktion genauer untersucht, relevante Dokumente werden in die Wissensbasis aufgenommen und schließlich wird die Benutzeranfrage beantwortet.

Abbildung 2.1 auf Seite 18 zeigt die Architektur eines Domänen–Experten, sowie die einzelnen Schritte zur Beantwortung von Benutzeranfragen (Pfeile markieren den Kontrollfluss). Diese sind ihrer Reihenfolge entsprechend mit Nummern versehen. Klammern bezeichnen optionale Schritte. Die Bearbeitung einer Benutzeranfrage erfolgt in folgenden Schritten: Eine mit Hilfe der *Benutzungsschnittstelle* komponierte domänenspezifische Anfrage wird zunächst zum *Anfragenprozessor* gesandt (1). Dieser beauftragt den *Wissens–Sammler* (2), initiales Wissen von externen Wissensbasen zu erlangen. Der Sammler schickt dazu eine Anfrage an eine oder mehrere klassische Suchmaschinen (oder

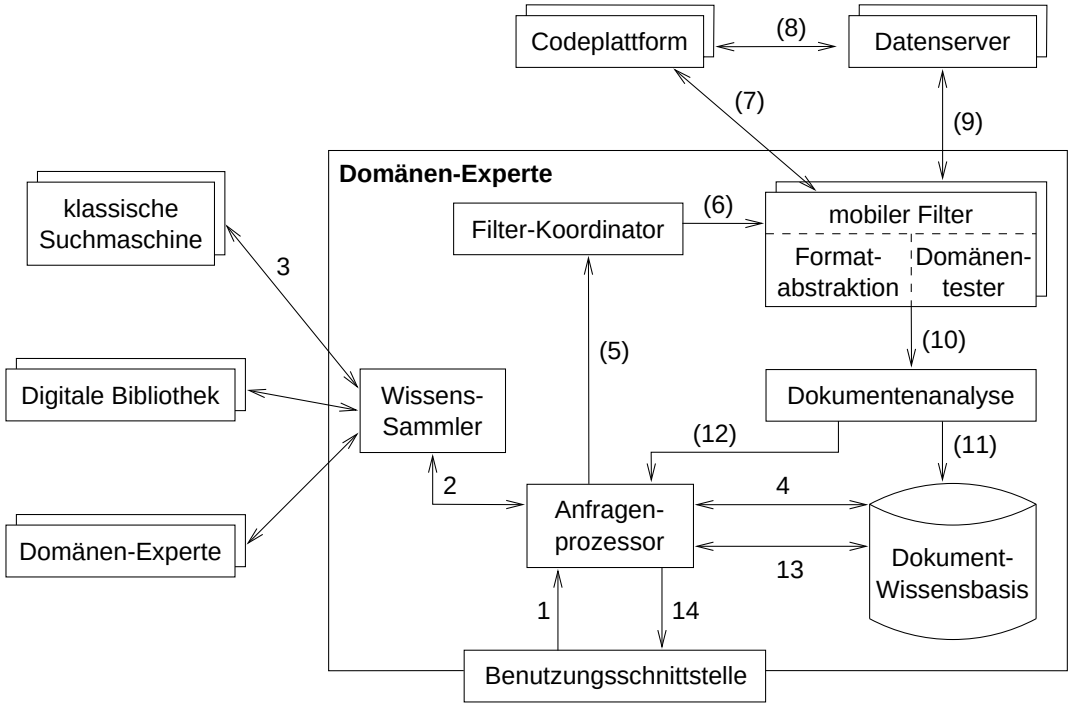


Abbildung 2.1: Architektur eines Domänen–Experten; Anfragenbearbeitung

Meta–Suchmaschinen). Dazu muss er allerdings zunächst die domänenspezifische Anfrage, die z.B. einige domänenspezifische Attribute oder auch spezielle Meta–Beschreibungen umfassen kann, in ein allgemeines Anfrageformat umwandeln, das von den externen Suchmaschinen verstanden wird. Die zurückgelieferten URLs werden schließlich zur weiteren Bearbeitung dem Anfragenprozessor übergeben. Man beachte, dass aufgrund der Anfragenumwandlung nur eine Teilmenge der referenzierten Dokumente der Spezifikation der ursprünglichen Anfrage genügt. In der Regel wird sich nur ein kleiner Teil dieser Dokumente als relevant für die Domäne des Experten erweisen und ein nochmals geringerer Teil als relevant im Sinne der Benutzeranfrage. Der Anfragenprozessor überprüft nun, ob die gelieferten Dokumentenverweise in der *Dokument–Wissensbasis* schon bekannt sind (4). Verweise auf unbekannte Dokumente und solche deren Zeitstempel sich geändert haben werden an den *Filter–Koordinator* übergeben (5). Dieser ist nun dafür verantwortlich, für jedes referenzierte Dokument zu entscheiden, ob es zur Domäne des Experten gehört oder nicht. Der herkömmliche, allerdings sehr ineffiziente Ansatz dazu wäre, alle Dokumente über das Netzwerk zu laden und sie mit der domänenspezifischen Filterfunktion zu untersuchen. Allerdings werden bei einem solchen Vorgehen zahlreiche Dokumente über das Netzwerk transferiert, nur um nach der Filterung sofort verworfen zu werden. Zur Behebung dieser Ineffizienz benutzt der Filter–Koordinator daher *mobile Filter*, de-

ren genauer Einsatz und Koordination (Schritte 7–9) im nächsten Abschnitt beschrieben sind. Die Teilkomponenten *Formatabstraktion* und *Domänentester* werden in Abschnitt 2.3 erläutert. Jeder Filter liefert als Ergebnis die Dokumente, die zur Domäne des Experten gehören (10). Diese Dokumente werden von der *Dokumentenanalyse* genauer untersucht, d.h. es werden Stichwörter, Attribute und weitere Meta–Beschreibungen bestimmt und in die Dokument–Wissensbasis integriert (11). Schließlich wird der Anfragenprozessor benachrichtigt, dass die Untersuchung der referenzierten Dokumente abgeschlossen ist (12). Dieser berechnet dann anhand der Wissensbasis (13) die für die Benutzeranfrage relevanten Dokumente und liefert die endgültige Antwort an den Benutzer (14).

Man beachte, dass die Filterung und Analyse von Dokumenten (Schritte 5–12) nicht notwendig erfolgt. Zu Beginn des Lebenszyklus des Domänen–Experten ist es sehr wahrscheinlich, dass zur Beantwortung einer Benutzeranfrage neue, entfernte Dokumente untersucht werden müssen. Wenn die Dokument–Wissensbasis in späteren Stadien angewachsen ist, können mehr und mehr Anfragen direkt beantwortet werden, ohne entfernte Dokumente zu analysieren. Benutzer, für die eine geringe Antwortzeit wichtiger ist als ein vollständiges und hochaktuelles Ergebnis, können natürlich auch sofort eine Antwort erhalten. In diesem Fall erfolgt die Analyse neuer, entfernter Dokumente (Schritte 2–12) unabhängig von der Anfrage, also nur zum Ausbau der internen Wissensbasis.

Neben dem durch Benutzeranfragen gesteuerten Wissenserwerb kann ein Domänen–Experte seine Wissensbasis auch selbständig zu erweitern suchen. Die hierzu notwendigen Mechanismen werden in Abschnitt 2.2.3 beschrieben.

2.2.2 Verteilte Dokumentenfilterung

Das Szenario der verteilten Dokumentenfilterung (die Schritte 7–9 aus der in Abbildung 2.1 dargestellten Anfragenbearbeitung) lässt sich wie folgt charakterisieren: Der Domänen–Experte kennt eine Reihe von Referenzen (URLs) auf Dokumente, die er mit seiner Filterfunktion untersuchen will. Letztendlich verwertet er aber nur die relevanten Dokumente, die den Filter passieren. Daher wäre es vorteilhaft, wenn die Filterung auf den jeweiligen Daten–Servern stattfinden könnte und nur noch die relevanten Dokumente über das Netzwerk übertragen werden müssten. Da ein Domänen–Experten seine spezifische Filterfunktion nicht auf allen Daten–Servern im Internet installieren kann, bietet sich als Ausweg der Einsatz von mobilen Programmen an, die auf entfernte Rechner übertragen werden können und dann dort zur Ausführung kommen. Eine genauere Definition

von mobilen Programmen erfolgt in Kapitel 4.1. Solche mobilen Programme können allerdings nur auf Rechner transferiert werden, die eine spezielle Ablaufumgebung dafür bieten. Ein solcher Rechner sei im Folgenden *Codeplattform*$^{\uparrow}$ genannt. Abermals kann man realistischerweise nicht davon ausgehen, dass jeder Daten–Server im Internet gleichzeitig eine Plattform für mobile Programme bietet. Deshalb wird an dieser Stelle nur die Existenz einer gewissen Zahl von Codeplattformen vorausgesetzt, und es wird versucht, die Filterfunktion möglichst „nahe“ an die jeweiligen Daten–Server heranzubringen.

Der Filter–Koordinator bekommt die Aufgabe zu entscheiden, zu welchen Codeplattformen ein mobiler Filter in einem konkreten Szenario gesandt werden soll und welche Dokumente ein jeder Filter zu untersuchen hat. Abbildung 2.2 zeigt ein mögliches Aussendungsszenario (mehrere Dokumente auf einem Daten–Server sind hierin zusammengefasst, da sie gleich behandelt werden können):

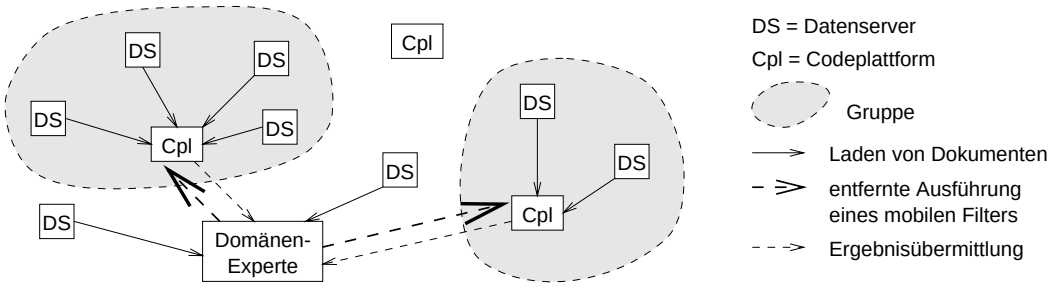


Abbildung 2.2: Gruppenweise Untersuchung von Dokumenten durch mobile Filter

In diesem Beispiel werden zwei mobile Filter ausgesandt, die jeweils Gruppen von zwei bzw. vier Daten–Servern untersuchen (entspricht Schritt 7 aus Abbildung 2.1). Es kann auch vorkommen, dass es sich am effizientesten erweist, einen Daten–Server direkt von der Plattform des Domänen–Experten aus zu untersuchen. Jeder Filter lädt die Dokumente aus seinem Zuständigkeitsbereich, hier als Gruppe von Daten–Servern markiert (Schritt 8). Schließlich senden die Filter die als relevant eingestuften Dokumente als Ergebnis ihrer Analyse zum Domänen–Experten und beenden ihre Arbeit (Schritt 9).

Die detaillierten Verfahren zur Aussendung mobiler Filter werden in Kapitel 4 vorgestellt und diskutiert. Da die mobilen Programme möglichst „nahe“ an bestimmte Daten–Server gebracht werden sollen, brauchen sie Wissen über Entfernungen im Netzwerk. Die Bestimmung solcher Entfernungen wird in Kapitel 3 behandelt und in Kapitel 5 evaluiert. Eine Evaluation der durch die Aussendung mobiler Programme erreichbaren Reduktion an Kommunikationkosten erfolgt in Kapitel 6. Kapitel 7 schließlich zeigt Kosteneinsparungen, die in einer konkreten Fallstudie zu einem Domänen–Experten erreicht wurden.

2.2.3 Wissensaktualisierung und -ausbau

Anstatt sich auf eine externe, klassische Suchmaschine zu verlassen, kann ein Domänen–Experte auch selbständig überprüfen, ob die von ihm registrierten Dokumente geändert oder gelöscht wurden. Hierbei erweist sich die Einschränkung auf eine Domäne als sehr hilfreich, durch die die Zahl der von einem Experten registrierten Dokumente um Größenordnungen geringer ist als bei einer klassischen Suchmaschine. Dadurch kann eine Aktualitätsüberprüfung wesentlich öfter als in einer klassischen Suchmaschine erfolgen, und es wird somit eine hoch aktuelle Wissensbasis erreicht.

Zusätzlich zur Wissenssammlung, die durch Benutzeranfragen initiiert wird, kann ein Domänen–Experte auch proaktiv nach relevanten Dokumenten suchen. Damit wird der Aufbau der Dokument–Wissensbasis beschleunigt und die Antwortzeit für Benutzeranfragen kann bereits in frühen Stadien des Experten reduziert werden. Zudem ist es auch möglich dabei Dokumente zu entdecken, die von der durch den Wissens–Sammler benutzten Suchmaschine nicht erfasst wurden, womit der Abrufwert des Experten steigt. Proaktive Suche kann auf folgende Arten erfolgen:

- Die Umgebung eines relevanten Dokumentes, d.h. Dokumente auf demselben Server mit ähnlichem Zugriffspfad (URLs die sich nur im hinteren Teil unterscheiden), kann nach weiteren relevanten Dokumenten durchsucht werden. Eine solche Heuristik kann z.B. in der Domäne „wissenschaftliche Artikel“ recht erfolgreich sein, da Dokumente, die von einer Institution oder auf einer Konferenz veröffentlicht wurden, in der Regel „nahe“ beieinander liegen.
- *Hyperlinks* im Web können von relevanten Seiten aus im Stile eines Roboters weiterverfolgt werden. [Chakrabarti et al., 1999] haben den hohen Nutzen dieser Heuristik für einige Domänen gezeigt.
- Relevante Dokumente können nach interessanten Hinweisen untersucht werden, mit Hilfe derer neue Dokumente gesucht werden. In der Domäne „wissenschaftliche Artikel“ können z.B. Namen von Autoren oder Konferenzen, wie sie im Literaturverzeichnis auftauchen, genutzt werden, um neue Suchanfragen für die Wissens–Sammlung zu formulieren. In anderen Domänen mag es auch nützlich sein, häufig vorkommende und somit typische Schlüsselwörter aus einem Dokument zu extrahieren und nach diesen in anderen Dokumenten zu suchen.
- Schließlich kann ein Domänen–Experte natürlich auch noch auf andere, externe Wissensquellen wie z.B. Digitale Bibliotheken oder andere Domänen–Experten zugreifen

und von diesen neues Wissen erwerben. Dieses Vorgehen wurde bereits von einigen spezialisierten Suchmaschinen eingesetzt (siehe z.B. [Shakes et al., 1997]).

Ein wichtiger Unterschied zwischen diesen vier Strategien besteht darin, dass die ersten beiden für beliebige Domänen angewandt werden können, ohne dass irgendwelche domänenspezifischen Festlegungen getroffen werden müssen. Im Gegensatz dazu sind die letzten beiden Strategien sehr domänenspezifisch und müssen individuell für jede Domäne implementiert werden.

2.3 Ein konfigurierbares Framework

Ein zentrales Problem von spezialisierten Suchmaschinen ist der Aufwand zur Erstellung derselben. Da im Gegensatz zu universellen Ansätzen dieser Aufwand für jede einzelne Suchmaschine (jeden Themenbereich) aufzubringen ist, kommt der Minimierung dieses Aufwandes eine große Bedeutung für die praktische Umsetzbarkeit des Konzeptes zu (vgl. mit Abschnitt 2.1.5).

Zur einfachen Erstellung von Domänen–Experten wurde darum in [Arnold, 1999] ein Framework entwickelt, das bereits über die meisten Funktionen eines Domänen–Experten verfügt und in dem ein Großteil der domänenspezifischen Eigenschaften mit Hilfe einer Konfigurationsdatei sehr einfach definiert werden kann.

Dieser Abschnitt beschreibt nun die einzelnen Komponenten des Frameworks für Domänen–Experten und zeigt auf, welche Teile domänenspezifisch konfiguriert werden können und welche Teile zur Erzeugung eines vollständigen Experten noch individuell implementiert werden müssen.

Generisches Dokumentenmodell

Kern eines jeden Domänen–Experten ist seine Dokument–Wissensbasis, in der er Wissen über alle ihm bekannten und für relevant befundenen Dokumente ablegt. Das generische Dokumentenmodell des Frameworks sieht hierzu zwei Datenstrukturen vor: Zum einen umfasst es einen Index, der zu jedem in einem beliebigen Dokument vorkommenden Wort speichert, in welchen Dokumenten und an welchen Positionen dieses Wort jeweils vorkommt. Zu jedem Vorkommen eines Wortes können zusätzlich beliebig viele domänenspezifischen Attribute angegeben werden. Daneben erlaubt das Dokumentenmodell die Speicherung (fast beliebiger) domänenspezifischer Meta–Informationen zu einem Dokument im sogenannten *Dokumentdatensatz*.

Die Definition der domänenspezifischen Anteile des Dokumentenmodells erfolgt in der Konfigurationsdatei. Abbildung 2.3 zeigt ein Beispiel für die Definition eines konkreten Dokumentenmodells für die Domäne „wissenschaftliche Artikel“. Im ersten Teil dieser

```
BeginIndexWordAttributes
   a_title, a_authors, a_abstract, a_references
EndIndexWordAttributes

BeginDocumentDataRecord
   d_language : String;
   d_size : Integer;
   d_timestamp : Date;
   d_url : Url;
   d_title : String;
   d_authors : String;
EndDocumentDataRecord
```

Abbildung 2.3: Beispieldefinition eines Dokumentenmodells

Definition werden die domänenspezifischen Attribute des Indexes durch einfache Bezeichner festgelegt. Im Beispiel lässt sich also für jedes Dokumente in der Wissensbasis genau beschreiben, welche seiner Wörter zum Titel, zu den Autorennamen, zur Kurzfassung oder zum Literaturverzeichnis gehören.

Der zweite Teil der Definition legt den Dokumentdatensatz fest. Jedes Element dieses Datensatzes muss mit einem vordefinierten Typ versehen werden. Natürlich kann das Framework um neue Typen ergänzt werden, hierzu muss allerdings der Programmcode erweitert werden. Im angeführten Beispiel sind Meta–Beschreibungen über die Sprache, den Umfang, den Zeitstempel der letzten Änderung und die URL eines jeden Dokumentes vorgesehen. Man beachte, dass alle diese Beschreibungen nicht direkt aus den Wörtern des Dokumentes abgeleitet werden können, sondern nur indirekt zu bestimmen sind. Der Grund für das Aufnehmen der Elemente `d_title` und `d_authors`, die als Attribute bereits im Index auftauchen, in den Dokumentdatensatz wird im nächsten Abschnitt über die Benutzungsschnittstelle erläutert.

Neben dem Wissen über relevante Dokumente muss ein Domänen–Experte auch noch Verweise (URLs) und Zeitstempel von untersuchten, aber als irrelevant erkannten Dokumenten speichern. Dies ist notwendig, damit solche Dokumente nicht immer wieder untersucht werden, wenn sie z.B. wiederholt in der Ergebnisliste einer angefragten externen

Suchmaschine auftauchen. Man beachte, dass ein Domänen–Experte diese Informationen nicht über alle im Internet verfügbaren Dokumente speichern muss, sondern nur über die von ihm untersuchten. Da die von einer externen Suchmaschine gelieferten Dokumente immer zumindest einem Teil einer Benutzeranfrage genügen müssen, erfolgt über die Anfrage schon eine gewisse Vorfilterung und es muss vom Experten letztendlich nur ein kleiner Teil des Internets untersucht werden. Gleiches gilt für die proaktive Suche, die sich immer auf einen sehr begrenzten Anteil des Internets beschränkt.

Generische Benutzungsschnittstelle

Wenn ein Domänen–Experte spezifisches Wissen über die von ihm erfassten relevanten Dokumente hat, so muss dieses auch vom Benutzer ausnutzbar sein. Insbesondere muss er bei der Formulierung einer Anfrage auf solche Spezifika eingehen können. Das Framework für Domänen–Experten erlaubt hierzu die Definition einer domänenspezifischen, HTML-basierten Anfragemaske. Abbildung 2.4 gibt hierzu ein Beispiel, abermals für die Domäne der „wissenschaftlichen Artikel“.

```
BeginGUIDefinition
   "Keywords", 40 : indexSearch{}, smCriterion;
   "Authors", 40 : indexSearch{a_author};
   "Reference", 40 : indexSearch{a_references};
   "Language", 20 : documentDataRecordSearch{d_language};
EndGUIDefinition
```

Abbildung 2.4: Beispieldefinition einer Anfragemaske

Jede Zeile dieser Definition definiert eine Zeile in der Anfragemaske. Die führende Zeichenkette wird als Beschreibungstext jedem Eingabefeld vorangestellt. Nach der Länge des Eingabefeldes wird dann definiert, wie in diesem Feld eingegebene Suchwörter zu behandeln sind. Dazu wird als Erstes festgelegt, ob die betreffenden Wörter im Index oder im Dokumentdatensatz zu suchen sind. Für den Index kann optional noch die Einschränkung auf solche Wörter erfolgen, die ein bestimmtes, im Dokumentenmodell definiertes Attribut haben. In der dritten Zeile der definierten Anfragemaske werden z.B. nur Wörter aus dem Index selektiert, die auch das entsprechende Attribut `a_author` haben. Für Stichwörter, die im Dokumentdatensatz zu suchen sind, muss das entsprechende, ebenfalls im Dokumentenmodell definierte Feld angegeben werden (wie z.B. in Zeile 5, `"Language"`). Die Bedeutung des Ausdrucks `smCriterion` wird im Abschnitt zum Wissens–Sammler erklärt.

Neben der domänenspezifischen Eingabemaske erlaubt das Framework auch eine Spezifikation der Ausgabe. Abbildung 2.5 gibt dazu ein Beispiel. Mit dieser Definition kann

```
BeginResultDefinition
   d_authors ":" \wwwLink{d_url}{d_title} "," \\
   d_size "kBytes" d_language
EndResultDefinition
```

Abbildung 2.5: Beispieldefinition einer Ergebnisdarstellung

das Aussehen eines einzelnen Ergebniseintrages in der auf eine Anfrage gelieferten Ergebnisseite festgelegt werden. Zur Definition können neben konstanten Zeichenketten nur Elemente des Dokumentdatensatzes sowie weitere zu definierende Funktionen über diesen Elementen herangezogen werden. Im angegebenen Beispiel beginnt die Ausgabe eines Ergebniseintrags mit den Autorennamen. Es folgt die Funktion `wwwLink`, die einen *Hyperlink* erzeugt, der durch den Titel des Dokumentes beschrieben ist und auf die angegebene URL verweist. Nach dem Zeilenumbruch (\\) wird schließlich noch die Größe des Dokumentes und die verwendete Sprache angegeben. Die Definition der Ergebnisdarstellung zeigt nun auch den Sinn der redundanten Definition der Felder `d_title` und `d_authors` im Dokumentdatensatz. Ohne diese Definition wären die entsprechenden Ausgaben nicht möglich, da die Ausgabe aller Indexwörter, die ein gegebenes Attribut erfüllen (z.B. `a_title` oder `a_authors`), vom Framework nicht zugelassen wird, um überlange und evtl. chaotische Ausgaben von vorneherein zu unterbinden.

Wissens–Sammler

Diese Komponente hat zwei domänenspezifische Aufgaben. Zum einen müssen Anfragen an den Domänen–Experten so umgewandelt werden, dass sie an die externe klassische Suchmaschine (ggf. Suchmaschinen) weitergeleitet werden können. Hierzu kann in der Definition der Anfragemaske zu jeder Anfragezeile mit dem Ausdruck `smCriterion` festgelegt werden, dass die entsprechenden Schlüsselwörter für die Anfrage an die klassische Suchmaschine verwendet werden sollen (siehe als Beispiel Abbildung 2.4). In der Regel wird man dies für alle Wörter des Indexes festlegen. Daneben kann der Wissens–Sammler auch noch durch einige Stichwörter konfiguriert werden, mit denen Anfragen an die klassische Suchmaschine gegenüber der ursprünglichen Benutzeranfrage angereichert werden. In der Domäne englischsprachiger „wissenschaftlicher Artikel“ könnten dies z.B. die Stichwörter „abstract“ und „references“ sein, da diese in sehr vielen Artikeln auftauchen.

Zum anderen muss der Zugang zu externen, speziellen Informationsdiensten, die im Rahmen einer proaktiven Suche genutzt werden, realisiert werden. Dieser Zugang muss individuell für jeden benutzten Informationsdienst programmiert werden.

Filterfunktion und Dokumentenanalyse

Die einzige Komponente, die zwingend einer domänenspezifischen Implementierung bedarf, ist die Filterfunktion. Allerdings kann der hierzu erforderliche Aufwand begrenzt werden, indem die Filterfunktion in zwei Teile zerlegt wird: in die allgemeine *Formatabstraktion*, die das aktuelle Dokumentenformat in eine einheitliche Repräsentation konvertiert, und den eigentlichen *Domänentester*, der die Relevanz eines in dieser Repräsentation vorliegenden Dokumentes bewertet (vgl. mit Abbildung 2.1 auf Seite 18). Der Domänentester wiederum könnte anhand von Standard–Klassifikationskomponenten zusammengestellt werden (siehe z.B. [Miller & Bharat, 1998]) oder auch auf lernbasierte Ansätze zurückgreifen, in dem die Funktion zur Erkennung der Relevanz anhand von Beispieldokumenten erlernt wird [Yang & Liu, 1999; Mladenic, 1999]. Nochmals sei betont, dass eine genauere Analyse dieser Aspekte nicht Teil der vorliegenden Arbeit ist.

Die Dokumentenanalyse ist dafür zuständig, ein für relevant befundenes Dokument in die Dokument–Wissensbasis einzutragen. Hierzu muss sie neben dem entsprechenden Aufbau des Indexes, der für alle Domänen einheitlich erfolgt, alle domänenspezifischen Attribute und Meta–Beschreibungen, wie im Dokumentenmodell spezifiziert, berechnen. Im einfachsten Fall ist ein Domänen–Experte bereits voll funktionsfähig und nützlich, wenn keinerlei domänenspezifische Attribute oder Meta–Beschreibungen definiert sind. In diesem Fall muss lediglich die Filterfunktion domänenspezifisch implementiert werden. Der Nutzen durch den Experten besteht dann in der für den Benutzer gemachten Klassifikation sowie in der höheren Aktualität des Indexes. Seinen vollen Nutzen entfaltet ein Domänen–Experte allerdings erst, wenn domänenspezifische Attribute und Meta–Beschreibungen definiert sind — deren Bestimmung muss dann aber von der Dokumentenanalyse implementiert werden.

Anfragenprozessor und Filter–Koordinator

Für den Anfragenprozessor ist die konkrete Domäne eines Experten vollständig transparent. Seine Funktionalität, die im Wesentlichen die Ablaufsteuerung zur Bearbeitung einer Benutzeranfrage und die Berechnung der letztlich relevanten Dokumente umfasst, ist komplett im Framework implementiert und braucht nicht domänenspezifisch angepasst zu

werden. Details zu dieser Implementierung, wie z.B. die Berechnung von Relevanzgraden einzelner Dokumente bezüglich einer Benutzeranfrage, finden sich in [Arnold, 1999].

Der Filter–Koordinator ist ebenfalls komplett domänenunabhängig. Seine Funktion besteht aus den in Kapitel 4 entwickelten Algorithmen zur Aussendung mobiler Programme.

2.4 Diskussion

Das Konzept der Domänen–Experten löst alle drei in Abschnitt 2.1.5 identifizierten Probleme spezialisierter Suchmaschinen. Die Relevanz von Dokumenten kann mittels einer domänenspezifischen Filterfunktion genau bestimmt werden. Inwieweit für eine konkrete Domänen eine solche Filterfunktion realisierbar bzw. effektiv ist, hängt vom jeweiligen Stand der Wissenschaft ab. Jedoch erlauben Domänen–Experten jederzeit den Einsatz der aktuellsten Methoden und erreichen somit die bestmögliche Lösung. Durch Rückgriff auf eine allgemeine Suchmaschine sowie durch den Einsatz mobiler Programme wird zudem eine effiziente Entdeckung von relevanten Dokumenten erreicht. Und schließlich wird der Aufwand zur Erstellung eines Domänen–Experten durch ein konfigurierbares Framework wesentlich reduziert.

Einbettung in eine IR–Infrastruktur

Domänen–Experten stellen eine nützliche Ergänzung zu bestehenden IR–Systemen dar. Gleichwohl werden sie diese nicht völlig verdrängen, da es nie zu jeder beliebigen Domäne einen entsprechenden Experten geben wird, sei es weil keine effektive Filterfunktion realisierbar ist oder weil der Aufwand zur Erstellung des Experten im Vergleich zu seinem Nutzen für diese Domäne als zu hoch erachtet wird. Daher werden die klassischen Ansätze der Suchmaschinen und der Kataloge weiterhin Bestand haben. Erstere werden ja bereits von Domänen–Experten benötigt und sind für einen Benutzer in vielen Fällen der einfachste und schnellste Weg, um an eine Information zu kommen. Letztere bieten aufgrund der redaktionellen Einordnung eine auf technischem Wege nicht erreichbare Qualität.

Für den Benutzer wäre eine Integration all dieser Ansätze (Suchmaschinen, Kataloge und Domänen–Experten) durch eine Meta–Suchmaschine höchst wünschenswert. Da ein Benutzer nicht alle verfügbaren Suchwerkzeuge und insbesondere Domänen–Experten kennen sollte bzw. kennen kann, muss der Zugang zu diesen durch eine Meta–Instanz unterstützt werden, die Anfragen entgegennimmt und an das passendendste Suchwerkzeug weiterreicht. Hierbei sind drei Aspekte zu lösen, die an dieser Stelle kurz angerissen (aber

nicht gelöst) werden sollen. Zunächst muss die Meta–Instanz über alle existierenden Suchwerkzeuge und deren Fähigkeiten unterrichtet werden. Dies kann entweder auf Initiative der Suchwerkzeuge erfolgen, die sich bei der Meta–Instanz registrieren müssen. Dieser Ansatz wird z.B. im System *Harvest* [Bowman et al., 1995] verfolgt. Alternativ ist die Meta–Instanz für die Erfassung der verfügbaren Suchwerkzeuge selbst zuständig (siehe z.B. [Manber & Bigot, 1997; Selberg & Etzioni, 1995; Dreilinger & Howe, 1997]).

Ausgestattet mit dem notwendigen Wissen über verfügbare Suchwerkzeuge muss die Meta–Instanz eingehende Benutzeranfragen bearbeiten. Hierzu muss sie zunächst den für eine Anfrage relevanten Suchwerkzeuge bestimmen und dann die Anfrage auf die (domänen-)spezifische Benutzungsschnittstelle abbilden. Dies kann halbautomatisch erfolgen (wie z.B. im *Search Broker* [Manber & Bigot, 1997] oder im System *Pharos* [Dolin et al., 1997]), indem der Benutzer ein Suchwerkzeug aus einer vorgeschlagenen Liste auswählt und seine Anfrage dann in ein (domänen-)spezifisches Formular eingetragen wird, das vom Benutzer noch modifiziert werden kann. Wie bei einer vollkommen transparenten Vermittlung (z.B. bei den Meta–Suchmaschinen *MetaCrawler* [Selberg & Etzioni, 1995] und *SavySearch* [Dreilinger & Howe, 1997]) eine automatische Abbildung der Anfrage auf die (domänen-)spezifische Benutzerschnittstelle zu erreichen ist, bleibt Forschungsgegenstand.

Bevor nun in Kapitel 4 beschrieben werden kann, wie die Aussendung mobiler Filterprogramme optimal zu koordinieren ist, wird vorher noch in Kapitel 3 das Konzept der Netzwerk–Distanzkarten präsentiert. Die mit diesen Karten schätzbaren Distanzen zwischen Netzwerkrechnern sind eine notwendige Voraussetzung für die Darstellung der Aussendungsverfahren in Kapitel 4. Eine Fallstudie zum Ansatz der Domänen–Experten wird in Kapitel 7 vorgestellt.

Kapitel 3

Dynamische Distanzkarten vom Internet

Karten im alltäglichen Leben sollen ihren Benutzern einen Überblick über ihnen unbekanntes Gelände geben und somit ein schnelles Vorankommen zu anderen Orten ermöglichen. Analog sollen Netzwerk–Distanzkarten einen Überblick über Computernetzwerke bieten, insbesondere soll anhand von ihnen die Bestimmung der netzwerktechnischen Distanz (Netzwerkdistanz$^{\uparrow}$) zwischen beliebigen an das Netzwerk angeschlossenen Rechnern, im Folgenden Endsysteme$^{\uparrow}$ genannt, möglich werden. Im Gegensatz zu alltäglichen Karten ist die Angabe von Kilometer-Distanzen in Netzwerken nicht besonders sinnvoll, sondern es sind vielmehr technische Charakteristiken einer Netzwerkverbindung, die für den Benutzer relevant sind, die aber auch Eigenschaften eines klassischen Distanzmaßes aufweisen. Ein weiterer Unterschied zu alltäglichen Karten besteht darin, dass Entfernungen in Netzwerken permanenten Änderungen unterworfen sind bzw. sein können, die sich im Extremfall sogar im Sekundenbereich abspielen können. Damit wird eine ständige Anpassung einer einmal erstellten Karte an die sich verändernden Bedingungen erforderlich. In diesem Sinne sind Netzwerk–Distanzkarten sehr dynamische Gebilde. Aufgrund der momentan überragenden Bedeutung des Internets konzentriert sich dieses Kapitel auf dieses spezielle Computernetzwerk. Gleichwohl sind die vorgestellten Verfahren (in ihrem Kern bereits in [Theilmann & Rothermel, 2000a] beschrieben) prinzipiell in beliebigen Netzwerken anwendbar (vgl. hierzu das Systemmodell in Abschnitt 3.3.1).

Der Rest dieses Kapitels ist wie folgt gegliedert: Im nachfolgenden Abschnitt 3.1 werden zur Motivation von Distanzkarten einige Anwendungen vorgestellt, die Distanzinformati-

on über Computernetzwerke benötigen. Anschließend werden einige grundlegende Eigenschaften von Distanzmaßen sowie vom Internet besprochen, die für das tiefere Verständnis des Problems der Distanzbestimmung hilfreich sind. Nach einer Diskussion existierender Ansätze zur Distanzbestimmung in Abschnitt 3.2 wird in Abschnitt 3.3 die Idee des Kartenansatzes mit dem zugrundeliegende Systemmodell eingeführt. Die Algorithmen zur Erstellung und Anpassung von Karten auf Basis einzelner Distanzmessungen werden dann in Abschnitt 3.4 erklärt. Abschließend wird in Abschnitt 3.5 gezeigt, wie die verteilte Datenerhebung (Distanzmessung) und Datenaggregation angemessen koordiniert werden kann und wie Distanzkarten für beliebige Klienten repliziert werden. Eine Evaluation des Ansatzes erfolgt in Kapitel 5.

3.1 Problemstellung

3.1.1 Motivation

Wie fast jede Systemressource ist auch die Ressource Netzwerkbandbreite ein knappes Gut, das es effizient zu nutzen gilt. Neben der Optimierung der auf Netzwerkebene eingesetzten Techniken zu möglichst effizientem *Routing*[1], zur Stauvermeidung u.ä gibt es auch auf Anwendungsebene einige Ansätze, um die Effizienz bzw. die Geschwindigkeit der Kommunikation über das Netzwerk zu verbessern. Diese können zwar das Verhalten des Netzwerkes nicht selbst beeinflussen, aber immerhin versuchen, die Distanz zwischen Kommunikationspartnern zu minimieren.

Die Technologie der mobilen Programme oder auch mobilen Agenten[2] ist explizit dazu vorgesehen, die Distanz zwischen Klient und Server zu verringern, indem Funktionalität (Programme) vom einen zum anderen oder möglichst nahe zu ihm übertragen wird, um die eigentliche Kommunikation dann lokal vornehmen zu können [Rothermel et al., 1997]. Ein Szenario zur nutzbringenden Verwendung mobiler Programme wird in Kapitel 4 vorgestellt. Hier werden mobile Programme ausgesandt, um verteilt vorliegende Dokumente effizient zu filtern. Dabei wird versucht, die Programme möglichst nahe an die zu untersuchenden Server zu bringen, wozu eben die Kenntnis von Netzwerkdistanzen erforderlich ist. Eine Anwendung, die von der Aussendung solcher mobilen Filterprogramme profitiert,

[1] Die deutsche Übersetzung dieses Begriffes (Leitwegsbestimmung) ist derart ungebräuchlich, dass sie in dieser Arbeit nicht verwendet wird. Gleiches gilt für den Begriff Router$^{\uparrow}$ (zu deutsch Vermittlungsknoten).

[2] Eine genaue Definition dieser Konzepte erfolgt in Kapitel 4.1.

wurde mit den Domänen–Experten bereits in Kapitel 2 vorgestellt. Ein Domänen–Experte müss zur effizienten Lokalisierung der für ihn relevanten Dokumente gezielt einzelne, verteilt vorliegende Dokumente mit einer Filterfunktion überprüfen. Die Überprüfung kann durch den Einsatz mobiler Filterprogramme sehr effizient und mit einer minimalen Netzwerkbelastung erfolgen.

In einer von [Ranganathan et al., 1997] vorgestellten Anwendung der Technologie mobiler Agenten wird ein *chat*–Server[3] dynamisch so verlagert, dass die maximale Distanz (die Latenzzeit) zu seinen Klienten minimiert wird.

Eine der verbreitetsten Techniken zur Optimierung der Netzwerkkommunikation auf Anwendungsebene ist die Replikation populärer Daten oder Dienste in die Nähe von großen Benutzergruppen. Damit können Benutzer auf solche Daten/Dienste über wesentlich kürzere Distanzen zugreifen, womit der Zugriff (in der Regel) schneller erfolgen kann und zudem das Netzwerk weniger stark belastet wird. Beide in diesem Szenario beteiligten Parteien benötigen Distanzinformationen über das zugrundeliegende Netzwerk. Ein Replikations–Server benötigt Wissen über die Entfernungen zu anderen Replikations–Servern, auf die er seine Daten/Dienste replizieren kann, und zu den Benutzern, die diese Daten/Dienste benötigen (siehe z.B. [Gwertzman & Seltzer, 1995; Gwertzman & Seltzer, 1996; Bestavros, 1997; Rabinovich et al., 1999]). Benutzer von replizierten Entitäten müssen die Entfernung zu einzelnen Replikations–Servern kennen (und natürlich wissen, welche Entitäten wo verfügbar sind), um den nächstgelegenen Server auszuwählen. Neben dem Replikationsmanagement gibt es noch einige andere Anwendungen, in denen Wissen über Netzwerkdistanzen ausgenutzt wird. So wird beispielsweise in [Lagoze et al., 1998] vorgeschlagen, das Routing von Benutzeranfragen in verteilten Digitalen Bibliotheken entsprechend der jeweiligen Netzwerkdistanzen zu optimieren.

Manche Anwendungen benötigen zwar keine expliziten Distanzinformationen, aber wenigstens eine gewisse Kenntnis von Netzwerkregionen, also von Bereichen in denen die enthaltenen Endsysteme enger miteinander verbunden sind als mit Endsystemen aus anderen Regionen. Hierzu gehören Verfahren zur hierarchischen Organisation von Caches [Chankhunthod et al., 1996] oder auch Systeme mit verteilten Objektspeichern [Steen et al., 1997]. Ebenfalls auf einer Regionenhierarchie basieren manche Ansätze zur Anmeldung neuer Mitglieder in zuverlässigen Multicast–Gruppen [Rothermel & Maihöfer, 1999].

[3] Der *chat*–Dienst ermöglicht das Einrichten und Betreiben von Unterhaltungsforen im Internet. Der zentrale *chat*–Server empfängt dabei Gesprächsbeiträge der einzelnen Teilnehmer und leitet diese an die anderen weiter.

Ein großes Problem für alle diese Anwendung besteht darin, Distanzinformationen zu erlangen, ohne dabei das Netzwerk durch eigene Messungen so sehr zu belasten, dass der Effekt der angestrebten Entlastung letztlich ausbleibt oder sich sogar ins Gegenteil verkehrt. Zum besseren Verständnis dieses Problems werden in den nachfolgenden Abschnitten zunächst einige Grundlagen zu Netzwerkdistanzen und zur Funktionsweise des Internets erklärt, bevor dann in Abschnitt 3.2 existierende Lösungsansätze diskutiert werden.

3.1.2 Distanzmaße und Distanzmetriken

Als Distanzmaß wird in dieser Arbeit ein beliebiges Maß zur Beschreibung der Verbindungscharakteristik zwischen zwei Endsystemen in einem Netzwerk bezeichnet. Ein paar netzwerktechnisch relevante Maße seien hier beispielhaft erwähnt: Die *Latenzzeit* (engl. *delay*) beschreibt den Zeitaufwand zur Übertragung eines einzelnen Bits auf einen entfernten Rechner. Die *Umlaufzeit*$^{\uparrow}$ (engl. *round trip time – rtt*) zählt dagegen die komplette Zeit zur Übertragung eines Datagramms bis zu dessen Bestätigung. Die *Knotenzahl*$^{\uparrow}$ (engl. *hop count*) bezeichnet die Zahl der bei einer Übertragung durchlaufenen Router$^{\uparrow}$. Die *verfügbare Bandbreite* beschreibt die theoretisch in einem Übertragungskanal verfügbare Bandbreite. Dagegen bezeichnet die *Restbandbreite* nur noch die wirklich verfügbare Bandbreite, also abzüglich der durch anderen Datenverkehr schon belegten Bandbreite. Weitere interessante Charakteristiken sind die Paket–Verlustrate, also der Prozentsatz der im Mittel auf einem Weg verworfenen oder verlorenen Datenpakete, oder auch die Veränderlichkeit der Latenzzeit (engl. *jitter*).

Verschiedene Distanzmaße können je nach Kontext interessant sein. Für Benutzer von Börseninformationsdiensten hat z.B. die Latenzzeit absolute Priorität, für Web–Surfer dagegen bekommt der Durchsatz eine höhere Bedeutung und bei der Übertragung von Multi–Media–Strömen steht die Veränderlichkeit der Umlaufzeit im Vordergrund. Schließlich können einzelne Distanzmaße auch zur (technisch einfachen) Schätzung anderer Maße herangezogen werden. Bei zuverlässigen Transportprotokollen, z.B. dem im Internet verwendeten *Transmission Control Protocol*$^{\uparrow}$ (TCP), hängt die erreichbare Bandbreite wesentlich von der Geschwindigkeit, mit der Paketbestätigungen eintreffen, also von der Umlaufzeit, ab. Allgemein kann die Knotenzahl (im Internet evtl. nach Hierarchieebenen gewichtet) als grober Schätzwert für die Umlaufzeit genutzt werden. Es sei an dieser Stelle betont, dass die Entwicklung und Bewertung von Distanzmessverfahren nicht Teil dieser Arbeit ist.

Um mit Distanzmaßen praktisch umgehen zu können werden oft die Eigenschaften einer Metrik gefordert, bei deren Gültigkeit man von einer Distanzmetrik spricht.

Definition 3.1 (Distanzmetrik): Ein Distanzmaß $|*,*|$ ist für eine Menge von Punkten eine Metrik, genau dann wenn für drei beliebige Punkte x, y, z gilt:

1. $|x, y| = 0 \Leftrightarrow x = y$
2. Symmetrie: $|x, y| = |y, x|$
3. Dreiecksungleichung: $|x, z| \leq |x, y| + |y, z|$

Hieraus lässt sich insbesondere ableiten, dass $|x, y| \geq 0$.

Während Distanzmaße aus dem täglichen Leben (wie z.B. das Meter–Maß) eine Metrik bilden, ist dies für Netzwerkdistanzen nicht oder nur eingeschränkt der Fall. Das 1. Kriterium ist in diesem Zusammenhang unkritisch, da seine Gültigkeit im Wesentlichen von der Definition des Extremfalles $|x, x|$ abhängt, was für Netzwerkdistanzmaße, wie den oben angeführten, keine Beschränkung darstellt.

Für die Gültigkeit der Dreiecksungleichung muss ein Netzwerkdistanzmaß zwei Voraussetzungen erfüllen. Erstens muss es additiv sein, d.h. die Distanz $|x, z|$ zwischen zwei Endsystemen x, z, die nur über einen Router y miteinander verbunden sind, muss gleich $|x, y| + |y, z|$ sein. Dies gilt z.B. für die Distanzmaße Umlaufzeit und Knotenzahl, nicht aber für die Bandbreite, bei der in obiger Konstellation gilt: $|x, z| = \min\{|x, y|, |y, z|\}$. Zweitens müssen die zugrundeliegenden Routing–Verfahren bezüglich des Distanzmaßes optimal sein, also immer den kürzesten Weg wählen. Diese Voraussetzung wird im Internet sicherlich nie perfekt erfüllt sein, es bleibt jedoch zu untersuchen wie groß der durch diese Annahme entstehende Fehler ist.

Die Symmetrieannahme ist in aller Regel nicht perfekt erfüllt. Für das Maß Umlaufzeit ergibt sie sich noch trivial aus der Definition. Für andere Maße hingegen ist sie aufgrund des potenziell asymmetrischen Routings und der asymmetrischen Auslastung der Übertragungskanäle nicht erfüllt. Abermals bleibt der Fehler dieser Annahme im konkreten Fall zu untersuchen.

3.1.3 Grundlagen des Internet

Das Internet ist, wie sein Name *Inter–net* schon sagt, ein (paketvermittelndes) Netzwerk, das verschiedene Netzwerke verknüpft und integriert, womit ein globales Netzwerk

entsteht. Basis dieser Verknüpfung ist das sogenannte *Internet Protocol*↑ (IP), das das Format von Datagrammen und Rechneradressen regelt. Eine Adresse besteht aus drei Teilen: einem Teil zur Bezeichnung des Netzwerkes, einem optionalen Teil zur Bezeichnung des Subnetzes und einem Teil zur Identifikation des Rechners innerhalb des spezifizierten Netzes bzw. Subnetzes. Organisatorisch betrachtet besteht das Internet aus einer Menge von *autonomen Systemen*↑, das sind Teilnetze, die ein oder mehrere Netzwerke (im Sinne der IP–Adressen) umfassen und unter einer administrativen Autorität stehen.

Für das Routing zwischen autonomen Systemen wird im Internet ein einheitliches Protokoll verwendet, momentan das *Border Gateway Protocol*↑ (BGP) [Rekhter & Li, 1995]. BGP ist im Kern ein Distanz–Vektor–Routing–Protokoll, in dem jeder Router die von ihm erreichbaren Subnetze sowie die hierzu zu durchlaufenden Pfade von autonomen Systemen mit seinen benachbarten Routern austauscht. BGP–Router können mit einer Fülle von spezifischen Kriterien konfiguriert werden, die die Politik des Routings beeinflusse. So lassen manche Router nur Transitverkehr für „befreundete“ Router zu oder vermeiden, dass ihre Daten zu nicht vertrauenswürdigen Routern (z.B. von konkurrierenden Firmen, Institutionen oder von feindlichen Ländern) gesandt werden. Aus den von Nachbarn eingehenden Pfaden kann ein BGP–Router dazu diejenigen herausfiltern, die seiner Politik genügen. Aus den verbleibenden Pfaden wird dann der kürzeste ausgewählt, also derjenige mit der geringsten Zahl an zu traversierenden autonomen Systemen.

