

Prüfer: Prof. Dr. Rothermel

Betreuer: Dipl.-Inform. Fritz Hohl

Begonnen am: 1.2.1996

Beendet am: 31.7.1996

CR-Klassifikation: C.2.4, H.4.3

Studienarbeit Nr. 1539

Finden von mobilen Agenten in einem weitverteilten System

Klaus Röhrle

Kurzfassung

Mobile Agenten sind Programme, die Aufträge für einen Benutzer ausführen. Dazu sind sie in der Lage, autonom zu handeln und von einem Rechner auf einen anderen zu migrieren, vorausgesetzt, daß auf beiden Rechnern ein „Agentensystem“ installiert ist.

In einem solchen Agentensystem ist ein Dienst hilfreich, der es erlaubt, den aktuellen Aufenthaltsort eines Agenten zu bestimmen. Ein solcher Dienst wird in dieser Arbeit Agent-Locator genannt. Die Aufgabe dieser Studienarbeit ist es, Verfahren zu finden und zu vergleichen, die als Grundlage für die Realisierung eines Agent-Locators verwendet werden können. Im einzelnen wurden folgende Möglichkeiten untersucht:

- Die Verwendung eines Broadcasts um einen Agenten zu finden.
- Das periodische Senden von Informationen über die an einem Ort anwesenden Agenten an jeden anderen Ort.
- Die Verwendung eines Energiekonzeptes, bei dem die Agenten sich melden müssen um neue Energie zu erhalten.
- Das Speichern und Verfolgen der Route, die ein Agent durchs System genommen hat.
- Die Möglichkeit Agenten durch andere Agenten suchen zu lassen.
- Das Heimatregisterverfahren, bei dem ein Ort immer über den Aufenthaltsort eines Agenten informiert ist.
- Das Speichern der Aufenthaltsorte der Agenten in einem Name Service.

Die einzelnen Verfahren und Kombinationen aus ihnen wurden aufgrund vorher festgelegter Kriterien, wie z.B. Korrektheit, Skalierbarkeit, Sicherheit und Kommunikationsaufwand miteinander verglichen und bewertet.

Die Studienarbeit beschreibt darüber hinaus den Entwurf und die Implementierung eines Agent-Locators für MOLE, einem Agentensystem, welches in der Abteilung Verteilte Systeme entwickelt wird. Der Implementation liegt das Heimatregisterverfahren zugrunde.

Inhaltsverzeichnis

1 EINFÜHRUNG.....	1
1.1 WAS SIND MOBILE AGENTEN?.....	1
1.2 WAS IST EIN AGENTENSYSTEM?	2
1.3 WAS IST DAS ZIEL DIESER ARBEIT?.....	3
2 DIE AUFGABEN EINES AGENT-LOCATORS.....	5
3 ANWENDUNGEN FÜR EIN AGENTENSYSTEM.....	7
3.1 ANWENDUNG 1: ACTIVE MAIL	7
3.2 ANWENDUNG 2: INFORMATIONSRECHERCHE	8
3.3 ANWENDUNG 3: DER VIRTUELLE MARKTPLATZ.....	9
3.4 ANWENDUNG 4: NETZWERKMANAGEMENT	10
4 BEWERTUNGSKRITERIEN.....	11
4.1 NETZPARTITIONIERUNG	11
4.2 KORREKTHEIT DES ERGEBNISSES.....	11
4.3 SKALIERBARKEIT	12
4.4 ANTWORTZEIT (KOSTEN EINER ANFRAGE).....	12
4.5 KOSTEN ZUR LAUFZEIT	13
4.6 KOSTEN DER IMPLEMENTIERUNG.....	14
4.7 SICHERHEIT/DATENSCHUTZ	14
4.8 EIGNUNG FÜR DIE BEISPIEL-ANWENDUNGEN.....	15
5 ALLGEMEINE PROBLEME	17
5.1 SYSTEMVORAUSSETZUNGEN.....	17
5.2 MIGRATION DES AGENTEN WÄHREND EINER ANFRAGE.....	17
5.3 MIGRATION DES AGENTEN DIREKT NACH EINER ANFRAGE.....	17
5.4 DAS „MOVING TARGET“-PROBLEM	18
5.5 LOKALISIEREN DES HEIMATORTES EINES AGENTEN	18
6 VERFAHREN ZUR REALISIERUNG	20
6.1 BROADCAST	20
6.1.1 Verfahren.....	20
6.1.2 Bewertung.....	20
6.2 PERIODISCHE BROADCASTS.....	23
6.2.1 Verfahren.....	24
6.2.2 Bewertung.....	25
6.3 ENERGIEKONZEPT.....	27
6.3.1 Verfahren.....	27
6.3.2 Bewertung.....	28
6.4 VERFOLGEN DER MIGRATIONSROUTE	30
6.4.1 Verfahren.....	30
6.4.2 Bewertung.....	33
6.5 STRAßENSPERREN.....	36
6.5.1 Verfahren.....	36
6.5.2 Bewertung.....	37
6.6 HEIMATREGISTER	39
6.6.1 Verfahren.....	39
6.6.2 Bewertung.....	41
6.7 NAME SERVICE.....	45
6.7.1 Verfahren.....	45
6.7.2 Bewertung.....	46
6.8 SONSTIGE VERFAHREN.....	49
7 ZUSAMMENFASSENDE BEWERTUNG.....	51