Innerhalb von autonomen Systemen ist im Internet kein spezielles Routing–Protokoll vorgeschrieben. Es gibt lediglich ein empfohlenes Protokoll, das *Open Shortest Path First*–Protokoll↑ (OSPF) [Moy, 1997]. OSPF basiert auf dem Link–Zustands–Verfahren, in dem jeder Router den Zustand aller seiner Verbindungsleitungen an alle anderen Router im gleichen autonomen System überträgt. Zur Verbesserung der Skalierbarkeit in großen autonomen Systemen erlaubt OSPF die Einführung einer weiteren Hierarchieebene, sogenannte Bereiche (engl. *areas*). Bei Routing–Protokollen innerhalb von autonomen Systemen kommen zahlreiche Distanzmaße zum Einsatz, z.B. Verzögerungszeit, Bandbreite oder auch die geographische Distanz.

Eine gute Einführung in Struktur und Funktionsweise von Computernetzwerken im Allgemeinen und vom Internet im Besonderen findet sich in [Tanenbaum, 1996].

3.2 Existierende Ansätze zur Distanzbestimmung

Einige Internetanwendungen, die keine expliziten Distanzen sondern lediglich Wissen über die grobe Zuordnung von Endsystemen zu Regionen benötigen (vgl. mit Abschnitt 3.1.1),

nehmen eine administrative Einteilung des Internets in Regionen einfach an (z.B. [Steen et al., 1997; Rothermel & Maihöfer, 1999]) bzw. führen diese für Teilbereiche selbst durch [Chankhunthod et al., 1996]. Eine Festlegung von Regionen im globalen Maßstab ist bisher nicht erfolgt.

Schätzung durch geographische Distanzen

Eine scheinbar einfache Möglichkeit zur Bestimmung von Netzwerkdistanzen besteht darin, diese durch die geographische Distanz zwischen den jeweiligen Endsystemen zu schätzen. [Gwertzman & Seltzer, 1995] evaluieren diesen Ansatz im Kontext der Replikation populärer Dokumente. Sie stellen jedoch fest, dass eine Korrelation zwischen der geographischen Distanz und der als Netzwerkdistanzmaß gewählten Knotenzahl nur sehr eingeschränkt vorliegt, nämlich nur innerhalb von einzelnen und relativ kleinen autonomen Systemen. Das Problem, wie zu einem beliebigen Endsystem im Internet dessen geographische Koordinaten bestimmt werden können, wird von ihnen nicht behandelt.

Ausnutzung von Routing–Tabellen

Eine naheliegende Idee besteht darin, Distanzinformationen aus den von Routern erstellten und gepflegten Routing–Tabellen zu entnehmen. Ein auf dieser Methode basierender Replikationsdienst wurde z.B. von [Rabinovich et al., 1999] entwickelt. Will man jedoch die Distanz zwischen zwei beliebigen über das Internet miteinander verbundenen Endsystemen bestimmen, tauchen eine Reihe von Problemen auf:

- Zunächst einmal ist das Internet aus Gründen der Fehlertoleranz und der Skalierbarkeit dezentral aufgebaut. Damit kennt jeder Router immer nur einen sehr begrenzten Ausschnitt aus der Topologie und dem Zustand des Netzwerkes. Entfernungsinformationen müssen also aus einer Fülle von verteilt vorliegenden Einzelinformationen zusammengesetzt werden.
- Daneben ist man auf die von den Routing–Verfahren benutzten Distanzmaße angewiesen. Beispielsweise wird für das Routing zwischen autonomen Systemen lediglich die Zahl der traversierten Systeme als Entfernungsmaß benutzt [Rekhter & Li, 1995], was für viele Anwendungen ein ziemlich unnützes Entfernungsmaß ist, da die Größe von autonomen Systemen und damit die Kosten zu ihrer Traversierung extrem unterschiedlich sein können. Zusätzlich werden in verschiedenen Teilnetzen des Internets (in verschiedenen autonomen Systemen oder Bereichen) durchaus unterschiedliche Distanzmaße verwendet, was die Zusammensetzbarkeit einzelner Distanzinformationen zu einer globalen Distanz stark einschränkt.

- Schließlich ist der Zugriff auf die Routing–Tabellen nicht öffentlich, sondern dem jeweiligen Betreiber vorbehalten. Aufgrund der großen Zahl von autonomen Systemen ist es daher praktisch unmöglich, von allen diesen eine spezielle Zugriffserlaubnis zu bekommen.
- Ein orthogonales Problem in der Distanzbestimmung besteht in der Dynamik von Netzwerkdistanzen. Damit das Internet auch bei Ausfall oder Überlast von einzelnen Routern seinen Dienst noch möglichst gut erbringt, müssen Routing–Pfade dynamisch angepasst werden. Dadurch kann sich die Distanz zwischen zwei Endsystemen in sehr kurzen Zeiträumen stark ändern, und man muss die Informationen aus den Routing–Tabellen dementsprechend oft neu laden.

Aufgrund dieser Probleme ist der Ansatz von [Rabinovich et al., 1999] auch nur zum Einsatz innerhalb einzelner autonomer Systeme gedacht, ohne Beachtung von Subdomänen oder Bereichen. Die Replikation erfolgt dabei nur zu Endsystemen der eigenen Domäne oder zu Rechnern am „Domänenrand“, also zu Rechnern die von fremden Domänen aus schnell erreichbar sind, ohne die eigene Domäne wesentlich zu belasten.

Messbasierte Ansätze

Zahlreiche Ansätze basieren auf lokal von einer Anwendung vorgenommenen Entfernungsmessungen, z.B. von Klienten zu mehreren Replikations–Servern [Carter & Crovella, 1996; Guyton & Schwarz, 1995] und umgekehrt [Bestavros, 1997] oder auch von einem *chat*–Server zu seinen Klienten [Ranganathan et al., 1997]. Auf diese Weise kann ein Endsystem seine Distanz zu beliebigen anderen Endsystemen bestimmen, es kann aber nicht die Distanz zwischen zwei entfernten Endsystemen herausbekommen. Letzteres wäre insbesondere für Replikations–Server notwendig, da diese die Distanz zwischen anderen Servern und Klienten bestimmen müssen. Ein zweiter Nachteil ist der unnötig hohe erzeugte Netzverkehr, der entsteht, wenn jedes Endsystem bzw. jede Anwendung individuell seine Entfernungsmessungen vornimmt. So machen z.B. zwei nahe beieinanderliegende Klienten eines Replikations–Dienstes, die von ihrer gegenseitigen Existenz nichts wissen, fast dieselben Messungen, ohne voneinander profitieren zu können.

Von [Moore, 1998] und [Francis, 1997] wurden Protokolle für zwei Dienste (SONAR bzw. HOPS) zur Verteilung von Distanzinformationen vorgeschlagen. Während ein SONAR–Server lediglich Distanzen zwischen sich selbst und anderen Endsystemen liefert, kann der HOPS–Dienst prinzipiell beliebige Distanzen liefern. Hierzu werden HOPS–Server in einer Hierarchie ähnlich der des *Domain Name Systems*[↑] (DNS) angeordnet. Das Problem der skalierbaren Erhebung von Distanzdaten wird von beiden Diensten nicht behandelt.

Offensichtlich ist die komplette Ausmessung aller Distanzen, welche bei n Endsystemen einen Aufwand von n^2 hätte, nicht akzeptabel.

Der bisher einzige Ansatz zu einer messbasierten Schätzung von globalen Umlauf–Distanzen stammt von [Francis et al., 1999]. Sie schlagen die Verwendung einer Reihe von Mess–Servern vor, die über das Internet verteilt sind, und die zunächst ihre Distanzen untereinander ausmessen. Aus der resultierenden Datenstruktur von quadratischem Umfang werden alsdann Kanten, die sich durch additive Konkatenation anderer Kanten approximieren lassen, eliminiert. Die Zuordnung von Endsystemen zum jeweils nächstgelegenen Mess–Server erfolgt durch einen zufallsgesteuerten Algorithmus, bei dem Mess–Server regelmäßig ihre Distanz zu zufällig ausgewählten Endsystemen bestimmen und diesen zugeordnet werden, wenn die Distanz geringer als zum bisher zugeordneten Mess–Server ist. Die Bestimmung der Distanz zwischen zwei Endsystemen erfolgt dann über die zwei jeweils zugeordneten Mess–Server und durch Berechnung des kürzesten Pfades zwischen diesen beiden.

Leider werden die einzelnen Mechanismen lediglich angerissen, jedoch nicht als komplette Verfahren ausgearbeitet und vorgestellt. Es erscheint zumindest fragwürdig, inwieweit eine effektive Vereinfachung der Distanzmatrix mit obigen Mechanismen möglich ist. Die Diskussion des Ansatzes erfolgt rein über den von einer Distanz–Karte benötigten Speicherbedarf. Die erzeugte Netzlast hingegen wird nicht beachtet. Das Problem der Anpassung an sich ändernde Netzcharakteristiken wird kaum angesprochen. Eine experimentelle Evaluation des Ansatzes findet nicht statt. Schließlich stellt sich das prinzipielle Problem, dass die Verwendung von Algorithmen zur Berechnung kürzester Pfade nur dann sinnvoll ist, wenn das betrachtete Distanzmaß additiv ist (Bandbreite ist z.B. kein additives Maß) und zudem von der zugrundeliegenden Routing–Strategie optimiert wird.

Anstatt Skalierbarkeit über die nachträgliche Beschneidung einer Distanzmatrix von quadratischem Umfang und durch Pfadaggregation anzustreben, basiert der in dieser Arbeit vorgestellte Ansatz auf einer Ballung von Mess–Servern in eine Hierarchie von Regionen. Damit werden sowohl Netzwerk- als auch Speicherkosten minimiert. Zudem ist der Ansatz für nahezu beliebige Distanzmaße anwendbar.

3.3 Ansatz der Netzwerk–Distanzkarten

Dieser Abschnitt stellt nun den Ansatz der Distanzkarten von Computernetzwerken vor. Dazu wird zunächst das zugrundeliegende Systemmodell beschrieben und es werden einige

Anforderungen diskutiert. Danach folgt die dem Kartenansatz zugrundeliegende Idee und die sich daraus ergebende Datenstruktur zur Repräsentation einer Distanzkarte. Die konkreten Algorithmen zur Kartenerstellung werden in Abschnitt 3.4 präsentiert.

3.3.1 Systemmodell

Netzwerk. Das im Folgenden benutzte Modell eines Computernetzwerkes besteht aus einer Menge von Endsystemen $\mathcal{E}$ zusammen mit einer Funktion $\Delta(x, y)$, die jedem Paar von Endsystemen $x, y \in \mathcal{E}$ einen nicht-negativen und symmetrischen Distanzwert zuweist.

Die Motivation der Symmetrieannahme erfolgt am Ende dieses Abschnittes im Paragrafen über die Netzwerksicht sowie im Abschnitt 3.3.3. Die Gültigkeit der Annahme wird anhand der Experimente in Kapitel 5 überprüft.

Da Netzwerk–Distanzkarten prinzipiell für beliebige Distanzmaße erstellt werden können, wird vom Netzwerkmodell keinerlei spezielles Distanzmaß gefordert. Netzwerk- oder Knotenfehler, die dazu führen, dass zwei Endsysteme (temporär) nicht miteinander verbunden sind, werden zunächst nicht beachtet, um die Darstellung des Ansatzes möglichst einfach zu halten. In Abschnitt 3.6.1 wird dann explizit diskutiert, inwieweit Netzwerk–Distanzkarten solche Fehler wiedergeben können und wie die vorgestellten Algorithmen und Protokolle fehlertolerant gemacht werden können. Dabei wird sich zeigen, dass die Beachtung von Fehlern keine Einschränkung des Kartenansatzes mit sich bringt.

Das Modell eines Netzwerkes beschreibt immer nur einen einzelnen globalen Zustand. Zeitliche Veränderungen von Distanzen werden nicht modelliert. Sie können jedoch als Folge von Zuständen, d.h. als Folge von Distanzkarten, beschrieben werden.

Eine mögliche Vereinfachung bei der Modellierung des Internets bestünde darin, die Granularität der Netzwerkbetrachtung von Endsystemen auf Subnetze zu reduzieren, also alle Endsysteme innerhalb eines Subnetzes zusammenzufassen. Damit ließe sich die Zahl der zu behandelnden Netzwerkendpunkte und damit der Aufwand zur Kartenerstellung erheblich reduzieren. Der Verlust an Genauigkeit der Distanzinformationen wäre wahrscheinlich gering. [Francis et al., 1999] diskutieren verschiedene Ansätze, wie eine solche Reduktion auf Subnetze technisch realisiert werden könnte. Für die Verfahren zur Kartenerstellung ist die gewählte Granularität allerdings unerheblich und wird daher im Weiteren auch nicht mehr beachtet.

Netzwerksicht. Realistischerweise kann man nicht davon ausgehen, Zugang auf jedes im Netzwerk befindliche Endsystem zu haben, um von dort Distanzmessungen vorzunehmen. Deshalb wird angenommen, dass eine Menge von *Mess–Servern*$^{\uparrow}$ $\mathcal{M} \subseteq \mathcal{E}$ existiert, die Distanzmessungen zu beliebigen Endsystemen ermöglichen. Mit diesen bekommt man immerhin eine, wenn auch eingeschränkte, Sicht auf das Netzwerk. Im Folgenden wird ein Endsystem als *einfaches Endsystem* bezeichnet, wenn deutlich gemacht werden soll, dass es sich bei diesem nicht um einen Mess–Server handelt.

Auch wenn grundsätzlich von der Messbarkeit beliebiger Distanzen ausgegangen wird, ist dies in der Praxis nicht immer möglich. Für den Ansatz der Distanzkarten stellt sich dieser Fall jedoch genauso dar wie ein Netzwerkfehler und kann daher mit den in Abschnitt 3.6.1 besprochenen Mechanismen zur Fehlertoleranz behandelt werden.

In realen Netzwerken kann es natürlich Endsysteme geben, zu denen prinzipiell keine Distanzmessung möglich ist, z.B. weil sie hinter einer *Firewall*[4] in einem privaten Netzwerk liegen. Daher versteht sich die Menge der im Netzwerk betrachteten Endsysteme immer als die Menge der öffentlich zugänglichen Systeme. Nichtsdestotrotz kann auch die Distanzinformation bis zu einer *Firewall* für außerhalb liegende Endsysteme durchaus nützlich sein.

Aufgrund der realistischerweise angenommenen beschränkten Netzwerksicht können Distanzen zwischen Mess–Servern und Endsystemen immer nur aus Sicht der Mess–Server bestimmt werden. Daher macht es im Netzwerkmodell keinen Sinn, diese Distanzen nach ihrer Richtung zu unterscheiden.

3.3.2 Anforderungen

Genauigkeit der Distanzinformationen

Die von Netzwerk–Distanzkarten gelieferten Distanzen sollen natürlich so genau wie möglich mit dem aktuellen Netzzustand übereinstimmen. Die praktisch erreichbare Qualität hängt aber wesentlich von der, durch die Kartenerstellung bedingten, tolerierbaren Netzwerklast ab. Diese Toleranzgrenze wiederum wird durch einige externe Faktoren bestimmt, die nichts mit den Verfahren zur Kartenerstellung zu tun haben: Zuallererst

[4] Eine *Firewall* dient der Kontrolle des Netzwerkverkehrs zwischen einem externen Netzwerk (z.B. dem Internet) und einem internen Netzwerk (z.B. einem Firmennetz). Dabei soll unerlaubter oder unerwünschter Datenverkehr herausgefiltert werden, wodurch letztlich das interne Netzwerk vor externen Netzwerkattacken geschützt werden soll.

bestimmt die Zahl der Nutzer von Distanzkarten, d.h. eigentlich vielmehr der Nutzen den sie aus den Karten ziehen, die zur Kartenerstellung tolerierbare Netzwerkbelastung. Selbst bei gegebener Toleranzgrenze hängt die erzielbare Qualität noch wesentlich vom gewählten Distanzmaß und dem Messverfahren ab. Z.B. belastet die Messung der aktuell verfügbaren Restbandbreite ein Netzwerk wesentlich mehr als die Bestimmung der momentanen Umlaufzeit [Carter & Crovella, 1996]. Daneben spielt noch die Stabilität des gewählten Distanzmaßes eine entscheidende Rolle, da diese die Häufigkeit bestimmt, mit der eine Messung wiederholt werden muss. Beispielsweise ist das Distanzmaß Knotenzahl wesentlich stabiler als die Umlaufzeit (siehe Kapitel 5).

Um dem Ersteller bzw. Betreiber einer Netzwerk–Distanzkarte ein Maximum an Flexibilität zu ermöglichen, sollten Verfahren zur Kartenerstellung für beliebige Genauigkeitsanforderungen einstellbar sein.

Skalierbarkeit

Der Ansatz der Netzwerk–Distanzkarten muss Skalierbarkeit bezüglich der Netzwerkgröße, also der Zahl der existierenden Endsysteme, auf drei Ebenen erreichen: Erstens muss die durch Distanzmessungen verursachte Netzwerkbelastung eingeschränkt werden. Da große Netzwerke wie das Internet Millionen von Endsystemen miteinander verbinden, ist es offensichtlich inakzeptabel von jedem Endsystem zu jedem anderen die Entfernung explizit zu messen. Zweitens sollte der Speicherbedarf von Distanzkarten möglichst gering sein, damit diese auf zahlreiche Server repliziert werden können, ohne diese zu sehr zu belasten. Und drittens sollte ein konkreter Distanzwert aus einer Distanzkarte heraus schnell bestimmbar sein, damit ein Karten–Server möglichst viele Distanz–Anfragen in kurzer Zeit beantworten kann.

3.3.3 Idee

Die prinzipielle Idee zur Erstellung von Netzwerk–Distanzkarten besteht darin, (1) eine Menge von Mess–Servern zu verwenden, (2) die Distanzen zwischen diesen Servern und zu einfachen Endsystemen zu messen, (3) Endsysteme ihrem jeweils nächstgelegenen Mess–Server zuzuordnen und (4) die Distanz zwischen zwei beliebigen Endsystemen durch die Distanz zwischen deren zugeordneten Mess–Servern zu schätzen.

Natürlich liegt dieser Idee die Hoffnung zugrunde, dass Netzwerkdistanzen möglichst gut die Eigenschaften einer Metrik oder sogar einer Geometrie erfüllen. Die erreichbare Genauigkeit der Distanzschätzung hängt wesentlich von der Zahl und der Verteilung der

Mess–Server ab. Setzt man eine gleichmäßige Verteilung der Mess–Server voraus, dann führt die Hinzunahme von weiteren Mess–Servern zu einer Verringerung der durchschnittlichen Distanz zwischen einem Endsystem und dessen nächstgelegenem Mess–Server womit schließlich die endgültige Distanzschätzung genauer wird. Abbildung 3.1 skizziert dieses Szenario, wobei Distanzen durch die planare Geometrie definiert sind.

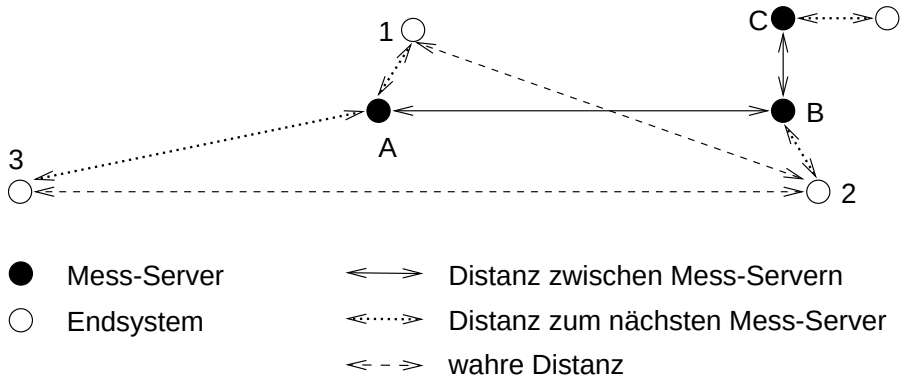


Abbildung 3.1: Distanzschätzung mit Hilfe von Mess–Servern

Die Distanz zwischen den Endsystemen 1 und 2 sowie zwischen 2 und 3 wird hier durch die Distanz zwischen den Mess–Servern A und B geschätzt. Während die erste Schätzung recht genau ist, ist die zweite Schätzung aufgrund der großen Distanz zwischen Endsystem 3 und Mess–Server A ziemlich unpräzise.

Der beschriebene Ansatz birgt zwei Probleme im Sinne der Skalierbarkeit der durch Messungen verursachten Netzbelastung. Zum einen erfordert die Zuordnung von einfachen Endsystemen zu ihrem jeweils nächstgelegenen Mess–Server, dass jeder Mess–Server seine Distanz zu jedem Endsystem messen muss. Zum anderen wächst die Zahl der Messungen zwischen Mess–Servern quadratisch mit der Zahl der zur Verfügung stehenden Server. Von diesen Messungen sind viele anhand von anderen Messungen schätzbar. Ein einfaches Beispiel hierzu zeigt ebenfalls Abbildung 3.1: Dort bräuchte die Distanz zwischen A und C nicht gemessen zu werden, sondern könnte durch die Distanz A–B geschätzt werden, wenn man wüsste, dass C nahe bei B und B relativ gesehen weit entfernt von A liegt.

Zur Lösung dieser Probleme werden die Mess–Server einem rekursiven *Ballungsprozess*$^{\uparrow}$ (engl. *clustering*) unterzogen, der zu einer hierarchischen Einteilung der Server in Regionen und Teilregionen von nahe beieinander liegenden Servern führt. Für jede Region wird ein Mess–Server als Repräsentant auserwählt. Damit kann die Zuordnung eines einfachen Endsystems zu seinem nächstgelegenen Server sehr effizient erfolgen, indem man dieses rekursiv in die Regionenhierarchie einordnet. Ausgehend vom Wurzelknoten der Regionenhierarchie werden die Distanzen zwischen dem Endsystem und den Repräsentanten

der Kindregionen des Wurzelknotens bestimmt. Die Region mit dem nächstgelegenen Repräsentanten wird ausgewählt und der Einfügeprozess wird rekursiv für die Kindregionen der ausgewählten Region fortgesetzt. Damit bei der Ballung der Mess–Server nicht alle Distanzen zwischen diesen gemessen werden müssen, wird ein *gemischter Algorithmus* verwendet, der zunächst eine Ballung anhand einer Teilmenge der verfügbaren Server berechnet und die anderen Server danach passend einsortiert.

Eine wichtige Konsequenz aus dem Ansatz der Ballung in Regionen besteht darin, dass Netzwerkdistanzen symmetrisch sein müssen. Andernfalls könnte man nicht entscheiden, ob zwei Objekte zusammen gruppiert werden sollen, insbesondere wenn die Distanzen in beiden Richtungen erheblich voneinander abweichen. Als Beispiel betrachte man den Prozess der Integration einfacher Endsysteme. Wenn nun, aus der Sicht eines Endsystems, eine Region R_1 im Vergleich zu einer zweiten Region R_2 sehr nahe liegt, aus der Sicht der Regionen aber gerade das Umgekehrte der Fall ist, so kann man keine sinnvolle Zuordnung des Endsystems zu einer der beiden Regionen treffen.

3.3.4 Datenstruktur und Distanzableitung

Netzwerk–Distanzkarten werden durch eine hierarchische Datenstruktur, den sogenannten *Netzwerkbaum*$^{\uparrow}$, dargestellt (siehe hierzu Abbildung 3.2). Innere Knoten dieses Baumes repräsentieren Netzwerkregionen, Blattknoten stehen für einzelne Mess–Server. Sozusagen als Blattknoten zweiter Ordnung sind Endsysteme jeweils ihrem möglichst nächstgelegenen Mess–Server zugeordnet. Für ein Endsystem, das gleichzeitig Mess–Server ist, ist diese Zuordnung natürlich trivial. Der nur aus Mess–Servern und Regionen bestehende Teilbaum wird als *Regionenbaum*$^{\uparrow}$ bezeichnet.

Diese Datenstruktur wird um Distanzwerte zwischen Geschwisterknoten im Regionenbaum sowie zwischen Endsystemen und ihrem jeweils zugeordneten Mess–Server erweitert. Der Distanzwert zwischen zwei verschwisterten Regionen (notiert durch $\|*,*\|$) ist eine Schätzung für die durchschnittliche Distanz zwischen Mess–Servern, die diesen Regionen angehören. Die Distanz zwischen zwei verschwisterten Mess–Servern ergibt sich direkt aus einer Netzwerkmessung. Gleiches gilt für die Distanzen zwischen Endsystemen und ihrem jeweils zugeordneten Mess–Server. Ist ein Endsystem selbst Mess–Server, so ist die Distanz zwischen diesen beiden als 0 definiert. Man beachte, dass ein Netzwerkbaum in keinster Weise mit der Topologie des zugrundeliegenden Netzwerkes korrespondieren muss. Er gibt lediglich Distanzmerkmale zwischen Endsystemen und Mess–Servern wieder.

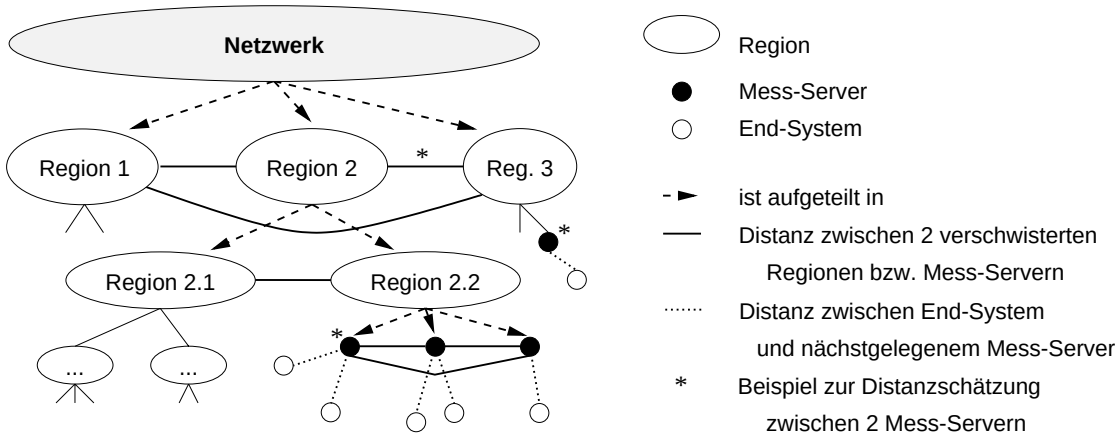


Abbildung 3.2: Repräsentation einer Netzwerk–Distanzkarte

Distanzen zwischen beliebigen Endsystemen können wie folgt aus dieser Darstellung abgeleitet werden. Sie werden ebenfalls mit $\|*,*\|$ bezeichnet. Zur Verbesserung der Übersichtlichkeit nachfolgender Formeln wird die Funktion Λ benutzt, die jedem Knoten des Netzwerkbaumes seinen Elternknoten zuweist. Λ^n bezeichnet dementsprechend den Vorfahren der n-ten Generation. Die Distanz zwischen zwei Mess–Servern m_1 und m_2 wird durch die Distanz zwischen den jeweiligen Kindknoten des speziellsten gemeinsamen Vorfahren von m_1 und m_2 geschätzt (siehe auch das mit einem Stern * markierte Beispiel in Abbildung 3.2):

$$\|m_1, m_2\| := \|\Lambda^i(m_1), \Lambda^j(m_2)\|, \qquad \text{so dass } i,j \text{ minimal und } \Lambda^{i+1}(m_1) = \Lambda^{j+1}(m_2).$$

Hierauf aufbauend lässt sich nun die Distanzschätzung für beliebigen Paare von Endsystemen e_1, e_2 definieren:

$$\|e_1, e_2\| := \text{worst}\{\|\Lambda(e_1), \Lambda(e_2)\|,\ \Delta(e_1, \Lambda(e_1)),\ \Delta(e_2, \Lambda(e_2))\},$$

wobei Δ wieder die durch Messung gewonnene Distanz zwischen Endsystemen und Mess–Servern beschreibt und *worst* eine Funktion ist, die je nach zugrundeliegendem Distanzmaß das Schlechteste seiner Argumente auswählt. Das erste Argument ist die aus dem Netzwerkbaum abgeleitete Distanz zwischen den beiden Mess–Servern, denen die Endsysteme e_1 und e_2 zugeordnet sind. Das zweite bzw. dritte Argument ist die aus einer Messung gewonnene Distanz zwischen dem Endsystem e_1 (bzw. e_2) und seinem zugeordneten Mess–Server. Bei additiven Distanzmaßen wie der Knotenzahl oder der Umlaufzeit wählt *worst* einfach das maximale Element. Bei Distanzmaßen wie der Bandbreite wird dagegen das Minimum ausgewählt. Die Anwendung der *worst*–Funktion ermöglicht eine genauere Schätzung für Endsysteme, die demselben Mess–Server zugeordnet sind, und für

solche, deren zugeordneter Mess–Server weit entfernt liegt, die also von der Distanzkarte noch nicht so gut erfasst werden.

Die Ableitung beliebiger Distanzen benötigt einen in der Baumtiefe linearen Zeitaufwand. Sei k der Grad[5] des Regionenbaumes. Dann hat ein vollständig ausgeglichener[6] Regionenbaum die Tiefe $t = \lceil \log_k |\mathcal{M}| \rceil$, wobei die Tiefe der Wurzel als 0 definiert ist. Ein maximal unausgeglichener, voller[6] Baum hat die Tiefe $t = \lceil (|\mathcal{M}| - 1)/(k-1) \rceil$.

Der Speicherbedarf eines Netzwerkbaumes kann wie folgt geschätzt werden: Unter der Annahme eines vollen Regionenbaumes, besteht dieser aus $\lceil (|\mathcal{M}| * k - 1)/(k-1) \rceil$ Knoten. Da jeder Knoten Distanzwerte zu seinen Geschwistern speichert (höchstens $k-1$) beträgt der gesamte Speicherbedarf eines Netzwerkbaumes

$$O\left(|\mathcal{E}| + \lceil \frac{|\mathcal{M}| * k - 1}{k-1} \rceil * (k-1)\right) = O\left(|\mathcal{E}| + |\mathcal{M}| * k\right).$$

Hierbei bestimmt der Parameter k den Kompromiss zwischen der Genauigkeit einer Distanzkarte und ihrem Speicherbedarf. Der Extremfall $k = \mathcal{M}$ entspricht der exakten Abbildung einer Netzwerksicht, lediglich die Endsysteme sind nur mit ihrem jeweils zugeordneten Mess–Server verbunden. Will man beispielsweise ein Netzwerk mit 10.000 Mess–Servern und einer Million Endsysteme durch einem Baum mit $k = 10$ repräsentieren, so beträgt der hierzu erforderliche Speicherbedarf $1,1 * 10^6$ Elemente. Die komplette Speicherung einer Netzwerksicht bräuchte dagegen $|\mathcal{M}| * |\mathcal{E}| = 10^{10}$ Elemente.

Regionen–Repräsentant

Wie schon in Abschnitt 3.3.3 angesprochen, dienen Repräsentanten von Regionen einer schnellen Einordnung von Endsystemen und Mess–Servern in einen Regionenbaum. Die Auswahl eines Repräsentanten sollte dazu zwei in gegenseitigem Konflikt stehenden Zielen genügen. Zum einen sollte er wirklich „repräsentativ“ sein, d.h. seine Distanz zu Endsystemen außerhalb seiner Region sollte gleich der mittleren Distanz eines beliebigen Endsystems seiner Region sein. Zum anderen sollte seine Bestimmung nicht eine Unmenge von zusätzlichen Netzwerkmessungen erfordern. Letzteres ist insbesondere für die Aktualisierung von Netzwerkbäumen wichtig.

Daher wurde als Repräsentant einer Region (eines Baumknotens) das *Regionenzentrum* gewählt, das sich aus dem Kindknoten ergibt, dessen maximaler Abstand zu einem Geschwisterknoten minimal ist. Für eine Menge von Kindknoten K und Distanzen $\Delta(*,*)$

[5] Der Grad eines Baumes ist definiert als die maximale Zahl an Kindknoten, die ein Baumknoten hat.

[6] Ein Baum vom Grad k heißt *voll*, wenn alle inneren Baumknoten genau k Kindknoten haben. Ein Baum heißt *vollständig ausgeglichen*, wenn er voll ist und alle Blattknoten dieselbe Tiefe haben.

zwischen diesen ist das Zentrum z also definiert als

$$z \in K \quad \text{, mit } \max_{k \in K, k \neq z} \Delta(z,k) = \min_{k \in K} \max_{h \in K, k \neq h} \Delta(k,h).$$

Ist der Kindknoten z ein Mess–Server, so wird dieser auch als Repräsentant gewählt. Andernfalls wird rekursiv das Regionenzentrum des Kindknotens z berechnet und auch als Repräsentant der aktuellen Region gewählt. Damit wird sicher gestellt, dass als Repräsentant in jedem Fall ein Mess–Server ausgewählt wird.

Damit kann die Bestimmung des Repräsentanten allein anhand der im Netzwerkbaum bereits bekannten Distanzen erfolgen, womit insbesondere auch bei Baumaktualisierungen kein Mehraufwand entsteht. Daneben führt diese Festlegung zu Repräsentanten, die durch ihre kurzen Regionen–internen Distanzen charakterisiert sind. Es lässt sich daher erwarten, dass Repräsentanten nicht überdurchschnittlich schlecht an Endsysteme außerhalb ihrer Region angebunden sind, wodurch sich ihre Eignung für den Einfügeprozess von Mess–Servern ergibt (vgl. mit den Algorithmen zur gemischten Ballung, Abschnitt 3.4.2) und ebenso zur Integration und Aktualisierung einfacher Endsysteme (siehe Abschnitte 3.4.3 und 3.4.4).

3.4 Algorithmen zur Kartenerstellung und -anpassung

3.4.1 Ballungskriterien und -berechnug

Das erste Problem bei der Berechnung eines Netzwerkbaumes besteht in der Bestimmung einer effektiven Ballung (engl. *clustering*) der verfügbaren Mess–Server. [Francis et al., 1999] diskutieren hierzu die Ballung gemäß autonomer Systeme, bei der jedes autonome System eine Netzwerkregion definiert und Endsysteme dementsprechend diesen Regionen zugeordnet werden. Allerdings verwerfen sie diesen Ansatz, da ein autonomes System sich im Sinne der Umlaufzeit über sehr große Distanzen erstrecken kann während gleichzeitig Endsysteme aus verschiedenen autonomen Systemen nahe beieinander liegen können.

Entsprechend der in Abschnitt 3.3.4 vorgestellten Methode zur Distanzableitung wäre das optimale Ballungskriterium das Kriterium *Min-k-Fehler*. Dieses minimiert den mittleren Schätzfehler, also den Ausdruck

$$\sum_{\substack{i,j \in [1..k] \\ i<j}} \sum_{\substack{x \in B_i \\ y \in B_j}} |\Delta(x,y) - \Delta(B_i, B_j)|.$$

Hierbei bezeichnet k die Zahl der gewünschten Ballungen. Dieser Parameter wird a priori festgelegt, da er die Genauigkeit eines Regionenbaumes und damit den Aufwand zu dessen Erstellung bestimmt. Mit B_i wird eine einzelne Ballung und mit $\Delta(B_i, B_j)$ der mittleren Abstand zwischen Elementen der beiden Ballungen B_i und B_j bezeichnet. Leider ist uns kein polynomial berechenbarer Lösungweg ja nicht einmal eine polynomiale Approximierung dieses Optimierungsproblemes bekannt.

Daher wurden heuristische Ballungskriterien untersucht, die zwei unterschiedlichen Strategien folgen. Die eine versucht Ballungen zu finden, die sich durch ihre geringe Ausdehnung auszeichnen, für die also die Distanzen zwischen ihren Mitgliedern möglichst klein sind. Die andere sucht nach Ballungen, die möglichst weit voneinander entfernt sind, bei denen der Abstand zwischen Mitgliedern unterschiedlicher Ballungen also möglichst groß ist. Bei beiden Ansätzen hat man zudem die Wahl, entweder den Extremwert oder den Durchschnittswert zu optimieren. Mit ersterem optimiert man den schlechtest möglichen Fall (den größten bzw. kleinsten Abstand in bzw. zwischen Ballungen), mit letzterem den durchschnittlich auftretenden Fall. In der Kombination erhält man also vier Ballungskriterien, die im Folgenden diskutiert werden und deren Optimierungsresultate in Tabelle 3.1 zusammengefasst sind. Zu jedem Kriterium werden sein Name, die Zielfunktion (die entweder das Minimum oder das Maximum des nachfolgenden Ausdrucks über allen möglichen Ballungen sucht) sowie die Berechenbarkeits- bzw. Approximationsresultate angegeben. Der Parameter n bezeichnet immer die Zahl der zu ballenden Objekte (z.B. Mess–Server). Zur Kennzeichnung eines Kriteriums wird eine dreiteilige Kurzschreibweise eingeführt, die mit *Max* oder *Min* beginnt, je nachdem ob der Optimierungsausdruck zu maximieren oder minimieren ist, vom Formparameter k gefolgt wird, der für die Zahl der gewünschten Ballungen steht und ggf. auch durch einen konkreten Zahlenwert ersetzt wird, und mit einem Bezeichner schließt (*Trennung*, *Ausdehnung*, *Schnitt* oder *Ausdehnungssumme*), der das jeweilige Kriterium beschreibt.

Von allen vier Kriterien ist allein das Kriterium *Max-k-Trennung* nicht NP-hart$^{\uparrow}$. Abbildung 3.3 auf Seite 48 gibt einen Algorithmus zu seiner Berechnung an, der eine Laufzeit von $O(n^2 * \log n)$ hat. Hierin bezeichnet S die Menge der zu ballenden Objekte. Interpretiert man die zu ballenden Objekte und die Distanzen zwischen ihnen als vollständig verknüpften Graphen, so ist der in [Theilmann & Rothermel, 1999b] beschriebene Algorithmus prinzipiell identisch zur Berechnung eines minimalen Spannbaumes. Der Unterschied besteht lediglich darin, dass er bereits abgebrochen wird, wenn ein Wald von k minimalen Teilspannbäumen erreicht ist.

[7] Unter Annahme der Dreiecksungleichung.

Modus \ Ziel	maximale Trennung	minimale Ausdehnung
Optimierung nach Extremwert	**Max-k-Trennung** $\max \min_{\substack{i,j\in[1..k],i<j \\ x\in B_i,y\in B_j}} \Delta(x,y)$ berechenbar in $O(n^2 * \log n)$	**Min-k-Ausdehnung** $\min \max_{\substack{i\in[1..k] \\ x,y\in B_i}} \Delta(x,y)$ NP-hart$^{\uparrow}$; optimale Approximation[7] mit Faktor 2 berechenbar in $O(n * k)$
Optimierung nach Durchschnitt	**Max-k-Schnitt** $\max \sum_{\substack{i,j\in[1..k] \\ i<j}} \sum_{\substack{x\in B_i \\ y\in B_j}} \Delta(x,y).$ NP-hart; bester bekannter Approximationsfaktor ist $1/(1-1/k+2\ln k/k^2)$	**Min-k-Ausdehnugssumme** $\min \sum_{i=1}^{k} \sum_{x,y\in B_i} \Delta(x,y)$ NP-hart; beste bekannte Approximation[7] mit Faktor 2 berechenbar in $O(n^k)$

Tabelle 3.1: Optimierung einiger heuristischer Ballungskriterien

Im Einzelnen arbeitet der Algorithmus wie folgt: Mit jedem Schleifendurchlauf wird das Objektpaar mit der geringsten Distanz ausgewählt (Schritte 3–4). Um diese Auswahl möglichst effizient zu gestalten, werden alle Distanzen in Schritt 1 vorsortiert. Die ausgewählten Objekte können, da sie die geringste aller Distanzen haben, unmöglich verschiedenen Ballungen angehören und werden darum zusammengefasst (Schritt 5). Die Distanzen dieser Zusammenfassung zu anderen Objekten bzw. Zusammenfassungen werden entsprechend dem Kriterium der minimalen Trennung als Minimum der Distanzen der einzelnen Objekte bestimmt (Schritt 6). Am Ende des Verfahrens repräsentieren die k Objekte der Menge S die k trennungsmaximalen Ballungen. Protokolliert man die Verschmelzungen aus Schritt 5 so lassen sich hieraus direkt die einzelnen Ballungen ableiten.

Die beste bekannte Approximation des Kriteriums *Max-k-Schnitt* erreicht einen Approximationsfaktor[8] von $1/(1-1/k+2\ln k/k^2)$ [Frieze & Jerrum, 1995], was gegenüber dem Approximationsfaktor einer Zufallsballung von $1/(1-1/k)$ keine wesentliche Verbesserung

[8]Der Approximationsfaktor gibt an, um welchen Faktor die Güte der approximativ bestimmten Lösung maximal von der Güte der optimalen Lösung abweicht. Gütemaß ist dabei der zu optimierende (maximierende oder minimierende) Ausdruck.

(1) sortiere Distanzen für Paare (x, y) aus S
(2) **while** $(|S| > k)$
(3) $D := \min\{\Delta(x, y) | x, y \in S\}$
(4) wähle x, y mit $\Delta(x, y) = D$
(5) verschmelze x, y zu $\tilde{x}$:
$S := S \setminus \{x, y\} \cup \{\tilde{x}\}$
(6) bestimme Distanzen zu $\tilde{x}$: $\forall z \in S \setminus \{\tilde{x}\}$:
$\Delta(\tilde{x}, z) := \min\{\Delta(x, z), \Delta(y, z)\}$
(7) **endwhile**

Abbildung 3.3: Algorithmus zur Berechnung einer *Max-k-Trennung*

darstellt.

Approximationsalgorithmen, die auf den Kriterien *Min-k-Ausdehnung* oder *Min-k-Ausdehnungssumme* basieren, setzen die Gültigkeit der Dreiecksungleichung (siehe Abschnitt 3.1.2) voraus. Während das erste Kriterium für den bestmöglichen Approximationsfaktor von 2 mit $O(n*k)$ noch recht effizient berechnet werden [González, 1985], ist der Aufwand von $O(n^k)$ zur Berechnung des zweiten Kriteriums [Guttmann-Beck & Hassin, 1998] für die bei Netzwerk–Distanzkarten relevanten Problemgrößen inakzeptabel.

Abbildung 3.4 beschreibt kurz den von [González, 1985] entwickelten *farthest point* Algorithmus zur Berechnung einer *Min-k-Ausdehnung*. Die hierin berechnete Menge B umfasst am Ende gerade k Objekte, die untereinander maximalen Abstand haben. Für alle nicht in B enthaltenen Objekte beschreibt die Funktion *zuordnung* das nächstgelegene Objekt aus B und *dist* die zugehörige Distanz. Am Ende des Verfahrens repräsentieren die Objekte aus B die k ausdehnungsminimalen Ballungen und aus der Funktion *zuordnung* können die Elemente einer jeden Ballung gewonnen werden.

Die Experimente in Kapitel 5 wurden mit den beiden nach Extremwerten optimierenden Ballungskriterien durchgeführt, *Max-k-Trennung* und *Min-k-Ausdehnung*, da nur diese mit akzeptablem Aufwand und akzeptabler Güte berechnet werden können. Optimierungen anhand von Extremwerten können bisweilen zu überraschenden Ergebnissen führen, da das gesamte Ergebnis durch ein einzelnes Datum bestimmt sein kann. Um derartige Effekte zu illustrieren, zeigt Abbildung 3.5 zwei Beispiele. Distanzen sind hierbei durch die planare Geometrie definiert. Die bemerkenswerten Effekte sind, dass das Kriterium *Max-k-Trennung* zu sehr schlecht balancierten Ballungen führen kann, während das Kriterium *Min-k-Ausdehnung* in sehr nahe beieinander liegenden Objekten resultieren kann, die aber

```
(1)  B := ∅
(2)  ∀x ∈ S : dist(x) := ∞
(3)  while (|B| < k)
(4)      D := max{dist(x)|x ∈ S \ B}
(5)      wähle x̃ ∈ S \ B mit dist(x̃) = D
(6)      B := B ∪ {x̃}
(7)      forall (x inS \ B})
(8)          if (Δ(x, x̃) < dist(x))
(9)              dist(x) := Δ(x, x̃)
(10)             zuordnung(x) := x̃
(11) endwhile
```

Abbildung 3.4: Algorithmus zur Berechnung einer *Min-k-Ausdehnung*

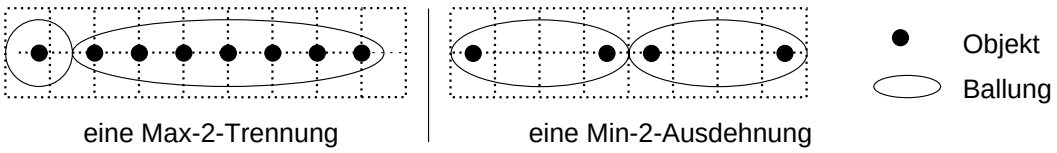


Abbildung 3.5: Zwei extreme Ballungsergebnisse

verschiedenen Ballungen angehören.

3.4.2 Berechnung des Regionenbaumes

Die Berechnung eines kompletten Regionenbaumes erfolgt durch die rekursiv wiederholte Anwendung des gewählten Ballungsverfahrens, d.h. zunächst werden die verfügbaren Mess–Server in grobe Regionen eingeteilt (geballt), und danach wird jede dieser Regionen weiter verfeinert. Wie bereits in Abschnitt 3.3.3 erwähnt werden hierzu aus Effizienzgründen nicht alle Distanzen zwischen allen Mess–Servern bestimmt, sondern ein einzelner Ballungsprozess wird jeweils nur auf einer Teilmenge der verfügbaren Mess–Server durchgeführt. Die anderen Server werden danach passend einsortiert. Dieser *gemischte Algorithmus* ist durch 2 Parameter gekennzeichnet: Der *Baumgrad* k bestimmt die Zahl der Ballungen in die eine Menge von Mess–Servern jeweils aufgeteilt werden sollen. Die *Selektivität* s bestimmt die Anzahl der für einen einzelnen Ballungsprozess auszuwählenden Mess–Server. Eine Selektivität von $s = |\mathcal{M}|$ bewirkt eine *optimale Ballung*, die aber auch die maximale Zahl an Distanzmessungen erfordert. Kleinere Werte für s führen zu suboptimalen Ballungen. Man beachte, dass das Attribut „optimal“ hier relativ zum gewählten

procedure baumberechnung (Mess–Server M, Baumgrad k, Selektivität s)
(1) wähle Menge S von s Mess-Servern zufällig aus M
(2) messe Distanz zwischen allen Elementen aus S
(3) berechne Ballung von S in k Regionen $B_1, \ldots, B_k$
(4) bestimme Distanzen zwischen Regionen als Durchschnittswert
(5) berechne (vorläufige) Regionenzentren $r_1, \ldots, r_k$
(6) **forall** ($m \in M \setminus S$)
(7) messe Distanz $\Delta(m, r_i)$ zu allen Regionenzentren r_i
(8) ordne m der Region mit nächstgelegenem Zentrum zu
(9) aktualisiere Distanzen zwischen Regionen
(10) **forall** (Regionen B_i mit mehr als k Mess–Servern M_i)
(11) setze Ballung rekursiv fort: baumberechnung (M_i, k, s)
(12) **forall** (Regionen B_i mit höchstens k Mess–Servern M_i)
(13) messe noch unbekannte Distanzen zwischen Servern aus M_i und erweitere Regionenbaum
(14) berechne endgültige Regionenzentren für alle Regionen B_i

Abbildung 3.6: Algorithmus zur Berechnung eines Regionenbaumes

Ballungskriterium zu sehen ist. Nimmt man ein Kriterium als gegeben hin, so führt die Festlegung $s = |\mathcal{M}|$ zur relativ bestmöglichen Ballung. Das optimale Ballungskriterium ist natürlich weiterhin das *Min-k-Fehler*–Kriterium aus Abschnitt 3.4.1.

Abbildung 3.6 beschreibt den kompletten Algorithmus zur Berechnung eines Regionenbaumes. Man beachte, dass die Berechnung der vorläufigen Regionenzentren (Schritt 5) komplett anhand der in Schritt 2 gemachten Messungen erfolgt. Gleiches gilt für die Berechnung der endgültigen Regionenzentren (Schritt 14), die anhand der Distanzen zwischen den jeweiligen Teilregionen bestimmt werden können. Während die Zentrumsbestimmung in Schritt 5 direkt auf einer Menge von Mess–Servern durchgeführt werden kann, kommt in Schritt 14 die rekursive Zentrumsberechnung zum Tragen (vgl. mit Abschnitt 3.3.4 auf Seite 44).

Die Einordnung eines Mess–Servers in eine Region (Schritt 8) genügt im Rahmen der gemachten Messungen zu Regionenzentren sowohl dem Kriterium der Trennungsmaximierung als auch dem der Ausdehnungsminimierung.

Die Aktualisierung von Distanzen zwischen Regionen (Schritt 9) betrifft alle Distanzen zwischen der Region, in die Mess–Server m eingeordnet wurde, und allen anderen,

verschwisterten Regionen. Da die Distanz zwischen Mess–Server m und allen Regionenzentren gemessen wurde, kann die Distanzschätzung zwischen diesen Regionen verfeinert werden. Hierzu bezeichne B die Region, in die m eingeordnet wurde, und d_i die Distanz zwischen m und dem Regionenzentrum von Region B_i. Die Verfeinerung der Distanzschätzung zwischen Region B und Region B_i ($B_i \neq B$) erfolgt dann nach $\|B, B_i\| := \|B, B_i\| + (d_i - \|B, B_i\|) * \frac{1}{w}$, wobei der Gewichtungsfaktor w die Zahl der Messungen angibt, die bereits zwischen den Regionen B und B_i durchgeführt wurden. Damit wird gewährleistet, dass jede Distanzmessung den gleichen Beitrag zur Regionendistanzschätzung erbringt.

Der Baumgrad k bestimmt die Granularität des Regionenbaumes und damit die Genauigkeit der Distanzschätzung zwischen einzelnen Mess–Servern. Die Selektivität s bestimmt die Wahrscheinlichkeit, dass die zufällig gewählte Menge S (Schritt 1) die Berechnung von repräsentativen Ballungen ermöglicht, in die sich die zusätzlichen Mess–Server sinnvoll einordnen lassen.

Messaufwand

Bei der Analyse des Messaufwandes werden im Folgenden vollständig ausgeglichene Regionenbäume mit $|\mathcal{M}| = k^t$ (t bezeichnet wieder die Baumtiefe) und $s = k^n$ ($0 < n \leq t$) angenommen.

Um die erste Ebene eines Regionenbaumes zu berechnen, muss zunächst eine optimale Ballung über einer Teilmenge S der Mess–Server berechnet werden. Nutzt man die Symmetrieeigenschaft des Distanzmaßes aus, so sind hierzu $s*(s-1)/2$ einzelne Distanzmessungen erforderlich. Die nachträgliche Einordnung der übrigen Mess–Server erfordert $k*(|\mathcal{M}|-s)$ Messungen. Auf tieferen Baumebenen entsteht der analoge Aufwand in jedem Knoten, allerdings für entsprechend kleinere Mengen von Mess–Servern. Als Optimierung können auf tieferen Baumebenen jedoch Messungen, die auf höheren Ebenen erfolgten, wieder verwendet werden. Führt ein Ballungsverfahren auf Ebene i zu gleichmäßig großen Ballungen, so sind für alle k Ballungsverfahren auf Ebene $i+1$ jeweils die Distanzen zwischen s/k Mess–Servern wieder verwendbar. Ab Ebene $(t-n+1)$ lassen sich alle benötigten Distanzen aus den übergeordneten Ballungsverfahren gewinnen.

Damit beträgt der Gesamtaufwand an Distanzmessungen zur Berechnung eines Regionenbaumes

$$\frac{s*(s-1)}{2} + k*(|M|-s) + \sum_{i=1}^{t-n} k^i * \left(\frac{s*(s-1)}{2} + k*\left(\frac{|M|}{k^i} - s \right) - \frac{s/k*(s/k-1)}{2} \right),$$

was sich vereinfachen lässt zu

$$\sum_{i=0}^{t-n} k^i * \frac{s*(s-1)}{2} - \sum_{i=1}^{t-n} k^i * \frac{s/k*(s/k-1)}{2} + (t-n+1)*k*(|M|-s).$$

In dieser Summe dominiert der Aufwand des ersten (positiven) Summanden sowohl den zweiten (negativen) als auch den dritten Summanden. Damit kann der Gesamtausdruck abgeschätzt werden zu

$$O(k^{t-n} * s^2) \; = \; O(|\mathcal{M}| * s).$$

Den Nutzen des gemischten Ballungsverfahrens mag folgendes, exakt durchgerechnetes Zahlenbeispiel verdeutlichen: Will man für 1000 Mess–Server einen Regionenbaum mit $k = 10$ und $s = 100$ berechnen, so sind hierzu 62100 Distanz-Messungen erforderlich, vorausgesetzt der entstehende Baum ist vollständig ausgeglichen. Im Gegensatz dazu bräuchte eine komplette Ausmessung der Distanzen zwischen allen Mess–Servern 499500 Messungen, was um den Faktor 8 größer ist.

Bei einem maximal unausgeglichenen, vollen Baum werden bei jedem Ballungsprozess alle bis auf $(k-1)$ Mess–Server derselben Ballung zugeordnet. Damit sind in diesem schlechtesten Fall $\frac{|\mathcal{M}|*(|\mathcal{M}|-1)}{2} = O(|\mathcal{M}|^2)$ Distanzmessungen erforderlich, es ergibt sich also keinerlei Vorteil gegenüber dem optimalen Ballungsverfahren mit $s = |\mathcal{M}|$.

Berechnungskomplexität

Die Berechnung eines kompletten Regionenbaumes besteht in der rekursiv wiederholten Anwendung des gemischten Ballungsverfahrens. Die Komplexität des ersten (nichtrekursiven) Durchlaufs des gemischten Ballungungsverfahrens setzt sich aus dem Aufwand der folgenden drei (rechenintensivsten) Schritte zusammen: dem Aufwand $O(s^2)$ zur Messung aller Distanzen zwischen Elementen der Menge S (Schritt 2), dem Ausführungsaufwand der eigentlichen Ballung (Schritt 3) sowie dem Aufwand $O((|\mathcal{M}|-s)*k)$ zur Einordnung der weiteren Mess–Server. Ignoriert man den Faktor $\log n$ bei der Berechnungskomplexität des Kriteriums *Max-k-Trennung*, so ist für beide in Abschnitt 3.4.1 ausgewählten Ballungskriterien ihr Berechnungsaufwand bereits durch den Aufwand $O(s^2)$ aus Schritt 2 erfasst. Damit beträgt der Gesamtaufwand des ersten Durchlaufes des gemischten Ballungungsverfahrens $O(s^2+(|\mathcal{M}|-s)*k)$. Der Aufwand zur kompletten Berechnung eines vollständig ausgeglichenen Regionenbaumes beträgt dann

$$O\left(\sum_{i=0}^{t-n} k^i * \left(s^2 + k*(\frac{|M|}{k^i} - s)\right) + \sum_{i=t-n+1}^{t-1} k^i * \left(\frac{|M|}{k^i}\right)^2\right)$$

und mit ($|M| = k^t$, $s = k^n$)

$$\begin{aligned} & O\left(\sum_{i=0}^{t-n}\left(k^i * (k^{2n} - k^{n+1}) + k^{t+1}\right) + \sum_{i=t-n+1}^{t-1} k^{2t-i}\right) \\ = \; & O\left(k^{t-n} * (k^{2n} - k^{n+1}) + (t-n+1) * k^{t+1} + k^{t-n+1}\right) \\ = \; & O\left(k^{t+n} + (t-n+1) * k^{t+1}\right) \\ \leq \; & O(|\mathcal{M}| * s * (\log_k |\mathcal{M}| - \log_k s)). \end{aligned}$$

Dieser Aufwand hängt in der Praxis allerdings wesentlich davon ab, ob ein Ballungsprozess zu zahlenmäßig ausgeglichenen Ballungen führt. Je unausgeglichener die entstehenden Ballungen sind, desto öfter muss ein Ballungsprozess auf großen Objektmengen angewandt werden (vgl. Abbildung 3.5 auf Seite 49). Der schlechtest mögliche Fall ist wieder dadurch gekennzeichnet, dass bei jedem Ballungsprozess alle bis auf $(k-1)$ Mess–Server derselben Ballung zugeordnet werden. Es wird also auf jeder Baumebene i nur ein Ballungsprozess durchgeführt, dieser allerdings über $(|\mathcal{M}| - i*(k-1))$ Mess–Servern. Ein solcher Baum hat die Tiefe $t = \lceil \frac{|\mathcal{M}|-1}{k-1} \rceil$. Vernachlässigt man die Situation, dass auf unteren Baumebenen Ballungen mit weniger als s Elementen durchgeführt werden, und nimmt auch hier Mengen von s Elementen an, so beträgt der Gesamtaufwand im schlechtest möglichen Fall

$$\begin{aligned} & O\left(\sum_{i=0}^{t-1} s^2 + k * (|\mathcal{M}| - i * (k-1) - s)\right) \\ = \; & O\left(t * (s^2 - k * s) + k * t * \left(|\mathcal{M}| - \frac{t-1}{2} * (k-1)\right)\right) \\ & \text{und mit } t = \lceil \tfrac{|\mathcal{M}|-1}{k-1} \rceil \\ = \; & O\left(|\mathcal{M}| * \frac{s^2}{k} + |\mathcal{M}|^2\right). \end{aligned}$$

3.4.3 Integration einfacher Endsysteme

Die Integration einfacher Endsysteme in einen Regionenbaum erfolgt ähnlich zu dem im vorigen Abschnitt beschriebenen Einfügevorgang für Mess–Server. Auf oberster Baumebene werden hierzu die Distanzen zwischen dem Endsystem und den jeweiligen Regionenzentren gemessen. Der Prozess setzt sich rekursiv für die Region mit dem nächstgelegenen Zentrum fort. Damit erfordert die Integration eines einfachen Endsystems in einen vollständig ausgeglichenen Regionenbaum der Tiefe t und vom Grad k insgesamt $t * k$ Distanzmessungen.

Um die Integration von Endsystemen gegenüber einzelnen ungenau gemessenen Distanzen robuster zu gestalten, wurde obiger Algorithmus erweitert. Dabei wird für ein Endsystem der Regionenbaum in einer Tiefensuche durchlaufen. Diese wird jedoch abgebrochen, wenn der momentan betrachtete Baumknoten bzw. sein Repräsentant eine signifikant größere Distanz zum Endsystem hat als der bisher gefundene, nächstgelegene Mess–Server. Die Signifikanz des Distanzunterschiedes wird durch die Wahl einer *Ähnlichkeitsschranke* festgelegt. Die Distanz zu einer Region darf maximal um diesen Faktor größer sein als die bisher optimale Distanz. Wird die Ähnlichkeitsschranke auf $1 - \epsilon$ gesetzt, so wird bei der Integration eines Endsystems genau ein Pfad im Regionenbaum verfolgt. Ein Wert von nahezu Unendlich führt zu einem kompletten Durchlauf des gesamten Baumes.

Der detaillierte Algorithmus ist in Abbildung 3.7 wiedergegeben. Zur Vereinfachung der Darstellung bezeichnet hierin δ eine Funktion, die zu jedem Endsystem die optimale (bisher gefundene) Zuordnungsdistanz zu einem Mess–Server liefert. Man beachte, dass

```
procedure integration (Endsystem e, Region r, Ähnlichkeitsschranke a)
(1)   Liste l := {Teilregionen von r}
(2)   messe Distanzen zwischen e und Repräsentanten aller Regionen aus l
(3)   sortiere Elemente aus l aufsteigend nach Distanzen zu e
(4)   i := 1
(5)   d_1 := Distanz zwischen e und (nächstgelegener) Region R_1 ∈ l
(6)   loop
(7)       if (R_i ist kein Mess–Server)
(8)           setze Integration rekursiv fort: integration(e, R_i, a)
(9)       else if (d_i < δ(e))
(10)          aktualisiere Zuweisung: δ(e) := d_i
(11)      i := i + 1
(12)      d_i := Distanz zwischen e und Region R_i
(13)  until (d_i > a * δ(e))
```

Abbildung 3.7: Algorithmus zur Integration einfacher Endsysteme

die Teilregionen im Extremfall auch einzelne Mess–Server sein können. Zur Vereinfachung der Darstellung wurde auf diese Unterscheidung verzichtet. Als Repräsentant eines Mess–Servers (Schritt 2) wird natürlich der Mess–Server selbst genommen. Die Schleife (Schritte 6–13) realisiert das Überprüfen einer Menge von verschwisterten Regionen in der Reihenfolge ihres Abstandes zum Endsystem e. Ggf. wird in Schritt 8 die Integration rekursiv für eine Teilregion verfeinert. Die Schleife wird mit der Bedingung in Schritt

13 beendet, falls die Distanz zur nächsten Region signifikant schlechter als das bisherige Optimum ist.

Neben diesem Verfahren mit Tiefensuchstrategie wurden noch zwei weitere Verfahren ausprobiert. Das erste verfolgt eine Strategie der Breitensuche. Hierbei werden mehrere Pfade im Regionenbaum parallel verfolgt, bis sich einer als der bessere erweist. Genau wie im obigen Verfahren mit Tiefensuchstrategie bestimmt eine *Ähnlichkeitsschranke* den Abbruch der Breitensuche. Da sich dieses Verfahren in den Experimenten in Kapitel 5.2 als wesentlich schlechter erweisen wird, soll es an dieser Stelle nicht detaillierter beschrieben werden. Schließlich wurde auch noch ein Zufallsalgorithmus erprobt, bei dem zufällig einige Mess–Server ausgewählt werden, die Distanz zwischen diesen und dem Endsystem bestimmt wird und das Endsystem dann dem nächstgelegenen Server zugewiesen wird.

Die Evaluierung der drei Verfahren bezüglich der Qualität der Mess–Server–Zuweisung und bezüglich des Messaufwands erfolgt empirisch in Kapitel 5.2.

3.4.4 Aktualisierung von Distanzkarten

Da sich Netzwerkdistanzen mit der Zeit ändern (durch Änderung der aktuellen Last oder durch Veränderungen in der Topologie), müssen einmal erstellte Karten permanent an diese Veränderungen angepasst werden. Die triviale Lösung dieses Problems besteht in der regelmäßigen, kompletten Neuberechnung der Karte, allerdings wird hiermit eine unnötig hohe Belastung des Netzwerkes erzeugt. Daher wurden Verfahren entwickelt, die das bereits in einer Karte steckende Wissen ausnutzen, um diese Karte möglichst kostengünstig an Veränderungen anzupassen.

Die Festlegung, wie oft und in welchem Umfang Aktualisierungen durchgeführt werden sollen, ist nicht Gegenstand dieser Arbeit. Einige Ideen hierzu werden jedoch in Abschnitt 3.6.2 vorgestellt.

Anpassung des Regionenbaumes

Regionenbäume werden durch die erneute Ballung von einzelnen Teilbäumen aktualisiert. Der Umfang dieser Aktualisierung wird durch die Baumtiefe t festgelegt, ab der die erneute Ballung durchgeführt werden soll. Hierzu werden für jede Region auf Tiefe t die in ihr enthaltenen Mess–Server gesammelt und erneut dem Prozess zur Berechnung eines Regionenbaumes (in diesem Fall eines Teilbaumes) unterzogen. Eine erneute Ballung auf Tiefe 0 ist identisch zu einer kompletten Neuberechnung des Regionenbaumes. Die

Zuordnung von Endsystemen zu Mess–Servern bleibt von der Anpassung des Regionenbaumes unberührt. Der komplette Algorithmus ist in Abbildung 3.8 wiedergegeben. Man

procedure baumanpassung (Tiefe t, Selektivität s)
(1) **forall** (Baumknoten K mit Tiefe t)
(2) $M :=$ Menge aller Mess-Server in K
(3) Teilbaum $B :=$ baumberechnung (M, k, s)
(4) ersetze Teilbaum mit Wurzel K durch Teilbaum B
(5) passe ggf. Regionenzentren in Oberknoten mit Tiefe $< t$ an

Abbildung 3.8: Algorithmus zur Anpassung des Regionenbaumes

beachte, dass die Neuballung auf unteren Ebenen auch zu einer Änderung von Regionenzentren auf höheren Ebene führen kann und zwar genau dann, wenn eine höhergelegene Region als Zentrum ein Zentrum einer tiefergelegenen Region hat, welches sich im Zuge der Baumanpassung verändert hat. Dieses wird in Schritt 5 überprüft.

Der Aufwand zur Neuballung eines vollständig ausgeglichenen Regionenbaumes auf Tiefe t beträgt k^t mal den Aufwand zur Berechnung eines Baumes für eine Menge von $|\mathcal{M}|/k^t$ Mess–Servern. Setzt man das Beispiel aus Abschnitt 3.4.2 fort, nimmt also 1000 Mess–Server und einen Baumgrad von $k = 10$, so erfordert eine optimale Neuballung auf Tiefe 1 49500 Distanzmessungen was um den Faktor 10 niedriger ist als der Aufwand zur kompletten Neuberechnung. Dieser Unterschied wird dagegen für suboptimale Baumberechnungen mit verhältnismäßig kleinen Selektivitätswerten recht gering. Wie in Abschnitt 3.4.2 gesehen erfordert z.B. eine komplette Neuberechnung für $s = 100$ lediglich 62100 Distanzmessungen.

Mögliche Verbesserungen dieses sehr simplen Verfahrens zur Anpassung von Regionenbäumen werden in Abschnitt 3.6 diskutiert.

Anpassung der Zuordnung von Endsystemen

Neben dem Regionenbaum muss natürlich auch die Zuordnung von Endsystemen zu Mess–Servern aktualisiert werden. Dem hierzu benutzten Verfahren liegt die Heuristik zugrunde, nach der die Umordnung eines Endsystems zu einem vorher schon in der Nähe liegendem Mess–Server wahrscheinlicher ist als zu einem weit entfernten.

Das Verfahren vereinigt zwei prinzipielle Vorgehensweisen: Nach der ersten wird der Integrationsprozess für ein Endsystem einfach von einer übergeordneten Region aus wiederholt. Der zweite Ansatz startet eine Umgebungssuche, bei der alle Mess–Server in einem

beschränkten Umkreis dahingehend überprüft werden, ob sie näher zum betrachteten Endsystem liegen als der aktuell zugeordnete Mess–Server. Beide Vorgehensweisen erlauben die freie Einschränkung des Bereiches, innerhalb dessen eine Aktualisierung stattfindet. Damit können lokale Aktualisierungen häufiger durchgeführt werden als globale, was genau der obigen Heuristik entspricht.

Der engültige Algorithmus wird durch zwei Parameter festgelegt: dem *Radius*, d.h. der Zahl der benachbarten Mess–Server bzw. Regionen, die in der Reihenfolge ihrer Entfernung untersucht werden sollen, und der *Ähnlichkeitsschranke*, die im Integrationsprozess von übergeordneten Regionen aus eine Rolle spielt. Die Zuordnung eines Endsystems wird schließlich geändert, wenn ein neuer nähergelegener Mess–Server durch den Algorithmus gefunden wurde. Der komplette Algorithmus ist in Abbildung 3.9 wiedergegeben.

```
procedure zuordnungsanpassung (Endsystem e, Radius r, Ähnlichkeitsschranke a)
(1)   messe Distanz δ(e) zwischen e und zugeordnetem Mess–Server K := Λ(e)
(2)   while (K ≠ nil and r > 0)
(3)       for (i = 1 TO min{r, Anzahl der Geschwister von K})
(4)           G := i-nächster Geschwisterknoten von K
(5)           messe Distanz d zwischen e und G bzw. Zentrum von G
(6)           if (G ist Mess–Server)
(7)               if (d < δ(e)) // G ist näher zu e als bisheriges Optimum
(8)                   ändere Zuweisung von e: Λ(e) := G, δ(e) := d
(9)           else
(10)              if (d ≤ a * δ(e))
(11)                  integriere e probeweise in Teilbaum mit Wurzel G
(12)                  if (Probeintegration erbringt bessere Zuweisung für e)
(13)                      ändere Zuweisung von e
(14)      endfor
(15)      K := Λ(K) (Elternknoten von K im Regionenbaum)
(16)      r := r−(Zahl der Geschwister von K)
(17)  endwhile
```

Abbildung 3.9: Algorithmus zur Anpassung des Zuweisung von Endsystemen

Die Evaluierung dieses Verfahrens bezüglich der Qualität der Mess–Server–Zuweisung und bezüglich des Messaufwands erfolgt empirisch in Kapitel 5.3. Man beachte jedoch, dass der Aufwand nicht gleichmäßig mit steigendem Radius r anwächst, sondern immer dann

sprunghaft steigt, wenn die Suche nach Nachbarn in einer nächsthöheren Baumebene fortgesetzt wird.

3.5 Koordination der Distanzmessungen und Kartenreplikation

Die bisher vorgestellten Verfahren leisten die Berechnung einer Netzwerk–Distanzkarte anhand von gemessenen Distanzen. Die tatsächliche Durchführung dieser Messungen erfolgt jedoch von verteilten Mess–Servern. Daher müssen die einzelnen Messungen entsprechend koordiniert werden. Ein zentraler Server wäre mit dieser Koordination hoffnungslos überlastet, weshalb diese Aufgabe auf möglichst viele Server verteilt werden soll. Zweckmäßigerweise berechnet jeder dieser Server den Teil der Distanzkarte, der aus den von ihm koordinierten Messungen resultiert. Damit ist aber das Wissen über die endgültige Distanzkarte ebenfalls auf mehrere Server verteilt. Um dieses Wissen effizient zusammenzuführen und auch interessierten Klienten zur Verfügung zu stellen, werden zwei Multicast$^{\uparrow}$–Gruppen aufgebaut. Über diese werden jeweils Informationen über den Regionenbaum und über die Zuordnung von Endsystemen ausgesandt. Von den Multicast–Gruppen wird angenommen, dass sie zuverlässig sind, jede Nachricht also auch bei allen Empfänger ankommt. Ein Vergleich verschiedener Protokolle zur Realisierung von zuverlässigen Multicast–Gruppen findet sich in [Levine & Garcia-Luna-Aceves, 1996].

1. Die Multicast–Gruppe MC_{reg} dient zur Übermittlung von Regionenbäumen. Als einziger Nachrichtentyp wird hierin ein (Teil-)Regionenbaum übermittelt.
2. Über die Multicast–Gruppe MC_{es} werden Nachrichten über die Zuordnung von Endsystemen zu Mess–Servern übermittelt. Diese Nachrichten bestehen immer aus einem Tripel (Endsystem, Mess–Server, Distanz).

Der Grund für den Aufbau von zwei verschiedenen Multicast–Gruppen, findet sich darin, dass diese beiden unterschiedliche Empfänger haben. Der Gruppe MC_{reg} gehören sowohl Klienten der Distanzkarte als auch Mess–Server und die nachfolgend vorgestellten Mess–Koordinatoren an. Dagegen wird die Gruppe MC_{es} lediglich von Klienten der Distanzkarte abgehört.

Des Weiteren wird eine Menge von *Mess–Koordinatoren*$^{\uparrow}$ angenommen, von denen aus die Koordination einzelner Messungen sowie die Berechnung von Teilen der Distanzkarte erfolgen kann. Diese Koordinatoren können (aber müssen nicht) auch Mess–Server sein.

Jeder Koordinator ist Empfänger der Multicast–Gruppe MC_{reg}, wodurch er jederzeit über den aktuellen Regionenbaum verfügt.

Nachfolgend wird nun gezeigt, wie der Aufbau und die Aktualisierung von Netzwerk–Distanzkarten geeignet koordiniert werden kann. Netzwerk- oder Knotenfehler werden hierbei nicht explizit beachtet. Die Bewertung der durch die Koordination zusätzlich zur Distanzmessung erzeugten Netzwerkbelastung erfolgt einfach durch die Zahl der notwendigen Kontrollnachrichten (Messaufträge oder -ergebnisse). Die über Multicast–Gruppen verschickten Nachrichten werden dabei nicht beachtet. Zum einen hängt der durch sie hervorgerufene Netzwerkverkehr wesentlich von der Zahl und der Verteilung der Empfänger ab und ist daher nur im Einzelfall mit einzelnen Unicast$^{\uparrow}$–Nachrichten zu vergleichen. Zum anderen müssen die Multicast–Nachrichten in jedem Fall ausgesandt werden, um interessierte Klienten mit der aktuellen Distanzkarte zu versorgen. Je mehr Klienten es dabei gibt, desto unwichtiger wird im Verhältnis der Nachrichtenaufwand für die Mess–Server und Mess–Koordinatoren.

3.5.1 Koordination der Regionenbaumberechnung

Eine einzelne (nichtrekursive) Ballung einer Menge M von Mess–Servern in Regionen wird immer durch einen Mess–Koordinator gesteuert. Dieser wählt zunächst die Teilmenge S der im optimalen Ballungsprozess zu benutzenden Server aus. Für diese stellt er die jeweils durchzuführenden Messaufgaben in sogenannten Messpaketen zusammen, sendet jedem Server sein Messpaket, wartet auf die Ergebnisse aller Messungen und berechnet daraus schließlich eine Ballung in Regionen. Als nächstes erstellt er ein Messpaket, das alle Regionenzentren umfasst. Dieses Paket wird an alle übrigen Server aus $M \setminus S$ geschickt. Anhand der zurückgelieferten Ergebnisse kann der Mess–Koordinator die zusätzlichen Mess–Server den einzelnen Regionen zuweisen. Das Ergebnis der kompletten Ballung wird an die Multicast–Gruppe MC_{reg} geschickt. Für jede Region wird alsdann ein Mess–Koordinator ausgewählt, der optimalerweise in der entsprechenden Region liegen sollte und der mit der rekursiven Fortsetzung des Ballungsprozesses für seine Teilregion beauftragt wird. Abbildung 3.10 auf der nächsten Seite gibt die von einem Mess–Koordinator auszuführenden Schritte detailliert wieder.

Die Festlegung der Messpakete (Schritt 2) garantiert, dass jede Distanz nur in einer Richtung gemessen wird und dass der Umfang der einzelnen Messpakete optimal ausgeglichen ist.

procedure ballungskoordination (Mess–Server M, Baumgrad k, Selektivität s)

(1) wähle zufällig $S := \{m_0, \ldots, m_{s-1}\} \subseteq M$

(2) erstelle Messpakete P_i für $0 \leq i < s$: (OBdA s gerade)

$\forall i \in \{0, \ldots, s/2-1\} : P_i := \{m_{(i+1) \bmod s}, m_{(i+2) \bmod s}, \ldots, m_{(i+s/2) \bmod s}\}$

$\forall i \in \{s/2, \ldots, s\} : P_i := \{m_{(i+1) \bmod s}, m_{(i+2) \bmod s}, \ldots, m_{(i+s/2-1) \bmod s}\}$

(3) $\forall i \in \{0, \ldots, s-1\}$: verschicke Paket P_i an m_i und warte auf Ergebnisse

(4) berechne Ballung, Regionenzentren und Regionenabstände

(5) erstelle Messpaket P bestehend aus allen Regionenzentren

(6) verschicke P an alle Server aus $M \setminus S$ und warte auf Ergebnisse

(7) integriere zusätzliche Mess–Server in Regionenbaum

(8) sende Regionenteilbaum an MC_{reg}

(9) wähle für jede Region R_i einen Mess–Koordinator K_i

(10) **forall** (Regionen R_i mit mehr als k Mess–Servern M_i)

(11) starte ballungskoordination(M_i, k, s) auf K_i

(12) **forall** (Regionen R_i mit höchstens k Mess–Servern M_i)

(13) beauftrage K_i mit Ausmessung der Distanzen zwischen Servern aus M_i

Abbildung 3.10: Steuerung einer Ballung durch einen Mess–Koordinator

Die Auswahl der Mess–Koordinatoren (Schritt 9) kann völlig beliebig erfolgen. Aus Effizienzgründen sollten aber jeweils Koordinatoren gewählt werden, die in den von ihnen zu koordinierenden Regionen liegen.

Die Zahl der mit diesem Protokoll nötigen Kontrollnachrichten pro Ballung beträgt $2 * s + 2 * |M \setminus S| = 2 * |M|$, was klar unter der Zahl der notwendigen Distanzmessungen $(\frac{s*(s-1)}{2} + k*|M \setminus S|)$ liegt. Die Auslastung der einzelnen Mess–Server ist recht gleichmäßig. Jeder Server muss maximal k bzw. $s/2$ Messungen durchführen und eine Kontrollnachricht empfangen und versenden. Der Mess–Koordinator muss dagegen $|M|$ Kontrollnachrichten versenden und empfangen. Um diese Last gleichmäßiger zu verteilen, könnte er Stellvertreter verwenden und diese mit der Integrationssteuerung einer Teilmenge der Mess–Server aus $M \setminus S$ beauftragen.

Aufgrund des geringeren Aufwandes können tiefer gelegene Baumregionen häufiger neu geballt werden als höher gelegene Regionen. Hierzu müssen jedoch die einzelnen Neuballungsprozesse so koordiniert werden, dass sie sich nicht gegenseitig beeinflussen. Die Koordination kann durch Uhren erfolgen. Bei jeder Berechnung einer Ballung (nach dem in Abbildung 3.10 dargestellten Verfahren) legt der Mess–Koordinator hierbei fest, wann die von ihm in Schritt 9 ausgewählten Koordinatoren der einzelnen Teilbäume ihre Teil-

ballung wiederholen sollen. Diese Zeitpunkte müssen alle vor dem nächsten Zeitpunkt der Neuberechnung durch den aktuellen Koordinator liegen. Haben die Zeitpunkte der Neuberechnung auf verschiedenen Hierarchieebenen einen ausreichenden Abstand, so spielen Abweichungen zwischen den Uhren der einzelnen Koordinatoren keine Rolle.

Empfänger der Multicastgruppe MC_{reg} sind selbst dafür verantwortlich, eine konsistente Sicht aus den einzelnen Nachrichten über Regionenteilbäume zusammenzustellen. Insbesondere sollten sie Neuballungen auf höheren Baumebenen erst dann berücksichtigen, wenn diese bis zu den Blättern des Regionenbaumes verfeinert wurden, wenn also jede über MC_{reg} versandte Ballung entweder weiter verfeinert wurde oder aus höchstens k Elementen besteht. Daneben müssen sie die jeweils endgültigen Regionenzentren aus dem Regionenbaum ableiten.

3.5.2 Koordination der Zuordnung von Endsystemen

Initiale Integration

Die Integration von Endsystemen in eine Netzwerk–Distanzkarte, die nach dem in Abschnitt 3.4.3 vorgestellten Verfahren entlang der Hierarchie des Regionenbaumes erfolgt, wird ebenfalls durch Mess–Koordinatoren gesteuert. Die von einem Koordinator durchzuführenden Schritte zur Integration einer Menge von Endsystemen E in eine Region R sind in Abbildung 3.11 auf der nächsten Seite wiedergegeben. Die Funktionen δ und Λ liefern wieder zu jedem Endsystem die optimale (bisher gefundene) Zuordnungsdistanz zu einem Mess–Server bzw. den zugeordnete Mess–Server selbst. Initial wird die Integrationssteuerung auf einem beliebigem Mess–Koordinator für den Wurzelknoten des Regionenbaumes gestartet. In dieser Situation sind natürlich noch keine Zuordnungen für einzelne Endsysteme festgelegt. Die am Ende der Integration zurückgelieferten Ergebnisse werden an die Multicast–Gruppe MC_{es} sowie an die zugewiesenen Mess–Server gesandt.

Die Zahl der bei diesem Vorgehen nötigen Kontrollnachrichten hängt neben der Struktur des Regionenbaumes stark von der Wahl der Ähnlichkeitsschranke a ab. Für $a = 1-\epsilon$ wird für jede Region maximal einmal ein Mess–Koordinator beauftragt. In einem vollständig ausgeglichenen Baum werden von jedem Koordinator k Messpakete versandt. Für größere Werte von a kann die Zahl der Kontrollnachrichten wesentlich größer sein, bleibt aber bei einer genügend großen Menge von Endsystemen E sehr gering gegenüber der Zahl der notwendigen Distanzmessungen.

Wendet man das Verfahren der Integrationssteuerung direkt für alle einfachen Endsysteme, so entsteht eine immense Belastung für die Regionenzentren, insbesondere für solche

procedure integrationssteuerung (Region R, Endsysteme E, $\delta(E)$, $\Lambda(E)$, Ähnlichkeitsschranke a)

(1) **forall** (Kindknoten K von R, die keine Region sind)
(2) sende Messpaket E an K
(3) erwarte Ergebnisse auf alle in (1-2) verschickten Pakete und aktualisiere ggf. (δ, Λ)
(4) $r :=$ Anzahl der Teilregionen von Region R
(5) sende Messpaket $(E, \delta(E))$ an alle Zentren der Teilregionen R_i von R $(1 \leq i < r)$
(6) erwarte Ergebnisse über Endsysteme mit Zuordnungsverbesserung
(7) bestimme für alle $e \in E$ Reihenfolge der Teilregionen gemäß ihrer Entfernung zu e
(8) // durchlaufe Ordnungsindex für die in (7) bestimmte Teilregionenordnung:
for $(o = 1$ **TO** $r)$
(9) // teste Integration für alle End–Systeme in ihre jeweils o-nächstgelegene
// Teilregion (gruppiert nach Teilregionen):
for $(t = 1$ **TO** $r)$ // Schleife über Teilregionen
(10) $U := \{e \in E | R_t$ ist o-nächste Teilregion für e und $\Delta(e, R_t) \leq \delta(e) * a\}$
(11) starte integrationssteuerung$(R_t, U, \delta(U), \Lambda(U), a)$ auf Koordinator für R_t
(12) **endfor**
(13) warte auf Ergebnisse auf alle in (9-12) verschickten Pakete
(14) aktualisiere Zuordnungen (δ, Λ)
(15) **endfor**
(16) sende Zuordnungsverbesserungen von (δ, Λ) an Aufrufer

Abbildung 3.11: Steuerung der Integration von Endsystemen

auf hohen Baumebenen. So muss jedes Zentrum auf Baumebene 1 die Distanz zu allen Endsystemen messen, was für die Größe des Internets sicher nicht akzeptabel ist. Zum einen sollte daher die betrachtete Granularität des Internets auf Subnetze eingeschränkt werden (vgl. Abschnitt 3.3.1). Zum anderen sollten zu allen Regionenzentren genügend viele Stellvertreter bestimmt werden, die auch möglichst zentral in ihrer Region liegen und von denen Messungen zu Endsystemen alternativ ausgeführt werden können.

Die Belastung der Mess–Koordinatoren durch Kontrollnachrichten ist im Vergleich relativ unerheblich. Für einen einzelnen Koordinationsauftrag müssen sie gerade mal k Kontrollnachricht empfangen und versenden. Mehrere Koordinationsaufträge für ein und dieselbe Region können perfekt auf alle Koordinatoren dieser Region verteilt werden.

Anpassung

Die Anpassung der Zuweisung eines Endsystems zum nächstgelegenen Mess–Server wird

immer durch den momentan zugewiesenen Mess–Server initiiert. Hierzu wird angenommen, dass jeder Mess–Server Empfänger der Multicast–Gruppe MC_{reg} ist und aus deren Nachrichten die Struktur aller ihm übergeordneten Regionen speichert. Zusätzlich muss er natürlich die ihm zugewiesenen Endsysteme kennen, was er im Rahmen der initialen Integration von Endsystemen ja auch mitgeteilt bekommt.

Die Zuordnungsanpassung ist ein in jedem Mess–Server permanent laufender Prozess. Dieser ist in Abbildung 3.12 auf der nächsten Seite dargestellt. Pro Schleifendurchlauf wird zunächst der zu untersuchende Radius r bestimmt (Schritt 3). Ist dieser gleich 0, so werden die Zuordnungsentfernungen einfach durch den lokalen Mess–Server erneut gemessen und ggf. aktualisiert. Andernfalls wird der r-nächste Nachbar des lokalen Mess–Servers im Regionenbaum bestimmt und es wird überprüft, ob von diesem eine geringere Distanz zu den jeweiligen Endsysteme vorliegt (Schritte 8-13). Endsysteme mit Distanzverbesserung werden dem jeweiligen Mess–Server neu zugeordnet (Schritt 16). Schließlich werden von anderen Mess–Servern übernommene Endsysteme dem lokalen Server zugeordnet (Schritt 18). Man beachte allerdings, dass weder die Zeitabstände zwischen einzelnen Schleifendurchläufen noch die Auswahlstrategie für den zu untersuchenden Radius (Schritt 3) festgelegt sind. Diese sind für das prinzipielle Koordinationverfahren unerheblich und können je nach Distanzmaß und gewünschter Kartengenauigkeit variieren.

Für die Zahl der bei diesem Vorgehen nötigen Kontrollnachrichten gilt Gleiches wie beim obigen Verfahren zur Integrationssteuerung. Sie hängt stark von der Wahl des Parameters a ab, ist aber selbst bei $|E| = 1$ höchstens doppelt so hoch wie die Zahl der durchzuführenden Distanzmessungen. Für größere Mengen E wird die Zahl der Kontrollnachrichten gegenüber den Messungen vernachlässigbar.

Die Belastung der einzelnen Mess–Server durch die Anpassungssteuerung ist sehr gleichmäßig verteilt, vorausgesetzt dass alle ungefähr gleichviele Endsysteme zugeordnet haben.

Synchronisation mit Regionenbaumanpassung

Im Rahmen der Zuordnungsanpassung kann es zu einem Konfliktfall kommen, wenn parallel zu einer von einer Region R ausgehenden Integrationssteuerung ebendiese Region R oder Kindregionen von R einem erneuten Ballungsprozess unterzogen werden. Um hier evtl. Inkonsistenzen zu vermeiden, werden Regionenteilbäume, die über MC_{reg} versandt werden, mit einem global eindeutigem Bezeichner versehen. Zu jeder Integration von Endsystemen in einen Teilbaum wird nun zusätzlich die Instanz des Teilbaumes anhand des globalen Bezeichners vermerkt. Mess–Koordinatoren müssen Teilbäume auch dann

```
procedure anpassungssteuerung (Ähnlichkeitsschranke a)
(1)  E := Menge der dem lokalen Mess–Server zugeordneten Endsysteme
(2)  loop
(3)      wähle Radius r
(4)      if (r = 0)
(5)          messe und aktualisiere Distanzen zu allen Endsystemen aus E
(6)      else
(7)          N := r-nächster Nachbar von M
(8)          if (N ist Mess–Server)
(9)              sende Messpaket E zu N und erwarte Ergebnis
(10)         else // N ist also eine Region
(11)             wähle Regionenkoordinator K für N
(12)             starte integrationssteuerung(N, E, δ(E), Λ(E), a) auf Koordinator K
(13)         endif
(14)         U := Menge der Endsysteme, für die eine bessere Zuordnung gefunden wurde
(15)         streiche U aus E: E := E \ U
(16)         übergebe Kontrolle über Endsysteme aus U an entsprechende Mess–Server
(17)     endif
(18)     erweitere E um neu erhaltene Endsysteme
(19)     sende neue Zuordnungen und Distanzen für Endsysteme aus E an MC_es
(20) endloop
```

Abbildung 3.12: Steuerung der Anpassung von Endsystemen durch einen Mess–Server

noch für eine gewisse Zeit lang speichern, wenn diese bereits aktualisiert worden sind. Dann können sie nämlich Integrationsprozesse, die noch auf veralteten Regionenbäumen ausgeführt werden, korrekt zu Ende führen. Anhand des globalen Bezeichners stellen sie hierzu die jeweils für eine Integration relevante Version des Regionenteilbaumes fest.

3.6 Diskussion und Erweiterungen

Die vorgestellten und recht einfachen Verfahren zur Berechnung und Wartung von Netzwerk–Distanzkarten bergen sicherlich noch ein enormes Optimierungspotenzial. Gleichwohl skalieren sie aufgrund der hierarchischen Struktur der Karten für nahezu beliebige Problemgrößen in allen drei in Abschnitt 3.3.2 genannten Aspekten. Bei im Wesentlichen ausgeglichenen Regionenbäume sind Distanzschätzungen sind mit einem Zeitaufwand, der

sich logarithmisch zur Zahl der Mess–Server verhält, möglich. Der Speicherbedarf einer Distanzkarte ist linear in der Zahl der Endsysteme und der Mess–Server. Am Schwersten ins Gewicht fällt die durch Distanzmessungen entstehende Netzwerkbelastung. Diese ist in jedem Fall mindestens linear in der Zahl der Endsysteme, was bei Netzwerken wie dem Internet schon eine signifikante Last bedeutet. Allerdings ist auch kein messbasiertes Verfahren vorstellbar, das Distanzschätzungen mit geringerem Grundaufwand ermöglicht. Der zur Anpassung von Distanzkarten erforderliche Aufwand hängt wesentlich von der Stabilität des betrachteten Distanzmaßes sowie von der gewünschten Genauigkeit der Distanzschätzung ab. Die Protokolle zur Koordination der Kartenerstellung führen zu einer gleichmäßig unter den Mess–Servern bzw. –Koordinatoren verteilten Last an Netzwerkinteraktionen.

Der Ansatz der Netzwerk–Distanzkarten ist prinzipiell für beliebige Distanzmaße und Computernetzwerke anwendbar. Seine Effektivität lässt sich hingegen nur durch Experimente in konkreten Szenarien feststellen.

Die Kalibrierung der Verfahren, d.h. die optimale Wahl der bestimmenden Parameter wie z.B. der Ähnlichkeitsschranke, wird in dieser Arbeit nicht behandelt und muss individuell in Abhängigkeit des betrachteten Distanzmaßes sowie der Anforderungen an die Güte der Distanzschätzung erfolgen.

3.6.1 Behandlung von Knoten- und Netzwerkfehlern

Bei der Berücksichtigung von Fehlern (Knoten- und Netzwerkfehlern) kann man drei Bereiche unterscheiden: (1) die Datenstruktur der Netzwerk–Distanzkarten und die Frage, inwieweit sie überhaupt Fehler wiedergeben können, (2) die zur Kartenerstellung verwendeten Algorithmen und (3) die zur verteilten Distanzmessung und Kartenreplikation eingesetzten Protokolle.

Wiedergabe von Fehlern durch Netzwerk–Distanzkarten

Zur Schätzung der Distanz zwischen zwei Endsystemen wird von Distanzkarten ein Wert herangezogen, der dem Abstand zwischen zwei benachbarten Netzwerkregionen entspricht, denen die beiden Endsysteme angehören. Dieser Abstand wiederum ergibt sich als Mittelwert aus einzelnen Distanzmessungen, die zwischen Mess–Servern der beiden Regionen durchgeführt wurden. Somit gibt eine Distanzschätzung immer nur ein mittleres Verhalten (noch dazu aus der Vergangenheit) wieder und kann singuläre Fehler nicht erfassen.

Hierzu wäre letztendlich eine komplette Ausmessung des gesamten Netzwerkes notwendig. Zudem müsste die Ausmessung auch noch möglichst zeitnah zur Schätzung erfolgen.

Einzig eine Netzwerkpartition, bei der ein Netzwerk in zwei Regionen aufgespalten ist, zwischen denen keine Kommunikation möglich ist, kann von einer Distanzkarten wiedergegeben werden. In diesem Fall stellt der Fehler gerade das Charakteristikum der Distanzmessung zwischen allen Mess–Server der beiden Regionen dar.

Fehlertolerante Algorithmen zur Kartenerstellung

Die zur Kartenerstellung verwendeten Algorithmen könne problemlos so erweitert werden, dass sie Fehler, das heißt nicht verfügbare Distanzwerte, tolerieren. Diese Erweiterung war für die Experimente in Kapitel 5 sogar notwendig, da die dort verwendete Messmethode nicht in allen Fällen einen Distanzwert lieferte.

Im Einzelnen müssen die Algorithmen folgendermaßen erweitert werden: Bei den Verfahren zur optimalen Ballung aus Abschnitt 3.4.1 werden nicht messbare Distanzen schlichtweg ignoriert und damit auch nicht von der Distanzkarte wiedergegeben. Einzig eine Netzwerkpartition würde von den Ballungsverfahren durch zwei unverbundene Regionen wiedergegeben. Um fehlende Distanzwerte zu Regionenzentren zu überbrücken, werden pro Zentrum 2 Stellvertreter bestimmt, die sich ebenfalls durch ihre zentrale Lage innerhalb der Region auszeichnen. Sollte also ein Messwert zu einem Zentrum nicht verfügbar sein, so wird stattdessen die Distanz zu einem Stellvertretern herangezogen. Ist bei der Integration von Mess–Servern (im gemischten Verfahren aus Abschnitt 3.4.2) oder Endsystemen (Abschnitt 3.4.3) weder die Distanz zu einem Regionenzentrum noch zu seinen Stellvertretern bestimmbar, wird die entsprechende Region für die Integration nicht weiter in Betracht gezogen. Da zur Anpassung von Regionenbäumen ebenfalls das gemischte Verfahren aus Abschnitt 3.4.2 verwendet wird, sind hier keine weiteren Maßnahmen zur Realisierung der Fehlertoleranz notwendig. Falls im Rahmen der Anpassung von Endsystemen ein Distanzwert zu einem momentan zugeordneten Mess–Server nicht verfügbar ist, so wird die Umgebung des Servers solange durchsucht, bis wieder ein regulärer Mess–Server gefunden wurde.

Fehlertolerante Protokolle zur Distanzmessung und Kartenreplikation

Die Realisierung fehlertoleranter Protokolle ist prinzipiell natürlich möglich, die Herausforderung liegt hier eher in der Bestimmung von Mechanismen, die einen möglichst geringen Mehraufwand bedeuten. Die Bestimmung solcher Mechanismen ist nicht Gegenstand

dieser Arbeit, es sollen aber einige Ansätze aufgezeigt werden, mit denen Fehlertoleranz erreicht werden kann.

Die fehlertolerante Replikation von Karteninformationen durch zuverlässige Multicast–Gruppen stellt hier kein besonderes Problem dar, da es hierzu bereits umfangreiche Arbeiten gibt (siehe z.B. [Levine & Garcia-Luna-Aceves, 1996]).