8 KOMBINATION DER VERFAHREN.....	54
8.1 HEIMATREGISTER & NAME SERVICE.....	54
8.2 HEIMATREGISTER & BROADCAST.....	54
8.3 NAME SERVICE & BROADCAST.....	55
8.4 ENERGIEKONZEPT & STRAßENSPERREN.....	55
8.5 MIGRATIONSROUTE & ENERGIEKONZEPT.....	56
8.6 MIGRATIONSROUTE & STRAßENSPERREN.....	56
8.7 STRAßENSPERREN & BROADCAST.....	56
8.8 HEIMATREGISTER & VERFOLGEN DER MIGRATIONSROUTE.....	57
9 PROTOTYPISCHE IMPLEMENTIERUNG.....	58
9.1 KURZE EINFÜHRUNG IN MOLE.....	58
9.2 AUSWAHL EINES VERFAHRENS.....	59
9.3 ENTWURF.....	60
9.3.1 Agent-Locator API.....	61
9.3.2 Der RegistrationAgent.....	63
9.3.3 Zugriffsmethoden.....	64
9.4 IMPLEMENTIERUNG.....	65
9.4.1 Der RegistrationAgent.....	65
9.4.2 Zugriffsmethoden.....	66
9.4.3 Der Display Agent.....	68
9.4.4 Der Agent Walker.....	70
9.4.5 Debug Ausgabe.....	70
9.5 TEST.....	71
9.6 NACHTEILE UND ERWEITERUNGSMÖGLICHKEITEN.....	73
10 ZUSAMMENFASSUNG.....	75
11 AUSBLICK.....	76
12 ANHANG.....	77
A ÜBERSICHT ÜBER DIE NEUEN KLASSEN.....	77
B DIE METHODEN DER NEUEN KLASSEN.....	78
C GLOSSAR.....	81
13 LITERATURVERZEICHNIS.....	85

Abbildungsverzeichnis

ABBILDUNG 1-1: AGENTENSYSTEM.....	2
ABBILDUNG 2-1: AGENT LOCATOR API.....	5
ABBILDUNG 6-1: PERIODISCHER BROADCAST.....	24
ABBILDUNG 6-2: VERKÜRZEN DER MIGRATIONSROUTE.....	32
ABBILDUNG 6-3: AKTUALISIERUNGSNACHRICHTEN 1. VERSION.....	40
ABBILDUNG 6-4: AKTUALISIERUNGSNACHRICHTEN 2. VERSION.....	40
ABBILDUNG 6-5: AKTUALISIERUNGSNACHRICHTEN 3. VERSION.....	41
ABBILDUNG 9-1: MOLE SYMBOL.....	58
ABBILDUNG 9-2: MOLE ENGINE.....	58
ABBILDUNG 9-3: AGENT-LOCATOR API.....	60
ABBILDUNG 9-4: MOLE KLASSENBAUM FÜR AGENTEN.....	67
ABBILDUNG 9-5: DISPLAYWINDOW FENSTER.....	69

1 Einführung

In diesem Kapitel beschäftigt sich mit der Frage, was Agentensysteme und mobile Agenten sind. Außerdem wird erläutert, was in dieser Studienarbeit erreicht werden soll.

1.1 Was sind Mobile Agenten?

Ein Agent ist „jemand, der im Auftrag für einen anderen eine Aufgabe erledigt“ [Fuenf95]. Diese allgemeine Definition für einen Agenten ist auch für einen mobilen Agenten gültig. Mobile Agenten sind ein neuer Ansatz im Bereich der Verteilten Systeme und bieten eine Alternative zur herkömmlichen Kommunikation mit Nachrichten.

Mobile Agenten sind kleine Programme, häufig in einer Script-Sprache geschrieben, die Aufträge für ihre Benutzer ausführen. Sie haben keine Wissensbasis und unterscheiden sich damit erheblich von Agenten, wie sie im Bereich der künstlichen Intelligenz verwendet werden. Mobile Agenten werden entweder von einem (menschlichen) Benutzer programmiert oder von einem Programm bei Bedarf erzeugt. Das „Wissen“ des Agenten ist implizit in seinem Programmcode enthalten. Mobile Agenten haben die Möglichkeit ihre Ausführung zu unterbrechen und durch ein Netzwerk zu anderen Computern zu reisen. Dort werden sie dann weiter ausgeführt. Dieser Vorgang wird als Migration des Agenten bezeichnet.

Ein Agent führt immer einen Auftrag seines Benutzers aus. Dieser wird in dieser Arbeit auch als Besitzer des Agenten bezeichnet. Der Besitzer hat die Verantwortung für den Agenten. Durch die Möglichkeit der Migration kann ein Agent von einem Ort zu einem anderen geschickt werden und dort einen Dienst für seinen Besitzer in Anspruch nehmen. Danach kann er wieder an seinen Ausgangspunkt zurückkehren, um die Ergebnisse zu abzuliefern. Es kann auch nötig sein, daß der Agent zur Erfüllung seiner Aufgabe mehrere Orte besuchen muß. Agenten müssen jedoch nicht zwingend migrieren. Sie können auch immer an einem Ort bleiben, z.B. weil sie sehr groß sind oder ihre Aufgabe keine Migrationen erfordert. Der Ort, an dem ein Agent erzeugt wird, nenne ich in dieser Arbeit „Heimatort“ des Agenten. Dies bedeutet jedoch nicht, daß der Heimatort auch öfter vom Agenten besucht wird. Es kann vorkommen, daß der Agent nur bei seiner Erzeugung am Heimatort ist und danach zu anderen Orten migriert, ohne jemals zurückzukommen.

Dieser Ansatz unterscheidet sich grundlegend von den bisherigen Lösungen für Client-Server Verarbeitung, da bei der Kommunikation durch mobile Agenten keine permanente Kommunikationsverbindung zwischen den Partnern existiert. Bisher wurden immer Nachrichten zwischen Clientprogramm und Serverprogramm über eine solche Verbindung ausgetauscht. Bei der Kommunikation mit mobilen Agenten besteht nun die Möglichkeit ein Programm zum Kommunikationspartner

zu transferieren und dort lokal ausführen zu lassen. Damit ist es möglich, daß Benutzer nicht immer mit dem Agentensystem verbunden sein müssen, sondern sich von Zeit zu Zeit einwählen können um Agenten abzuschicken oder zu empfangen.

Zwar kann man mit diesem neuen Paradigma nichts tun, was mit den alten Lösungen nicht auch schon möglich gewesen wäre, doch bringt der Einsatz von mobilen Agenten für viele Anwendungsbereiche Vorteile [Harrison94]. Einige Beispiele für die Einsatzmöglichkeiten von mobilen Agenten werden im Kapitel 3. „Anwendungen für ein Agentensystem“ näher beschrieben.

1.2 Was ist ein Agentensystem?

Das Ausführen von fremdem Programmcode auf dem eigenen Computer gilt als riskant. Woher soll ein Anbieter eines Dienstes, der sich mobiler Agenten bedient, wissen, ob der Programmcode Anweisungen enthält (absichtlich oder unabsichtlich), die Schaden an seiner Anlage verursachen? Aus diesem Grund muß das System vor den Agenten geschützt werden.