Bei den Protokollen zur Distanzmessung muss allein die Arbeit der Mess–Koordinatoren und der Mess–Server fehlertolerant realisiert werden. Erstere sind für die Erstellung eines (Teil-)Regionenbaumes sowie für die Integration von Mess–Servern und einfachen Endsystemen verantwortlich (siehe Abbildungen 3.10 und 3.11 auf den Seiten 60 bzw. 62). Letzteren fällt im Rahmen der Anpassung einfacher Endsysteme die entscheidende Steuerungsrolle zu (siehe Abbildung 3.12 auf Seite 64). In beiden Fällen kann Fehlertoleranz durch Einführung von „Beobachtungsrechnern“ erreicht werden. Diese kontrollieren den Fortschritt des jeweils beobachteten Mess–Koordinators bzw. Mess–Servers. Fällt ein Koordinator/Server aus, so können seine Beobachter dies feststellen und einen neuen Koordinator/Server auswählen und mit der Arbeit beauftragen. Ein Beispiel zur Umsetzung eines solchen Mechanismus', wenn auch in anderem Kontext eingesetzt, findet sich in [Straßer & Rothermel, 1998]. Wesentlich effizienter als diese explizite Beobachtung könnte eine implizite Beobachtung anhand der Multicast–Gruppen MC_{reg} und MC_{es} sein. An diesen kann der Fortschritt der Arbeit eines jeden Mess–Koordinators und Mess–Servers beobachtet werden. Fällt z.B. ein Mess–Koordinator aus, so wird der von ihm zu berechnende (Teil-)Regionebaum nicht über die Multicast–Gruppe MC_{reg} übermittelt. Fällt ein Mess–Server aus, so finden für die ihm zugeordneten Endsysteme keine Aktualisierungen mehr statt. Am Ausbleiben dieser Informationen könnten Beobachtungsrechner also einen Fehlerfall erkennen und dann beheben. Da die Erstellung und Anpassung von Netzwerk–Distanzkarten keine zeitkritische Aufgabe ist, spielt die größere Zeitverzögerung dieser indirekten Beobachtungsmöglichkeit keine Rolle.

Die Abbarbeitung von Messpaketen bzw. die Durchführung einzelner Distanzmessungen durch Mess–Server bedarf keiner besonderen Mechanismen zur Fehlertoleranz, da die Algorithmen zur Kartenerstellung nicht auf einzelne Messergebnisse angewiesen sind.

Der Fehlerfall einer Netzwerkpartitionen kann mit obigen Verfahren natürlich nicht behoben werden. In diesem Fall kann eine Distanzkarte nur innerhalb der nicht partitionierten Teile eines Netzwerkes erstellt bzw. aktualisiert werden.

Zusammenfassend lässt sich feststellen, dass der Ansatz der Netzwerk–Distanzkarten auch bei auftretenden Netzwerk- oder Knotenfehlern funktioniert und dass der Mehraufwand zur Realisierung fehlertoleranter Protokolle sehr gering gehalten werden kann.

3.6.2 Flexible Anpassung von Distanzkarten

Ein besonders wichtiger Optimierungsaspekt für Netzwerk–Distanzkarten soll nachfolgend kurz angerissen werden. Das in Abschnitt 3.4.4 beschriebene Verfahren zur Anpassung von Regionenbäumen ist äußerst simpel und nicht sehr effizient, da mit ihm immer komplette Teilbäume neu berechnet werden müssen. Wesentlich effizienter könnten u.U. Verfahren sein, mit denen Struktur und Distanzen in Regionenbäumen getrennt voneinander aktualisiert werden können. So sollte es zum Beispiel möglich sein, lediglich die Distanzschätzung für zwei benachbarte Regionen zu aktualisieren, ohne den zugehörigen Teilbaum komplett neu berechnen zu müssen. Ebenso sind Verfahren vorstellbar, mit denen einzelne Knoten oder komplette Teilbäume im Regionenbaum verschoben werden ohne dabei aber den Rest des Baumes zu verändern.

Schließlich wäre es äußerst wünschenswert, wenn die Anpassung von Netzwerkbäumen nicht in vordefinierten Zeitintervallen erfolgt, sondern Mess–Server ihre „Umgebung“ selbständig beobachten und erkennen, wann umfangreichere Anpassungen notwendig werden. Nach ähnlichen Mechanismen könnte die Anpassung einfacher Endsysteme optimiert werden. Ein Mess–Server bräuchte z.B. Zuweisungsanpassungen für ein ihm zugeordnetes Endsystem nur dann durchzuführen, wenn sich seine Distanz zum Endystem vergrößert oder seine Distanz zu einem entfernten Mess–Server verringert hat.

Das nachfolgende Kapitel 4 beschreibt nun eine Anwendung, die unter Ausnutzung von Netzwerk–Distanzkarten, ihre Netzwerkkommunikation erheblich optimieren kann. Dabei handelt es sich um mobile Programme, die verteilt vorliegende Daten mit einer Filterfunktion untersuchen sollen. Die Aussendung dieser Programme wird durch die von Netzwerk–Distanzkarten ermöglichte Schätzung beliebiger Netzwerkdistanzen optimiert. Das in Kapitel 2 vorgestellte Konzept einer spezialisierten Suchmaschine wiederum benutzt mobile Programme zur effizienten Analyse von potenziell relevanten Dokumenten.

Kapitel 4

Aussendung mobiler Programme zur verteilten Informationsfilterung

Will ein Benutzer in einem Computernetzwerk Daten, die auf einem entfernten Server vorliegen, mit einem Programm analysieren, so muss er die Daten üblicherweise auf seinen Rechner herunterladen, um dann das Programm auf sie anwenden zu können. Anstatt die Daten auf den Rechner des Programmes zu kopieren, kann man allerdings auch umgekehrt das Programm zu den Daten senden, also das Programm auf den Rechner kopieren, auf dem die Daten vorliegen, es dort zur Ausführung bringen und das Ergebnis schließlich an den Benutzer zurücksenden. Ein Ziel, das sich mit dem Einsatz solcher *mobilen Programme* verbindet, ist die Reduktion der Kommunikationskosten. Diese sind im einfachsten Fall als die Menge der über das Netzwerk zu übertragenden Daten definiert. Eine Reduktion ist im Prinzip immer dann möglich, wenn das zu untersuchende Datenvolumen größer ist als das zu versendende Programm plus dem Umfang des aus der Analyse resultierenden Ergebnisses. Da also insbesondere das Ergebnisvolumen geringer als das Datenvolumen sein muss, spricht man auch von einem Szenario der Informationsfilterung und von dem mobilen Programm als Filterprogramm. Der letztlich mit der Reduktion der Kommunikationskosten verfolgte Zweck besteht in der Verringerung der Netzwerkbelastung oder auch in der Reduzierung der Latenzzeit für eine Anwendung, die auf das Ergebnis der Datenanalyse wartet.

Verfeinert man die Betrachtung der Kommunikationskosten, so kann man neben dem reinen Datenvolumen auch noch die Netzwerkdistanz betrachten, über die die Daten jeweils übertragen werden müssen. Eine solche Betrachtung ist insbesondere deshalb notwendig,

weil man nicht davon ausgehen kann, dass alle Daten–Server bereit sind, mobile Programme aufzunehmen und auszuführen. Daher kann ein mobiles Programm normalerweise auch nur in die „Nähe“ eines Daten–Servers gesandt werden und kann dann diesen (immer noch entfernten) Server über eine nun allerdings kürzere Netzwerkdistanz hin untersuchen. Ob mit der Aussendung eines mobilen Programmes wirklich eine Reduktion der Kommunikationskosten erreichbar ist, hängt dann neben den jeweiligen Datenmengen entscheidend von den Netzwerkdistanzen zwischen den beteiligten Rechnern ab. Allgemein gilt, je näher ein mobiles Programm zu einem Daten–Server gebracht werden kann, desto größer ist sein Nutzen, also die mit ihm erreichbare Reduktion der Kommunikationskosten.

Die Verallgemeinerung dieses Szenarios besteht darin, nicht nur Daten auf einem sondern auf mehreren Servern mit einer Filterfunktion zu untersuchen, die als mobiles Programm zu den Daten–Servern gesandt werden kann. Die Berechnung einer optimalen Strategie zur Aussendung eines solchen mobilen Filterprogrammes (in [Theilmann & Rothermel, 1999a] und [Theilmann & Rothermel, 2000c] bereits beschrieben und in [Theilmann & Rothermel, 2000b] auf das Gebeit des elektonischen Handels angewandt) ist der Gegenstand dieses Kapitels. Eine Anwendung, die einer solchen Filterung bedarf, wurde bereits in Kapitel 2 vorgestellt. Dabei handelt es sich um spezialisierte Suchmaschinen, sogenannte *Domänen–Experten*, die über das Internet verteilte Dokumente auf ihre Relevanz für das spezielle Themengebiet der Suchmaschine hin analysieren müssen. Die in Kapitel 3 vorgestellten Netzwerk–Distanzkarten bieten mit der Schätzung beliebiger Netzwerkdistanzen die entscheidende Voraussetzung, um die Aussendung des mobilen Filterprogrammes geeignet koordinieren zu können. Es erweist sich als keineswegs optimal, jeweils eine Instanz des mobilen Filterprogrammes möglichst nah zu jedem zu untersuchenden Daten–Server zu senden. Stattdessen muss die Aussendung koordiniert werden, damit ein möglichst großer Nutzen erreicht wird. Zum Beispiel wird sich in den nachfolgenden Untersuchungen zeigen, dass es in einigen Situationen durchaus sinnvoll ist, Daten, die auf mehrere Server verteilt sind, von ein und demselben Rechner aus zu untersuchen.

Die weiteren Abschnitte dieses Kapitels sind wie folgt gegliedert: Zum besseren Verständnis mobiler Programme und zur Klärung einiger Begriffe wird in Abschnitt 4.1 eine kurze Einführung hierzu gegeben. Nach einer Formalisierung und Analyse des Problems der Aussendung mobiler Filterprogramme in Abschnitt 4.2 werden in Abschnitt 4.3 einige verwandte Ansätze diskutiert. In den Abschnitten 4.4 und 4.5 werden dann zwei Lösungsverfahren vorgestellt, wobei das zweite eine Erweiterung des ersten darstellt und bessere Skalierungseigenschaften aufweist. Abschnitt 4.6 fasst das Kapitel zusammen und erläutert einige mögliche Erweiterungen der vorgestellten Verfahren. Eine Evaluation des

Ansatzes erfolgt in Kapitel 6.

4.1 Grundlagen mobiler Programme

Begriffsdefinitionen

Ein Programm, das aus der Initiative eines Rechners A heraus von Rechner A über ein Netzwerk auf einen entfernten Rechner B gebracht und dort automatisch gestartet bzw. fortgesetzt werden kann, wird als *mobiles Programm*$^{\uparrow}$ bezeichnet. Diese Definition schließt Techniken des *Code on Demand*, bei denen ein Programm dynamisch auf den lokalen Rechner geladen und ausgeführt wird (wie z.B. bei *Java Applets*), explizit aus, da in diesem Fall die Initiative zum Programmtransfer vom Zielrechner ausgeht.
Ein Rechner, der die Aufnahme und Ausführung mobiler Programme zulässt, wird als *Codeplattform*$^{\uparrow}$ bezeichnet.
Ein System, das eine einheitliche Schnittstelle für Codeplattformen, mobile Programme und den Transfer von Programmen zwischen Codeplattformen definiert, heißt *Mobile–Code–System*.

Technische Grundlagen

Erste Voraussetzung für den Einsatz mobiler Programme ist, dass diese auf den jeweiligen Codeplattformen auch ausführbar sind. Um hier keinen Einschränkungen zu unterliegen sollte ein mobiles Programm daher in maschinenunabhängiger Form formuliert sein.

Für die technische Realisierung des Transfers eines mobilen Programmes gibt es verschiedene Mechanismen (nach [Rothermel et al., 1997]). Der Mechanismus der *Remote Execution*$^{\uparrow}$ transferiert ein Programm mitsamt einiger Aufrufparameter auf den entfernten Rechner und startet das Programm dort.

Der Mechanismus der *(starken) Migration*$^{\uparrow}$ erlaubt den Transfer eines mobilen Programmes zur Laufzeit, es wird also genau genommen eine Programminstanz transferiert. Die Programmierumgebung bietet hierzu ein spezielles Kommando an (z.B. einen Befehl `go(Codeplattform xy)`), das vom Programmierer an beliebigen Stellen des Programmes eingefügt werden kann. Sobald die Abarbeitung des Programmes auf dieses Kommando stößt, wird das Programm auf den spezifizierten Zielrechner übertragen und dort mit dem nachfolgenden Kommando fortgesetzt. Bei der Migration wird neben dem Programm und seinen Daten auch noch der komplette Ausführungszustand übertragen. Damit erfolgt der Programmtransfer für den Programmierer völlig transparent.

Neben diesen beiden Mechanismen gibt es noch eine Mischform, genannt *schwache Migration*. Wie bei der starken Migration wird hier der Transfer einer Programminstanz zur Laufzeit ermöglicht. Allerdings erfolgt dieser Transfer nicht mehr vollkommen transparent. Vielmehr muss der Programmierer eine spezielle Einsprungsprozedur festlegen, die nach einem Transfer zu einer neuen Codeplattform aufgerufen wird. Daneben wird auch nicht mehr der komplette Ausführungszustand, sondern lediglich der Inhalt der globalen Variablen übertragen. Über diese muss ein Programmierer den Ausführungszustand so weit wie nötig codieren.

Geschichte und Nutzen der mobilen Programme

Die Idee, mobile Programme über ein Netzwerk zu entfernten Rechnern zu senden und dort auszuführen, lässt sich bis in die Anfänge des Internets zurückverfolgen. So wurde bereits von [Rulifson, 1969] eine *Decode-Encode-Language* entwickelt, mit der interaktive Programme auf entfernten Terminals ausgeführt werden konnten.

In den späten siebziger Jahren kam das Forschungsgebiet der Prozessmigration auf (nach [Milojičić et al., 1999]). Ziel hierbei war es, Prozesse in einer homogenen Umgebung automatisch auf andere Rechner verlagern zu können, um hierdurch Last dynamisch zu verteilen oder auch die Fehlertoleranz von Systemen zu erhöhen.

Mit dem in den neunziger Jahren aufgekommenen Begriff der *mobilen Agenten* verbindet sich die Idee mobiler Programme, die im Auftrag ihres Besitzers autonom durch ein Netzwerk wandern und dabei Aufgaben für ihren Benutzer erfüllen. Dabei entstand die Vision eines globalen Agentensystems, in dem eine Vielzahl von Agenten sich gleichzeitig bewegen und untereinander kooperieren kann. Daneben wurde unter dem Banner der mobilen Agenten auch die Forschung über mobile Programme im Allgemeinen wieder aufgenommen, nun verstärkt im Kontext sehr großer Computernetzwerke wie dem Internet.

Neben der bereits angesprochenen Reduktion von Kommunikationskosten gibt es eine Reihe von anderen Vorteilen, die sich mit mobilen Programmen bzw. Agenten ergeben sollen. Hierzu gehören das asynchrone Arbeiten (z.B. das Versenden von Arbeitsaufträgen in Form eines Programmes von einem mobilen, nur zeitweise mit einem Netzwerk verbundenem Endgerät) und die automatische Software–Installation. Gute und kritische Diskussionen zu diesen und anderen Vorteile finden sich in [Chess et al., 1995] oder auch [Rothermel et al., 1997].

4.2 Problemstellung

4.2.1 Erweitertes Systemmodell

Das im Folgenden benutzte Modell eines Computernetzwerkes ähnelt dem im Kontext der Netzxwerk–Distanzkarten eingeführten Modell aus Kapitel 3.3.1. Es besteht wieder aus einer Menge von Endsystemen $\mathcal{E}$ zusammen mit einer Funktion $\Delta(x, y)$, die jedem Paar von Endsystemen $x, y \in \mathcal{E}$ einen nicht-negativen Distanzwert zuweist.

Im Gegensatz zu Kapitel 3.3.1 wird im Zusammenhang der Aussendung mobiler Programme keine Symmetrie bei den Distanzwerten gefordert und Distanzen, die aufgrund eines Netzwerk- oder Knotenfehlers zeitweilig nicht bestimmbar sind, können durch den Wert Unendlich modelliert werden. Da diese Annahmen schwächer als die in Kapitel 3.3.1 sind, steht einer Anwendung von Distanzkarten bei der Koordination mobiler Filterprogramme nichts im Wege.

Jedes Endsystem wird gleichzeitig *Daten–Server*$^{\uparrow}$ genannt, da es potenziell Daten für den öffentlichen Zugriff bereitstellt. Die Menge der Daten–Server wird mit $\mathcal{D}$ bezeichnet und es gilt somit $\mathcal{D} = \mathcal{E}$.

Ein Endsystem, das die Ausführung mobiler Programme ermöglicht, wird *Codeplattform*$^{\uparrow}$ genannt. Die Menge der Codeplattformen wird mit $\mathcal{C}$ bezeichnet. Es wird explizit nicht angenommen, dass jedes Endsystem bzw. jeder Daten–Server gleichzeitig eine Codeplattform darstellt. Eine solche Annahme wäre aus zahlreichen Gründen unrealistisch: Zum eine haben viele Rechnerbetreiber überhaupt gar kein Interesse daran, ihren Rechner für fremde mobile Programme zur Verfügung zu stellen. Daneben wird es, einen Erfolg des Paradigmas der mobilen Programme vorausgesetzt, sicherlich noch einige Zeit dauern, bis sich aus der Vielzahl der *Mobile-Code-Systeme* ein (universeller) Standard heraus etabliert. Bis dahin werden verschiedene Systeme miteinander konkurrieren und jeweils nur auf einigen Rechnern verfügbar sein. Schließlich macht es das bisher nur unzureichend gelöste Sicherheitsproblem von *Mobile-Code-Systemen* (siehe z.B. [Vigna, 1998]) wahrscheinlich, dass sich verschiedene Anbietergruppen bilden, die jeweils nur mobile Programme von Rechnern der jeweils eigenen Gruppe akzeptieren.

Kommunikationskosten werden definiert als Funktion des Datenvolumens, das über eine Netzwerkverbindung übertragen wird, und einer Netzwerkdistanz, die die Charakteristik der jeweiligen Verbindung beschreibt. Der Ansatz zur Aussendung mobiler Programme ist unabhängig von der gewählten Funktion und auch unabhängig vom gewählten Distanzmaß. Eine kurze Diskussion verschiedener Distanzmaße erfolgt in Kapitel 3.1.2. Zur

Vereinfachung der Darstellung wird im Folgenden wie auch in den Experimenten in Kapitel 6 die Produktfunktion verwendet. Andere Funktionen, in denen zum Beispiel ein fester Kostenfaktor pro Kommunikationsverbindung addiert wird, wären gleichermaßen möglich.

4.2.2 Formalisierung des Aussendungsproblems

Ein Szenario der Informationsfilterung lässt sich folgendermaßen beschreiben:

1. eine Menge von zu untersuchenden Daten–Servern $\mathcal{D}_U \subseteq \mathcal{D}$,
2. dem pro Server zu untersuchenden Datenvolumen $v_U : \mathcal{D}_U \rightarrow \mathbb{N}$,
3. das pro Server aus der Untersuchung resultierende Datenvolumen $v_R : \mathcal{D}_U \rightarrow \mathbb{N}$,
4. dem Umfang des mobilen Filterprogrammes $v_F \in \mathbb{N}$ und
5. der Ausgangsplattform $A \in \mathcal{C}$, auf der das Filterprogramm initial vorliegt und zu der alle Ergebnisse geliefert werden müssen.

Bei Bedarf ließe sich diese Modellierung problemlos um die Zahl der pro Daten–Server zu untersuchenden Dokumente (und damit der notwendigen Anfragen bzw. das mit diesen verbundene Datenvolumen) verfeinern. Zur Vereinfachung der folgenden Diskussion wurde eine solche Modellierung allerdings unterlassen.

Ein *Aussendungsplan*$^{\uparrow}$ beschreibt nun eine Möglichkeit, wie eine gegebene Menge von Daten–Servern durch ein mobiles Filterprogramm untersucht werden kann. Er legt im Einzelnen fest, zu welchen Codeplattformen das Programm gesandt werden soll und wie die hierzu nötigen Transfers aussehen. Dies kann durch eine Baumstruktur beschrieben werden. Die Wurzel dieses Baumes stellt die Ausgangsplattform dar, Knoten des Baumes beschreiben Codeplattformen und Kanten beschreiben Programmtransfers. Daneben legt ein Aussendungsplan noch fest, von welcher der durch das mobile Filterprogramm besuchten Codeplattformen die Daten–Server jeweils untersucht werden.

Abbildung 4.1 zeigt ein Beispiel eines Aussendungsplanes. Man beachte hierbei, dass Transfers des Filterprogrammes sowohl von der Ausgangsplattform also auch von anderen Codeplattform aus stattfinden. Daneben kann es durchaus sinnvoll sein, manche Daten–Server direkt von der Ausgangsplattform aus zu untersuchen, und natürlich müssen nicht alle verfügbaren Codeplattformen in jedem Fall genutzt werden.

Es wird davon ausgegangen, dass es im Sinne der Kommunikationskosten am effizientesten ist, die auf den einzelnen Codeplattformen anfallenden Ergebnisse immer direkt zur

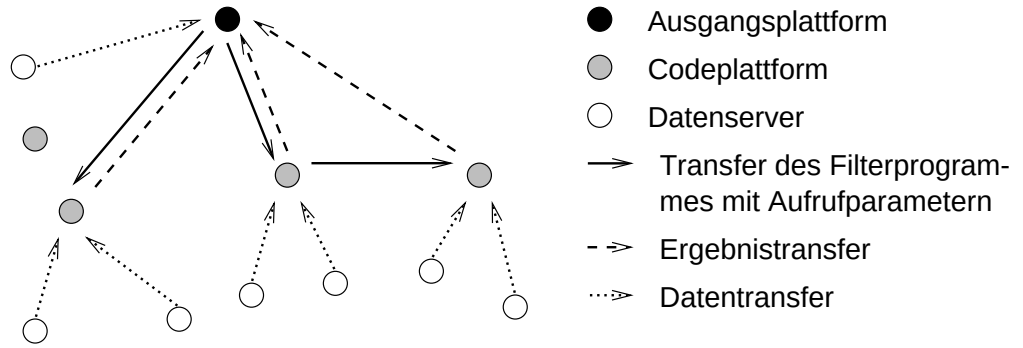


Abbildung 4.1: Beispiel zur Aussendung eines mobilen Filterprogrammes

Ausgangsplattform zurückzusenden. Setzt man bei Netzwerkdistanzen die Gültigkeit der Dreiecksungleichung voraus (vgl. mit Kapitel 3.1.2), so ist diese Annahme auf jeden Fall erfüllt, wenn Kommunikationskosten über die Produktfunktion definiert sind und auch für den Fall, dass noch ein zusätzlicher konstanter Faktor pro Kommunikationsverbindung hinzuaddiert wird.

Ein Aussendungsplan kann formal folgendermaßen beschrieben werden:

1. $\mathcal{C}_B \subseteq \mathcal{C}$ legt die Menge der durch das Filterprogramm besuchten Codeplattformen fest (mit $A \in \mathcal{C}_B$).
2. Die Funktion $u : \mathcal{D}_U \rightarrow \mathcal{C}_B$ weist jedem zu untersuchenden Daten–Server eine Codeplattform zu. Zur Untersuchung eines Servers muss das Filterprogramm zu der jeweils zugewiesenen Codeplattform transferiert werden.
3. Die Funktion $q : \mathcal{C}_B \setminus \{A\} \rightarrow \mathcal{C}_B$ beschreibt die Quellplattform von der jede besuchte Codeplattform das Filterprogramm jeweils bekommt. Sie muss so definiert sein, dass für jede besuchte Codeplattform ein Pfad von Filtertransfers von der Ausgangsplattform bis zur Codeplattform hin existiert.

Ein Aussendungsplan beschreibt nicht explizit die notwendigen Replikationen des Filterprogrammes.[1] Insbesondere legt er nicht fest, ob von einer besuchten Codeplattform c aus jeweils ein Replikat zu allen über die Funktion q spezifizierten Nachfolgeplattformen gesandt werden soll oder ob zumindest eine Replikation ausgespart wird, indem das Filterprogramm auf Plattform c zunächst seine lokalen Aufgaben wahrnimmt und danach zu einer der Nachfolgeplattformen transferiert wird. Beide Alternativen weisen dieselben

[1] Der Begriff des *Filterprogrammes* bezeichnet im Folgenden sowohl das Programm an sich, welches durch eine Filterfunktion definiert ist, als auch die einzelnen Replikate, d.h. die einzelnen auszusendenden Programminstanzen. Es wird aus dem Kontext ersichtlich sein, welche Bedeutung jeweils gemeint ist.

Kommunikationskosten auf, jedoch ist die erste Alternative trotzdem zu bevorzugen, da mit ihr die Gesamtzeit zur Erledigung der globalen Filteraufgabe minimiert wird.

Der Spezialfall eines Aussendungsplanes ist der triviale Aussendungsplan, bei dem alle Daten–Server von der Ausgangsplattform untersucht werden, der also durch $\mathcal{C}_B = \{A\}$, $u \equiv A$ und $q = \emptyset$ beschrieben wird.[2]

Die mit der Ausführung eines allgemeinen Aussendungsplanes verbundenen Kommunikationskosten betragen:

$$\underbrace{\sum_{c \in \mathcal{C}_B \setminus \{A\}} \Delta(q(c), c) * v_F}_{\text{Filtertransfers}} + \underbrace{\sum_{d \in \mathcal{D}_U} \Delta(d, u(d)) * v_U(d)}_{\text{Datentransfers}} + \underbrace{\sum_{c \in \mathcal{C}_B \setminus \{A\}} \Delta(c, A) * \sum_{d \in \mathcal{D}_U, u(d) = c} v_R(d)}_{\text{Ergebnistransfers}}, \tag{4.1}$$

wobei $\Delta(*, *)$ wieder die Netzwerkdistanz zwischen zwei Endsystemen im Netzwerk bezeichnet. Man beachte, dass mit dieser Formel auch die Kosten der klassischen Klient–Server–Lösung bestimmt werden können, indem die Werte für den trivialen Aussendungsplan eingesetzt werden.

Das eigentliche *Aussendungsproblem* besteht nun darin, zu einem gegebenen Szenario einer Informationsfilterung einen Aussendungsplan zu finden, der die obige Kostenfunktion minimiert.

4.2.3 Komplexität des Aussendungsproblems

Leider ist das Aussendungsproblem NP-hart$^{\uparrow}$ in der Zahl der verfügbaren Codeplattformen $|\mathcal{C}|$. Dies lässt sich wie folgt durch eine polynomiale Reduktion des Steinerbaum–Problems zeigen, welches nach [Karp, 1972] NP–vollständig ist. Das Steinerbaum–Problem basiert auf einem ungerichteten, gewichteten Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ und einer Menge von Knoten $\mathcal{N} \subseteq \mathcal{V}$. Das Problem besteht darin, einen minimalen Spannbaum zu finden, der alle Knoten aus $\mathcal{N}$ umfasst. Ausgehend von einem beliebigen Steinerbaum–Problem mit $\mathcal{N} = \{n_1, \ldots, n_k\}$ kann ein äquivalentes Aussendungsproblem folgendermaßen konstruiert werden (die Konstruktion ist in polynomial beschränkter Zeit berechenbar):

$$v_U \equiv 1,\ v_R \equiv 0,\ v_F = 1,\ \mathcal{C} := \mathcal{V},\ A \text{ beliebig} \in \mathcal{N},$$

[2] Zur Spezifikation von Funktionen (hier der Funktion q) wird bisweilen eine Mengenschreibweise verwendet, in der eine Funktion als Menge von Tupeln aus Argument und Funktionswert dargestellt wird. Der Operator $\equiv$ dient zur Definition von konstanten Funktionen die für beliebige Argumente denselben Funktionswert liefern.

$$\mathcal{D}_U := \{d_1, \ldots, d_k\} \text{ mit } \mathcal{D}_U \cap \mathcal{C} = \emptyset,$$
$$\forall a, b \in \mathcal{C} : \Delta(a,b) := \begin{cases} |(a,b)|_{\mathcal{G}} & \text{, falls } (a,b) \in \mathcal{E} \\ \infty & \text{, sonst} \end{cases}$$
$$\forall d_i \in \mathcal{D}_U, \forall c \in \mathcal{C} : \Delta(d_i,c) := \begin{cases} 0 & \text{, falls } c = n_i \\ \infty & \text{, sonst} \end{cases}$$

In dieser speziellen Version des Aussendungsproblems ist jeder Daten–Server mit genau einer Codeplattform verbunden, wobei die Kosten zum Transfer der Daten 0 sind. Dadurch sind die Gesamtkosten des Aussendungsproblems lediglich durch die Kosten zum Transfer des Filterprogrammes bestimmt. Eine optimale Lösung ist gerade der minimale Spannbaum über den Codeplattformen, die mit einem der zu untersuchenden Daten–Server verbunden sind. Also ist jede Lösung des allgemeinen Steinerbaum–Problems gleichzeitig eine Lösung des korrespondierenden speziellen Aussendungsproblems und umgekehrt.

Eine naheliegende Idee zur Lösung des Aussendungsproblems besteht in der Anwendung von Approximationsverfahren zum Steinerbaum–Problem. Allerdings lässt sich das Aussendungsproblem exakt nur durch ein Steinerbaum–Problem für gerichtete Graphen beschreiben. Letzteres kann nach [Zelikovsky, 1997] nicht effizient für kleine Approximationsfaktoren$^{\uparrow}$ gelöst werden.

Schließlich bleibt selbst bei Lösungsverfahren mit polynomialem Zeitaufwand zu untersuchen, wie gut diese für große Problemgrößen mit evtl. tausenden von Codeplattformen und Daten–Servern skalieren.

4.3 Verwandte Ansätze

4.3.1 Reduktion der Kommunikationskosten durch mobile Programme

Es gibt bereits eine ganze Reihe von Anwendungen, in denen mit Hilfe von mobilen Programmen Kommunikationskosten[3] reduziert werden. Prinzipiell gibt es hierzu drei Ansatzpunkte.

[3]Der Begriff *Kommunikationskosten* wird an dieser Stelle ganz allgemein verwandt und nicht nach der speziellen Definition aus dem vorangehenden Abschnitt 4.2.

Reduktion des Datenvolumens

Die erste Möglichkeit besteht darin, die Menge an Daten, die über das Netzwerk übertragen wird, zu reduzieren. Dieser Ansatz wurde z.B. von zwei Anwendungen verfolgt, in denen ein mobiles Programm zur Analyse von Wetterdatenbanken (siehe [Johansen, 1998]) bzw. Web–Servern (siehe [Kato et al., 1999]) verwandt wurde. Beiden Anwendungen gemein ist, dass sie nur Szenarien betrachten, in denen ein einzelner entfernter Daten–Server zu untersuchen ist. Von [Whitebread & Jameson, 1995] werden mobile Agenten in einem komplexeren, militärischen Szenario eingesetzt. Ein Militär, der z.B. Informationen über ein Gebiet sammeln will, stellt einen speziellen Agenten zusammen, der diese Informationen von einer Reihe von Daten–Servern, die nur über ein sehr schmalbandiges Netzwerk verbunden sind, sukzessive einsammelt. Leider wird der Nutzen der sukzessiven Bearbeitung der Daten–Server nicht erläutert. Da der Agent das auf seinem Weg erworbene Wissen bei später besuchten Daten–Servern nicht anwendet, erscheint ein direktes Zurücksenden der Teilergebnisse oder auch ein paralleles Aussenden mehrerer Agenten zu den einzelnen Daten–Servern sinnvoller. Eine Abschätzung des erreichbaren Performanzgewinns erfolgt nicht.

Reduktion der Zahl an Interaktionen

Die zweite prinzipielle Möglichkeit zur Reduktion von Kommunikationskosten besteht darin, die Zahl der Interaktionen zwischen über das Netzwerk verteilten Rechnern zu minimieren. Ein derartiger Ansatz wurde zum Beispiel von [Dasgupta et al., 1999] im Kontext des elektronischen Handels vorgeschlagen. Hierin können Kunden einen mobilen Agenten zu einem oder mehreren Händlern schicken, um mit diesen über die Bedingungen einer angestrebten Handelstransaktion zu verhandeln. Zur Erfüllung eines Auftrages oder auch zur Festlegung eines Angebotes müssen Händler u.U. ihrerseits wieder als Kunden bei anderen Händlern auftreten. Leider werden mögliche Performanzgewinne weder quantifiziert noch wird ein allgemeines Modell aufgestellt, mit dem man entscheiden könnte, wann die Aussendung mobiler Programme sinnvoller als der klassische Klient–Server–Ansatz ist.

Reduktion der Kommunikationsdistanz

Die dritte Möglichkeit besteht darin, die Netzwerkdistanz zwischen Kommunikationspartnern zu reduzieren. Eine Anwendung, die dieser Idee folgt, wurde von [Ranganathan et al., 1997] vorgestellt. Hierin wird ein *chat*–Server als mobiles Programm ausgelegt, das sich zwischen den Rechnern seiner Klienten bewegen kann. Die Klienten messen permanent die Latenzzeit zu allen anderen Klienten und senden die Ergebnisse an eine zentrale Stelle,

von der die optimale Platzierung des *chat*–Servers berechnet wird, derart dass die maximal Latenzzeit zwischen Server und einem Klienten minimiert wird. Auf diese Weise kann die Platzierung des Servers bei sich ändernden Netzwerkcharakteristiken ständig angepasst werden. Es ist jedoch fraglich, inwieweit die Methode der kompletten Distanzmessung zwischen allen Klienten für große *chat*–Gruppen skaliert. Die Experimente basieren lediglich auf Gruppen von vier Klienten.

Performanzmodelle

Von [Straßer & Schwehm, 1997] wurde ein allgemeines Performanzmodell entwickelt, das die Kosten einzelner Interaktionen (entfernter Prozeduraufruf oder Migration plus lokaler Prozeduraufruf) sowie einer ganzen Folge von Interaktionen beschreibt. Die Netzwerkcharakteristiken wie Latenzzeit und Bandbreite sowie das Datenvolumen von Anfragen und Ergebnissen werden als bekannt vorausgesetzt. Im Gegensatz zum Aussendungsproblem wird eine feste Zuordnung von Daten–Servern zu Agentenplattformen angenommen, und die Reihenfolge der einzelnen Interaktionen ist ebenso festgelegt.

[Baldi & Picco, 1998] untersuchen ein spezielles Netzwerk–Management–Szenario, in dem eine zentrale Station und n Gerätestationen existieren. Das betrachtete, statische Interaktionsmuster umfasst ein Programm, das bei der Zentrale vorliegt, und Daten bei den n Gerätestationen, die mit dem Programm analysiert werden sollen. Untersucht werden die Kosten im klassischen Klient–Server–Fall, bei dem alle Daten zur Zentrale übertragen werden, und in zwei Szenarien, in denen mobile Programme eingesetzt werden. Im ersten von ihnen werden parallel n Programme zu den jeweiligen Gerätestationen gesandt und die Ergebnisse werden direkt zurückgeschickt. Im zweiten Fall besucht ein Programm nacheinander alle Gerätestationen, sammelt so alle relevanten Daten und sendet sie von der letzten Station zurück zur Zentrale. Die aufgestellten Modelle werden anhand von Fallstudien an kleinen Netzwerken evaluiert. Auch die von [Baldi & Picco, 1998] untersuchten Szenarien setzen im Gegensatz zum Aussendungsproblem eine feste Reihenfolge der einzelnen Interaktionen voraus.

4.3.2 Aussendungsverfahren

Das zur Aussendung mobiler Programme analoge Problem besteht in der Server-initiierten Replikation von Dokumenten hin zu großen Klientengruppen. [Bestavros, 1997] schlägt hierzu ein Verfahren vor, mit dem populäre Dokumente entlang einer Hierarchie von

Replikationsservern repliziert werden. Auf jeder Stufe dieser Hierarchie erfolgt eine Optimierung, derart dass möglichst viele der zu erwartenden zukünftigen Anfragen direkt beantwortet werden können. Eine Hierarchie muss für jeden Daten–Server von Hand aufgebaut werden. Netzwerkdistanzen werden nicht beachtet. Im Gegensatz zum Aussendungsproblem wird für alle auszusendenden Daten ein fester Weg angenommen. Die Optimierung wirkt sich lediglich darauf aus, in welchen Pfaden der Hierarchie die Daten wie weit ausgesandt werden.

[Gwertzman & Seltzer, 1995] schlagen eine Partitionierung aller Klienten in Regionen von geographisch möglichst nahe beieinander liegenden Klienten bzw. Rechnern vor. Für jede Region wird dann individuell entschieden, ob sie ein Replikat vom Basisserver aus erhalten soll. Im Gegensatz hierzu werden beim Aussendungsproblem Abhängigkeiten zwischen einzelnen Replikations- bzw. Transferentscheidungen beachtet. Als Quelle eines Transfers kommen alle Codeplattformen in Betracht, die das mobile Programm im Rahmen der Aussendung erhalten. Daneben können Daten–Server auch von Programmen, die in andere Regionen gesandt werden, untersucht werden. Schließlich wird in dem in Abschnitt 4.5 vorgestellten Verfahren eine Hierarchie von Regionen verwendet, wodurch die Berechnung eines Aussendungsplanes wesentlich besser skaliert als bei einer einstufigen Regionenmenge.

Neben der Dokumentenreplikation scheint auch das Problem des Multicast–Routings (siehe z.B. [Tanenbaum, 1996, Seite 372 ff.]) Gemeinsamkeiten mit dem Aussendungsproblem zu haben. Hier wie dort gilt es, eine Nachricht von einem Sender zu mehreren Empfängern zu senden, abgesehen davon dass beim Aussendungsproblem die Empfänger (d.h. die zu benutzenden Codeplattformen) erst noch bestimmt werden müssen. Im Gegensatz zum Aussendungsproblem, das sich für jede Kombination von Daten–Servern und Filterfunktion neu stellt, bleibt eine Multicast–Gruppe jedoch über einen längeren Zeitraum (über viele Nachrichten) hinweg gleich bzw. ändert sich nur wenig. Auch wenn einzelne Mitglieder wechseln, so bleibt doch die Gruppe als Ganzes und ihre Adresse erhalten. Daher können Lösungen zum Multicast–Routing schrittweise an die Netzwerkbedingungen angepasst und somit optimiert werden. Der Aufbau einer neuen Multicast–Gruppe für jedes Aussendungsproblem würde eine extreme Netzwerkbelastung bedeuten, da die Gruppenmitglieder in aller Regel weit über das Netzwerk verteilt sind. Der Aufbau einer vordefinierten Multicast–Gruppe für jede mögliche Kombination von zu besuchenden Codeplattformen hingegen würde eine in der Zahl der Codeplattformen exponentielle Zahl von Gruppen erfordern. Schließlich bliebe bei beiden Vorgehensweisen das Problem, zu einem konkreten Aussendungsproblem die passende Gruppe von Codeplattformen zu be-

stimmen.

4.4 Allgemeines Aussendungsverfahren

In diesem Abschnitt wird nun ein einfacher Algorithmus zur Berechnung von effizienten Aussendungsplänen vorgestellt. Dieser, im Folgenden *Basisalgorithmus* genannt, löst das NP–harte Aussendungsproblem in polynomialer Zeit. Die von ihm bestimmte Lösung ist natürlicherweise nicht optimal, erweist sich in den Experimenten in Kapitel 6 aber trotzdem als effizient. Im nachfolgenden Abschnitt 4.5 wird dann ein zweites Verfahren zur Berechnung von Aussendungsplänen vorgestellt, welches besser mit der Zahl der verfügbaren Codeplattformen skaliert. Beide Verfahren erfordern die Kenntnis aller Parameter, die das Aussendungsproblem charakterisieren, inklusive der relevanten Netzwerkdistanzen.

Der Basisalgorithmus zur Berechnung von suboptimalen, aber trotzdem effizienten Aussendungsplänen folgt einer Greedy–Heuristik$^{\uparrow}$. Er versucht eine bereits gefundene Lösung schrittweise auszubauen, indem zur Menge der mit dem Filterprogramm zu besuchenden Codeplattformen eine weitere Codeplattform hinzufügt wird, derart dass die neue Codeplattform den größten Nutzen in Bezug auf die resultierenden Kommunikationskosten verspricht.

Die initiale Lösung, von der der Basisalgorithmus ausgeht, ist der triviale Aussendungsplan, in dem alle Daten–Server von der Ausgangsplattform aus untersucht werden. Er wird durch $\mathcal{C}_B = \{A\}$, $u \equiv A$ und $q = \emptyset$ beschrieben.

Ausgehend von einem beliebigen Aussendungsplan (charakterisiert durch $\mathcal{C}_B$, u und q) kann ein optimal erweiterter Aussendungsplan ($\mathcal{C}'_B$, u' und q'), der die zusätzliche Codeplattform $\tilde{c} \notin \mathcal{C}_B$ enthält, folgendermaßen bestimmt werden:
Zunächst wird $\tilde{c}$ in die Menge der besuchten Codeplattformen integriert:

$$\mathcal{C}'_B := \mathcal{C}_B \cup \{\tilde{c}\}.$$

Danach wird geprüft, ob irgendwelche Daten–Server vorzugsweise von dieser neuen Codeplattform aus untersucht werden sollten:

$$\forall d \in \mathcal{D}_U : u'(d) := \begin{cases} u(d)\,, \text{falls } \Delta(d,u(d)) * v_U(d) + \Delta(u(d),A) * v_R(d) \\ \qquad\qquad \leq \Delta(d,\tilde{c}) * v_U(d) + \Delta(\tilde{c},A) * v_R(d) \\ \tilde{c} \quad\;\; \text{, sonst.} \end{cases}$$

Schließlich wird die zu $\tilde{c}$ nächstgelegene Codeplattform bestimmt, die das Filterprogramm im Rahmen des bisherigen Aussendungsplanes bereits erhalten hat und von der das Programm zur neuen Plattform transferiert werden kann:

$$q' := q \cup \{(\tilde{c}, c_{\text{quelle}})\} \qquad \text{, mit } c_{\text{quelle}} \in \mathcal{C}_B \text{ so dass } \Delta(\tilde{c}, c_{\text{quelle}}) = \min_{c \in \mathcal{C}_B} \Delta(\tilde{c}, c).$$

Anhand der Kostenfunktion 4.1 können die mit einem jeden Aussendungsplan verbundenen Kommunikationskosten berechnet werden. Ebenso ist ein Vergleich der Effizienz zweier Aussendungspläne möglich. Damit kann der Basisalgorithmus zur Berechnung von Aussendungsplänen vollständig beschrieben werden. Er erweitert den trivialen Aussendungsplan solange um die jeweils bestmögliche Codeplattform, bis keine Verringerung der Kommunikationskosten mehr möglich ist. Abbildung 4.2 gibt den gesamten Algorithmus wieder, wobei der triviale Aussendungsplan (Schritt 1) und seine Erweiterungen (Schritt 7) jeweils durch die Menge $\mathcal{C}_B$ angedeutet werden.

---

procedure basisverfahren (Codeplattformen $\mathcal{C}$, Daten–Server $\mathcal{D}$, Ausgangsplattform A)

(1) nehme trivialen Aussendungsplan als initiale Lösung: $\mathcal{C}_B := \{A\}$
(2) **loop**
(3) **forall** $c \in (\mathcal{C} \setminus \mathcal{C}_B)$:
(4) berechne um c erweiterten Aussendungsplan
(5) wähle $c \in (\mathcal{C} \setminus \mathcal{C}_B)$ so dass Kommunikationskosten des um c erweiterten Aussendungsplanes minimal
(6) **if** (Aussendungsplan mit c besser als aktueller Plan))
(7) füge c zum aktuellen Aussendungsplan hinzu ($\mathcal{C}_B := \mathcal{C}_B \cup \{c\}$)
(8) **else**
(9) Abbruch, da keine weitere Verbesserung möglich
(10) **endloop**

Abbildung 4.2: Basisalgorithmus zur Aussendungsplanberechnung

Die Bewertung der mit dem Basisverfahren erreichten Reduktion der Kommunikationskosten erfolgt in Kapitel 6.

Berechnungskomplexität

Die Berechnung eines erweiterten Aussendungsplanes und der mit ihm verbundenen Kommunikationskosten erfordert $O(|\mathcal{D}_U| + |\mathcal{C}_B|)$ Berechnungsschritte. In jeder Iteration der

Hauptschleife des Basisalgorithmus werden $|\mathcal{C}| - |\mathcal{C}_B|$ erweiterte Aussendungspläne berechnet. Damit hat die Berechnung eines Aussendungsplanes, der m Transfers des Filterprogrammes beinhaltet, den Aufwand

$$\sum_{i=1}^{m+1} (|\mathcal{C}| - i) * O(|\mathcal{D}_U| + i).$$

Eine obere Schranke dieses Aufwandes, welche für $m \ll |\mathcal{C}|$ ziemlich genau ist, beträgt

$$(m+1) * |\mathcal{C}| * O(|\mathcal{D}_U| + m) \; = \; O(m * |\mathcal{C}| * (|\mathcal{D}_U| + m)). \tag{4.2}$$

Da maximal $|\mathcal{C}|$ Filtertransfers erfolgen können, entsteht im schlechtesten Fall ein Aufwand von

$$O(|\mathcal{C}|^2 * (|\mathcal{D}_U| + |\mathcal{C}|)). \tag{4.3}$$

Die Experimente in Kapitel 6 zeigen jedoch, dass dieser Extremfall in der Regel nicht einmal annähernd erreicht wird.

Man sieht, dass mit dem sehr einfachen Basisalgorithmus bereits ein recht effizientes Verfahren zur Berechnung von Aussendungsplänen gefunden wurde. Sind jedoch sehr viele Codeplattformen verfügbar, so kann der Berechnungsaufwand insbesondere bei vielen zu untersuchenden Daten–Servern trotzdem sehr groß werden. Betrachtet man z.B. ein Szenario mit 100 Daten–Servern, 1000 Codeplattformen sowie 20 Filtertransfers, so beträgt der Aufwand des Basisverfahrens bereits 2,4 Millionen Berechnungsschritte. Dieser Aufwand ist insbesondere für Server von Bedeutung, die den Dienst der Berechnung von Aussendungsplänen anbieten und die möglichst viele Anfragen in möglichst kurzer Zeit beantworten sollen.

4.5 Hierarchisches Aussendungsverfahren

Um die Skalierbarkeit des Basisalgorithmus bezüglich der Zahl der verfügbaren Codeplattformen zu verbessern, wurde ein *hierarchischer Algorithmus* entworfen, der einer Teile–und–Herrsche Strategie folgt und hierzu die hierarchischen Regionen von Netzwerk–Distanzkarten (vgl. mit Kapitel 3) ausnutzt. Zur Erinnerung sei deren Struktur kurz wiederholt: Eine Distanzkarte repräsentiert ein Computernetzwerk durch eine Hierarchie von Regionen, den sogenannten Netzwerkbaum.[4] Jedes Endsystem des Netzwerkes (und

[4] Im Gegensatz zu Kapitel 3 werden im Kontext der Aussendungsverfahren auch Mess–Server als Regionen bezeichnet.

somit auch jeder Daten–Server und jede Codeplattform) wird genau einem vollständigen Pfad in diesem Netzwerkbaum zugeordnet, es gehört also gleichzeitig mehreren Regionen an, die untereinander eine Inklusionskette bilden. Ein Netzwerkbaum ermöglicht die Distanzschätzung zwischen beliebigen Regionen und Endsystemen.