Dieser Schutz ist eine der Aufgaben des Agentensystems. Unter einem Agentensystem versteht man ein Programm, welches den Agenten die nötige Infrastruktur zur Verfügung stellt und den Rest des Systems vor ihnen abschirmt. Auf jedem Computer, der mit Agenten arbeiten will, muß eine Instanz des Agentensystems installiert sein. Gelegentlich bezeichnet man auch die Gesamtheit aller Computer, auf denen ein bestimmtes Agentensystem läuft, als Agentensystem.

Ein Agentensystem „sitzt“ zwischen dem Betriebssystem und der mit Hilfe von Agenten realisierten Anwendung.

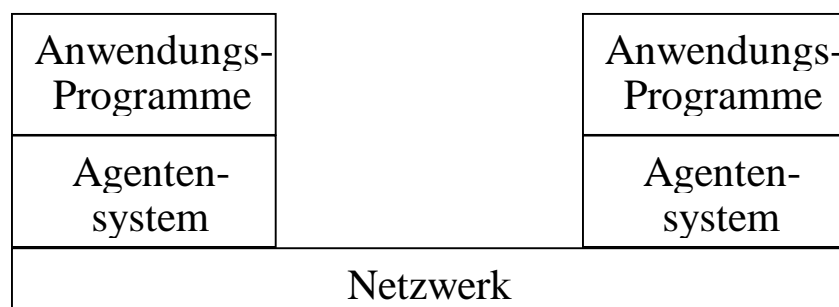


Abbildung 1-1: Agentensystem

Das Agentensystem sorgt dafür, daß die Agenten keinen direkten Zugriff auf Ressourcen des Computers bekommen, auf dem sie ausgeführt werden. Der Agent hat also keinen direkten Zugriff auf das Dateisystem, den Hauptspeicher oder auf Betriebssystemroutinen. Um sicherzustellen, daß solche direkten Zugriffe nicht möglich sind, müssen die Anweisungen im Agentencode während der Ausführung des Agenten überprüft werden. Dies ist wesentlich einfacher, wenn der Agentencode nicht in Maschinensprache vorliegt, sondern durch einen Interpreter inter-

pretiert wird. Die meisten Agentensysteme verwenden daher interpretierbare Sprachen wie z.B. JAVA, (Save-) Tcl oder Smalltalk. Die Verwendung von interpretierbaren Sprachen für das Agentensystem hat außerdem den Vorteil, daß das Agentensystem auch in einer heterogenen Umgebung funktioniert, wenn der Interpreter vorher für die entsprechenden Architekturen portiert wurde. Bereits in Maschinensprache übersetzte Programme laufen dagegen nur auf der Plattform, für die sie übersetzt wurden.

Das Agentensystem muß ebenfalls sicherstellen, daß die Agenten sich nicht gegenseitig manipulieren können. Auch der Schutz der Agenten vor dem Agentensystem ist eine wichtige Aufgabe. Dies ist insbesondere dann wichtig, wenn die Agenten virtuelles Geld oder sensible Informationen mit sich tragen. Dies ist z.B. im Telescript-System von General Magic der Fall, in welchem ein Agentensystem verwendet wird, um einen virtuellen Marktplatz zu realisieren [White94]. Ein solcher Schutz ist jedoch mehr als schwierig und stellt ein ungelöstes Problem im Bereich der mobilen Agenten dar.

Das Agentensystem stellt außerdem die gesamte Infrastruktur für die Agenten zur Verfügung. Dies beinhaltet die Bereitstellung von „Orten“ (Plätzen, Räume...). Orte sind die elektronischen Pendanten zu physikalischen Räumlichkeiten. Agenten können sich nur an Orten aufhalten. Ein mobiler Agent hält sich daher immer an einem Ort auf oder migriert gerade von einem Ort zu einem anderen. Im einfachsten Fall hat der Agent zu Beginn seiner Reise eine Liste von Orten, die er besuchen will. Interessanter ist es natürlich, wenn der Agent aufgrund von Informationen, die er unterwegs erhält, seine Route beliebig festlegen kann. In allen mir bekannten Agentensystemen ist die zweite Möglichkeit gegeben.