Die Idee des hierarchischen Algorithmus besteht darin, aus dem Netzwerkbaum der Netzwerk–Distanzkarte einen modifizierten Baum zu berechnen, dessen Regionen die Verteilung der Codeplattformen widerspiegeln. Die Berechnung erfolgt anhand der folgenden Transformationsregeln:

1. Mit jeder Region werden die in ihr enthaltenen Daten–Server als ein *virtueller Daten–Server* assoziiert.
2. Ebenso werden mit jeder Region die in ihr enthaltenen Codeplattformen als eine *virtuelle Codeplattform* assoziiert.
3. Daten–Server bzw. Regionen werden aus dem Baum entfernt, wenn mit der übergeordneten Region keine Codeplattform assoziiert ist.
4. Regionen, die genau eine Codeplattform enthalten, werden zu Blattknoten, die ggf. um einen (virtuellen) Daten–Server erweitert werden. D.h. ihre ursprünglichen Kindknoten werden aus dem Baum entfernt.

Im modifizierten Baum werden fortan lediglich die inneren Knoten als Regionen bezeichnet.

Abbildung 4.3 zeigt ein Beispiel zur Transformation eines Netzwerkbaums gemäß den obigen Regeln. Anstelle der ursprünglichen 10 Regionen sind hierin gemäß der Regeln (3) und (4) nur noch drei Regionen (innere Baumknoten) enthalten. Die Daten–Server 1 und 2 sind zu einem virtuellen Server 1-2 zusammengefasst und auf nächsthöherer Ebene zusammen mit Server 3 weiter zu einem virtuellen Server 1-3. Auf gleiche Weise werden die Codeplattformen B, C und A zu virtuellen Codeplattformen zusammengefasst, mit dem Unterschied dass sie auch als einzelne Blattknoten bestehen bleiben.

Der hierarchische Algorithmus traversiert diesen modifizierten Baum und wendet den Basisalgorithmus aus dem vorangehenden Abschnitt 4.4 für jede Region d.h. für die in ihr enthaltenen (virtuellen) Daten–Server und Codeplattformen getrennt an. Distanzen zwischen virtuellen Codeplattformen ergeben sich direkt aus der Distanzschätzung zwischen den jeweils zugeordneten Regionen. Die Kommunikationskosten zum Herunterladen der Daten eines virtuellen Daten–Servers auf eine (virtuelle) Codeplattform ergeben sich als Summe aller Kommunikationskosten für die einzelnen Daten–Server, wobei die Distanzen

zu diesen ebenfalls aus dem Netzwerkbaum bestimmt werden. Das Basisverfahren wird dann entsprechend angepasst und mit virtuellen Codeplattformen und Daten–Servern durchgeführt.

Die Traversierung des modifizierten Baumes startet bei der Ausgangsplattform und wird mittels der beiden folgenden elementaren Aktionen durchgeführt:

1. Für jede Region, die die Ausgangsplattform A umfasst, wird der angepasste Basisalgorithmus ausgeführt. Hierzu wird auf unterster Ebene A und auf den höheren Ebenen die entsprechende A übergeordnete Region als (virtuelle) Ausgangsplattform interpretiert.

2. Wird für eine virtuelle Codeplattform c beschlossen, dass diese das Filterprogramm erhalten soll, so wird diese Entscheidung anhand der c entsprechenden Region r weiter verfeinert. Dazu wird für alle Kindregionen von r (bzw. die r zugeordneten Codeplattformen) jeweils angenommen, dass diese das Filterprogramm bekommen, und es werden die hieraus resultierenden Kommunikationskosten berechnet. Beispielsweise müsste eine Verfeinerung der virtuellen Codeplattform A-C in Abbildung 4.3 die Plattformen A und B-C berücksichtigen.
   Danach wird die Kindregion/Codeplattform $\tilde{r}$ mit den geringsten Kosten ausgewählt und das angepasste Basisverfahren wird für Region r ausgeführt, wobei die Kindregion/Codeplattform $\tilde{r}$ als Ausgangsplattform fungiert. In Fortsetzung des obigen Beispiels würde bei einer Auswahl der Codeplattform A also das angepasste Basisverfahren mit den Plattformen/Servern A, B-C, 1-2 und 3 durchgeführt, wobei A als Ausgangsplattform fungieren würde.

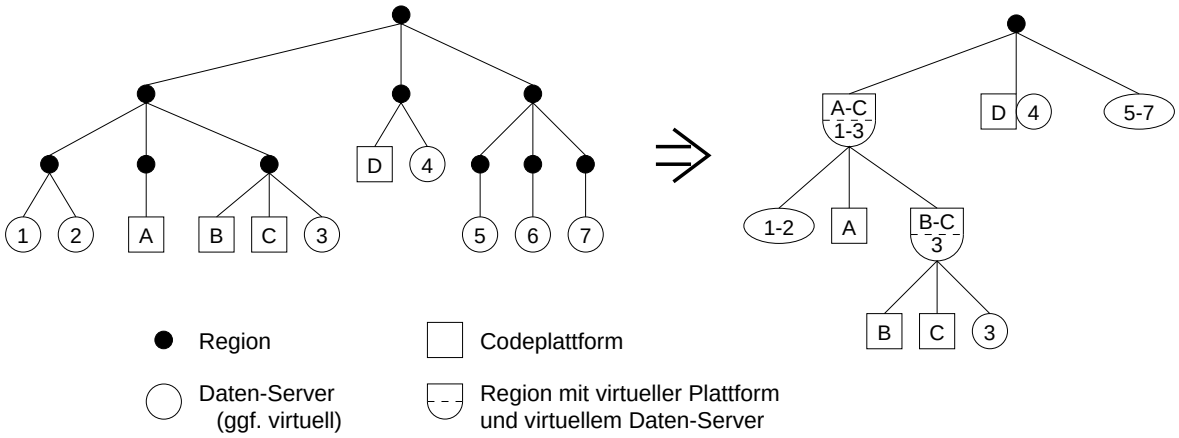


Abbildung 4.3: Beispiel zur Transformation eines Netzwerkbaums (links) in einen modifizierten Baum aus virtuellen Daten-Servern und Codeplattformen (rechts)

Ist $\tilde{r}$ eine Region, so wird die Entscheidung über den Transfer des Filterprogrammes solange weiter verfeinert, bis eine eindeutige Codeplattform ausgewählt werden kann.

Eine konkrete Traversierung des in Abbildung 4.3 auf Seite 85 dargestellten modifizierten Baumes könnte wie folgt aussehen: Ausgangsplattform sei D. Es sollen alle Daten–Server untersucht werden. Dann wird zunächst der Basisalgorithmus für den Wurzelknoten ausgeführt, d.h. mit der Ausgangsplattform D, der virtuellen Codeplattform A-C sowie den (virtuellen) Daten–Servern 1-3, 4 und 5-7. Angenommen das Basisverfahren führt zur Entscheidung, zu A-C ein weiteres Filterprogramm zu senden und die Server 5-7 von Plattform D aus zu untersuchen. Die anderen Server werden von den jeweils in ihrer Region liegenden Codeplattformen aus untersucht, also 1-3 vin A-C und 4 von D. Die Entscheidung ein Filterprogramm zu A-C zu senden muss dann verfeinert werden. Es werden hierzu die Varianten verglichen, einen Filter zur Plattform A oder B-C zu senden, wobei jeweils die Kosten zur Analyse der Daten–Server 1,2 und 3 bestimmt werden. Angenommen A erweise sich als die bessere Variante, dann wird das Basisverfahren nun mit der Ausgangsplattform A, der Codeplattform B-C und den Daten–Servern 1-2 und 3 durchgeführt. Es führe zur Entscheidung 1-2 von A aus zu untersuchen und einen weiteren Filter zu B-C zu senden (dieser wird nun allerdings nicht von der Ausgangsplattform D transferiert sondern von der nähergelegenen Plattform A). Der vollständige Aussendungsplan, der denätzlichen Transfer des Filters zur Plattform C vorsieht, wird formal folgendermaßen spezifiziert: $\mathcal{C}_B := \{\mathsf{A}, \mathsf{C}, \mathsf{D}\}$, $q := \{(\mathsf{A},\mathsf{D}), (\mathsf{C},\mathsf{A})\}$ und $ex := \{(1,\mathsf{A}), (2,\mathsf{A}), (3,\mathsf{C}), (4,\mathsf{D}), (5,\mathsf{D}), (6,\mathsf{D}), (7,\mathsf{D})\}$.

Abbildung 4.4 gibt das komplette Verfahren der Baumtraversierung nochmals wieder. Man beachte, dass in den Schritten (7, 8, 13) je nach Baumstruktur reale oder virtuelle Codeplattformen und Daten–Server verwendet werden.

Die Analyse der mit dem hierarchischen Algorithmus erreichbaren Effizienz von Aussendungsplänen erfolgt empirisch in Kapitel 5. Insbesondere wird dort untersucht, inwieweit sich die Abstraktion auf virtuelle Codeplattformen und Daten–Server als nachteilig gegenüber dem Basisalgorithmus erweist.

Berechnungskomplexität

Die folgende Diskussion benutzt zwei Parameter, um die Größe eines modifizierten Baumes zu beschreiben: t bezeichnet die Baumtiefe und k den Baumgrad, d.h. die maximale Zahl von Kindknoten, die ein Knoten haben kann. Es wird angenommen, dass jeder Blattknoten genau eine Codeplattform (evtl. inklusive einiger Daten–Server) repräsentiert. In

```
procedure hierarchisches_verfahren (Ausgangsplattform A)
(1)   R := speziellste A zugeordnete Region
(2)   while (R ≠ null)
(3)       starte angepasstes_basisverfahren(R, A)
(4)       A := R // nächste virtuelle Ausgangsplattform
(5)       R := Elternregion von R
(6)   endwhile
procedure angepasstes_basisverfahren (Region R, (virtuelle) Ausgangsplattform A)
(7)   bestimme (virtuelle) Codeplattformen C für alle Nachbarn (Regionen) von A
(8)   bestimme (virtuelle) Daten–Server D für alle Nachbarn (Regionen) von A sowie
                  Kommunikationskosten bezüglich aller Codeplattformen aus C
(9)   starte basisverfahren(C, D, A)
(10)  forall (neue Zielplattform c)
(11)      if (Zielplattform ist virtuell)
(12)          starte verfeinerung (der c zugeordneten Region)
procedure verfeinerung (Region R)
(13)  bestimme (virtuelle) Daten–Server D und Codeplattformen C für alle
                                                  Kinder (Regionen) von R
(14)  forall (c ∈ C)
(15)      berechne Kosten zum Herunterladen aller Daten von Servern aus D
                                                  auf Plattform c
(16)  wähle c ∈ C, so dass Kosten minimal
(17)  starte angepasstes_basisverfahren(R, c)
(18)  if (c ist virtuell)
(19)      starte verfeinerung(der c zugeordneten Region)
```

Abbildung 4.4: Hierarchischer Algorithmus zur Aussendungsplanberechnung

einem vollständig ausgeglichenen modifizierten Baum, wo also alle inneren Baumknoten genau k Kindknoten und alle Blattknoten dieselbe Tiefe haben, gilt $t = \log_k |\mathcal{C}|$.[5].

Zur Optimierung der Rechenzeit des hierarchischen Algorithmus wird eine Vorberech-

[5]Prinzipiell ist es natürlich möglich, dass eine Blattregion mehr als k Codeplattformen umfasst. Im schlimmsten Fall befinden sich sogar alle Codeplattformen und alle zu untersuchenden Daten–Server innerhalb derselben Blattregion, womit der Aufwand identisch zu dem des Basisalgorithmus wird. Dieser Extremfall soll hier aber nicht beachtet werden, sondern es werden maximal k Codeplattformen pro Blattregion angenommen. Der Fall ließe sich auch dadurch umgehen, dass jede Codeplattform zusätzlich als Mess–Server fungiert.

nung durchgeführt, die für alle Baumknoten (Regionen und Codeplattformen) den zu ihnen gehörenden virtuellen Daten–Server mitsamt der Kommunikationskosten bezüglich benachbarter Knoten berechnet. Hierbei werden natürlich nur die für ein konkretes Aussendungsproblem relevanten Daten–Server betrachtet. Jeder Daten–Server ist in maximal t virtuellen Servern enthalten. Die Kommunikationskosten pro Daten–Server und pro Baumknoten müssen maximal $k-1$ mal berechnet werden, da jeder Knoten maximal k Kindknoten hat und einer von diesen immer als Ausgangsplattform fungiert. Somit beträgt der Gesamtaufwand dieser Vorberechnung $O(t * k * |\mathcal{D}_U|)$.

Als ein Erweiterungsschritt wird im Folgenden die Durchführung eines einzelnen Schleifendurchlaufes des auf virtuelle Codeplattformen und virtuelle Daten–Server angepassten Basisalgorithmus bezeichnet. Die komplette Schleife verbirgt sich hinter dem Kommando der Zeile 9 aus Abbildung 4.4, ein Schleifendurchlauf besteht dann aus den Schritten 3–9 des in Abbildung 4.2 auf Seite 82 dargestellten Basisverfahrens. Da der Grad eines Regionenbaumes auf k beschränkt ist, hat die Durchführung eines Erweiterungsschrittes den Aufwand $O(k^2)$.

Selbst wenn die Ausführung des hierarchischen Algorithmus zu keinen Transfers des Filterprogrammes führt, müssen t Erweiterungsschritte durchgeführt werden, die jeweils die wahre Ausgangsplattform bzw. deren übergeordneten Regionen als Ausgangsplattform betrachten. Für jede zusätzliche Transferentscheidung zu einer (virtuellen) Codeplattform muss diese Entscheidung maximal t mal verfeinert werden, wobei jeder Verfeinerungsschritt den gleichen Aufwand wie ein Erweiterungsschritt hat. Zusätzlich muss nach jedem Verfeinerungsschritt ein Erweiterungsschritt mit der jeweiligen (virtuellen) Codeplattform als Ausgangsplattform durchgeführt werden. Führt eine Traversierung des Regionenbaumes zu m Transfers des Filterprogrammes, so sind hierzu also maximal

$$t + m * (t + t) \; = O\,(m * t)$$

Erweiterungs- bzw. Verfeinerungsschritte notwendig.

Fasst man diese Ergebnisse zusammen, so erhält man als obere Schranke des Aufwandes zur Regionenbaumtraversierung

$$O\,(\; \underbrace{t \;*\; k \;*\; |\mathcal{D}_U|}_{\text{Vorberechnung}} + \underbrace{k^2 \;*\; m \;*\; t}_{\text{Erweiterungen}} \;) = O(t * k * (|\mathcal{D}_U| + m * k)). \tag{4.4}$$

Bei der maximal möglichen Zahl an Filtertransfers muss der komplette Regionenbaum traversiert werden, wobei für jeden inneren Baumknoten $(k-1)$ Erweiterungsschritte

durchzuführen sind. Unter der Annahme, dass jeder innere Knoten des Regionenbaumes genau k Kindknoten hat, besteht der komplette Baum aus $(|\mathcal{C}|-1)/(k-1)$ inneren Knoten. Der Gesamtaufwand beträgt dann

$$O(t * k * |\mathcal{D}_U| \; + \; |\mathcal{C}| * k^2). \tag{4.5}$$

Diese Formeln zeigen, dass im Vergleich zum Basisalgorithmus (siehe Gleichungen 4.2 und 4.3 auf Seite 83) das hierarchische Verfahren wesentlich effizienter ist. Bei ausgeglichenen Bäumen und gegebener Zahl an Filtertransfers hängt der Aufwand nur noch logarithmisch von der Zahl der verfügbaren Codeplattformen ab (über den Parameter t). Daneben ist nun der von der Zahl der Daten–Server abhängige Aufwand unabhängig von der Zahl der resultierenden Filtertransfers. Dies wirkt sich insbesondere positiv auf den Extremfall mit der maximalen Zahl an Filtertransfers auf. Hier ist der Aufwand des hierarchischen Verfahrens (Gleichung 4.5) um Größenordnungen geringer als der des Basisverfahrens (Gleichung 4.3).

Zur Illustration dieser Ergebnisse sei noch die Beispielrechnung zum Basisverfahren für das hierarchische Verfahren fortgesetzt. Nimmt man also wieder 100 Daten–Servern, 1000 Codeplattformen, einen Regionenbaum mit $k = 10$ und $t = 3$ sowie 20 Filtertransfers, so beträgt der Aufwand des hierarchischen Verfahrens etwa 9000 Berechnungsschritte, der des Basisverfahrens dagegen 2,4 Millionen Schritte.

4.6 Diskussion und Erweiterungen

Die in diesem Kapitel vorgestellten Verfahren ermöglichen einen sehr flexiblen und gleichzeitig effizienten Einsatz mobiler Programme zur Erledigung einer Aufgabe der verteilten Informationsfilterung. Im Gegensatz zu herkömmlichen Verfahren werden dabei Kommunikationskosten explizit modelliert und optimiert. Dabei wird kein festes Interaktionsmuster vorgegeben, sondern es wird jeweils die für eine konkrete Filteraufgabe am besten geeignete Mischung von Programmtransfers und klassischem Herunterladen der relevanten Daten gewählt.

In den aufgestellten Funktionen über die Kommunikationskosten finden die Kosten zur Bestimmung der relevanten Netzwerkdistanzen (z.B. durch eine Netzwerk–Distanz–Karte) keine Beachtung. Allerdings lassen sich diese Kosten auch kaum in Beziehung zueinander setzen, da erhobene Netzwerkdistanzen auch von zahlreichen anderen Anwendungen nutzbringend eingesetzt werden können. Kapitel 3.1.1 beschreibt einige solche Anwendungen.

Daneben hängt der Aufwand zur Erhebung von präzisen und aktuellen Distanzwerten noch wesentlich vom gewählten Distanzmaß ab (siehe hierzu auch Kapitel 3.3.2) und kann daher nicht allgemein quantifiziert werden.

Auf zwei spezielle Anwendungs- bzw. Erweiterungsmöglichkeiten der vorgestellten Verfahren soll in den nachfolgenden Abschnitten noch kurz eingegangen werden.

Eine interessante Forschungsaufgabe für die Zukunft bestünde in der Optimierung des Einsatzes mobiler Programme in anderen Aufgabenbereichen, z.B. in Szenarien, in denen mobile Programme einen unvollständig bzw. mit Randbedingungen spezifizierten Weg (engl. *itinerary*) ablaufen müssen.

4.6.1 Allgemeine und effiziente 1:n–Kommunikation

Die vorgestellten Verfahren zur Berechnung von Aussendungsplänen sind nicht nur zur Aussendung mobiler Filterprogramme, sondern auch in anderen Kontexten nutzbringend einsetzbar. Ganz allgemein lässt sich die Kommunikation in beliebigen Szenarien, in denen eine Nachricht von einem Sender zu mehreren Empfängern geschickt werden soll, mit den vorgestellten Algorithmen optimieren. Eine Anwendung dieser Verfahren ist sogar dann möglich, wenn die Empfänger der Nachricht nicht explizit bekannt sind, sondern nur eine Menge von Endsystemen gegeben ist, in deren „Richtung“ die Nachricht geschickt werden soll. Dieser Fall tritt zum Beispiel im Aussendungsproblem für mobile Programme, aber auch bei der Aussendung von zu replizierenden Dokumenten auf.

Um die vorgestellten Algorithmen für ein Szenario der 1:n–Kommunikation anwenden zu können, bedarf es lediglich einiger spezieller Endsysteme, die Nachrichten weiterleiten können und die damit die Rolle der Codeplattformen aus dem Aussendungsproblem übernehmen. Die Empfänger der Nachricht übernehmen die Rolle der Daten–Server. Bei entsprechender Anpassung der Kostenfunktion (vgl. mit Gleichung 4.1 auf Seite 76) können die Verfahren zur Berechnung von Aussendungsplänen dann direkt angewandt werden.

Bei Empfängergruppen, die über zahlreiche Nachrichten hinweg ungefähr gleich bleiben, und bei sehr großen Empfängergruppen ist Multicast–Technologie natürlich effizienter als die Aussendungsverfahren, da die Kommunikationsoptimierung auf Netzwerk- und nicht auf Anwendungsebene stattfindet. Dies gilt allerdings nur, wenn die Empfänger einer Nachricht explizit bekannt sind.

4.6.2 Zusammenspiel mit Caching–Verfahren

Ein prinzipieller Kritikansatz gegen die Aussendung mobiler Filterprogramme besteht darin, dass deren Aussendung durch den Einsatz von großen Netzwerk–Caches (siehe z.B. [Chankhunthod et al., 1996]) nahezu wirkungslos bleibt oder sich sogar nachteilig auswirken kann. Während die mobilen Programme vom Klienten zu den Datenquellen wandern, werden die zu untersuchenden Daten durch Caching–Verfahren in Richtung der Klienten gebracht. Wird dieser Effekt nicht beachtet, so kann der Einsatz eines mobilen Filterprogrammes letztlich sogar teurer kommen, da die in nahegelegenen Caches vorhandenen Kopien der gesuchten Daten nicht beachtet bzw. genutzt werden.

Um aus beiden Mechanismen den optimalen Nutzen zu ziehen, muss daher ein mobiles Filterprogramm zunächst die jeweils relevanten Cache–Speicher durchsuchen, bevor es letztlich zu den ursprünglichen Datenquellen transferiert wird. Hierbei wird angenommen, dass die einem beliebigen Endsystem zugeordneten Cache–Speicher eine lineare Liste bilden, dass also bei einer beliebigen Dokumentenanfrage zunächst der erste Cache–Speicher konsultiert wird, bei Misserfolg dann der zweite Speicher usw. Diese Annahme ist insbesondere für hierarchische Caching–Systeme erfüllt, welches die momentan am Weitesten verbreitete Architektur von Caching–Systemen ist (siehe z.B. [Chankhunthod et al., 1996]). Im Einzelnen funktioniert das Zusammenspiel zwischen Filterprogramm und Cache–Speichern wie folgt: Das Filterprogramm wird mit seiner Liste von zu untersuchenden Daten–Servern bzw. Dokumenten zunächst zu dem lokal zugeordneten Cache–Speicher transferiert. Gesuchte Dokumente, die in diesem Speicher vorhanden sind, werden vom Filterprogramm geladen, analysiert und das Ergebnis wird an die Ausgangsplattform zurückgesandt. Gibt es noch weitere zu untersuchende Dokumente, die nicht im Cache–Speicher verfügbar waren, so wird das Filterprogramm zum nächsten Cache–Speicher transferiert und fährt dort mit seiner Untersuchung fort. Dies wird bis maximal zum letzten Speicher in der Cache–Speicher–Liste fortgeführt. Das Aussendungsverfahren wird schließlich für die Dokumente durchgeführt, die in keinem der besuchten Cache–Speicher gefunden wurden.

Es ist durchaus möglich, dass dieses Zusammenspiel mit Caching–Systemen den mit mobilen Programmen erzielbaren Performanzgewinn noch weiter steigert, da diese mit einem einzigen Transfer zu einem Cache–Speicher häufig schon eine ganze Reihe von Dokumenten lokal vorfinden. Im Falle von weit verteilten Daten–Servern ist dieses Verhältnis zwischen Anzahl der Programmtransfers und lokal damit erreichbaren Dokumenten um Einiges schlechter.

Schließlich gibt es im Falle des Internets zwei Gründe dafür, dass trotz der Caching–Verfahren immer noch ein großer Anteil der zu untersuchenden Dokumente in keinem Cache–Speicher verfügbar ist und somit im Rahmen des Aussendungsverfahrens Beachtung findet. Zum einen macht es die schiere Menge (und das weiterhin enorme Wachstum) an Dokumenten im Internet sehr unwahrscheinlich, dass Caching–Systeme einen Großteil dieser Dokumente speichern können. Daneben gibt es im Internet, d.h. insbesondere im World Wide Web, eine Vielzahl von „dynamischen Dokumenten“, die für jede Anfrage an einen Web–Server neu erzeugt werden und daher prinzipiell nicht von Caching–Systemen erfasst werden können.

Zusammenfassend lässt sich feststellen, dass die Aussendung mobiler Filterprogramme auch bei Existenz von großen Caching–Systemen höchst sinnvoll ist, ja dass im Zusammenspiel mit ihnen sogar ein noch höherer Performanzgewinn durch den Einsatz der mobilen Filterprogramme zu erwarten ist.

Kapitel 5

Experimente zur Kartenberechnung

Zur Validierung des Ansatzes der Netzwerk–Distanzkarten wurden umfangreiche Experimente durchgeführt. Diese basieren auf Distanzmessungen, die im Internet vorgenommen wurden. Wie bereits in Kapitel 3.1.2 erwähnt können einzelne Distanzmaße für unterschiedliche Zwecke von Bedeutung sein. Die im Folgenden vorgestellten Experimente basieren auf den beiden Distanzmaßen *Knotenzahl* (engl. *hop-count*) und *Umlaufzeit* (engl. *round trip time*). Dies sind die beiden einzigen Maße, für die Netzwerkmessungen im großen Maßstab möglich waren. Trotzdem sind sie beide relevant und eignen sich aufgrund ihrer unterschiedlichen Charakteristiken sehr gut zur Validierung des Kartenansatzes.

5.1 Methodik

Messverfahren

Mit Hilfe des Werkzeugs „Traceroute“ von [Jacobsen, 1988] wurden zahlreiche Distanzmessungen durchgeführt. Traceroute ermöglicht die Bestimmung des Routing–Pfades von einem Endsystemen im Internet zu einem anderen Endsystem, wozu es auf dem ersten der beiden Endsysteme ausgeführt werden muß. Zur Bestimmung werden einzelne UDP-Pakete$^{\uparrow}$ zum zweiten Endsystem gesandt, allerdings wird die Lebensdauer der Pakete (engl. *time to live*) hierbei eingeschränkt. Die Lebensdauer bezeichnet die maximale Zahl von Routern, die ein Paket durchlaufen darf. Wird sie überschritten, so wird das Paket vom entsprechenden Router verworfen und es wird eine ICMP–Nachricht$^{\uparrow}$ hierüber zum

Sender des Paketes übermittelt. Traceroute verschickt nun einzelne Pakete mit bei eins beginnender und sukzessiv steigender Lebensdauer. Das erste Paket wird dann vom ersten erreichten Router verworfen, was über eine ICMP–Nachricht angezeigt wird. Das zweite Paket wird vom zweiten Router auf dem Pfad zum adressierten Empfänger verworfen usw. Traceroute verschickt solange Pakete mit der nächsthöheren Lebensdauer, bis ein Paket nicht mehr verworfen wird, sondern beim Empfänger ankommt. Letzteres wird durch eine spezielle Fehlermeldung des Empfängers erkannt, der darauf reagiert, dass das Paket an einen ungültigen *Port*[1] geschickt wurde.

Auf diese Weise kann mit Traceroute die Knotenzahl zwischen zwei Endsystemen bestimmt werden. Zusätzlich kann auch noch die Umlaufzeit abgeleitet werden, indem für jedes von Traceroute ausgesandte Paket die Zeit bis zum Eintreffen einer Reaktionsnachricht gemessen wird. Da zu jeder gewählten Lebensdauer jeweils drei Pakete versandt werden, lässt sich diese Zeit auch noch über drei Einzelmessungen mitteln. Trotzdem ist eine solche Bestimmung der Umlaufzeit noch nicht sehr aussagekräftig, was die Schätzung von zukünftig zu erwartenden Umlaufzeiten betrifft. So haben [Acharya & Saltz, 1996] gezeigt, dass die Umlaufzeit eine große zeitliche Variationsbreite hat und dass ein Wert, der die Verteilung der Umlaufzeit gut charakterisiert, erst aus zahlreichen Einzelmessungen abgeleitet werden kann. Dabei konnten sie auch zeigen, dass der am häufigsten in einer empirischen Verteilung vorkommende Messwert eine solche Charakterisierung leisten kann und für ca. 45 Minuten gültig bleibt. Eine mehrfache Wiederholung aller Traceroute–Messungen, mit dem Ziel einen aussagekräftigen Schätzwert für die Umlaufzeit zu bestimmen, war im Rahmen der nachfolgend beschriebenen Datenerhebung nicht möglich, da die dort benutzten Mess–Server nicht übermäßig belastet werden sollten. Trotzdem wurde auch die Umlaufzeit in die nachfolgenden Experimente miteinbezogen, um die Effekte eines gänzlich anderen Distanzmaßes bzw. einer gänzlich anderen Distanzverteilung zu analysieren.

Eine zweite Ungenauigkeitsquelle für die mit Traceroute bestimmten Distanzen liegt darin begründet, dass nicht alle Rechner im Internet die von Traceroute benutzten Effekte (Beachtung der Lebenszeit, Reaktion auf ungültige Ports) unterstützen. So kann es bisweilen zu Situationen kommen, in denen Traceroute nicht exakt das erwartete Ergebnis liefert oder auch gar keine Messung möglich ist. Eine detaillierte Diskussion solcher Fehlerfälle erfolgt in [Gardner, 1998].

Es sei noch betont, dass sich Traceroute nicht als besonders effizient bei der Bestimmung

[1] Mit Ports werden die von der Internet–Transportschicht vorgesehenen Kommunikationsendpunkte auf einem Endsystem bezeichnet.

der Knotenzahl bzw. der Umlaufzeit erwiesen hat. Sein eigentlicher Zweck liegt ja auch vielmehr in der Bestimmung von kompletten Routing–Pfaden. Für die nachfolgenden Experimente spielt die Effizienz des Messverfahrens hingegen keine Rolle.

Datenerhebung

Zahlreiche Web–Server im Internet bieten die Durchführung von Traceroute–Messungen zu beliebigen anderen Endsystemen an. Für die nachfolgenden Experimente wurde eine Liste von 119 solchen Web–Servern zusammengestellt. Im Kontext der Experimente zu Distanzkarten werden diese von nun an Mess–Server genannt. Daneben wurden 460 weitere (einfache) Endsysteme ausgewählt, die sich über alle 5 Kontinente und zahlreiche autonome Systeme erstrecken.

Um bei diesen Messungen keinen Mess–Server zu überlasten und Seiteneffekte zwischen einzelnen Messungen auszuschließen, wurde pro Mess–Server ein zeitlicher Mindestabstand von 2 Minuten zwischen zwei aufeinanderfolgenden Messungen vorgesehen. Zusätzlich wurden die Messungen so koordiniert, dass jedes Endsystem höchstens alle 2 Minuten Ziel einer Messung war.

Eine erste Datenerhebung ($D1$) fand zwischen dem 1. und 11. Februar 1999 statt. Dabei wurden Distanzmessungen von jedem der 119 Mess–Server zu allen anderen 578 Endsystemen (118 Mess–Server + 460 einfache Endsysteme) durchgeführt. Die Zahl der Messungen belief sich somit auf 68782.

Zwischen dem 24. März und dem 1. April 1999 fand eine zweite Datenerhebung ($D2$) statt. Da von den in $D1$ benutzten Mess–Servern 9 ihren Dienst eingestellt hatten, konnten nur noch 110 Mess–Server benutzt werden. Immerhin konnten 7 der Server als Endsystem weiterhin angemessen werden, womit sich die Zahl der in $D2$ benutzten einfachen Endsysteme auf 467 erhöhte. Nachfolgende Vergleiche zwischen diesen beiden Datenerhebungen basieren immer auf den gemeinsamen Teilmengen von Mess–Servern und Endsystemen, welches exakt die in $D2$ benutzten sind.

Eine Unzulänglichkeit bei diesen Datenerhebungen besteht sicherlich in dem großen Zeitraum, über den sie sich jeweils erstrecken. Hierdurch ist eine konsistente Sicht auf einen Netzwerkzustand nur noch sehr eingeschränkt gegeben, da sich Netzwerkdistanzen und insbesondere die Umlaufzeit in erheblich kürzeren Zeiträumen signifikant ändern können. Beispielsweise führt der lange Messzeitraum dazu, dass die Zeitpunkte zweier symmetrischer Messungen erheblich voneinander abweichen und die Messwerte somit in der Regel auch nicht symmetrisch sind. Eine andere Inkonsistenz besteht darin, dass manche

Messwerte zu Zeiten hoher Netzwerkauslastung (tagsüber) andere hingegen bei geringer Auslastung des Netzwerkes (nachts) erfolgten. Aus diesen Gründen sind einige Annahmen, die dem Ansatz zur Berechnung von Distanzkarten zugrunde liegen, wie z.B. die Symmetrieannahme oder die Dreiecksungleichung, durch die Messwerte nicht (vollständig) erfüllt. Es wird sich jedoch in den Experimenten zeigen, dass die Verfahren zur Berechnung von Distanzkarten trotz dieser Fehlerquellen immer noch nützliche Ergebnisse liefern.

Datenanalyse

Abbildung 5.1 zeigt die Histogramme der 67961 erfolgreich gemessenen Distanzen aus $D1$. Um die Histogramme auf ihren zentralen Bereich zu beschränken, wurde die Zählung von Distanzen größer gleich der Knotenzahl 30 bzw. der Umlaufzeit $1500ms$ jeweils in einem einzigen Histogramm-Wert zusammengefasst, der ganz rechts dargestellt ist.

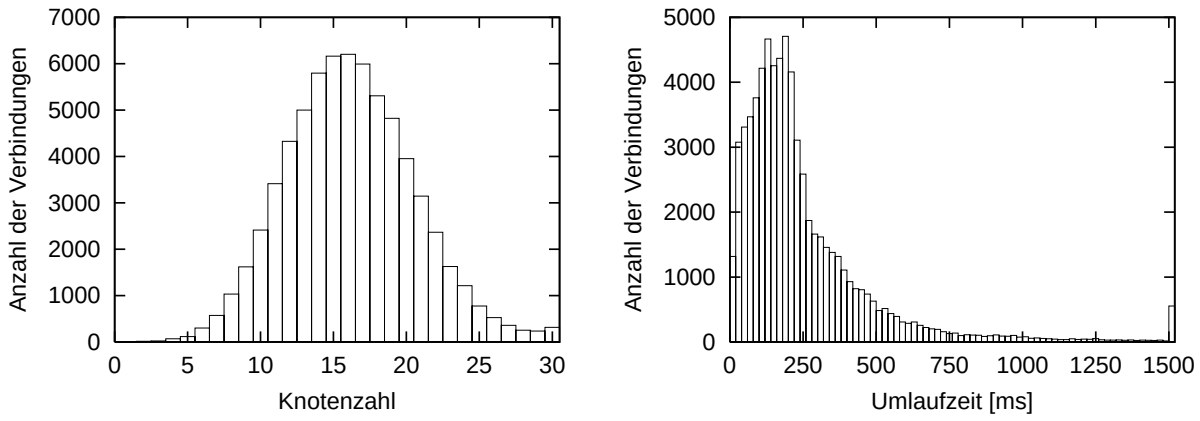


Abbildung 5.1: Histogramme der gemessenen Distanzen aus Datenerhebung $D1$

Während die Verteilung der Knotenzahl bei einem Mittelwert von 16,23 (Standardabweichung $\sigma = 4,43$) ungefähr einer Normalverteilung entspricht, ist die Verteilung der Umlaufzeit mehr durch ihre Schiefe charakterisiert. Diese drückt sich insbesondere dadurch aus, dass der Median mit $186ms$ wesentlich kleiner als der Mittelwert von $256ms$ ist. Qualitativ betrachtet zeigen diese Verteilungen sehr ähnliche Charakteristiken zu anderen Messungen auch aus anderen Jahren, z.B. denjenigen in [Carter & Crovella, 1996].

Als Erstes wurde anhand der gemachten Messungen überprüft, inwieweit die im Systemmodell in Kapitel 3.3.1 gemachte Annahme symmetrischer Distanzen gerechtfertigt ist. Anhand von $n = 6960$ erfolgreich in beiden Richtungen gemessenen Distanzen zwischen Mess–Servern wurde der mittlere Symmetriefehler bestimmt als

$$\frac{1}{n} * \sum_{m_1, m_2} |\Delta(m_1, m_2) - \Delta(m_2, m_1)|,$$

wobei jeweils über alle Paare von Mess–Servern summiert wird, zwischen denen die Distanz in beiden Richtungen messbar ist. Der mittlere Symmetriefehler beträgt für das Entfernungsmaß Knotenzahl 2,0 (Standardabweichung $\sigma = 2,3$) und für das Maß Umlaufzeit $119ms$ (Standardabweichung $\sigma = 266ms$). Obwohl Umlaufzeiten prinzipiell symmetrisch sein müssten, ist die nur schwache Gültigkeit der Symmetrieeigenschaft angesichts der unzureichenden Messmethode und des langen Messzeitraumes nicht sonderlich überraschend. Allgemein ist der Symmetriefehler ein Maß für diese Unzulänglichkeiten und beschreibt eine untere Schranke für die in den nachfolgenden Experimenten erreichbare Qualität der Distanzschätzung. Es wird sich jedoch zeigen, dass diese Schranke in einigen Fällen trotzdem unterschritten wird, da aufgrund von Mittelwertbildungen (z.B. bei der Distanzschätzung zwischen Regionen) Unzulänglichkeiten des Messverfahrens bis zu einem gewissen Grad ausgeglichen werden können. Schließlich muss man noch bedenken, dass absolute Fehler im Zusammenhang mit Umlaufzeiten aufgrund deren großer Variationsbreite bisweilen etwas irreführend sein können. Der mittlere relative Symmetriefehler, definiert als

$$\frac{1}{n} * \sum_{m_1, m_2} \frac{|\Delta(m_1, m_2) - \Delta(m_2, m_1)|}{\max(\Delta(m_1, m_2),\ \Delta(m_2, m_1))},$$

beträgt nämlich lediglich 24% ($\sigma = 25\%$).

Die nächste Untersuchung betraf die Gültigkeit der Dreiecksungleichung, die ja von einigen Ballungskriterien (vgl. Kapitel 3.4.1) vorausgesetzt wird und ebenso eine notwendige Voraussetzung dafür ist, dass ein Distanzmaß eine Metrik bildet. Bei den gut 7 Millionen Tripeln, die aus den Distanzmessungen zusammengestellt werden konnten, ist die Dreiecksungleichung für das Distanzmaß Knotenzahl in 99.5% der Fälle erfüllt und für die Umlaufzeit in 87.6% der Fälle.

Schließlich wurde untersucht, wie weit ein einfaches Endsystem durchschnittlich von seinem nächstgelegenen Mess–Server entfernt war. Der Durchschnittswert für die Knotenzahl beträgt 7,1 ($\sigma = 2,4$), was ca. zwei Fünftel der mittleren Knotenzahl zwischen beliebigen Endsystemen entspricht und einem 1,4%-Quantil[2] der Knotenzahlverteilung. Der Durchschnittswert für die Umlaufzeit beträgt $27ms$ ($\sigma = 56ms$), was lediglich ein Zehntel der mittleren Umlaufzeit zwischen beliebigen Endsystemen entspricht und einem 3,2%-Quantil der Umlaufzeitverteilung. Die Quantile zeigen, dass mit den Mess–Servern prinzipiell eine gleichmäßige und gute Abdeckung des Internets erreicht wurde. Trotzdem ist die mittlere Knotenzahl zwischen einfachen Endsystemen und ihrem nächstgelegenen

[2]Das α-Quantil ($\alpha \in (0,1)$) einer Stichprobe mit Werten $x_1, \ldots, x_n$ legt den Wert x_α fest, so dass mindestens der Anteil α der $x_i \leq x_\alpha$ und mindestens der Anteil $(1-\alpha)$ der $x_i \geq x_\alpha$ sind. Der Median ist gerade das 0,5-Quantil.

Mess–Server ziemlich groß, was aber an der Charakteristik der empirischen Verteilung liegt. Im Sinne der Umlaufzeit existiert dagegen zu einem durchschnittlichen einfachen Endsystem ein recht nahe gelegener Mess–Server.

Die analogen Untersuchungen für die Datenerhebung $D2$ brachten sehr ähnliche Ergebnisse. Zusätzlich wurde noch die Divergenz, d.h. die Distanzänderung, zwischen den beiden Datenerhebungen $D1$ und $D2$ untersucht. Zu jeder ausgemessenen Verbindung mit Distanz $d1$ in $D1$ und Distanz $d2$ in $D2$ wurde das Verhältnis $|d1 - d2| / \max(d1, d2)$ berechnet. Die Divergenzverteilung ist in Abbildung 5.2 dargestellt. Die mittlere Di-

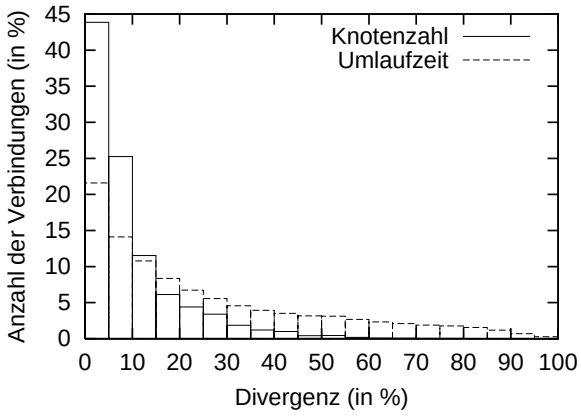


Abbildung 5.2: Divergenzverteilung zwischen den Datenerhebungen $D1$ und $D2$

vergenz für die Knotenzahl beträgt 8,2% ($\sigma = 10,7\%$) und 24,8% für die Umlaufzeit ($\sigma = 24.0\%$). Erwartungsgemäß stellt sich die Knotenzahl als wesentlich stabiler als die Umlaufzeit heraus.

5.2 Initiale Erstellung von Karten

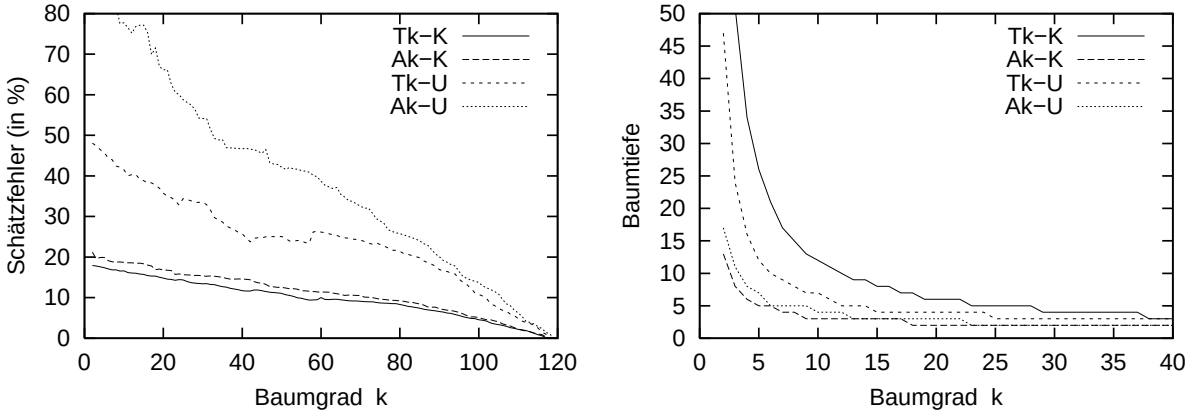
In diesem Abschnitt werden nun die Experimente beschrieben, die zur Evaluierung der Verfahren zur initialen Erstellung von Distanzkarten (siehe Abschnitte 3.4.1 bis 3.4.3) durchgeführt wurden. Alle Versuche basieren auf der Datenerhebung $D1$, wobei für Verbindungen, die in beiden Richtungen ausgemessen wurden, jeweils der Mittelwert der beiden Messwerte herangezogen wurde.

Optimales Ballungsverfahren

Als Erstes wurde ein Vergleich der beiden Ballungskriterien *Max-k-Trennung* und *Min-k-Ausdehnung* und ihrer Berechnungsverfahren aus Kapitel 3.4.1 durchgeführt. Hierzu

wurden Regionenbäume für beide Kriterien und für die beiden Distanzmaße Knotenzahl und Umlaufzeit berechnet. Der Baumgrad k wurde zwischen 2 und 119 variiert, die Selektivität auf ihren Maximalwert von 119 gesetzt. Für jeden Regionenbaum und jedes Paar von Mess–Servern wurde der Fehler der Distanzschätzung durch Vergleich mit der in $D1$ gemessenen Distanz ermittelt.

Der linke Teil von Abbildung 5.3 zeigt die in verschiedenen Situationen erreichte Güte der Distanzschätzung als relativen Schätzfehler, gemittelt über allen Paaren von Mess–Servern. Für einen Messwert m und einen Schätzwert e berechnet sich dieser Fehler zu $|m - e|/m$.



Bedeutung der verwendeten Kürzel:	
Tk-K	Max-k-Trennung; Knotenzahl
Ak-K	Min-k-Ausdehnung; Knotenzahl
Tk-U	Max-k-Trennung; Umlaufzeit
Ak-U	Min-k-Ausdehnung; Umlaufzeit

Abbildung 5.3: Optimale Ballung: Schätzfehler und Regionenbaumtiefe

Man sieht, dass die Schätzung der Knotenzahl in allen Konstellationen wesentlich genauer als die Schätzung der Umlaufzeit ist, was nach der Datenanalyse im vorigen Abschnitt allerdings keine Überraschung darstellt. In allen Fällen geht der Fehler mit wachsendem k gleichmäßig zurück, womit sich zeigt, dass der Ansatz der Ballung gut funktioniert. Da die Regionenbäume für $k = 119$ alle gemessenen Distanzen direkt enthalten, ist in diesem Fall der Fehler natürlicherweise 0.

Die ungefähr lineare Korrelation zwischen dem Baumgrad k und dem Schätzfehler erlaubt die Schlussfolgerung, dass keine bevorzugte „natürliche Ballung“ existiert, die eine

optimale Balance zwischen dem aus k folgendem Messaufwand und dem Schätzfehler erreicht. Zwar ist es vorstellbar, dass eine natürliche Ballung für eine größere Zahl von Mess–Servern existiert, z.B. gemäß der autonomen Systeme im Internet. Allerdings erlauben obige Experimente keine derartige Schlussfolgerung.

Von den beiden Ballungskriterien führt das Kriterium *Max-k-Trennung* zu präziseren Distanzschätzungen, insbesondere für die Schätzung der Umlaufzeit.

Ein zweites Gütekriterium von Regionenbäumen ist die Baumtiefe, da mit ihr der Aufwand zur Einordnung und Anpassung einfacher Endsysteme verknüpft ist. Je tiefer ein Regionenbaum desto mehr Netzwerkmessungen sind pro Endsystem erforderlich, um dieses möglichst optimal im Regionenbaum anzuordnen. Ähnliches sollte für die Probeintegration im Rahmen der Anpassung von Endsystemen gelten. Der rechte Teil von Abbildung 5.3 zeigt die Tiefe eines jeden der schon zur Bewertung der Distanzschätzung berechneten Regionenbäume. Die Baumtiefe erweist sich beim Kriterium *Min-k-Ausdehnung* als wesentlich geringer. So beträgt die Tiefe für das Maß Knotenzahl und $k = 10$ nur 2 während sie für das Kriterium *Max-k-Trennung* 11 beträgt. Offensichtlich tritt hier für letzteres Kriterium der in Kapitel 3.4.1 beschriebene Effekt schlecht balancierter Ballungen auf. Für das Distanzmaß Umlaufzeit ist dieser Effekt weniger stark ausgeprägt aber trotzdem signifikant.

Schließlich wurde noch der Effekt der hierarchischen Verfeinerung von Ballungen in Regionenbäumen untersucht. Tabelle 5.1 beschreibt hierzu einige charakteristische Parameter, die die ersten drei Ebenen von Regionenbäumen mit Baumgrad $k = 10$ beschreiben. Zu jeder Ebene mit Tiefe l werden dabei genau die Paare von Mess–Servern betrachtet, die einer gemeinsamen Region mit Tiefe l im Regionenbaum angehören. Pro Ebene wird aus diesen Paaren die mittlere gemessene Distanz sowie der entsprechende mittlere absolute Schätzfehler dargestellt. Der Schätzfehler für Ebene 0 entspricht genau dem in Abbildung 5.3 auf Seite 99 dargestellten Fehler, außer dass dort relative Fehler dargestellt sind.

	mittlere Netzwerkdistanz / absoluter Schätzfehler		
Distanzmaß, Kriterium	Ebene 0	Ebene 1	Ebene 2
Knotenzahl, Max-10-Trennung	15.6 / 2.47	15.1 / 2.46	14.7 / 2.45
Knotenzahl, Min-10-Ausdehnung	15.6 / 2.65	12.7 / 1.52	1.52 / 0.0
Umlaufzeit, Max-10-Trennung	$249ms$ / $91ms$	$163ms$ / $70ms$	$145ms$ / $67ms$
Umlaufzeit, Min-10-Ausdehnung	$249ms$ / $109ms$	$147ms$ / $39ms$	$80ms$ / $16ms$

Tabelle 5.1: Eigenschaften von Regionenbäumen auf den Ebenen 0–2

Die mit tieferen Ebenen geringer werdende mittlere Netzwerkdistanz belegt den Sinn der hierarchischen Einteilung eines Netzwerkes in Regionen und Teilregionen. Da der mittlere absolute Schätzfehler ebenfalls auf tieferen Ebenen abnimmt, reflektiert auch die Distanzschätzung diese zunehmende Lokalität. Beide Effekte sind für das Kriterium *Min-10-Ausdehnung* stärker ausgeprägt, was einmal mehr die besseren Balancierungseigenschaften dieses Kriteriums beweist. Die extrem niedrige mittlere Netzwerkdistanz für das Kriterium *Min-10-Ausdehnung* und das Distanzmaß Knotenzahl auf Ebene 2 erklärt sich dadurch, dass die meisten Regionen auf dieser Ebene lediglich einen einzigen Mess–Server umfassen. Beim Kriterium der *Max-10-Trennung* tritt genau der entgegengesetzte Effekt auf. Aufgrund der ungleichmäßigen Balancierung gibt es auf Ebene 2 immer noch Regionen die sehr viele Mess–Server umfassen und daher noch zu hohen Netzwerkdistanzen und Schätzfehlern führen. Analoge Experimente mit anderen Baumgraden haben im Wesentlichen zu den gleichen Resultaten geführt. Daher werden sie an dieser Stelle nicht mehr explizit aufgeführt.

Gemischtes Ballungsverfahren

Der nächste Evaluierungsaspekt war das gemischte Ballungsverfahren (vorgestellt in Kapitel 3.4.2), nach dem eine optimale Ballung jeweils nur auf einer Teilmenge der verfügbaren Mess–Server bestimmt wird und die weiteren Server danach passend in die bestimmten Regionen einsortiert werden. Hierzu wurde eine Testserie für Regionenbäume mit Baumgraden von $k = 10$ und $k = 40$ und mit Selektivitätswerten zwischen 10 (bzw. 40) und 119 durchgeführt. Alle nachfolgenden Ergebnisse sind jeweils über 100 Testläufen gemittelt. Abbildung 5.4 auf der nächsten Seite zeigt Ergebnisse zu Ballungsversuchen für das Distanzmaß Knotenzahl. Es werden die schon in Abbildung 5.3 eingeführten Kürzel benutzt, wobei der Parameter k jeweils durch einen konkreten Wert ersetzt wurde.
Der im linken Abbildungsteil gezeigte Schätzfehler fällt in allen vier Konstellationen konkav mit steigender Zahl der Netzwerkmessungen. Für beide Ballungskriterien ist dieser Abfall anfangs recht stark. Beim Kriterium *Min-k-Ausdehnung* hat der Schätzfehler dann aber auch schon fast sein Optimum erreicht, zum Beispiel wird bei $k = 10$ der Fehler mit 2500 Messungen (36% vom Maximalwert) bereits bis auf 0,7 Prozentpunkte dem minimalen Fehler von 18,8% angenähert. Dagegen ist beim Kriterium *Max-k-Trennung* auch im weiteren Kurvenverlauf ein gleichmäßiger, wenn auch nicht mehr so starker, Rückgang des Fehlers zu beobachten. Der Mindestaufwand an Messungen ist bei Regionenbäumen mit Baumgrad $k = 40$ naturgemäß um Einiges höher als für $k = 10$, da bei ihnen die Selektivität mindestens 40 beträgt.

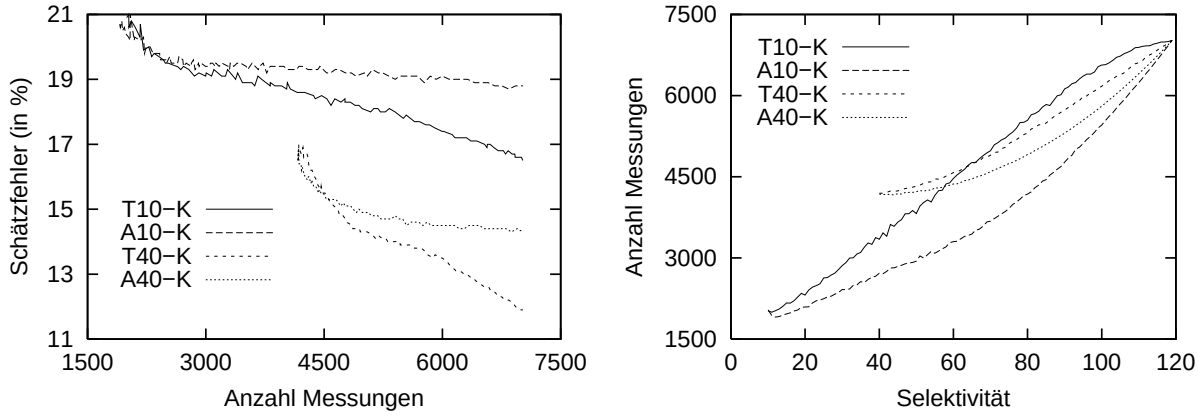


Abbildung 5.4: Gemischte Ballung (Knotenzahl): Schätzfehler und Messaufwand

Der rechte Abbildungsteil stellt die Relation zwischen der Wahl des Selektivitätswertes und der resultierenden Zahl der Distanzmessungen dar. Das monotone Ansteigen der Kurve zeigt, dass sich der mit dem gemischten Verfahren verbundene Aufwand und damit die erzielbare Qualität direkt durch den Selektivitätswert steuern lässt.

Bemerkenswert ist noch, dass auch beim gemischten Ballungsverfahren die Verwendung des Kriteriums *Max-k-Trennung* zu schlecht balancierten Bäumen führt. Für die minimale Selektivität von $k = 10$ bzw. $k = 40$ ist die Baumtiefe für beide Kriterien noch gleich. Während sie beim Kriterium *Min-k-Ausdehnung* ungefähr konstant über der Selektivität bzw. der Zahl der Messungen bleibt, steigt sie beim Kriterium *Max-k-Trennung* linear an.

Insgesamt erweist sich das gemischte Verfahren für das Kriterium *Min-k-Ausdehnung* als durchaus nützlich. Mit verhältnismäßig wenigen Messungen wird der mit dem optimalen Verfahren erreichbare Schätzfehler bereits sehr gut angenähert. Für das Kriterium *Max-k-Trennung* ist der Nutzen dagegen gering. Bei dieser Bewertung muss allerdings beachtet werden, dass die Zahl der in den Experimenten beteiligten Mess–Server sehr gering ist. Für größere Zahlen ist durchaus auch ein höherer Nutzen zu erwarten.

Der linke Teil von Abbildung 5.5 zeigt den Schätzfehler der analogen Versuche für das Distanzmaß Umlaufzeit. Auffällig ist die Verschlechterung des Fehlers mit steigender Zahl der Messungen für das Kriterium *Min-k-Ausdehnung* und $k = 10$. Selbst im Extremfall mit fast 7500 Messungen ist der Schätzfehler noch höher als zu Anfang. Dies zeigt jedoch nur, das das Ballungskriterium nicht optimal ist.

Für die anderen Konstellationen ergibt sich zwar keine Verschlechterung, jedoch ist der Nutzen des gemischten Verfahrens auch hier gering, da der Schätzfehler sehr gleichmäßig mit steigender Zahl der Messungen zurückgeht.

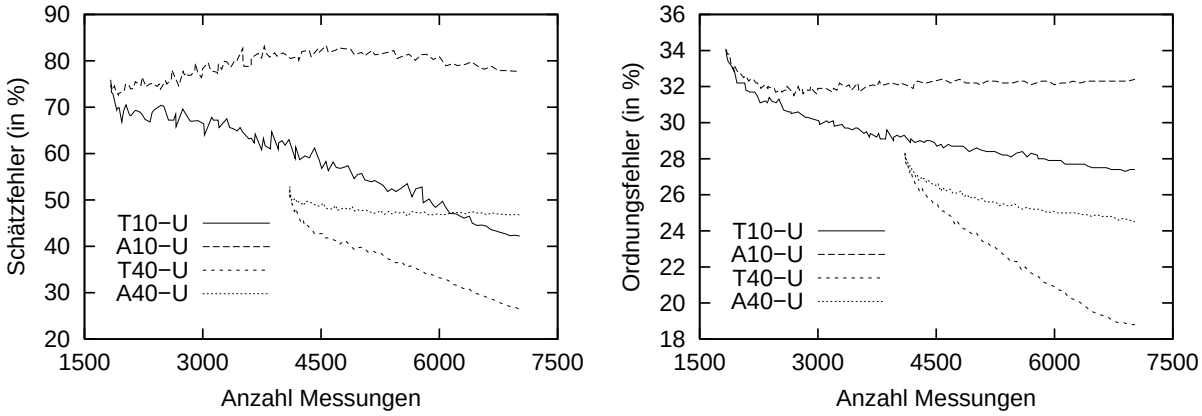


Abbildung 5.5: Gemischte Ballung (Umlaufzeit): Schätzfehler und Ordnungsfehler

Um zu untersuchen, ob diese Effekte evtl. an der zu ungenauen Messmethode für Umlaufzeiten liegen (vgl. mit Abschnitt 5.1), wurde ein zweites Fehlermaß zur Analyse herangezogen, welches eine stärkere Glättung der Daten bewirkt: der sogenannte *Ordnungsfehler*. Dieser ist für einen Messwert m und einen Schätzwert e definiert als $|m - e| / \max\{m, e\}$. Dieses Fehlermaß beschreibt den Faktor, um den Schätzwert und Messwert voneinander abweichen, Abweichungen nach oben und unten werden somit gleich behandelt. Ein Ordnungsfehler von 50% besagt also, dass der Schätzwert entweder um den Faktor 2 nach oben oder um den Faktor 1/2 nach unten vom Messwert abweicht. Der rechte Teil von Abbildung 5.5 zeigt nun den Ordnungsfehler des gemischten Verfahrens für das Distanzmaß Umlaufzeit. Interessanterweise sinkt der Ordnungsfehler konkav mit steigender Zahl der Messungen. Bis auf die Konstellation $k = 40$ und das Kriterium *Max-k-Trennung* erfolgt die Annäherung des optimalen Fehlers sogar recht schnell. Der Ordnungsfehler zeigt also, dass auch für Umlaufzeiten das gemischte Ballungsverfahren durchaus nützlich ist. Zur genaueren Untersuchung dieses Aspektes wären allerdings umfangreichere Experimente mit mehr Mess–Servern und insbesondere genaueren Mess–Methoden erforderlich.

Die in Abbildung 5.6 auf der folgenden Seite dargestellte Relation zwischen Selektivitätswerten und Messaufwand zeigt die gleichen Charakteristiken wie für das Distanzmaß Knotenzahl.

Integration einfacher Endsysteme

Das Ziel bei der Integration einfacher Endsysteme in einen Regionenbaum besteht darin, mit möglichst wenigen Messungen jeweils einen möglichst nah gelegenen Mess–Server zu finden. Die Qualität einer gefundenen Zuordnung wird mit der sogenannten *Approximationsratio* beschrieben. Für ein Endsystem e, einen gefundenen Mess–Server m und den

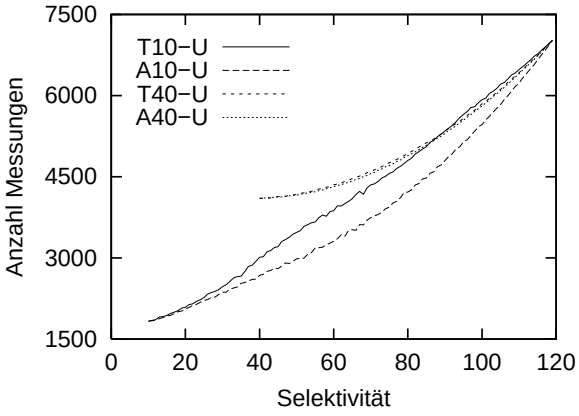


Abbildung 5.6: Gemischte Ballung (Umlaufzeit): Messaufwand

zu e nächstgelegenen Mess–Server *opt* ist die Approximationsratio der Lösung m definiert als das Verhältnis der Distanzen von e zu *opt* bzw. m, also als $\Delta(opt, e)/\Delta(m, e)$.

Nachfolgend werden die drei in Kapitel 3.4.3 vorgestellten Integrationsverfahren (Tiefensuche, Breitensuche und Zufallssuche) miteinander verglichen. Die Verfahren mit Tiefen- und Breitensuchestrategie wurden dazu mit verschiedenen Parametern für die Ähnlichkeitsschranke ausgeführt. Das Zufallsverfahren wurde mit allen möglichen Werten (zwischen 1 und 119) für die Zahl der zufällig auszuwählenden Mess–Server ausgeführt. Abbildung 5.7 zeigt für alle 3 Verfahren, beide Distanzmaße und beide Ballungskriterien die Approximationsratio als Funktion der Zahl der pro Endsystem notwendigen Distanzmessungen. Beide Werte sind jeweils über allen Endsystemen gemittelt. Beim Zufallsverfahren wurde zusätzlich noch eine Mittelung über 10 Durchläufen vorgenommen. Von allen Kurven wird nur der Teil gezeigt, der zu Approximationswerten über 50% führt. Der linkeste Wert einer jeden Kurve zur Tiefen- oder Breitensuche repräsentiert den Fall des einfachen Integrationsverfahrens, in dem immer nur genau ein Pfad durch den Regionenbaum verfolgt wird.

Für das Distanzmaß Knotenzahl (im linken Abbildungsteil dargestellt) erweist sich die Integration in Regionenbäume, die mit dem Kriterium *Min-k-Ausdehnung* berechnet wurden, sowohl bei der Tiefen- als auch bei der Breitensuche dem Kriterium *Max-k-Trennung* deutlich überlegen. Hier kommen offensichtlich die schlechten Balancierungseigenschaften des letzteren Kriteriums voll zum Tragen und führen zu einem erheblich höherem Integrationsaufwand. Für beide Kriterien ist die Tiefensuche etwas effektiver als die Breitensuche. Das Zufallsverfahren erreicht erstaunlich gute Approximationswerte, die zum Teil nur knapp unter denjenigen des Kriteriums *Min-k-Ausdehnung* liegen.

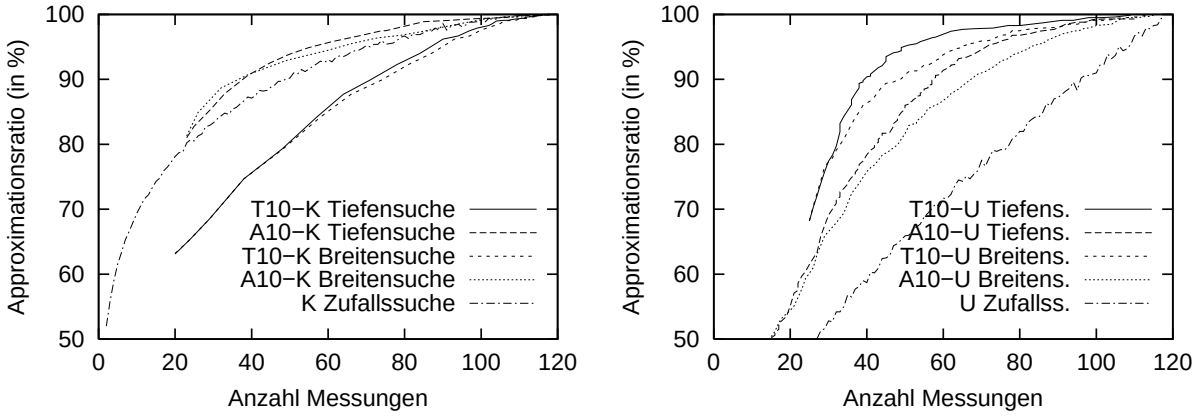


Abbildung 5.7: Integration einfacher Endsysteme: Approximationsratio für Knotenzahl (links) und Umlaufzeit (rechts)

Für das Distanzmaß Umlaufzeit hingegen erweist sich das Zufallsverfahren als das mit Abstand Schlechteste. Die Tiefensuche ist der Breitensuche klar überlegen. Die Integration bei Regionenbäumen, die mit dem Kriterium *Max-k-Trennung* berechnet wurden, erfolgt effizienter als bei dem Kriterium *Min-k-Ausdehnung*. Dies korrespondiert genau mit den Experimenten zur optimalen und gemischten Ballung, wo das Kriterium *Max-k-Trennung* ebenfalls zu besseren Ergebnissen beim Distanzmaß Umlaufzeit führte. Offensichtlich dominiert hier die höhere Schätzgüte den Effekt schlechter balancierter Bäume.

Nicht explizit in einer Abbildung dargestellt aber trotzdem erwähnenswert ist, dass sowohl der Messaufwand als auch die aus ihm resultierende Approximationsratio monoton mit steigender Ähnlichkeitsschranke wachsen. Damit erlaubt dieser Parameter eine gute Kontrolle über den gewünschten Aufwand bzw. die Qualität der Zuordnung einfacher Endsysteme.

Distanzschätzung für einfache Endsysteme

Nach dem Systemmodell der Netzwerksicht aus Kapitel 3.3.1 ist die Distanz zwischen einfachen Endsystemen nicht direkt bestimmbar, im Gegenteil ihre Schätzung stellt (neben der Regionenbestimmung) den zentralen Zweck von Netzwerk–Distanzkarten dar. Um die Güte dieser Schätzung dennoch evaluieren zu können, wurden verschiedene Karten berechnet, wobei jeweils ein verfügbarer Mess–Server unbenutzt gelassen wurde und nur als Endsystem Beachtung fand. Distanzen zwischen einem solchen unbenutzten Mess–Server und anderen (echten) einfachen Endsystemen sind somit sowohl messbar als auch schätzbar und erlauben also die Bewertung der Distanzschätzung zwischen einfachen Endsystemen. Auf diese Weise wurden für jede Konfiguration von Distanzmaß, Ballungskriterium

und Baumgrad 119 verschiedene Netzwerkkarten bestimmt. Nachfolgende Ergebnisse sind immer über diesen 119 Karten gemittelt.

Abbildung 5.8 zeigt den absoluten Schätzfehler für beide Distanzmaße, beide Ballungskriterien und für die ersten beiden Ebenen des Netzwerkbaumes. Der Fehler für Ebene l (mit Baumtiefe l) ist dabei jeweils über allen Messpaaren gemittelt, die einer gemeinsamen Region mit Tiefe l im Netzwerkbaum angehören.

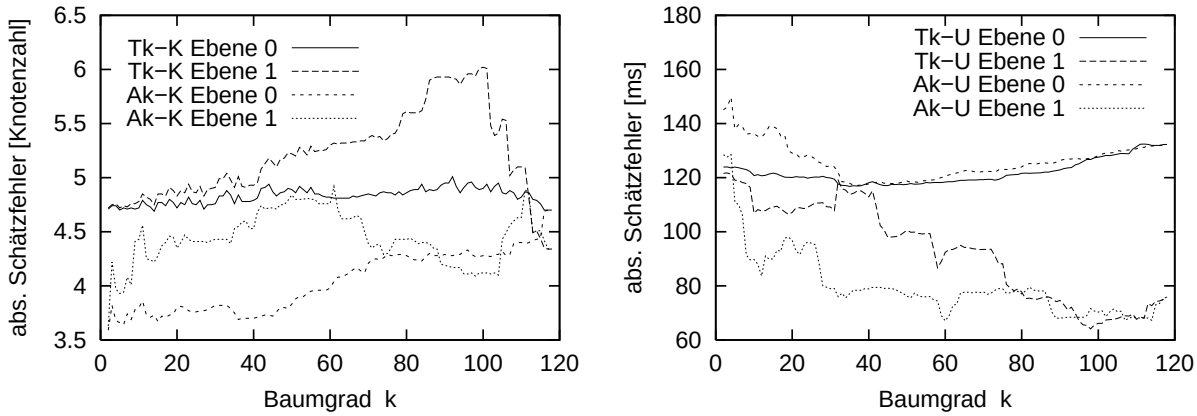


Abbildung 5.8: Absoluter Fehler der Distanzschätzung zwischen einfachen Endsysteme: Knotenzahl (links) und Umlaufzeit (rechts)

Der linke Teil der Abbildung zeigt, dass die Schätzung der Knotenzahl anhand des Kriteriums der *Min-k-Ausdehnung* durchweg besser ist als bei der *Max-k-Trennung*. Interessanterweise ist die Distanzschätzung auf Ebene 0 für beide Kriterien exakter als auf Ebene 1. Mit steigendem Baumgrad k wird in der Regel keinerlei Verbesserung der Schätzung erreicht, manchmal verschlechtert sich das Ergebnis sogar.

Ein Grund für diese Effekte findet sich in der empirischen Verteilung des Distanzmaßes Knotenzahl (vgl. mit der Datenanalyse aus Abschnitt 5.1). Hiernach beträgt die mittlere Knotenzahl zwischen einem einfachen Endsystem und seinem nächstgelegenen Mess–Server 7,1. Selbst bei perfekter Schätzung der Distanz zwischen 2 zugeordneten Mess–Servern bleibt also eine Unsicherheit über eine Knotenzahl von jeweils 7,1 für beide Enden der zu schätzenden Distanz. Auf Ebene 0 spielt diese Unsicherheit aufgrund der im Mittel größeren Distanzen dabei eine geringere Rolle als auf Ebene 1. Angesichts dieser Unsicherheit ist der durch obige Schätzwerte erreichte Fehler zwischen 3 und 6 sogar relativ gut, lässt sich allerdings durch höhere Baumgrade auch nicht mehr steigern.

Der rechte Teil von Abbildung 5.8 zeigt den absoluten Schätzfehler für das Distanzmaß Umlaufzeit. Diesmal erweist sich der Schätzfehler auf der tieferen Ebene als eindeutig

geringer. Für Ebene 1 sinkt er auch gleichmäßig mit zunehmendem Baumgrad. Auf Ebene 0 fällt der Fehler zunächst bis ungefähr zur Mitte der Kurve, steigt dann aber wieder leicht an. Während auf Ebene 0 beide Ballungskriterien zu ungefähr gleich guten Resultaten führen zeigt sich auf Ebene 1 ein leichter Vorteil beim Kriterium der *Min-k-Ausdehnung*.

Ein Grund für diesen Effekt könnte in den nur ungenau bestimmten Umlaufzeiten liegen (vgl. mit der in Abschnitt 5.1 beschriebenen Messmethode). Für größere Baumgrade wird die Zahl der Mess–Server pro Region immer kleiner. Damit wird aber auch die Schätzung der Distanzen zwischen Regionen weniger robust gegenüber einzelnen ungenau gemessenen Distanzen womit letztlich auch der mittlere Schätzfehler für einfache Endsysteme mit hohen Baumgraden zunimmt. Auf Ebene 1 tritt dieser Effekt aufgrund der zunehmenden Lokalität nicht auf.

Der in der jeweils optimalen Konstellation erreichte relative Schätzfehler beträgt 22,2% für das Distanzmaß Knotenzahl und 46,8% für die Umlaufzeit. Damit liegen die Schätzwerte zumindest von der Größenordnung her richtig.

5.3 Anpassung von Karten an Netzwerkänderungen

Zur Evaluation der Anpassungsverfahren wurden die beiden Datenerhebungen $D1$ und $D2$ verwendet. Mit der Ersten wurde eine initiale Netzwerk–Distanzkarte berechnet, die alsdann anhand der Distanzen aus $D2$ schrittweise aktualisiert wurde. Die aktualisierten Karten wurden jeweils mit der optimal aus $D2$ berechenbaren Karte verglichen. Da beim Vergleich nur die sowohl in $D1$ als auch $D2$ vorkommenden Mess–Server und einfachen Endsysteme miteinbezogen werden konnten, wurden die nachfolgenden Experimente nur noch mit 110 Mess–Servern und 467 einfachen Endsystemen durchgeführt.

Anpassung von Regionenbäumen

Regionenbäume werden durch Neuballung von einzelnen Teilbäumen an sich verändernde Netzwerkbedingungen angepasst. Zur Überprüfung der erreichbaren Anpassungsgüte wurden verschiedene Regionenbäume anhand der Datenerhebung $D1$ berechnet. Diese wurden mit Distanzmessungen aus $D2$ einer optimalen Neuballung auf den Ebenen 1 und 2 unterzogen (vgl. Verfahren zur Neuballung in Kapitel 3.4.4). Die Qualität der Distanzschätzung zwischen Mess–Servern wurde für diese Bäume anhand des vollständigen Distanzenwissens aus $D2$ evaluiert.

Tabelle 5.2 zeigt die Ergebnisse dieser Evaluation für Regionenbäume mit Baumgrad $k = 10$, für beide Distanzmaße und beide Ballungskriterien. Zu jeder Konstellation werden

	#Messungen	absoluter Schätzfehler		
		Ebene 0	Ebene 1	Ebene 2
Knotenzahl, *Max-10-Trennung*				
$D1$-Regionenbaum	-	2.69	2.7	2.69
Anpassung Ebene 2	4186	2.55	2.53	2.48
Anpassung Ebene 1	5050	2.51	2.48	2.46
$D2$-Regionenbaum	5995	2.47	2.46	2.44
Knotenzahl, *Min-10-Ausdehnung*				
$D1$-Regionenbaum	-	2.76	1.83	0.94
Anpassung Ebene 2	109	2.74	1.74	0.0
Anpassung Ebene 1	1060	2.71	1.56	0.0
$D2$-Regionenbaum	5995	2.66	1.49	0.0
Umlaufzeit, *Max-10-Trennung*				
$D1$-Regionenbaum	-	119	103	99
Anpassung Ebene 2	1369	115	94	83
Anpassung Ebene 1	2385	110	82	80
$D2$-Regionenbaum	5995	109	85	83
Umlaufzeit, *Min-10-Ausdehnung*				
$D1$-Regionenbaum	-	134	81	34
Anpassung Ebene 2	433	134	81	32
Anpassung Ebene 1	1227	128	53	9
$D2$-Regionenbaum	5995	139	55	0

Tabelle 5.2: Anpassung von Regionenbäumen auf Ebene 1 und 2

der ursprünglich aus $D1$ bestimmte Baum, die durch Anpassung auf den Ebenen 1 und 2 entstandenen Bäume sowie der optimal mit $D2$ korrespondierende Baum angegeben. Der $D2$-Regionenbaum entspricht einer Neuballung auf Ebene 0. Die Güte jedes Baumes wird durch den absoluten Fehler (bezüglich der Messungen aus $D2$) der Distanzschätzung auf den ersten drei Baumebenen angegeben (Spalten 3–5). Zusätzlich wird die Zahl der für jede Aktualisierung notwendigen Distanzmessungen angegeben (Spalte 2).

Die Zahlen belegen, dass Aktualisierungen auf tieferen Baumebenen durchaus effektiv sein können und auch die Distanzschätzung auf höhergelegenen Ebenen verbessern. Die Neuballung ist für Regionenbäume, die mit dem Kriterium *Min-k-Ausdehnung* berechnet

wurden, am effektivsten. Mit bis zu 1200 Messungen (ein Fünftel der bei der kompletten Neuberechnung notwendigen Messungen), werden die optimalen Schätzwerte bei beiden Distanzmaßen bereits sehr gut angenähert. Bei der Umlaufzeit führt die Neuballung auf Ebene 1 sogar zu einem besseren Ergebnis als die $D2$-Ballung. Dies zeigt allerdings nur, dass das Ballungskriterium nicht optimal ist. Die Anpassung beim Kriterium *Max-k-Trennung* erfordert dagegen mehr Distanzmessungen (insbesondere beim Distanzmaß Knotenzahl), was in der schlechten Balancierung dieser Bäume begründet liegt.

Absolut gesehen fällt die Reduktion des Schätzfehlers auf der obersten Baumebene relativ gering aus. Dies zeigt, dass eine Neuballung anhand der Messungen aus $D2$ noch nicht unbedingt nötig war. Für das Distanzmaß Knotenzahl ist es durchaus vorstellbar, dass ein zeitliches Intervall von knapp zwei Monaten (dem Zeitraum zwischen den beiden Datenerhebungen) keine besonders großen Veränderungen bringt. Demhingegen ist dieses Resultat für die Umlaufzeit etwas überraschend. Hierbei muss jedoch bedacht werden, dass beide Datenerhebungen sich jeweils über gut eine Woche erstreckt haben, und somit nur eine sehr ungenaue Sicht auf einen Netzwerkzustand liefern. Offensichtlich dominiert diese Ungenauigkeit die mit einer Neuballung erreichbare Verbesserung der Distanzschätzung.

Anpassung einfacher Endsysteme

Die Experimente zur Anpassung einfacher Endsysteme basieren auf Netzwerkbäumen, die aus neuen, aus $D2$ berechneten Regionenbäumen bestehen, aber noch die alten, aus $D1$ bestimmten Zuordnungen von Endsystemen zu Mess–Servern haben.

Für beide Distanzmaße, beide Ballungskriterien und den Baumgrad $k = 10$ zeigt Abbildung 5.9 die Ergebnisse des Anpassungsverfahrens, das bei der Probeintegration die Strategie der Tiefensuche verfolgt. Als Parameter für die bei der Probeintegration benötigte Ähnlichkeitsschranke wurde für das Distanzmaß Knotenzahl 1,2 und für das Distanzmaß Umlaufzeit 2,5 gewählt. Diese Parameter wurden aus einem ganzen Reihe von getesteten Parametern ausgewählt. Die automatische Auswahl solcher Parameter ist als Teil der Kalibrierung der Verfahren nicht Gegenstand dieser Arbeit. Durch Variation des Radius entstand jeweils ein ganzer Satz von Ergebnissen. Wie schon bei der Integration einfacher Endsysteme wird die Güte der Ergebnisse durch die Approximationsratio beschrieben, welche als Funktion der Zahl der pro Endsystem notwendigen Netzwerkmessungen dargestellt wird. Die Werte sind abermals über allen Endsystemen gemittelt.

Die Darstellung der Ergebnisse der Breitensuchstrategie wird weggelassen, da sie sich wie schon bei der Integration einfacher Endsysteme als schlechter herausstellt. Zum Vergleich

wurde dagegen noch ein Zufallsverfahren evaluiert, bei dem Distanzen zu zufällig gewählten Mess–Servern gemessen werden und die Zuordnung bei ggf. nähergelegenen Servern aktualisiert wird.

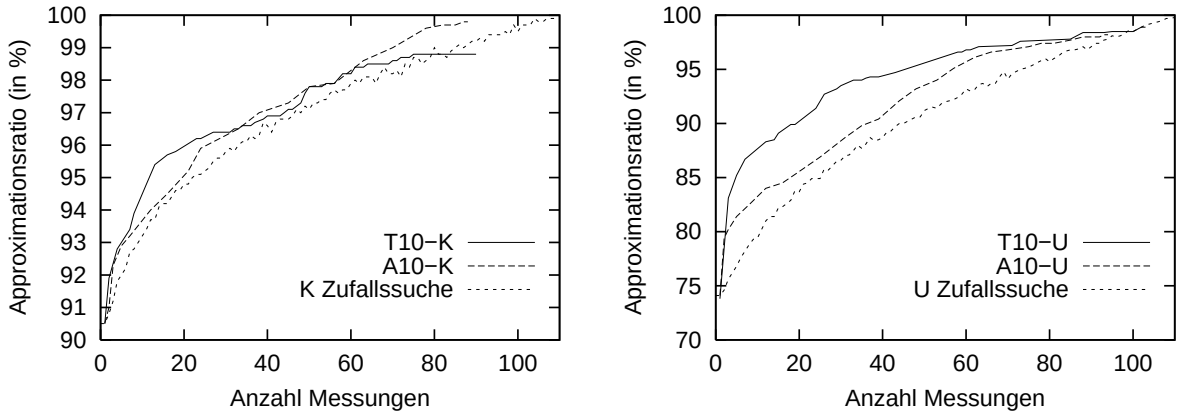


Abbildung 5.9: Anpassung einfacher Endsysteme: Approximationsratio vs. Messaufwand bei Knotenzahl (links) und Umlaufzeit (rechts).

Die initiale Approximation ist beim Distanzmaß Knotenzahl wesentlich besser als bei der Umlaufzeit. Dies korrespondiert exakt mit den unterschiedlichen in Abschnitt 5.1 festgestellten Divergenzverteilungen.

Für das Distanzmaß Knotenzahl ist anfangs das Kriterium *Max-10-Trennung* das etwas Bessere, danach ist es genau umgekehrt. Der Effekt schlecht balancierter Bäume scheint keine wesentliche Rolle mehr zu spielen. Bei der Umlaufzeit ist die *Max-10-Trennung* wie schon in den vorangegangenen Experimenten das bessere Ballungskriterium. Das Zufallsverfahren ist für beide Distanzmaße das Schlechteste, wenn auch sein Nachteil beim Distanzmaß Knotenzahl nur gering ausfällt.

Insgesamt erweist sich das Anpassungsverfahren als recht erfolgreich. Die Knotenzahl kann im Mittel mit 11 Messungen pro Endsystem auf 95% dem Optimum angenähert werden. 22 Messungen werden hierzu beim Zufallsverfahren benötigt. Die Umlaufzeit lässt sich mit immerhin 20 Messungen auf 90% annähern (44 Messungen beim Zufallsverfahren).

5.4 Diskussion

Die Experimente zeigen, dass die hierarchische Zerlegung in Regionen und Teilregionen bereits mit 120 Mess–Servern sowohl bezüglich der Knotenzahl als auch der Umlaufzeit

gut funktioniert und mit vertretetbarem Messaufwand zu bestimmen ist. Schon eine solche Zerlegung kann sich für viele Netzwerkanwendungen als sehr nützlich erweisen.

Die endgültige Schätzung der Distanzen zwischen einfachen Endsystemen ist hingegen noch nicht befriedigend. Zwar wird die Größenordnung der Distanzen richtig geschätzt, darüber hinaus ist der Schätzwert allerdings mit einem großen Fehler behaftet. Abermals ist festzustellen, dass bereits die korrekte Schätzung der Größenordnung für viele Netzwerkanwendungen sehr nützlich sein kann. Dies wird sich z.B. auch bei den nachfolgend vorgestellten Experimenten zur Aussendung mobiler Filterprogramme zeigen. Um genauere Schätzungen zu erlangen, müsste für das Distanzmaß Knotenzahl die Zahl der verfügbaren Mess–Server deutlich erhöht werden. Damit ließe sich die mittlere Distanz zwischen Endsystemen und ihrem jeweils nächstgelegenen Mess–Server verringern womit die Schätzung anhand der Distanz zwischen Mess–Servern letztlich genauer würde. Beim Distanzmaß Umlaufzeit ließe sich voraussichtlich bereits mit einem genaueren Messverfahren (z.B. wie in [Acharya & Saltz, 1996]) eine deutliche Verbesserung bei der Distanzschätzung erreichen.

Des weiteren haben die Experimente gezeigt, dass die Effizienz der einzelnen Verfahren stark von dem verwendeten Distanzmaß abhängt. Für die Knotenzahl hat sich das Ballungskriterium der *Min-k-Ausdehnung* aufgrund seiner guten Balancierungseigenschaften als das bessere erwiesen. Für die Umlaufzeit hingegen liefert das Kriterium der *Max-k-Trennung* genauere Ergebnisse. Von den Integrationsverfahren ist die Tiefensuche für beide Distanzmaße das effizientere. Die Kalibrierung der Verfahren wie z.B. die Wahl der Ähnlichkeitsschranke hängt wiederum wesentlich vom verwendeten Distanzmaß ab. Diese enge Verknüpfung zwischen Verfahrenseffizienz und Distanzmaß zeigt, dass für jedes weitere Distanzmaß, für das eine Netzwerk–Distanzkarte erstellt werden soll, zunächst die Effizienz der einzelnen Verfahren zu erproben ist, um dann die jeweils Besten auszuwählen. Die erfolgreiche Abbildung von zwei so unterschiedlichen Distanzmaßen in Distanzkarten legt es jedoch nahe, dass der Ansatz prinzipiell auch für andere Distanzmaße anwendbar ist.

Schließlich zeigen die Experimente, dass Distanzkarten bereits mit einem verhältnismäßig geringem Aufwand an sich ändernde Netzwerkbedingungen angepasst werden können. Um hier in Zukunft auch Aussagen über die notwendige Häufigkeit und den Umfang von Anpassungen gewinnen zu können, sind jedoch noch wesentlich umfangreichere Datenerhebungen, evtl. sogar kontinuierliche Messungen erforderlich.

Kapitel 6

Evaluierung der Aussendungsverfahren

Um die Effizienz der nach den Verfahren aus Kapitel 4 berechneten Aussendungspläne zu evaluieren, wurden zahlreiche Experimente unternommen. Diese basieren auf Distanzmessungen, die im Internet vorgenommen wurden. Auf Basis dieser Distanzen wurden sowohl die Aussendungsverfahren für verschiedene Szenarien durchgeführt als auch ihr Nutzen bewertet. Die Experimente bestehen also aus Simulationen, die auf Basis realer Daten durchgeführt wurden. Methodik und Durchführung der einzelnen Experimente werden im Folgenden beschrieben.

6.1 Methodik

Die den Experimenten zugrundeliegenden Distanzmessungen sind dieselben, die bereits zur Evaluierung von Netzwerk–Distanzkarten in Kapitel 5 verwendet wurden. Details zur Datenerhebung finden sich in Kapitel 5.1.

Alle Experimente dieses Kapitels basieren auf der Datenerhebung $D1$, welche Distanzmessungen für die Distanzmaße *Knotenzahl*$^{\uparrow}$ (engl. *hop-count*) und *Umlaufzeit*$^{\uparrow}$ (engl. *round trip time*) zwischen 119 Mess–Servern sowie von diesen zu 460 weiteren Endsystemen umfasst. Mess–Server und Endsysteme erstrecken sich jeweils über alle 5 Kontinente sowie über zahlreiche *autonome Systeme*$^{\uparrow}$. Von den 68782 Messungen, die für die Datenerhebung $D1$ durchgeführt wurden, waren 67961 erfolgreich bestimmbar. Nicht messbare Distanzen wurden für die folgenden Experimente durch den Median der jeweiligen Distanzmaße ersetzt. Der Median wurde anstelle des Wertes ∞ gewählt, da auf diese Weise

auch Daten–Server, zu denen keine erfolgreiche Messung von der Ausgangsplattform aus möglich war, in die Berechnung und Analyse von Aussendungsplänen mit einbezogen werden konnten. Der Median des Maßes Knotenzahl beträgt 16, der für die Umlaufzeit $168ms$.

Für die Experimente wurde angenommen, dass die Mess–Server als Codeplattform und die weiteren Endsysteme als Daten–Server fungieren. Die Einschränkung, dass nur Mess–Server als Codeplattform benutzt werden können, ergibt sich aus der Notwendigkeit, dass bei der Bewertung der mit den berechneten Aussendungsplänen erreichten Effizienz die wahre (d.h. gemessene) Distanz zwischen Codeplattformen und anderen Endsystemen bekannt sein muss. Im Übrigen lassen sich Codeplattformen ja auch sehr einfach zu Mess–Servern ausbauen, indem zu ihnen mobile Mess–Programme gesandt werden.

Einige Experimente basieren auf Distanzschätzungen durch Distanzkarten. Letztere wurden ebenfalls anhand der Datenerhebung $D1$ berechnet. Da jeder Mess–Server als Codeplattform fungiert, entfällt die in Kapitel 4.5 beschriebene Transformation der Regionenbäume.

Um nun ein Aussendungsverfahren für ein einzelnes Distanzmaß bewerten zu können, wurde jeweils eine ganze Versuchsreihe gestartet. Hierbei wurden verschiedene Testszenarien evaluiert, die sich jeweils in der Zahl der zu untersuchenden Daten–Server und im pro Server zu untersuchenden Datenvolumen unterscheiden. Zu jedem Testszenario wurden 100 Testläufe durchgeführt, jeweils mit einer zufällig gewählten Ausgangsplattform und zufällig gewählten Daten–Servern. Die dargestellten Ergebnisse sind immer über diese 100 Testläufe gemittelt. Der Umfang des mobilen Filterprogrammes v_F wurde auf 1 normiert. Datenvolumina werden immer als Vielfache hiervon beschrieben. Um die Zahl der freien Parameter zu reduzieren und damit die graphische Darstellung zu vereinfachen, wurde das pro Server zu untersuchende Datenvolumen v_U auf einen für alle Server gleichen Wert gesetzt, das Ergebnisvolumen v_R wurde auf 0 festgelegt. Experimente mit verschiedenen Werten für v_R führen zu den gleichen Charakteristiken. Daher bedeutet die Festlegung auf $v_R = 0$ an dieser Stelle keine wirkliche Einschränkung der Aussagekräftigkeit der Ergebnisse.

Die Bewertung eines Aussendungsplanes erfolgt durch die mit diesem erreichte Reduktion der Kommunikationskosten gegenüber der klassischen Klient–Server–Lösung, bei der alle zu untersuchenden Daten auf die Ausgangsplattform heruntergeladen werden müssen. Die Kostenreduktion berechnet sich zu

$$1 - \frac{\textit{mit dem Aussendungsplan verbundene Kommunikationskosten}}{\textit{Kommunikationskosten der Klient–Server–Lösung}}. \qquad (6.1)$$

Die theoretisch maximale Kostenreduktion von 100% würde dann erreicht, wenn die Ausführung des Aussendungsplanes gar keine Kommunikationskosten verursachen würde.

6.2 Basisalgorithmus mit gemessenen Distanzen

Als Erstes wurde die Effektivität des Basisalgorithmus aus Kapitel 4.4 auf Basis der direkt gemessenen Distanzen aus Datenerhebung $D1$ evaluiert. Dieses Szenario stellt also gewissermaßen den Optimalfall dar, in dem exaktes Distanzenwissen zugrundeliegt und das exaktere (wenn auch schlechter skalierende) der beiden Aussendungsverfahren verwendet wird.

Es wurden Experimente für unterschiedlich viele zu untersuchende Daten–Server (1, 10, 50, 100, ..., 400) und mit Datenvolumina v_U zwischen 0,25 und 4 durchgeführt. Abbildung 6.1 zeigt die Ergebnisse (Kostenreduktion und Zahl der notwendigen Filtertransfers) für das Distanzmaß Knotenzahl. Alle Ergebnisse sind jeweils über 100 Testläufen gemittelt.

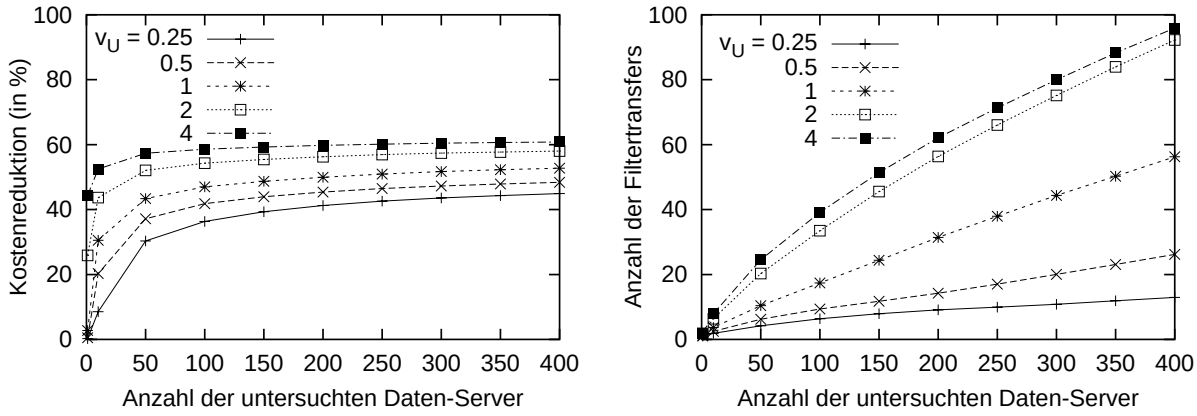


Abbildung 6.1: Basisalgorithmus (Knotenzahl): Kostenreduktion (links) und Anzahl der notwendigen Filtertransfers (rechts)

Man sieht, dass bereits bei einem Datenvolumen von $v_U = 0,25$, also einem Viertel des Umfanges des Filterprogrammes, und 50 zu untersuchenden Daten–Servern eine Kostenreduktion von mehr als 30% erreicht wird. Ist das Datenvolumen um das Vierfache größer als das Filterprogramm, so wird bei 50 Daten–Servern sogar eine Reduktion von beinahe 60% erreicht. Bei gegebenem Datenvolumen v_U kommt die erzielbare Kostenreduktion bereits bei 50–100 Daten–Servern ihrem Maximalwert sehr nahe. Mit weiteren Daten–Servern wird nur noch eine geringfügige prozentuale Verbesserung erreicht.

Die Zahl der notwendigen Filtertransfers wächst etwas schwächer als linear mit der Zahl der zu untersuchenden Daten–Server. Ebenso steigt die Zahl der Filtertransfers bei größeren Datenvolumina, was damit übereinstimmt, dass sich Filtertransfers bei größeren Datenvolumina eher lohnen sollten. Dieser Effekt verliert sich allerdings für große Datenvolumina (in der Abbildung für ein Volumen von $v_U = 4$), da in diesem Fall die Größe des Filterprogrammes nur noch einen geringen Einfluss auf die Transferentscheidung hat.

Da nach dem Systemmodell aus Kapitel 4.2.1 Daten–Server nicht gleichzeitig Codeplattformen sein müssen und es in den Experimenten auch nicht sind, muss es eine obere Schranke für die erreichbare Kostenreduktion geben, die sich aus dem Abstand zwischen Daten–Servern und ihren jeweils nächstgelegenen Codeplattformen bestimmt. Um diese obere Schranke abzuschätzen wurde ein weiteres Experiment mit 400 Daten–Servern und einem Datenvolumen von 100 durchgeführt. Die erzielte Reduktion liegt mit 64% nur knapp über der maximalen Reduktion aus Abbildung 6.1.