Das Agentensystem stellt auch die Kommunikationsmechanismen zur Verfügung. Es können dabei Kommunikationsbeziehungen zwischen Agenten, Orten und Partnern außerhalb des Agentensystems möglich sein. Da sich die Kommunikationspartner an einem Ort (lokal) oder an verschiedenen Orten (entfernt) aufhalten können, muß das Agentensystem verschiedene Kommunikationsmechanismen anbieten. Das Agentensystem ist auch für die Durchführung der Migrationen zuständig.

Ein Beispiel für ein Agentensystem ist MOLE, welches auf der Basis einer Diplomarbeit [Hohl95] in der Abteilung Verteilte Systeme am Institut für Parallele und Verteilte Höchstleistungsrechner der Universität Stuttgart entwickelt wird.

1.3 Was ist das Ziel dieser Arbeit?

In vielen Fällen wäre es wünschenswert, den aktuellen Aufenthaltsort eines Agenten herausfinden zu können. Alle mir bekannten Agentensysteme bieten jedoch keinen Dienst an, der den Aufenthaltsort eines Agenten bestimmt. Ein

Dienst, der eine solche Funktionalität besitzt, wird in dieser Arbeit Agent-Locator genannt.

Die Arbeit besteht aus zwei Teilen:

1. Zuerst soll untersucht werden, welche Verfahren es gibt, um einen Agent-Locator zu implementieren. Dazu sollen Mechanismen aus anderen Bereichen, wie z.B. der Mobiltelefonie auf ihre Verwendbarkeit in einem Agentensystem geprüft werden. Dabei soll berücksichtigt werden, daß die Skalierbarkeit bei Agentensystemen eine besondere Rolle spielt, weil diese aus potentiell vielen Agenten und Orten bestehen, die über eine große Zahl von Computern verteilt sind. Außerdem ist zu beachten, daß Agenten beliebig kurz an einem Ort verweilen können. Auch soll das Verfahren bei Netzpartitionierungen in den dann getrennten Teilnetzen weiter funktionieren.

Die Vor- und Nachteile der gefundenen Verfahren sollen verglichen und der jeweilige Kommunikationsaufwand geschätzt werden, der durch eine Verwendung des Verfahrens im Agentensystem entstehen würde.

2. Ein geeignetes Verfahren soll darüber hinaus für das Agentensystem MOLE als Anwendung implementiert werden.

2 Die Aufgaben eines Agent-Locators

Der Dienst, der von einem Agent-Locator erbracht wird, läßt sich am einfachsten durch eine Funktion beschreiben, die als Eingabeparameter einen Agentennamen erhält und als Ausgabe entweder den aktuellen Aufenthaltsort des Agenten oder eine Fehlermeldung zurückgibt (falls der Agent nicht gefunden werden konnte). Falls noch zusätzliche Informationen über den Status des Agenten vorliegen, können diese ebenfalls zurückgegeben werden. Auch die Fehlermeldung kann im Falle einer erfolglosen Suche zusätzliche Informationen enthalten, so z.B. ob der Agent bereits terminiert ist oder sich in einem gerade nicht erreichbaren Teil des Netzwerkes befindet. Beim Eingabeparameter ist zu unterscheiden ob nach jedem Agenten im System gesucht werden darf oder nur nach bestimmten Agenten. So könnte z.B. nur erlaubt sein nach Agenten zu suchen, deren Besitzer man ist. Auf jeden Fall wird vorausgesetzt, daß der Name des gesuchten Agenten bekannt ist.

Im folgenden wird der Aufruf des Agent-Locators als Anfrage eines Benutzers bezeichnet. Die ErgebnISRückgabe wird als Antwort des Agent-Locators bezeichnet, die wiederum für den Benutzer bestimmt ist. Als Benutzer ist hierbei entweder ein menschlicher Benutzer oder ein Programm (ein Agent) denkbar. Der Agent-Locator muß natürlich eine Schnittstelle anbieten, mit der es möglich ist eine Anfrage zu stellen und das Ergebnis zu erhalten. Diese Schnittstelle nenne ich Agent-Locator API (Application Programm Interface). Ist der Benutzer des Agent-Locator ein Programm (z.B. ein Agent) so kann er über diese API mit dem Agent-Locator kommunizieren. Für einen menschlichen Benutzer sollte es darüber hinaus ein Benutzerprogramm geben, welches eine grafische Benutzeroberfläche besitzen sollte und mit dem eine Anfrage an den Agent-Locator formuliert und die Antwort angezeigt werden kann.