Die analogen Experimente wurden für das Distanzmaß Umlaufzeit durchgeführt. Abbildung 6.2 zeigt die erzielten Ergebnisse. Was sofort auffällt, sind die im Vergleich zum

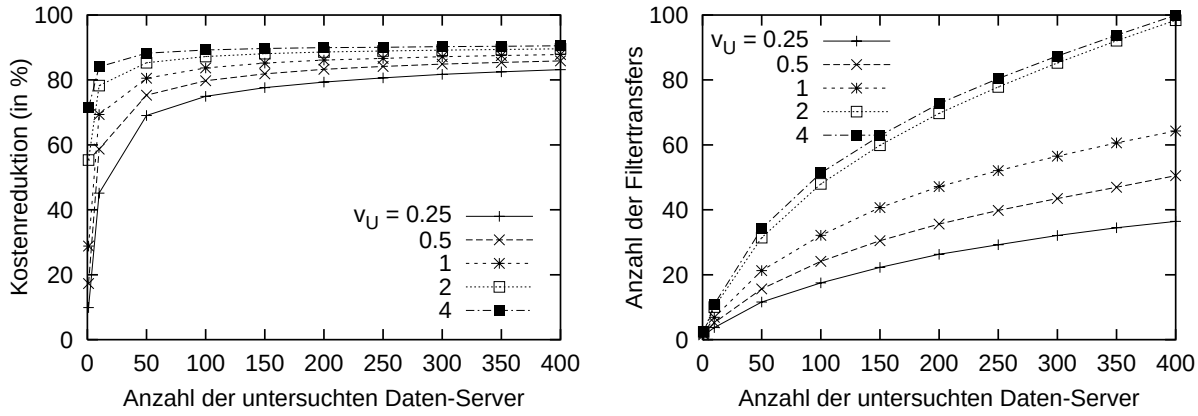


Abbildung 6.2: Basisalgorithmus (Umlaufzeit): Kostenreduktion (links) und Anzahl der notwendigen Filtertransfers (rechts)

Distanzmaß Knotenzahl wesentlich höheren Kostenreduktionen. Die Ursache für diesen Effekt findet sich in der unterschiedlichen empirischen Verteilung dieser beiden Distanzmaße (vergleiche mit Kapitel 5.1). Für das Distanzmaß Knotenzahl beträgt der mittlere Abstand zwischen Daten–Servern und ihren jeweils nächstgelegenen Codeplattformen 7,1, was ungefähr zwei Fünfteln der durchschnittlichen Knotenzahl zwischen beliebigen Endsystemen entspricht. Demgegenüber sind nach dem Distanzmaß Umlaufzeit Daten–Server im Mittel $27ms$ von ihrer nächstgelegenen Codeplattform entfernt, was lediglich

ein Zehntel der durchschnittlichen Umlaufzeit darstellt. Aus diesem Grund kann ein Filterprogramm im Sinne der Umlaufzeit wesentlich näher an die jeweiligen Daten–Server herangebracht werden, als dies beim Distanzmaß Knotenzahl möglich ist.

Für die Zahl der in den einzelnen Konstellationen notwendigen Filtertransfers gilt im Wesentlichen das Gleiche wie beim Distanzmaß Knotenzahl. Die mit einem Szenario von 400 Daten–Servern und einem Datenvolumen von 100 geschätzte maximal erreichbare Kostenreduktion beträgt 91%.

6.3 Basisalgorithmus mit geschätzten Distanzen

In den nachfolgend beschriebenen Experimenten wird abermals die Effektivität des Basisverfahrens untersucht, diesmal allerdings auf Basis der anhand von Netzwerk–Distanzkarten geschätzten Distanzen. Es werden jeweils Regionenbäume verwendet, die einen Baumgrad von $k = 10$ haben und mit den Ballungskriterien *Max-k-Trennung* und *Min-k-Ausdehnung* berechnet wurden (vgl. mit Kapitel 3.4.1). Die Bestimmung der jeweils erreichten Kostenreduktion erfolgt wieder anhand der gemessenen (wahren) Distanzen.

Abbildung 6.3 zeigt die mit dem Basisverfahren beim Distanzmaß Knotenzahl erreichbare Kostenreduktion, jeweils für die Ballungskriterien *Max-k-Trennung* und *Min-k-Ausdehnung*. Die erreichten Kostenreduktionen liegen signifikant unter denjenigen, die mit dem Basisalgorithmus für gemessene Distanzen erreicht wurden. Von den beiden Ballungskriterien führt die *Min-k-Ausdehnung* zu etwas effizienteren Aussendungsplänen. Es wird im besten Fall (bei 400 Daten–Servern und $v_U = 4$) eine Reduktion von 57% erreicht.

Um die Unterschiede zwischen den einzelnen Verfahren klarer herauszuheben, zeigt Abbildung 6.4 die für alle Testszenarien auftretende Differenz zwischen den erreichten Kostenreduktionen. Das Kriterium der *Max-k-Trennung* erweist sich gegenüber dem Kriterium der *Min-k-Ausdehnung* in fast allen Szenarien als unterlegen, wobei dies am stärksten für kleine Datenvolumina und zahlreiche Daten–Server zu Tage tritt. Das Kriterium der *Min-k-Ausdehnung* wiederum erreicht in allen Situationen schlechtere Resultate als der mit gemessenen Distanzen durchgeführte Basisalgorithmus. Der Abstand zwischen beiden Verfahren verringert sich mit zunehmendem Datenvolumen und mit steigender Zahl der Daten–Server bis auf 6 Prozentpunkte.

Die in den einzelnen Szenarien durchschnittliche Zahl der benötigten Filtertransfers zeigt ähnliche Charakteristiken wie bei den Experimenten im vorangehenden Abschnitt.

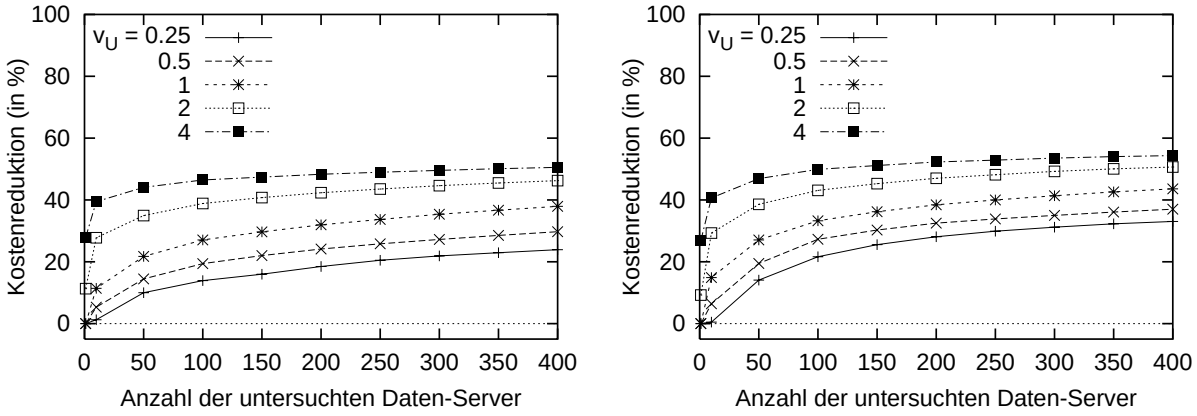


Abbildung 6.3: Basisalgorithmus (Knotenzahl) basierend auf Regionenbaum mit Kriterium *Max-k-Trennung* (links) und *Min-k-Ausdehnung* (rechts)

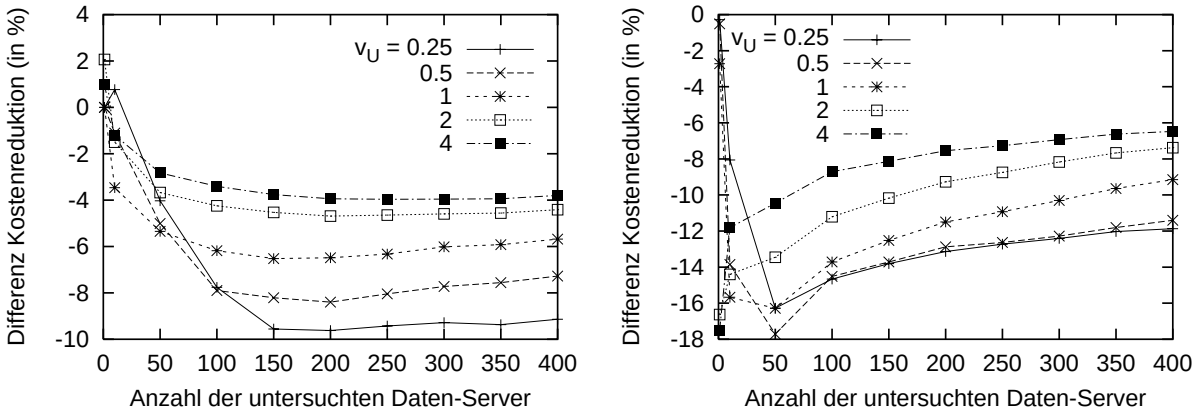


Abbildung 6.4: Vergleich Basisalgorithmus (Knotenzahl): *Max-k-Trennung* vs. *Min-k-Ausdehnung* (links) und *Min-k-Ausdehnung* vs. gemessene Distanzen (rechts)

Die Abbildungen 6.5 und 6.6 auf der nächsten Seite zeigen die Resultate zu den analogen Experimenten für das Distanzmaß Umlaufzeit. Qualitativ ergeben sich die gleichen Ergebnisse wie beim Distanzmaß Knotenzahl. Abermals erweist sich das Kriterium der *Min-k-Ausdehnung* dem Kriterium der *Max-k-Trennung* überlegen. In einem Szenario von 400 Daten–Servern und bei einem Datenvolumen von $v_U = 100$ wird mit der *Min-k-Ausdehnung* eine Kostenreduktion von 82% erreicht.

Für lediglich einen zu untersuchenden Daten–Server ergibt sich bei einem geringen Datenvolumen sogar eine negative Kostenreduktion (-1% beim Kriterium *Max-k-Trennung* und -2% beim Kriterium *Min-k-Ausdehnung*. Für einzelne Aussendungspläne ist so etwas

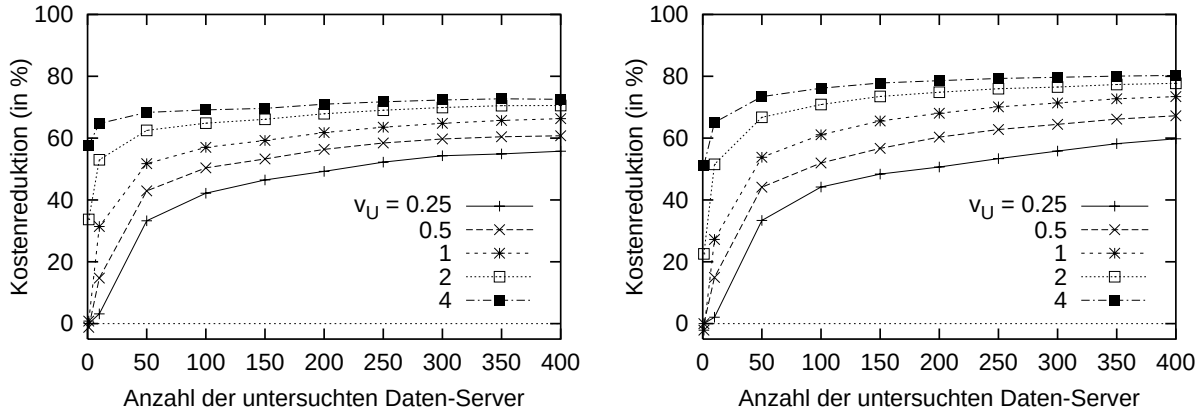


Abbildung 6.5: Basisalgorithmus (Umlaufzeit) basierend auf Regionenbaum mit Kriterium *Max-k-Trennung* (links) und *Min-k-Ausdehnung* (rechts)

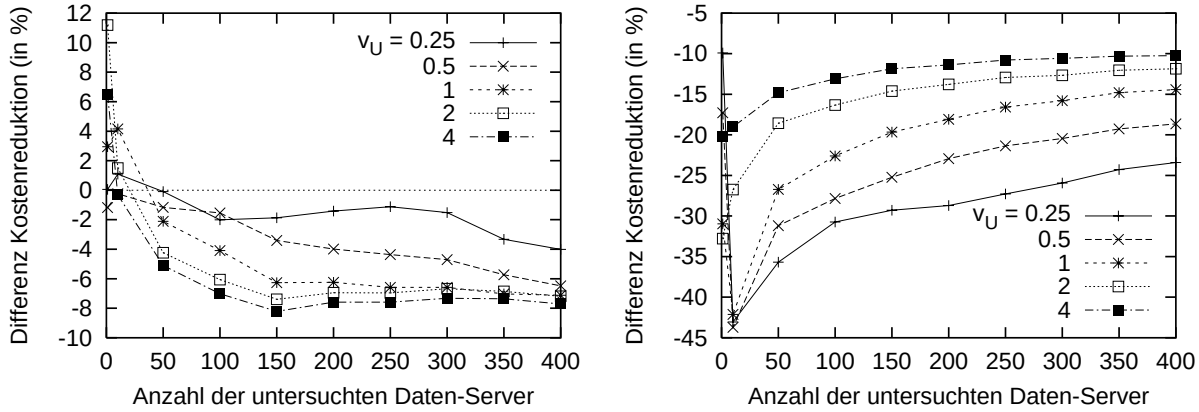


Abbildung 6.6: Vergleich Basisalgorithmus (Umlaufzeit): *Max-k-Trennung* vs. *Min-k-Ausdehnung* (links) und *Min-k-Ausdehnung* vs. gemessene Distanzen (rechts)

aufgrund von Schätzfehlern natürlich immer möglich. Dass ein solcher Fehler auch im Mittel von 100 Testläufen auftreten kann, weist auf einen leichten systematischen Fehler in der Distanzschätzung durch Netzwerk–Distanzkarten hin. In der Tat haben Experimente gezeigt, dass die Distanz zu Endsystemen tendenziell leicht unterschätzt wird.

Der Unterschied zwischen der durch die Distanzschätzung mit dem Kriterium der *Min-k-Ausdehnung* und der durch gemessene Distanzen erreichbaren Kostenreduktion fällt um Einiges größer als bei der Knotenzahl aus. Er geht aber ebenso mit zunehmendem Datenvolumen und mit steigender Zahl der Daten–Server zurück. Im besten Fall beträgt der Unterschied allerdings immer noch 10 Prozentpunkte.

Die in den einzelnen Szenarien durchschnittliche Zahl der benötigten Filtertransfers zeigt

ähnliche Charakteristiken wie bei den Experimenten im vorangehenden Abschnitt.

6.4 Hierarchischer Algorithmus

Die Experimente zum hierarchischen Algorithmus basieren notwendigerweise auf durch Netzwerk–Distanzkarten geschätzten Distanzen. Es wurden dieselben Regionenbäume wie im vorangehende Abschnitt verwendet, d.h. Bäume mit einen Baumgrad von $k = 10$, die mit den Ballungskriterien *Max-k-Trennung* und *Min-k-Ausdehnung* berechnet wurden. Die Bestimmung der jeweils erreichten Kostenreduktion erfolgt abermals anhand der gemessenen (wahren) Distanzen.

Abbildung 6.7 zeigt für das Distanzmaß Knotenzahl die mit den beiden Ballungskriterien jeweils erzielten Kostenreduktionen. Auch hier erweist sich das Kriterium der *Min-k-Ausdehnung* dem Kriterium der *Max-k-Trennung* überlegen, was sich auch im direkten Vergleich der beiden Kriterien in Abbildung 6.8 zeigt. Abermals wird ein negativer Reduktionswert erreicht, diesmal bei 10 zu untersuchenden Daten–Servern und einem Datenvolumen von $v_U = 0.25$. Er beträgt -3% für das Kriterium *Max-k-Trennung* und -2% für die *Min-k-Ausdehnung*.

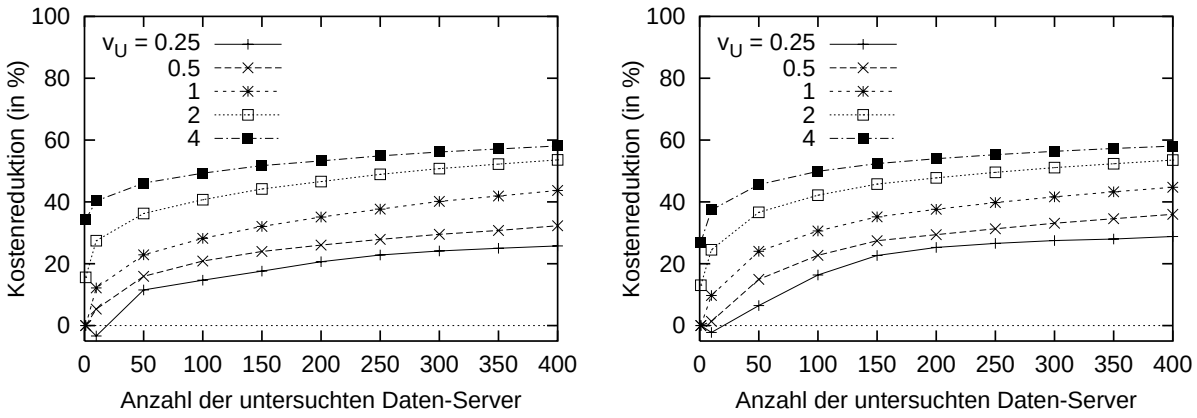


Abbildung 6.7: Hierarchisches Verfahren (Knotenzahl) basierend auf Regionenbaum mit Kriterium *Max-k-Trennung* (links) und *Min-k-Ausdehnung* (rechts).

Interessant ist der Vergleich zum auf Distanzschätzungen basierenden Basisverfahren im rechten Teil von Abbildung 6.8. Offensichtlich scheint sich die beim hierarchischen Algorithmus vorgenommene Abstraktion auf virtuelle Codeplattformen und Daten–Server, die jeweils ganze Regionen umfassen, nicht nachteilig auszuwirken. Im Gegenteil wird

für große Datenvolumina und zahlreiche Daten–Server sogar ein besseres Ergebnis als für das Basisverfahren erreicht. Für das Szenario mit 400 Daten–Servern und einem Datenvolumen von $v_U = 100$ wird sogar eine Reduktion von 63% erreicht, was nur um 1 Prozentpunkt unter dem Wert des Basisverfahrens mit gemessenen Distanzen liegt.

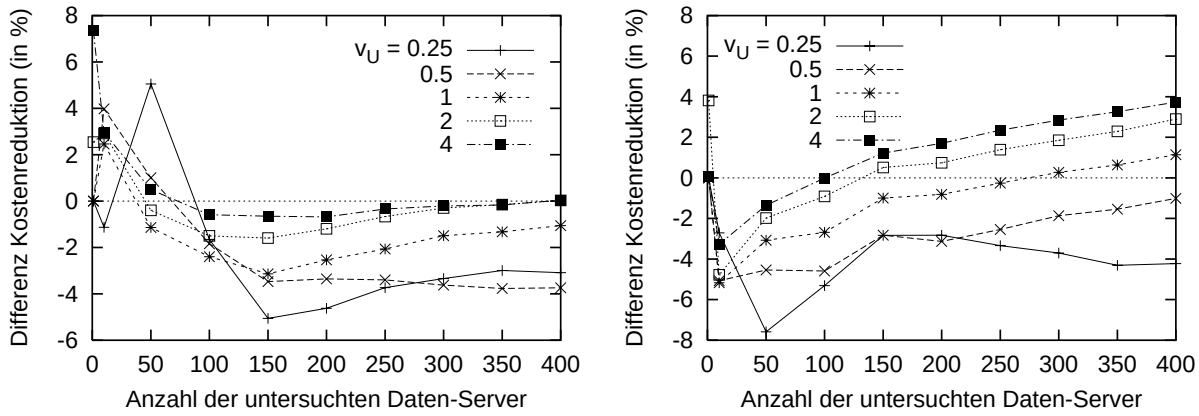


Abbildung 6.8: Vergleich (Knotenzahl): hierarchisches Verfahrens mit *Max-k-Trennung* vs. hierarchisches Verfahrens mit *Min-k-Ausdehnung* (links) und hierarchischen Verfahrens mit *Min-k-Ausdehnung* vs. Basisverfahren mit *Min-k-Ausdehnung* (rechts)

Die in den einzelnen Szenarien durchschnittliche Zahl der benötigten Filtertransfers zeigt die schon bekannten Charakteristiken.

Die Abbildungen 6.9 und 6.10 zeigen die analogen Resultate für das Distanzmaß Umlaufzeit. Qualitativ ergeben sich die gleichen Ergebnisse wie beim Distanzmaß Knotenzahl. Für das Szenario mit 400 Daten–Servern und einem Datenvolumen von $v_U = 100$ wird mit dem Kriterium der *Min-k-Ausdehnung* eine Kostenreduktion von 89% erreicht, womit das Optimum aus dem Basisalgorithmus mit gemessenen Distanzen bis auf 2 Prozentpunkte angenähert wird.

Die durchschnittliche Zahl der in den einzelnen Szenarien benötigten Filtertransfers zeigt die schon bekannten Charakteristiken.

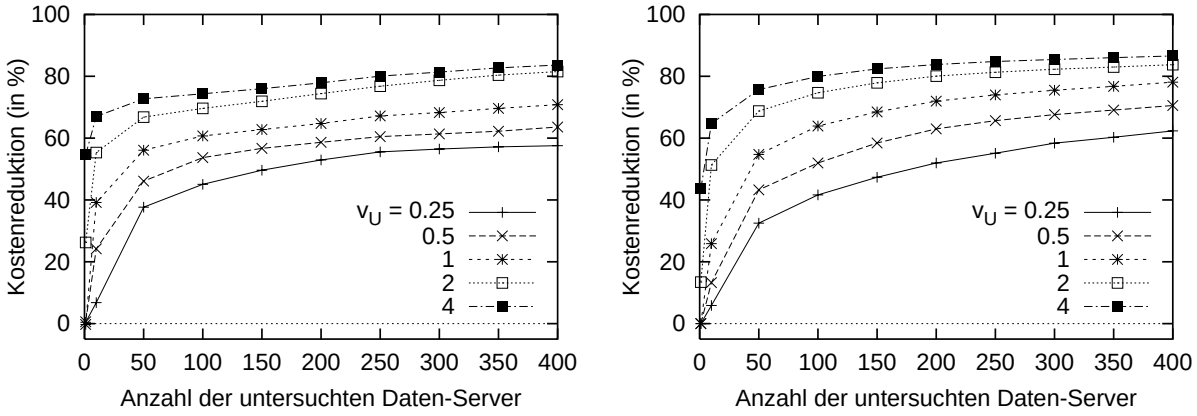


Abbildung 6.9: Hierarchisches Verfahren (Umlaufzeit) basierend auf Regionenbaum mit Kriterium *Max-k-Trennung* (links) und *Min-k-Ausdehnung* (rechts)

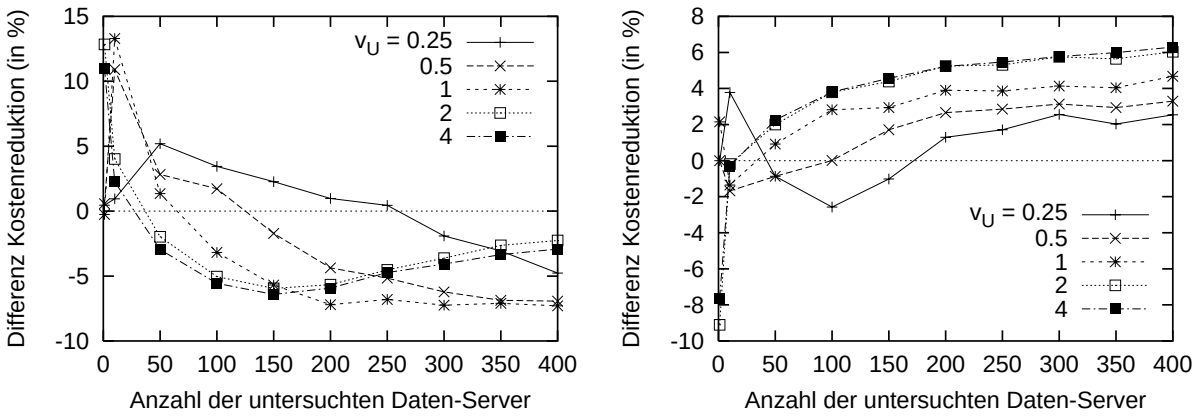


Abbildung 6.10: Vergleich (Umlaufzeit): hierarchisches Verfahrens mit *Max-k-Trennung* vs. hierarchisches Verfahrens mit *Min-k-Ausdehnung* (links) und hierarchischen Verfahrens mit *Min-k-Ausdehnung* vs. Basisverfahren mit *Min-k-Ausdehnung* (rechts)

6.5 Diskussion

Die Experimente haben gezeigt, dass die Filterung verteilter Datenbestände mit Hilfe mobiler Programme wesentlich effizienter als im herkömmlichen Klient–Server–Ansatz erfolgen kann. Je nach verwendetem Distanzmaß ist eine Reduktion der Kommunikationskosten bis zu 90% (Umlaufzeit) bzw. gut 60% (Knotenzahl) möglich.

Der Basisalgorithmus leistet dabei die Berechnung effizienter Aussendungspläne. Die mit ihm erreichbare Effizienz geht jedoch zurück, wenn anstelle der direkt gemessenen

Netzwerkdistanzen auf Distanzschätzungen aus Netzwerk–Distanzkarten zurückgegriffen werden muss. Aber auch hier werden noch signifikante Kostenreduktionen erzielt, die den Einsatz mobiler Programme auf jeden Fall rechtfertigen.

Während der Basisalgorithmus noch schlechte Skalierungseigenschaften aufweist, wird dieser Mangel vom hierarchischen Algorithmus überwunden. Interessanterweise zeigen die Experimente, dass durch ihn im Mittel genauso gute Kostenreduktionen erreicht werden wie beim Basisalgorithmus mit geschätzten Distanzwerten. Im Falle großer Datenvolumina werden sogar die gleichen Werten wie beim Basisverfahren mit gemessenen Distanzen erreicht.

Von den beiden Ballungskriterien, mit denen Regionenbäume bzw. Distanzkarten berechnet werden können, hat sich das Kriterium der *Min-k-Ausdehnung* als das etwas bessere erwiesen. Dies korrespondiert genau mit der besseren Distanzschätzung (insbesondere auf tieferen Baumebenen), die durch dieses Kriterium erreicht wird (vgl. mit Kapitel 5.2).

Kapitel 7

Fallstudie zur Domäne „wissenschaftliche Artikel“

Um den Ansatz der Domänen–Experten aus Kapitel 2 auch praktisch zu validieren, wurde eine Fallstudie für das Gebiet der „wissenschaftlichen Artikel“ durchgeführt. Hierzu wurde das von [Arnold, 1999] entwickelte *Framework* für Domänen–Experten um eine spezifische Filterkomponente erweitert, die in der Lage ist wissenschaftliche Artikel zu erkennen. Dieser *Artikel–Experte* arbeitet bisher ausschließlich für englischsprachige Texte, die als HTML-Datei vorliegen. Erweiterungen auf andere Sprachen und Dateiformate sind jedoch prinzipiell sehr einfach möglich, wie die Diskussion in Abschnittes 7.1 zeigen wird.

Natürlich ist der Begriff des „wissenschaftlichen Artikels“ nicht eindeutig definiert und es mag somit bei einzelnen Texten strittig sein, ob diese als wissenschaftlich zu bezeichnen sind. Trotzdem besteht ganz offensichtlich ein Bedarf nach Suchwerkzeugen für solche Artikel, was sich allein schon an der Vielzahl der Systeme zeigt, die für dieses Gebiet entwickelt bzw. betrieben werden. Zu diesen gehören zahllose Kataloge, die oft auf ein einzelnes Themengebiet abgestimmt sind, aber auch spezialisierte Suchmaschinen, wie die von [Monge & Elkan, 1996] und [Han et al., 1996]. Diese greifen beide auf eine Reihe von spezialisierten externen Diensten zurück, wobei es sich im Einzelnen um Bibliographieserver, sowie um Dienste zur Bestimmung von Institutionen und persönlichen *Homepages* handelt. Schließlich wurde von [Lawrence et al., 1999] ein spezialisiertes Suchwerkzeug entwickelt, das genau wie Domänen–Experten Hinweise mit einer klassischen Suchmaschine sammelt und die verwiesenen Dokumente danach genauer auf ihre Relevanz hin analysiert. Im Gegensatz zu Domänen–Experten erfolgt jedoch keinerlei Erweiterung des

Ansatzes zu einem allgemeinen Framework, noch wird das Problem der effizienten Lokalisierung angegangen.

Die weiteren Abschnitte dieses Kapitels sind wie folgt gegliedert: Nach der Beschreibung der für die Erkennung wissenschaftlicher Artikel entwickelten Filterfunktion in Abschnitt 7.1 wird in Abschnitt 7.2 deren Effektivität anhand einiger Beispielanfragen evaluiert. In Abschnitt 7.3 wird dann die mit dem Einsatz mobiler Filterprogramme erreichbare Reduktion der Kommunikationskosten evaluiert (abermals anhand der schon vorher benutzten Beispielanfragen). Abschnitt 7.4 fasst das Kapitel zusammen.

Die Resultate zur Filterfunktion und ihrer Effektivität wurden erstmals in [Theilmann & Rothermel, 1999c] vorgestellt.

7.1 Filterverfahren

Es muss betont werden, dass der entwickelte und nachfolgend vorgestellte Algorithmus zur Erkennung wissenschaftlicher Artikel in keinster Weise mit dem Stand der Wissenschaft im Bereich der künstlichen Intelligenz und der Inhaltsfilterung konkurrieren soll. Er dient lediglich der Validierung des Konzeptes der Domänen–Experten. Daher erfolgt die Vorstellung des Filterverfahrens auch nur in sehr knapper Form.

Allgemeiner Ablauf der Dokumentenfilterung

Wie bereits in Kapitel 2.3 auf Seite 26 angesprochen, erfolgt die Analyse eines Dokumentes durch die Filterfunktion in drei Schritten. Zunächst wird das Dokument in eine Folge von Tokens transformiert. Diese Tokens sind noch abhängig vom Dokumententyp. Bei einem HTML-Dokument treten zum Beispiel Formatierungs–Tokens wie „<H1>“ (leitet eine Überschrift der ersten Ordnung ein) aber auch ganz einfache Text–Tokens auf. Diese für den Dokumententyp spezifischen Tokens werden alsdann in eine Folge von domänenspezifischen Tokens transformiert, die noch genau die für den eigentlichen Filterungsprozess relevanten Informationen beinhalten. Auf Basis dieser Tokens bestimmt die eigentliche Filterfunktion zu einem untersuchten Dokument eine Bewertungszahl zwischen 0 und 1, mit der die Plausibilität ausgedrückt wird, dass das Dokument relevant im Sinne des Domänen–Experten ist. Eine vorher festzulegende *Akzeptanzschranke* entscheidet dann darüber, welche Dokumente schließlich als relevant klassifiziert werden.

Auf diese Weise kann bei der Filterung vom ursprünglichen Dokumententyp abstrahiert werden. Der Aufwand zur Anpassung bzw. Erweiterung der Filterfunktion auf einen neuen

Dokumententyp beschränkt sich damit auf die Entwicklung einer Transformationskomponente, mit der die für den Dokumententyp spezifischen Tokens in domänenspezifische Tokens umgewandelt werden können.

Ein (mobiles) Filterprogramm sendet nach Analyse der von ihm zu untersuchenden Dokumente die Klassifikation aller Dokumente sowie die akzeptierten/relevanten Dokumente an den Domänen–Experten.

Domänenspezifische Realisierung

Die für die Domäne der wissenschaftlichen Artikel spezifischen Tokens, die im Rahmen der Filterung erzeugt werden, bestehen aus einfachen Text–Tokens, die mit den folgenden drei Attributen versehen sind:

1. dem *Hervorhebungsgrad*, der sich aus der verwendeten Schriftgröße, dem Schriftstil und anderen Formatierungs–Tokens ergibt,
2. dem *Trennungsgrad*, der angibt, wie stark ein Text–Token von seinem Vorgänger–Token getrennt ist (z.B. durch einen Zeilenvorschub, einen Absatz oder eine horizontale Linie), und
3. dem *Ausrichtungsgrad*, der die Tiefe einer evtl. Einrückung des jeweiligen Text–Tokens wiedergibt.

Es sei nochmals betont, dass diese Attribute unabhängig von irgendwelchen Dokumententypen sind. Alle drei werden durch Zahlen ausgedrückt, die relativ zueinander interpretiert werden. Daneben sind sie auch nicht besonders spezifisch für die Domäne der wissenschaftlichen Artikel. Es ist also durchaus denkbar, dass sie (und damit auch die entsprechenden Transformationskomponenten) auch in anderen Domänen nützlich und anwendbar sind.

Basierend auf diesen attributierten Tokens versucht die Filterfunktion in einem zu untersuchenden Dokument Strukturelemente zu finden, die typischerweise in wissenschaftlichen Artikeln auftreten. Hierzu gehören der Titel, die Autorennamen, eine Kurzfassung, der Textkorpus bestehend aus mehreren Abschnitten, Zitierstellen sowie ein Abschnitt über die referenzierte Literatur. Die Erkennung von Kandidaten für Strukturelemente wird durch den Zustand des Filterprozesses (z.B. wird der Titel eines Artikels nur am Anfang des Prozesses gesucht) und durch die domänenspezifischen Tokens selbst getriggert (z.B. durch spezielle Schlüsselwörter wie „abstract“ oder „references“ oder durch einen sich ändernden Hervorhebungsgrad).

Um einen beliebigen Kandidaten für ein Strukturelement bewerten zu können, werden alle Kandidaten durch spezielle Attribute beschrieben. Solche Attribute sind z.B. die Zahl der Wörter, die Position innerhalb des Textes, der Hervorhebungsgrad u.a. Die konkrete Ausprägung dieser Attribute variiert natürlich zwischen einzelnen Dokumenten, sollte aber aufgrund der stereotypen Struktur von wissenschaftlichen Artikeln innerhalb eines typischen Bereichs bleiben. Um den jeweils zulässigen/typischen Bereich eines Strukturattributes auf robuste und fehlertolerante Weise zu beschreiben, werden unscharfe Mengen (engl. *fuzzy sets*) benutzt. Abbildung 7.1 gibt ein Beispiel, wie solche Mengen für Attribute, die zur Beschreibung von Dokumentabschnitten benutzt werden, aussehen können. Die ersten beiden Beispielattribute beziehen sich jeweils auf einzelne Abschnitte, das dritte Attribut auf die Gesamtmenge an Abschnitten. Der auf der y-Achse aufgetragene Parameter μ gibt für jeden Wert der x-Achse den Grad seiner Gültigkeit an.

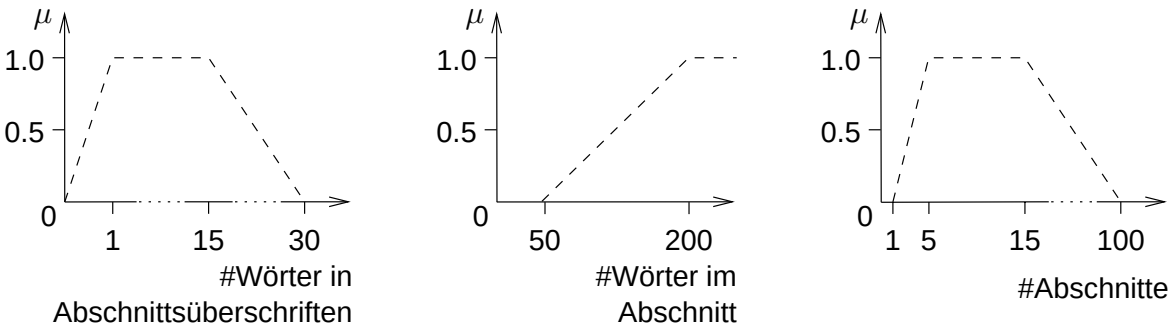


Abbildung 7.1: Unscharfe Mengen zur Definition typischer Strukturattribute zu den Abschnitten eines Dokuments

Mit Hilfe solcher unscharfen Mengen kann ein beliebiger Kandidat für ein Strukturelement bewertet werden und aus einer Vielzahl von Kandidaten kann somit der beste (d.h. der typischste) ausgewählt werden.

Hat man für alle Strukturelemente, die in einem wissenschaftlichen Artikel vorkommen sollen, den jeweils besten Kandidaten bestimmt, so kann man die Bewertungen von diesen mit Operationen der unscharfen Logik zu einer Gesamtbewertung des Dokumentes zusammenfassen. Diese gibt dann die Plausibilität wieder, dass das Dokument wirklich ein wissenschaftlicher Artikel, also relevant im Sinne des Domänen–Experten ist.

Ein schöner Nebeneffekt des beschriebenen Filterverfahrens ist, dass es gleichsam zur Dokumentenanalyse (vgl. mit Kapitel 2.2.1) genutzt werden kann. Mit ihm sind der Titel, der aus Autorennamen und -adressen bestehende Abschnitt, die Kurzfassung sowie einzelne Referenzen explizit bestimmbar.

Eine Anpassung der Filterfunktion auf weitere Sprachen neben Englisch ist einfach realisierbar, da die Filterfunktion im Wesentlichen sprachunabhängig ist. Einzig die sprachspezifischen Schlüsselwörter wie „abstract“ oder „references“, die zur Erkennung der jeweiligen Dokumentabschnitte benutzt werden, müssen auf andere Sprachen angepasst werden.

7.2 Effektivität des Filterverfahrens

Die Experimente zur Evaluation des Filterverfahrens hatten zwei Ziele: Zum einen sollte die Qualität des Verfahrens, d.h. der Prozentsatz der mit ihm korrekt klassifizierten Dokumente bestimmt werden. Daneben sollte aber auch ein Vergleich zu der mit herkömmlichen Suchmaschinen erzielbaren Suchqualität durchgeführt werden.

Methodik

Die nachfolgend vorgestellten Versuche basieren auf Beispielanfragen, die typischerweise im Kontext einer Suche nach wissenschaftlichen Artikeln auftreten könnten. Die Anfragen wurden an eine allgemeine Suchmaschine gesandt, die zurückgelieferten Dokumentenverweise (URLs) wurden gesammelt und die verwiesenen Dokumente wurden sowohl von Hand klassifiziert als auch dem Filterverfahren unterzogen. Die manuelle Klassifikation erlaubt dann die Bewertung des Qualität des Filterverfahrens.

Vorteil dieser Vorgehensweise ist, dass die Menge der untersuchten Dokumenten die enorme Vielfalt an Dokumenten im World Wide Web widerspiegelt. Der Nachteil dieser Methode besteht darin, dass aufgrund der mit einem hohen Aufwand verbundenen manuellen Klassifikation nur relativ wenige Dokumente in den Experimenten Berücksichtigung finden. Eine Alternative zu diesem Vorgehen bestünde darin, vorklassifizierte Dokumentenmengen (z.B. aus online verfügbaren Konferenzbänden) zu nehmen und dem Filterverfahren zu unterziehen. Allerdings hat ein derartiges Vorgehen den gravierenden Nachteil, dass solche Dokumentenmengen in der Regel eine sehr einheitliche Struktur aufweisen und damit nicht besonders repräsentativ für die im Web vorhandene Vielfalt sind.

Tabelle 7.1 auf der folgenden Seite beschreibt die drei zur Evaluation des Filterverfahrens durchgeführten Beispielanfragen. Die Syntax der Anfragen entspricht der von AltaVista[1], d.h. der '+'-Operator erzwingt das Vorkommen des nachstehenden Ausdrucks, Anführungsstriche umschließen Phrasen.

[1] AltaVista: URL: http://www.altavista.com

(1)	`+"mobile agent" +"information retrieval"`
(2)	`+multicast`
(3)	`+"software engineering" +"petri net" +invariant`

Tabelle 7.1: Drei Beispielanfragen

Alle drei Anfragen wurden an AltaVista gesandt und die zurückgelieferten Dokumentenverweise (URLs) wurden weiter untersucht. Tabelle 7.2 fasst einige charakteristische Werte zu den Anfragen zusammen: die Zahl der von AltaVista jeweils gelieferten Antworten, die Zahl der unter diesen Antworten gültigen Verweise (also der URLs die auf ein zugreifbares HTML-Dokument verweisen) sowie die durch manuelle Analyse bestimmte Zahl der wissenschaftlichen Artikel unter diesen Dokumenten. Für die zweite Anfrage wurden nur die ersten 200 der von AltaVista zurückgelieferten Verweise berücksichtigt.

Nr.	#Antworten	#gültige HTML-Dokumente	#wiss. Artikel
(1)	200	160	22
(2)	200 (>10.000)	127	29
(3)	113	66	4

Tabelle 7.2: Resultate zu den Beispielanfragen aus Tabelle 7.1

Qualität der Filterfunktion

Um die Qualität der Filterfunktion zu untersuchen, wurde zunächst die aus der Filterfunktion resultierende Gesamtbewertung für alle in den Beispielanfragen betrachteten Dokumente berechnet. Darauf aufbauend wurde analysiert, wie sich die Wahl der *Akzeptanzschranke* auf die abschließende Klassifikation der Dokumente auswirkt, d.h. für eine gegebene Akzeptanzschranke wie viele von den wissenschaftlichen Artikeln korrekt als solche erkannt werden und wie viele Dokumente irrtümlich als Artikel klassifiziert werden.

Abbildung 7.2 zeigt die Ergebnisse dieser Untersuchung für die Anfragen (1) und (2). Die x-Achse beschreibt jeweils die Akzeptanzschranke, die von einer Dokumentenbewertung mindestens erreicht werden muss, damit das Dokument als wissenschaftlicher Artikel klassifiziert wird. Eine Verringerung dieser Schranke erhöht zwangsläufig die Zahl der Positivklassifikationen, sowohl der korrekten als auch der irrtümlichen.

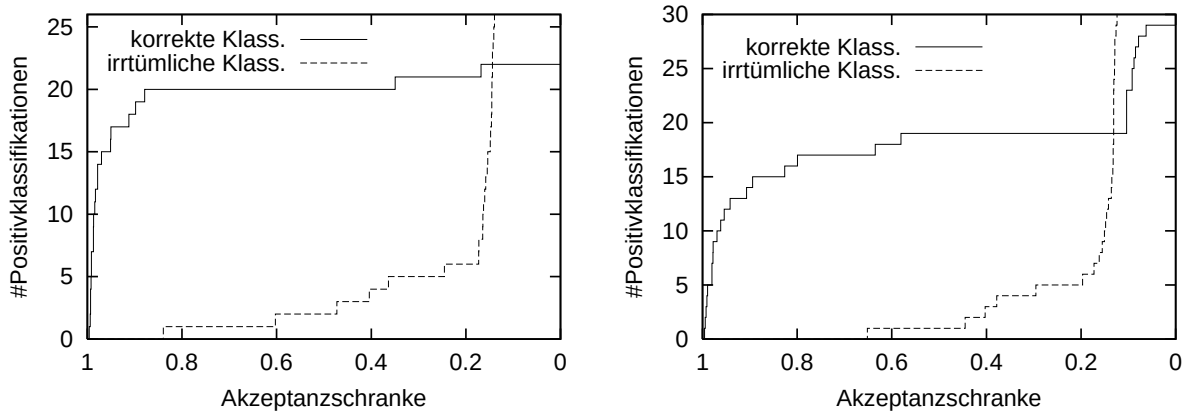


Abbildung 7.2: Erkennungsqualität für die Anfragen (1) und (2)

Die im linken Abbildungsteil dargestellten Ergebnisse zur Anfrage Nr.1 zeigen, dass das Filterverfahren sowohl effektiv als auch robust arbeitet. Mit einer sehr hohen Akzeptanzschranke von 0,87 werden bereits 20 Artikel korrekt erkannt ohne dass auch nur eine irrtümliche Positivklassifikation erfolgt. Selbst eine Akzeptanzschranke von 0,4 führt lediglich zu 4 irrtümlichen Positivklassifikationen.

Für die Anfrage Nr.2 tauchen einige Dokumente auf, die erst sehr spät erkannt werden (erst bei einer Akzeptanzschranke < 0.1). Eine genauere Analyse ergab, dass die betreffenden Dokumente (Internet-RFCs – *request for comments*) lediglich aus einem HTML-Rahmen bestanden, ansonsten aber als reine Textdatei gegeben waren. Somit ist es nicht verwunderlich, dass diese Dokumente nicht korrekt erkannt wurden, da die betreffende Transformationskomponente für HTML-Tokens keine verwertbaren Informationen aus diesen Daten ziehen konnte (vgl. mit Abschnitt 7.1). Um auch solche Dokumente korrekt klassifizieren zu können, müsste eine spezielle Transformationskomponente für Textdateien implementiert werden. Die anderen Dokumente wurden korrekt klassifiziert.

Abbildung 7.3 auf der nächsten Seite gibt die Ergebnisse der analogen Untersuchung für Anfrage Nr.3 wieder. Einmal mehr erweist sich das Filterverfahren als effektiv und robust. Irrtümliche Positivklassifikationen treten bis zu einer Akzeptanzschranke von 0,4 kaum auf.

Vergleich mit einer klassischen Suchmaschine

Neben der reinen Qualität der Filterfunktion ist natürlich auch der Nutzen, den ein Anwender aus einem Experten für wissenschaftliche Artikel ziehen kann, von großer Bedeutung. Der Nutzen eines Suchwerkzeuges wird üblicherweise durch sogenannte Präzision–

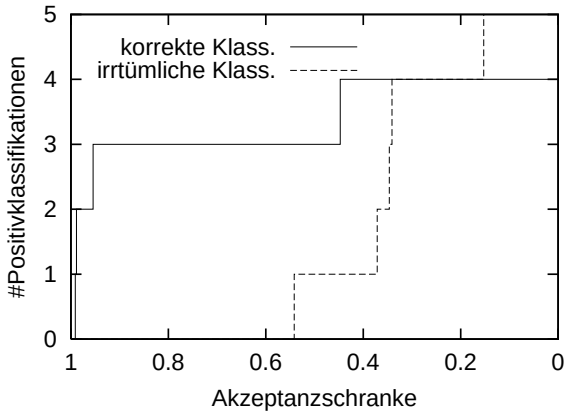


Abbildung 7.3: Erkennungsqualität für Anfrage (3)

Abruf–Diagramme dargestellt.[2] Für Anfragen, deren Ergebnisse als lineare, nach Relevanz geordnete Liste geliefert werden, beschreibt ein solches Diagramm, welche Präzision ein Benutzer in Kauf nehmen muss, um einen gewünschten Abruf bezüglich aller in der Ergebnisliste enthaltenen Dokumente zu erhalten. Mit anderen Worten, wie viele Dokumente der Benutzer anschauen muss, um $p\%$ der im Gesamtergebnis enthaltenen relevanten Dokumente zu finden.

Nachfolgend wird der Nutzen eines Artikel–Experten mit dem einer klassischen Suchmaschine verglichen. Benutzt ein Anwender eine solche Suchmaschine, so kann er in der Regel nur nach wichtigen inhaltlichen Stichwörtern (wie in den obigen Beispielanfragen) suchen. Eine direkte Suche nach Stichwörtern wie „scientific article“, „paper“ oder „publication“ ist nicht erfolgversprechend, da diese in aller Regel nicht explizit in solchen Dokumenten auftauchen. Ein cleverer Anwender mag immerhin auf die Idee kommen, seine Anfrage um die Stichwörter „abstract“ und „references“ zu erweitern, da diese typischerweise in einem Artikel vorkommen.

Abbildung 7.4 zeigt die Präzision–Abruf–Diagramme zu den Anfragen (1) und (3) für jeweils drei Konstellationen. Dies sind im Einzelnen die Anfrage an den Artikel–Experten sowie die *einfache* und die sogenannte *clevere Anfrage* an AltaVista, die sich von der einfachen Anfrage durch die Ergänzung um die Schlüsselwörter „abstract“ und „references“ unterscheidet. Man beachte, dass die Ergebnismenge für alle drei Konstellationen dieselbe

[2] Zur Erinnerung sei die Definition dieser beiden Begriffe nochmals kurz wiederholt: Die Präzision$^{\uparrow}$ ist definiert als Anteil der auf eine Anfrage gelieferten relevanten Dokumente gegenüber der Gesamtheit aller gelieferten Dokumente. Der Abruf$^{\uparrow}$ gibt den Anteil der auf eine Anfrage gelieferten relevanten Dokumente über allen (im jeweils betrachteten Dokumentenraum) existierenden relevanten Dokumenten wieder.

ist und sich diese lediglich in der Reihenfolge der einzelnen Ergebniseinträge unterscheiden. Beim Artikel–Experten ergibt sich diese Ordnung aus der Dokumentbewertung, für die Anfragen an AltaVista wird direkt eine (nach der durch AltaVista bestimmten Relevanz) geordnete Liste zurückgeliefert.

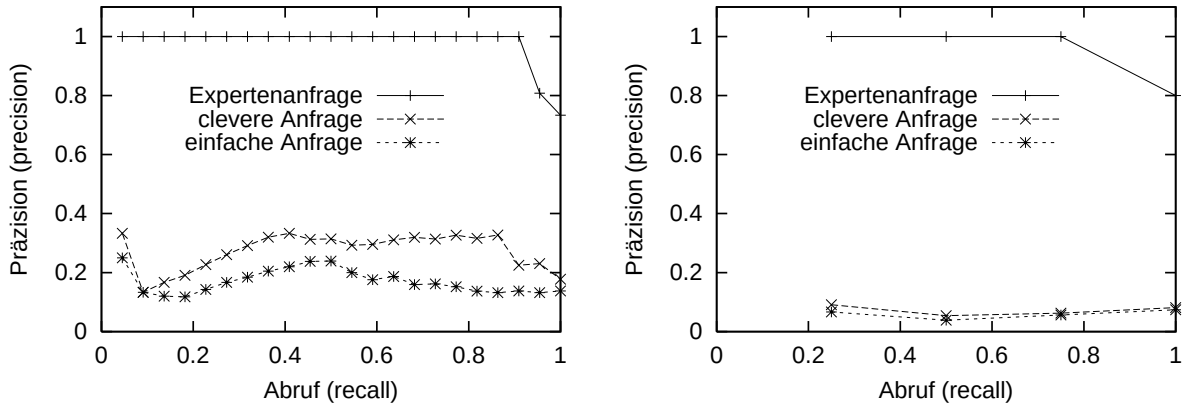


Abbildung 7.4: Präzision–Abruf–Diagramme für die Anfragen (1) und (3)

Man sieht für beide Anfragen, dass mit dem Artikel–Experten bei gleichem Abrufwert wesentlich höhere Präzisionswerte erreicht werden. Die clevere Anfrage zeigt sich der einfachen Anfrage leicht überlegen, ist aber trotzdem signifikant schlechter als der Experte.

Abbildung 7.5 auf der folgenden Seite zeigt die Resultate zur analogen Untersuchung für Anfrage Nr.2. Da bei dieser von AltaVista mehr als 10.000 Dokumente geliefert wurden (vgl. mit Tabelle 7.2 auf Seite 128) unterscheidet sich die Ergebnismenge für die cleveren Anfrage sowohl von der einfachen Anfrage als auch von der Expertenanfrage. Damit ist ein Vergleich dieser Ergebnismengen nicht möglich, weshalb in Abbildung 7.5 die Darstellung der cleveren Anfrage unterlassen wurde.

Für Abrufwerte bis 0,6 erreicht der Artikel–Experte wieder sehr gute Präzisionswerte. Danach gehen diese stark zurück, was sich entsprechend der Analyse des vorangehenden Abschnittes mit den nicht wirklich im HTML-Format vorliegenden Dokumenten erklärt.

7.3 Kostenreduktion durch mobile Filterprogramme

Abschließend wurde im Rahmen der Fallstudie noch untersucht, wie weit die Kommunikationskosten durch den Einsatz mobiler Filterprogramme bei konkreten Anfragen reduziert werden können.

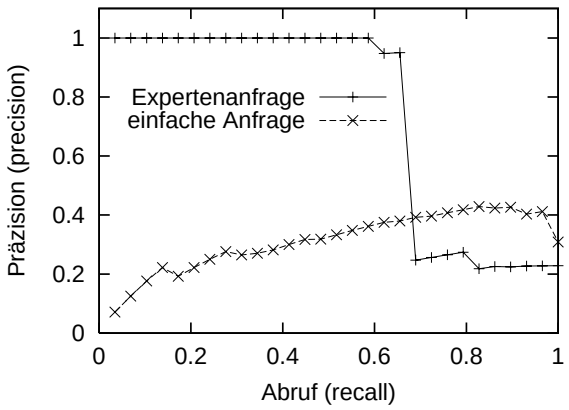


Abbildung 7.5: Präzision–Abruf–Diagramm für Anfrage (2)

Hierzu wurden die bereits im vorangehenden Abschnitt verwendeten Beispielanfragen (siehe Tabelle 7.1 und 7.2 auf Seite 128) herangezogen. Die Simulation der Aussendung des mobilen Filterprogrammes basiert auf folgenden Daten: Als Codeplattformen werden dieselben 119 Rechner wie in Kapitel 6 verwendet. Die Daten–Server ergeben sich direkt aus den Web–Servern, die Dokumente zu einer der drei Beispielanfragen bereitstellen. Sie sind eine Teilmenge der in Kapitel 6 benutzten Daten–Server. Netzwerkdistanzen zwischen Codeplattformen und zu Daten–Servern stammen ebenfalls aus der in Kapitel 6 beschriebenen Datenerhebung. Als Ausgangsplattform, auf der der Domänen–Experte ausgeführt wird und von dem die mobilen Filterprogramme ausgesandt werden, wird ein Rechner in Stuttgart (eine der Codeplattformen) benutzt. Der Umfang der Filterfunktion beträgt 16 KByte. Die für die Aussendungsverfahren verwendete Schätzung des zu untersuchenden Datenvolumens (bzw. der Dokumentgrößen) kann direkt aus den von AltaVista gelieferten Ergebnisseiten entnommen werden. Es wird angenommen, dass 10% der untersuchten Daten relevant sind, das geschätzte Ergebnisvolumen beträgt also jeweils ein Zehntel der Schätzung für das zu untersuchende Datenvolumen.

Bei der Analyse der erreichten Kostenreduktion wird natürlich auf die realen Datenvolumina zurückgegriffen. Das Ergebnisvolumen beträgt pro Dokument 2 Byte (für die Bewertungszahl durch die Filterfunktion) plus den Dokumentumfang, für die relevanten Dokumente. Es wird angenommen, dass das Filterprogramm exakt die relevanten Dokumente erkennt.

Tabelle 7.3 fasst die mit den in Kapitel 4 vorgestellten Aussendungsverfahren erzielbaren Kostenreduktionen für die drei Beispielanfragen zusammen. Wie in Kapitel 6 (Gleichung 6.1 auf Seite 113) gibt die Kostenreduktion an, um welchen Faktor die Kosten der klas-

sischen Klient–Server–Lösung mit Hilfe der Aussendungspläne reduziert werden. Beim hierarchischen Aussendungsverfahren wurde ein mit dem Kriterium *Min-k-Ausdehnung* berechneter Regionenbaum von Grad $k = 10$ zugrundegelegt. Zu jeder Anfrage werden die Selektivität, das ist der Prozentsatz der relevanten Dokumenten über allen untersuchten Dokumenten, und die Kostenreduktion für die beiden Distanzmaße Knotenzahl und Umlaufzeit sowie für den Basisalgorithmus und den hierarchischen Algorithmus wiedergegeben.

Anfragenr.	Selektivität	Kostenreduktion (Knotenzahl)		Kostenreduktion (Umlaufzeit)	
		Basisalg.	Hier.Alg.	Basisalg.	Hier.Alg.
(1)	14%	56%	52%	85%	74%
(2)	23%	48%	40%	78%	65%
(3)	6%	62%	57%	88%	85%

Tabelle 7.3: Reduktion der Kommunikationskosten bei drei Beispielanfragen

Es wird in allen Konstellationen eine signifikante Reduktion der Kommunikationskosten erreicht. Wie bereits bei den Experimenten in Kapitel 6 ist die Reduktion beim Distanzmaß Umlaufzeit und beim Basisalgorithmus jeweils höher als beim Distanzmaß Knotenzahl und dem hierarchischen Algorithmus.

7.4 Diskussion

Die Experimente zeigen, dass selbst mit einem sehr einfachen und naiven Algorithmus eine effektive und robuste Erkennung von wissenschaftlichen Artikeln möglich ist. Im Vergleich zur Benutzung einer klassischen Suchmaschine konnte der hohe Nutzen eines spezifischen Suchwerkzeuges nachgewiesen werden. Dabei wurden noch nicht einmal die spezifischeren Anfragemöglichkeiten berücksichtigt, die ein Artikel–Experte bietet. So erlaubt er z.B. die explizite Suche nach Artikeln bestimmter Autoren oder auch nach referenzierten Artikeln.

Wie bereits in Kapitel 2.1.5 angesprochen ist der zur Erstellung eines konkreten Domänen–Experten erforderliche Aufwand von großer Bedeutung für die praktische Umsetzbarkeit dieses Ansatzes. Der Quellcode des in der Fallstudie entwickelten Filterprogrammes umfasst 1000 Zeilen. Es ist zu erwarten, dass mit der Verwendung von Standardkomponenten

hier noch wesentliche Verbesserungen erreicht werden können. Für die meisten Domänen wird aber sicherlich jeweils ein gewisser Codeanteil neu erstellt werden müssen.

Nichtzuletzt konnte auch für die Beispiele aus der hier betrachteten Fallstudie gezeigt werden, dass mit dem Einsatz mobiler Programme die bei der Analyse der verteilt vorliegenden Dokumente entstehenden Kommunikationskosten erheblich reduziert werden können.

Eine generelle Schwäche an der durchgeführten Fallstudie besteht sicherlich in der sehr begrenzten Zahl an untersuchten Beispielanfragen und Dokumenten. Diese lässt eine Verallgemeinerung der erzielten Ergebnisse nur sehr eingeschränkt zu. Um die Aussagekraft der Ergebnisse zu verbessern, wären daher umfangreichere Versuche erforderlich. Allerdings bedeuten diese auch einen immensen Aufwand, da alle untersuchten Dokumente auch von Hand klassifiziert werden müssen.

Kapitel 8

Resümee

8.1 Zusammenfassung

In dieser Arbeit wurde mit dem Konzept der Domänen–Experten ein neuer Ansatz für Suchwerkzeuge vorgestellt, die auch in extrem großen Informationsräumen wie dem Internet eine präzise und umfassende Suche ermöglichen.

Der Kernidee des Ansatzes besteht in der Spezialisierung von Suchmaschinen auf einzelne Themengebiete, womit die Zahl der für eine solche Suchmaschine relevanten Dokumente erheblich reduziert und damit zu einer handhabbaren Größe wird. Außerdem kann durch die Verwendung von spezifischen Such- und Filtermöglichkeiten eine wesentlich höhere Präzision als bei herkömmlichen Suchmaschinen erreicht werden.

Der größte Nachteil dieses Ansatzes besteht in dem zur Erstellung eines Domänen–Experten erforderlichen Aufwand, der individuell für jeden neuen Experten zu erbringen ist. Hierzu gehören die spezifische Definition einer Dokument–Wissensbasis, die Festlegung von Anfrage- und Ergebnismasken und insbesondere die Realisierung einer Filterfunktion, mit welcher über die Relevanz von beliebigen Dokumenten entschieden werden kann. Mit Hilfe eines konfigurierbaren Frameworks konnte dieser Aufwand weitgehend auf die Erstellung der Filterfunktion reduziert werden. Alle anderen spezifischen Aspekte lassen sich anhand einer einfachen Konfigurationsdatei einstellen. Da es zudem im Gebiet der Inhaltsfilterung umfangreiche Forschungsarbeiten und zahlreiche erprobte Verfahren gibt, lässt sich erwarten, dass auch die Erstellung von spezifischen Filterfunktionen anhand von Standardkomponenten zur Inhaltsfilterung recht einfach gestaltet werden kann. Dieser Aspekt war jedoch nicht Untersuchungsgegenstand der vorliegenden Arbeit.

Um die für einen Domänen–Experten relevanten aber verteilt vorliegenden Dokumente effizient zu lokalisieren, wurde die Technologie der mobilen Programme eingesetzt. Hiermit kann die Filterfunktion zu potenziell relevanten Datenquellen gesandt werden, diese untersuchen und schließlich die relevanten Dokumente an den Domänen–Experten zurücksenden. Zur Aussendung der mobilen Filterprogramme wurden spezielle Verfahren entwickelt, die die Aussendung derart optimieren, dass die anfallenden Kommunikationskosten minimiert werden. Obwohl das Problem der optimalen Aussendung NP-hart ist, konnten skalierbare und effiziente Verfahren zur Aussendungskoordination entwickelt werden, die zwar keine feste Approximationsgüte garantieren können, sich in Experimenten aber trotzdem als sehr effektiv erwiesen haben. Auf Basis umfangreicher Netzwerkmessungen wurde gezeigt, dass der Einsatz mobiler Filterprogramme in vielen Situationen zu einer Reduktion der Kommunikationskosten um bis zu 90% gegenüber einem klassischen Klient–Server–Ansatz führt, bei dem alle zu untersuchenden Dokumente zum Domänen–Experten übertragen werden müssen.

Da zur Koordination der mobilen Filterprogramme die Kenntnis der jeweils relevanten Netzwerkdistanzen erforderlich ist, wurde schließlich mit den sogenannten Netzwerk–Distanzkarten ein Ansatz entwickelt, der die Schätzung beliebiger Netzwerkdistanzen im Internet auf skalierbare Art und Weise ermöglicht. Mit Hilfe einiger Mess–Server, die Distanzmessungen zu beliebigen Endsystemen ermöglichen, erfolgt hierzu eine hierarchische Einteilung des Internets in Regionen und Teilregionen, in die sich beliebige Endsysteme effizient einordnen lassen. Der Ansatz wurde anhand von umfangreichen Netzwerkmessungen evaluiert. Hierbei konnte nachgewiesen werden, dass die hierarchische Einteilung in Regionen und Teilregionen sowie die Zuordnung von Endsystemen sehr gut funktioniert. Die abschließende Distanzschätzung erreicht jedoch nur eine recht grobe Genauigkeit, sie kann daher im Wesentlichen nur zur Schätzung der Größenordnung von Netzwerkdistanzen genutzt werden. Interessanterweise zeigen die Experimente zur Aussendung mobiler Filterprogramme jedoch, dass bereits diese sehr grobe Schätzung äußerst nutzbringend zur Optimierung der Netzwerkkommunikation von Anwendungen eingesetzt werden kann.

Abschließend wurde das Konzept der Domänen–Experten im Rahmen einer Fallstudie zur Domäne der „wissenschaftlichen Artikel“ evaluiert, wobei insbesondere die entwickelten Verfahren zur Aussendung mobiler Filterprogramme und zur Schätzung von Netzwerkdistanzen zur Anwendung kamen. Hierbei wurde zum einen gezeigt, dass mit recht einfachen Mitteln eine effektive und nützliche Filterfunktion erstellt werden kann. Zum anderen wurde nachgewiesen, dass die Aussendung mobiler Filterprogramme auf Basis von durch Netzwerk–Distanzkarten geschätzten Netzwerkdistanzen zu einer erheblichen Reduktion

der Kommunikationskosten führt und damit signifikant zu einer effizienten Lokalisierung relevanter Dokumente beiträgt.

8.2 Bewertung und Ausblick

Domänen–Experten können in vielen Bereichen ein nützliches Werkzeug zur präzisen und umfassenden Informationssuche sein. Trotzdem werden sie herkömmliche Werkzeuge zur Informationssuche im Internet (i.W. herkömmliche Suchmaschinen und Kataloge) sicherlich nicht vollständig verdrängen. Suchmaschinen bieten aufgrund ihrer breiten Abdeckung in vielen Fällen den einfachsten und schnellsten Weg, um an eine Information zu kommen. Kataloge erzielen aufgrund der redaktionellen Einordnung eine auf technischem Wege nicht erreichbare Qualität. Ein wichtiger Forschungsaspekt für die Zukunft ist daher die Integration dieser Suchwerkzeuge, so dass für einen Benutzer ein einheitlicher Zugang zu ihnen möglich wird. Hierzu sind unter den Schlagwörtern der Metasuchmaschinen und der Wissensmakler bereits einige Aspekte erforscht worden, eine allgemeine und praxistaugliche Lösung wurde bisher allerdings nicht erreicht.

Die Technologie der mobilen Programme hat sich im Kontext der Filterung verteilter Informationen als sehr nützlich erwiesen, um Kommunikationskosten zu reduzieren. Die entwickelten Verfahren sind längst nicht nur im Kontext von Domänen–Experten anwendbar, sondern auch in beliebigen Filterszenarien und sogar in allgemeinen Situationen einer 1:n–Kommunikation. Mit ähnlichen Optimierungsverfahren sollte sich in Zukunft der Nutzen mobiler Programme auch für andere Interaktionsmuster nachweisen lassen.

Möglich wird der effiziente Einsatz mobiler Programme allerdings erst mit Hilfe der durch Netzwerk–Distanzkarten erreichten Schätzung beliebiger Netzwerkdistanzen. Abermals sind diese Karten nicht nur zur Optimierung des Einsatzes mobiler Programme nützlich, sondern auch für andere Anwendungen, die ihre Netzwerkkommunikation optimieren wollen. Obwohl die Distanzschätzung im Bereich der Aussendung mobiler Programme bereits zu guten Ergebnissen führt, ist die erzielte Qualität im Allgemeinen noch nicht befriedigend. Um hier systematische Fehler genauer identifizieren zu können ist eine Evaluierung mit einer weiterführenden Datenerhebung unabdingbar. Schließlich ist auch die Effizienz der Verfahren zur Aktualisierung von Netzwerk–Distanzkarten noch verbesserungsbedürftig. Eine Evaluation mit verteilten Mess–Servern, die permanente Messungen erlauben, wäre sehr wünschenswert.

Anhang A

Glossar und Abkürzungsverzeichnis

Abruf (engl. recall)
Ein Maß zur Beschreibung der Effektivität von IR–Systemen: der Anteil der auf eine Anfrage gelieferten relevanten Dokumente über allen (im jeweils betrachteten Dokumentenraum) existierenden relevanten Dokumenten.

Approximationsfaktor
Bei einem Optimierungsproblem beschreibt der Approximationsfaktor, um welchen Faktor die Güte der mit einem Approximationsverfahren bestimmten Lösung maximal von der Güte der optimalen Lösung abweicht. Gütemaß ist dabei der zu optimierende (maximierende oder minimierende) Ausdruck.

Aussendungsplan
Beschreibt eine Möglichkeit, wie eine gegebene Menge von Daten–Servern$^{\uparrow}$ durch ein mobiles Filterprogramm untersucht werden kann. Legt im Einzelnen eine Folge von Programmtransfers und -replikationen fest, durch die das mobile Programm$^{\uparrow}$ auf eine Menge von Codeplattformen$^{\uparrow}$ gesandt wird und von diesen aus die Daten–Server untersucht.

Autonomes System
Ein Teilnetz vom Internet, das als administrative Einheit betrachtet werden kann. Ein autonomes System wird von einer Institution (Universität, Firma, etc.) verwaltet.

Ballung (engl. clustering bzw. cluster)
Allgemeine Bezeichnung für eine Klasse von Algorithmen, die Objekte nach Ähnlichkeits- oder Nähekriterien in Gruppen anordnen. Sowohl der Prozess als auch eine hierdurch entstehende Gruppe wird als Ballung bezeichnet. Im Zusammenhang von Netzwerk–Distanzkarten werden Ballungsverfahren benutzt um Regionen$^{\uparrow}$ zu bestimmen.

BGP – Border Gateway Protocol
Protokoll, das im Internet zum Routing (zu deutsch Leitwegsbestimmung) zwischen autonomen Systemen↑ benutzt wird.

Codeplattform
Ein Rechner im Internet, der eine Plattform zur Aufnahme und Ausführung mobiler Programme bietet.

DNS – Domain Name System
Im Internet benutztes Rechner–Adressierungsschema, zu dem gleichzeitig ein Dienst angeboten wird, der beliebige Namen auf ihre IP–Adresse und umgekehrt abbildet.

Daten–Server
Ein Endsystem↑, das Daten bzw. Dokumente öffentlich, d.h. über das Internet, zur Verfügung stellt.

Endsystem (engl. host)
Ein Rechner im Internet, der im Gegensatz zu einem Router↑ keinerlei Netzwerk–Aufgaben erfüllt, sondern einfach als Endteilnehmer an das Netzwerk angeschlossen ist. Im Zusammenhang mit Netzwerk–Distanz–Karten bezeichnet ein *einfaches Endsystem* ein solches, das kein Mess–Server↑ ist.

FTP – File Transfer Protocol
Kommunikationsprotokoll im Internet zur Übertragung von Dateien. Bezeichnet gleichzeitig den damit verbundenen Dienst, mit dem auf sogenannten FTP–Servern Dateien im Internet „veröffentlicht“ werden können.

Greedy–Algorithmen
Algorithmen, die eine Lösungsstrategie benutzen, bei der die Lösung durch schrittweise Erweiterung eines Initialzustands gefunden wird.

HTML – HyperText Markup Language
Die im Web↑ vorrangig benutzte Dokumentenbeschreibungssprache, die u.a. Querverweise auf beliebige andere Dokumente im Web ermöglicht.

HTTP – HyperText Transfer Protocol
Das im Web↑ benutzte Transportprotokoll zur Übermittlung von HTML–Dokumenten↑.

ICMP – Internet Control Message Protocol
Protokoll im Internet, über das Fehlermeldungen und andere Kontrollnachrichten versandt werden.

IF – Information Filtering
Forschungsgebiet, das sich mit dem Herausfiltern relevanter Daten aus einem Strom von Daten befasst. Im Gegensatz zum *Information Retrieval*↑ wird der Informationsraum als eher dynamisch und das Informationsbedürfnis als statisch angesehen.

Index
Grundlegende Datenstruktur in vielen Suchwerkzeugen: Wissen über Dokumente wird wortorientiert abgelegt, d.h. zu jedem Wort wird gespeichert, wie oft und evtl. an welchen Positionen es in welchen Dokumenten vorkommt.

IP – Internet Protocol
Das Netzwerkschicht–Protokoll des Internets. Regelt u.a. das Format von Datagrammen und Rechneradressen.

IR – Information Retrieval
Forschungsgebiet, das sich mit dem Wiederauffinden von Daten bzw. Informationen in großen Datenbeständen bzw. Informationsräumen befasst. Im Gegensatz zum *Information Filtering*↑ wird der Informationsraum als eher statisch und das Informationsbedürfnis als dynamisch angesehen. Eine etwas vereinfachende Übersetzung von IR ist „Informationssuche“.

Knotenzahl (engl. hop count)
Die Zahl der von einem Datagramm durchlaufenen Router↑ auf dem Weg zu einem entfernten Endsystem↑.

Kommunikationskosten
Bezeichnen den mit einem Datentransfer verbundenen Aufwand bzw. die damit entstehende Belastung des Netzwerkes. Kommunikationskosten können unterschiedlich definiert werden. In dieser Arbeit werden sie als Produkt aus Netzwerkdistanz↑ und dem über diese Distanz zu übertragenden Datenvolumen definiert.

Mess–Koordinator
Ein Rechner im Internet, der im Zusammenhang mit Netzwerk–Distanz–Karten die Koordination von Distanzmessungen durch Mess–Server↑ sowie die Berechnung von Teilen der Netzwerk–Distanzkarte erlaubt.

Mess–Server
Ein Rechner im Internet, der im Zusammenhang mit Netzwerk–Distanz–Karten die Distanzmessung zu beliebigen Endsystemen↑ erlaubt.

Migration
Transfermechanismus für Instanzen mobiler Programme↑, der einen Transfer zur Laufzeit

erlaubt. Man unterscheidet zwischen der starken Migration, bei der durch den Transfer des kompletten Ausführungszustandes ein für den Programmierer vollkommen transparenter Transfer erreicht wird, und der schwachen Migration, bei der eine feste Einsprungsprozedur spezifiziert wird und nur globale Variablen übertragen werden.

Mobiles Programm
Bezeichnung einer Ausführungseinheit (Programm, Objekt), die aus der Initiative eines Senders heraus auf eine entfernte Codeplattform$^{\uparrow}$ übertragen und dort mit beliebigen Aufrufparametern gestartet bzw. fortgesetzt werden kann.

Multicast
Eine Technologie zur effizienten Versendung von Nachrichten an eine Gruppe aus mehreren Endsystemen. Eine der ersten Arbeiten zur Realisierung von Multicast–Kommunikation im Internet findet sich in [Deering & Cheriton, 1985].

Netzwerkbaum
Von einer Netzwerk–Distanzkarte benutzte Datenstruktur, die zusätzlich zum Regionenbaum$^{\uparrow}$ alle Endsysteme$^{\uparrow}$ und deren jeweilige Zuordnung zu einem möglichst nahe gelegenen Mess–Server umfasst.

Netzwerkdistanz
Beschreibt die Qualität einer Netzwerkverbindung zwischen zwei Endsystemen. Kann je nach Kontext unterschiedlich definiert sein, z.B. als Knotenzahl$^{\uparrow}$ oder als Umlaufzeit$^{\uparrow}$.

NP – Nichtdeterministisch Polynomial
Die Klasse NP bezeichnet alle Probleme, die durch einen nichtdeterministisch polynomialen Algorithmus berechenbar sind. Ein Problem p (nicht notwendig aus NP) heißt NP–hart, wenn alle Probleme $q \in NP$ höchstens so schwer wie p sind (q polynomial auf p reduzierbar ist). Ein Problem p heißt NP–vollständig, falls p NP–hart ist und $p \in NP$ gilt.

OSPF – Open Shortest Path First
Ein Protokoll, das für das Routing (deutsch: Leitwegsbestimmung) innerhalb eines autonomen Systems$^{\uparrow}$ eingesetzt wird.

Präzision (engl. precision)
Ein Maß zur Beschreibung der Effektivität von IR–Systemen: der Anteil der auf eine Anfrage gelieferten relevanten Dokumente gegenüber der Gesamtheit aller gelieferten Dokumente.

Remote Execution (Entfernte Ausführung)
Transfermechanismus für mobile Programme$^{\uparrow}$, mit dem ein Programm zusammen mit

einigen Aufrufparametern auf einen entfernten Rechner übertragen und dort gestartet werden kann.

Region
Eine Menge von Endsystemen, die untereinander enger, d.h. über eine geringere Distanz, verbunden sind als mit Endsystemen außerhalb dieser Region. Regionen werden mit Hilfe von Ballungsverfahren berechnet.

Regionenbaum
Von einer Netzwerk–Distanzkarte benutzte Datenstruktur, die alle Mess–Server$^{\uparrow}$ und Regionen$^{\uparrow}$ umfasst.

Roboter
Komponente einer Suchmaschine$^{\uparrow}$, die für das automatische Aufspüren und Erfassen von Dokumenten verantwortlich ist.

Router (Vermittlungsknoten)
Innerer Knoten eines Netzwerkes, der primär Vermittlungsaufgaben für die Kommunikation zwischen Endsystemen$^{\uparrow}$ übernimmt.

RTT – Round Trip Time
Siehe Umlaufzeit$^{\uparrow}$.

Suchwerkzeug
Allgemeinste Bezeichnung für einen System zur Informationssuche (Synonym zu IR–System).

Suchmaschine
Ein Suchwerkzeug, das sein Wissen vollständig automatisch erwirbt (im Gegensatz zu Katalogen).

Umlaufzeit (engl. round trip time – rtt)
Die erforderliche Zeit zur Übertragung eines Datagramms auf einen entfernten Rechner und zur Rückübertragung einer Bestätigung.

Unicast
Senden einer Nachricht an ein Endsystem$^{\uparrow}$.

TCP – Transmission Control Protocol
Im Internet benutztes zuverlässiges Unicast–Transport–Protokoll.

UDP – User Datagram Protocol
Im Internet benutztes unzuverlässiges Unicast–Transport–Protokoll.

URL – Uniform Resource Locator
Im Internet benutztes Schema zur global eindeutigen Bezeichnung von Ressourcen.

Usenet–News
Informationsdienst im Internet, der Diskussionsforen zu hierarchisch angelegten Themengruppen anbietet.

Web – World Wide Web
Informationsdienst im Internet, der das Publizieren von Dokumenten und das Navigieren durch diesen Dokumenten–Raum ermöglicht. In seiner momentanen Ausprägung ist das Web im Wesentlichen durch die Seitenbeschreibungssprache HTML$^{\uparrow}$ sowie das Übertragungsprotokoll HTTP$^{\uparrow}$ gekennzeichnet.

Anhang B

Mathematische Bezeichner

Bezeichner	Definition (Abschnitt der Einführung des Bezeichners)
$\mathcal{C}$	Menge der verfügbaren Codeplattformen im Netzwerk (4)
$\mathcal{D}$	Menge der existierenden Daten–Server im Netzwerk (4)
$\mathcal{E}$	Menge der existierenden Endsysteme im Netzwerk (3.3.1)
$\mathcal{M}$	Menge der verfügbaren Mess–Server im Netzwerk (3.3.1)
$\Delta(*,*)$	Reale (gemessene) Netzwerkdistanz zwischen zwei Endsystemen im Netzwerk (3.3.1)
$\|*,*\|$	Durch eine Netzwerk–Distanzkarte geschätzte Distanz zwischen zwei Endsystemen oder Regionen (3.3.4)
Λ	Funktion, die zu einem Knoten des Netzwerkbaumes, den übergeordneten Elternknoten liefert (3.3.4)
δ	Funktion, die zu einem Endsystem die Distanz zum momentan im Netzwerkbaum zugeordneten Mess–Server liefert (3.4.3)
k	Grad eines Regionenbaumes, d.h. maximale Kindknotenzahl eines Baumknotens (3.3.4)
t	Tiefe eines Regionenbaumes (3.3.4); die Tiefe der Wurzel ist als 0 definiert
m	Zahl der in einem Aussendungsplan vorgesehenen Transfers eines mobilen Filterprogrammes (4.4)
$D1, D2$	Erste/zweite zur Evaluierung von Netzwerk–Distanzkarten durchgeführte Datenerhebung (5.1)
σ	Allgemeiner Bezeichner für die Standardabweichung

Literaturverzeichnis

Acharya, A.; Saltz, J. (1996). "A study of internet round-trip delay". Technical Report CS-TR-3736, Department of Computer Science, University of Maryland, USA

Ardö, A.; Lundberg, S. (1998). "A regional distributed WWW search and indexing service - the desire way". In *Online Proc. 7th Int. World Wide Web Conf. (WWW7)*. URL: http://www7.conf.au/programme/fullpapers/1900/com1900.htm

Arnold, S. (1999). "Konzeption und Implementierung eines Frameworks für spezialisierte Suchmaschinen". Diplomarbeit Nr.1762, Fakultät Informatik, Universität Stuttgart, Germany

Baldi, M.; Picco, G. P. (1998). "Evaluating the tradeoffs of mobile code design paradigms in network management applications". In *Proc. 2nd Int. Conf. on Software Engineering (ICSE'98)*, pages 146–155. IEEE Computer Society Press

Bestavros, A. (1997). "WWW traffic reduction and load balancing through server-based caching". *IEEE Concurrency, Special Issue on Parallel and Distributed Technology*, 5(1):56–67

Bollacker, K. D.; Lawrence, S.; Giles, C. L. (1998). "CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications". In *Proc. 2nd Int. Conf. on Autonomous Agents*, pages 116–123. ACM Press

Bowman, C. M.; Danzig, P. B.; Hardy, D. R.; Manber, U.; Schwartz, M. F.; Wessels, D. P. (1995). "Harvest: A scalable, customizable discovery and access system". Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, Boulder, USA

Bowman, C. M.; Danzig, P. B.; Manber, U.; Schwartz, M. F. (1994). "Scalable internet resource discovery: Research problems and approaches". *Communications of the ACM*, 37(8):98–107

Carter, R. L.; Crovella, M. E. (1996). "Dynamic server selection using bandwidth probing in wide-area networks". In *Proc. 16th Conf. on Computer Communications (IEEE Infocom'97)*, pages 1014–1021. IEEE Computer Society Press. extended version as technical report BU-CS-96-007, Boston University, March 1996

Chakrabarti, S.; van den Berg, M.; Dom, B. (1999). "Focused crawling: A new approach to topic-specific web resource discovery". In *Online Proc. 8th Int. World Wide Web Conf. (WWW8'99)*. URL: http://www8.org/fullpaper.html

Chankhunthod, A.; Danzig, P.; Neerdaels, C.; Schwartz, M. F.; Worrell, K. J. (1996). "A hierarchical Internet object cache". In *Proc. USENIX Annual Technical Conference (USENIX'96)*, pages 153–163. USENIX Association.
URL: http://www.usenix.org/publications/library/proceedings/sd96/

Chess, D.; Harrison, C.; Kershenbaum, A. (1995). "Mobile agents: Are they a good idea?". IBM Research Report RC 19887 (88456)

Dasgupta, P.; Narasimhan, N.; Moser, L. E.; Melliar-Smith, P. (1999). "MAgNET: Mobile agents for networked electronic trading". *IEEE Transactions on Knowledge and Data Engineering*, 11(4):509–525

Deering, S.; Cheriton, D. (1985). "Host groups: A multicast extension to the internet protocol". Internet Engineering Task Force, Network Working Group, Request for Comments 966, URL: ftp://www.it.kth.se/docs/rfc/rfcs/rfc966.txt

Dolin, R.; Agrawal, D.; Abbadi, A. E.; Dillon, L. K. (1997). "Pharos: A scalable distributed architecture for locating heterogeneous information sources". In *Proc. 6th ACM Int. Conference on Information and Knowledge Management*, pages 348–355. ACM Press

Dolin, R.; Agrawal, D.; Abbadi, A. E.; Pearlman, J. (1998). "Using automated classification for summarizing and selecting heterogeneous information sources". *D-Lib Magazine*. ISSN 1082-9873

Doorenbos, R. B.; Etzioni, O.; Weld, D. S. (1997). "A scalable comparison-shopping agent for the world-wide web". In *Proc. Conf. on Autonomous Agents'97*, pages 39–48. ACM Press

Dreilinger, D.; Howe, A. E. (1997). "Experiences with selecting search engines using metasearch". *ACM Transactions on Information Systems*, 15(3):195–222

Francis, P. (1997). "A call for an Internet-wide host proximity service (HOPS)". White paper, URL: http://www.ingrid.org/hops/wp.html

Francis, P.; Jamin, S.; Paxon, V.; Zhang, L.; Gryniewicz, D. F.; Jin, Y. (1999). "An architecture for a global internet host distance estimation service". In *Proc. 18th Conf. on Computer Communications (IEEE Infocom'99)*. IEEE Computer Society Press. URL: http://www.ieee-infocom.org/1999/papers/02b_01.pdf

Francis, P.; Kambayashi, T.; Sato, S.-y.; Shimizu, S. (1995). "Ingrid: A self-configuring navigation infrastructure". In *Online Proc. 4th Int. World Wide Web Conference*. URL: http://www.w3.org/Conferences/WWW4/Program_Full.html

Frieze, A.; Jerrum, M. (1995). "Improved approximation algorithms for MAX k-CUT and MAX BISECTION". In *Proc. 4th Int. Conf. on Integer Programming and Combinatorial Optimization*, LNCS 920, pages 1–13. Springer–Verlag

Gardner, G. (1998). "Understanding traceroute". GeoNet Communications,Redwood City (CA), USA, white paper, URL: http://www.geo.net/support/traceroute.html

González, T. F. (1985). "Clustering to minimize the maximum intercluster distance". *Theoretical Computer Science*, 38:293–306. North-Holland

Guttmann-Beck, N.; Hassin, R. (1998). "Approximation algorithms for min-sum p-clustering". *Discrete Applied Mathematics*, 89(1-3):125–142. Elsevier

Guyton, J. D.; Schwarz, M. F. (1995). "Locating nearby copies of replicated Internet servers". In *Proc. ACM SIGCOMM'95*, volume 25:4 of *Computer Communication Review*. ACM SIGCOMM.
URL: http://www.acm.org/sigcomm/ccr/archive/1995/conf/guyton.ps

Gwertzman, J.; Seltzer, M. (1995). "The case for geographical push-cashing". In *Proc. 5th Conf. on Hot Topics in Operating Systems (HotOS'95)*, pages 51–55. IEEE Computer Society Press

Gwertzman, J.; Seltzer, M. (1996). "An analysis of geographical push-cashing". URL: http://www.eecs.harvard.edu/~vino/web/server.cache/icdcs.ps

Han, Y.; Loke, S. W.; Sterling, L. (1996). "Agents for citation finding on the World Wide Web". Technical Report 96/40, Department of Computer Science, University of Melbourne, Australia

Jacobsen, V. (1988). "Traceroute". URL: ftp://ftp.ee.lbl.gov/pub/traceroute.tar.Z

Johansen, D. (1998). "Mobile agent applicability". In *Proc. 2nd Int. Workshop on Mobile Agents (MA'98)*, LNCS 1477, pages 80–88. Springer–Verlag

Karp, R. M. (1972). "Reducibility among combinatorial problems". *Complexity of computer communications*, pages 85–103. Plenum Press

Kato, K.; Someya, Y.; Matsubara, K.; Toumura, K.; Abe, H. (1999). "An approach to mobile software robots for the WWW". *IEEE Transactions on Knowledge and Data Engineering*, 11(4):526–548

KAW (1999). *Online Proc. 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW'99).*
URL: http://sern.ucalgary.ca/KSI/KAW/KAW99/papers.html

Kosmynin, A. (1997). "From bookmark managers to distributed indexing: An evolutionary way to the next generation of search engines". *IEEE Communications Magazine*, pages 146–151

Lagoze, C.; Fielding, D.; Payette, S. (1998). "Making global digital libraries work: Collection services, connectivity regions, and collection views". In *Proc. 3rd ACM Conf. on Digital Libraries (DL'98)*, pages 134–143. ACM Press

Lawrence, S.; Bollacker, K.; Giles, C. L. (1999). "Indexing and retrieval of scientific literature". In *Proc. 8th Int. Conf. on Information and Knowledge Management (CIKM'99)*, pages 139–146. IEEE Computer Society Press

Lawrence, S.; Giles, C. L. (1998). "Searching the World Wide Web". *Science*, 280(5360):98–100

Lawrence, S.; Giles, C. L. (1999). "Accessibility of information on the web". *Nature*, 400(8):107–109

Levine, B. N.; Garcia-Luna-Aceves, J. J. (1996). "A comparison of known classes of reliable multicast protocols". In *Proc. IEEE Int. Conf. on Network Protocols (ICNP'96)*, pages 112–121. IEEE Computer Society Press

Manber, U.; Bigot, P. A. (1997). "The search broker". In *1st Usenix Symp. on Internet Technologies and Systems*. USENIX Association.
URL: http://www.usenix.org/publications/library/proceedings/usits97/full_papers/manber_search/manber_search.pdf

Miller, R. C.; Bharat, K. (1998). "SPHINX: A framework for creating personal, site-specific Web crawlers". In *Online Proc. 7th Int. World Wide Web Conference (WWW7'98)*.
URL: http://www.cs.cmu.edu/afs/cs.cmu.edu/user/rcm/WWW/www7/www7.html

Milojičić, D.; Douglis, F.; Wheeler, R., editors (1999). *Mobility – Processes, Computers, and Agents*. Addison–Wesley

Mladenic, D. (1999). "Text-learning and related intelligent agents: A survey". *IEEE Intelligent Systems & their applications*, 14(4):44–54

Monge, A. E.; Elkan, C. (1996). "The WebFind tool for finding scientific papers over the worldwide web". In *Proc. of Int. Congress on Computer Science Research*. URL: http://www-cse.ucsd.edu/users/amonge/Papers/ciicc96.ps

Moore, K. (1998). "SONAR - A network proximity service". Internet-Draft,
URL: ftp://ftp.isi.edu/internet-drafts/draft-moore-sonar-03.txt

Moy, J. T. (1997). "OSPF Version 2". Internet Engineering Task Force, Network Working Group, Request for Comments 2178,
URL: http://www.it.kth.se/docs/rfc/rfcs/rfc2178.txt

Rabinovich, M.; Rabinovich, I.; Rajaraman, R.; Aggarwal, A. (1999). "A dynamic object replication and migration protocol for an internet hosting service". In *Proc. 19th IEEE Int. Conf. on Distributed Computing Systems*, pages 101–113. IEEE Computer Society Press

Ranganathan, M.; Acharya, A.; Sharma, S.; Saltz, J. (1997). "Network-aware mobile programs". In *Proc. USENIX 1997 Ann. Technical Conf.* USENIX Association.
URL: http://www.usenix.org/publications/library/proceedings/ana97/full_papers/ranganathan/ranganathan.ps

Rekhter, Y.; Li, T. (1995). "A border gateway protocol 4 (BGP-4)". Internet Engineering Task Force, Network Working Group, Request for Comments 1771, URL: http://www.it.kth.se/docs/rfc/rfcs/rfc1771.txt

Rijsbergen, C. J. v. (1979). *Information Retrieval*. Butterworths, London, 2nd edition

Rothermel, K.; Hohl, F.; Radouniklis, N. (1997). "Mobile agent systems: What is missing?". In *Proc. 1st Int. Working Conf. on Distributed Applications and Interoperable Systems (DAIS'97)*, pages 111–124, London. Chapman & Hall

Rothermel, K.; Maihöfer, C. (1999). "A robust and efficient mechanism for constructing multicast acknowledgment trees". In *Proc. 8th Int. Conf. on Computer Communications and Networks (IC3N'99)*, pages 139–145. IEEE Computer Society Press

Rulifson, J. (1969). "DEL". Internet Engineering Task Force, Network Working Group, Request for Comments 5, URL: ftp://www.it.kth.se/docs/rfc/rfcs/rfc5.txt

Seacord, R. C.; Hissam, S. A.; Wallnau, K. C. (1998). "AGORA: A search engine for software components". *IEEE Internet Computing*, 2(6):62–70

Selberg, E.; Etzioni, O. (1995). "Multi-service search and comparison using the MetaCrawler". In *Online Proc. 4th Int. World Wide Web Conference*.
URL: http://www.w3.org/Conferences/WWW4/Papers/169/

Shakes, J.; Langheinrich, M.; Etzioni, O. (1997). "Dynamic reference sifting: A case study in the homepage domain". In *Online Proc. 6th Int. World Wide Web Conference (WWW6'97)*. URL: http://proceedings.www6conf.org/

Sheldon, M. A.; Duda, A.; Weiss, R.; Gifford, D. K. (1995). "Discover: A resource discovery system based on content routing". In *Proc. 3rd Int. World Wide Web Conference*, volume 27:6 of *Journal Computer Networks and ISDN Systems*, pages 953–972. Elsevier Science B.V.

Steen, M. v.; Homburg, P.; Tanenbaum, A. S. (1997). "The architectural design of Globe: A wide-area distributed system". Technical Report IR-422, Vrije University, Netherlands

Straßer, M.; Rothermel, K. (1998). "A fault-tolerant protocol for providing the exactly-once property of mobile agents". In *Proc. 17th IEEE Symposium on Reliable Distributed Systems (SRDS'98)*, pages 100–108. IEEE Computer Society Press

Straßer, M.; Schwehm, M. (1997). "A performance model for mobile agent systems". In Arabnia, H. R., editor, *Proc. Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA'97)*, volume 2, pages 1132–1140. CSREA Press

Tanenbaum, A. S. (1996). *Computer Networks.* Prentice Hall, New Jersey, 3rd edition

Theilmann, W.; Rothermel, K. (1998). "Domain experts for information retrieval in the World Wide Web". In *Proc. 2nd Int. Workshop on Cooperative Information Agents (CIA'98)*, LNAI 1435, pages 216–227. Springer–Verlag

Theilmann, W.; Rothermel, K. (1999a). "Disseminating mobile agents for distributed information filtering". In *Proc. Joint Symp. of 1st Int. Symp. on Agent Systems and Applications and 3rd Int. Symp. on Mobile Agents (ASA/MA'99)*, pages 152–161. IEEE Computer Society Press

Theilmann, W.; Rothermel, K. (1999b). "Efficient dissemination of mobile agents". In *Proc. 19th Int. Conf. on Distributed Computing Systems Workshop (ICDCSW'99), Electronic Commerce and Web-Based Applications*, pages 9–14. IEEE Computer Society Press

Theilmann, W.; Rothermel, K. (1999c). "Maintaining specialized search engines through mobile filter agents". In *Proc. 3rd Int. Workshop on Cooperative Information Agents (CIA'99)*, LNAI 1652, pages 197–208. Springer–Verlag

Theilmann, W.; Rothermel, K. (2000a). "Dynamic distance maps of the Internet". In *Proc. 19th Conf. on Computer Communications (IEEE Infocom'2000)*, volume 1, pages 275–284. IEEE Computer Society Press

Theilmann, W.; Rothermel, K. (2000b). "Effiziente Informationssuche mit Hilfe mobiler Programme". *Informatik Forschung & Entwicklung.* Springer–Verlag, to appear

Theilmann, W.; Rothermel, K. (2000c). "Optimizing the dissemination of mobile agents for distributed information filtering". *IEEE Concurrency*, 8(2):53–61

Vigna, G., editor (1998). *Mobile Agents and Security.* LNCS 1419. Springer–Verlag

Weider, C.; Fullton, J.; Spero, S. (1996). "Architecture of the Whois++ index service". Internet Engineering Task Force, Network Working Group, Request for Comments 1913, URL: ftp://www.it.kth.se/docs/rfc/rfcs/rfc1913.txt

Weiss, R.; Velez, B.; Sheldon, M. A.; Namprempre, C.; Szilagyi, P.; Duda, A.; Gifford, D. K. (1996). "HyPursuit: A hierarchical network search engine that exploits content-link hyper text clustering". In *Proc. 7th ACM Conference on Hypertext*, pages 180–193. ACM Press

Whitebread, K. R.; Jameson, S. (1995). "Information discovery in high-volume, frequently changing data". *IEEE Expert: Intelligent systems & their applications*, 10(5):51–53

Wondergem, B. C.; van Bommel, P.; Huibers, T. W.; van der Weide, T. P. (1997). "Towards an agent based retrieval engine". In *Proc. 19th Annual BCS IRSG Colloquium on IR Research*, pages 126–144. Robert Gordon University, Aberdeen

Yang, Y.; Liu, X. (1999). "A re–examination of text categorization methods". In *Proc. 22nd Ann. Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR'99)*, pages 42–49. ACM Press

Zelikovsky, A. (1997). "A series of approximation algorithms for the acyclic directed Steiner tree problem". *Algorithmica*, 18(1):99–110. Springer–Verlag