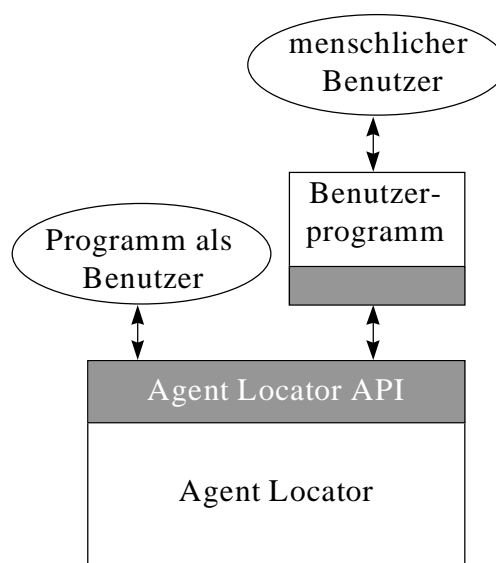


Abbildung 2-1: Agent-Locator API

In dieser Arbeit wird angenommen, daß der Agent-Locator kein Dienst ist, der zwingend für alle Agenten benötigt wird. Agenten (bzw. deren Besitzer) sollen selber entscheiden können, ob der Agent den Agent-Locator Dienst in Anspruch nehmen will oder nicht. Wenn kein Bedarf besteht, den Ort eines Agenten herauszufinden, wird der Dienst einfach ignoriert. Im anderen Fall können besondere Aktionen nötig sein, um den Agenten beim Agent-Locator „anzumelden“. Diese variieren je nach verwendetem Verfahren.

3 Anwendungen für ein Agentensystem

Agentensysteme lassen sich für viele Anwendungen einsetzen. Viele Agentensysteme sind daher auch nicht an bestimmte Anwendungen gebunden, sondern von ihrer Architektur her offen gehalten, damit sie vielseitig verwendbar sind. So werden in Agentensystemen häufig keine Annahmen darüber getroffen, wie lange Agenten leben, wie oft sie migrieren oder wie lange sie sich an einem Ort aufhalten [Beck95]. Die Beurteilung eines Verfahrens zur Ortsbestimmung von Agenten ist jedoch stark von diesen Eigenschaften abhängig. So kann ein bestimmtes Verfahren für eine Anwendung, bei der die Agenten sich lange an einem Ort aufhalten, hervorragend geeignet sein, dagegen bei einer anderen Anwendung, bei der die Agenten laufend migrieren, völlig versagen (z.B. weil das Verfahren zu langsam auf Migrationen reagiert).

Aus diesem Grund werden im folgenden einige Anwendungen für ein Agentensystem beschrieben. Diese Anwendungen unterscheiden sich bezüglich der oben genannten Eigenschaften oftmals erheblich voneinander. Die hier vorgestellten Anwendungen sind natürlich nur Beispiele in denen ein Agentensystem (sinnvoll) eingesetzt werden kann. Die Vorteile, die der Einsatz eines Agentensystems bei diesen und anderen Anwendungen hat, werden detaillierter in [Harrison94] erörtert. Alle Verfahren zur Ortsbestimmung eines Agenten werden später, unter anderem auch in Bezug auf diese Anwendungen, bewertet und beurteilt.

Am Ende der Beschreibung einer Anwendung findet sich jeweils eine Tabelle in der die Migrationsanzahl, die Aufenthaltsdauer an einem Ort und die Lebensdauer eines Agenten dargestellt wird. Diese Eigenschaften können dabei die drei Werte hoch, mittel und niedrig annehmen. Diese Einteilung ist allerdings sehr grob, und es lassen sich für jede Anwendung auch Grenzfälle konstruieren, bei denen die angegebenen Werte falsch sind. Doch sind die Angaben für die Mehrzahl der Agenten in der jeweiligen Anwendung richtig. Es lassen sich mit diesen Eigenschaften und ihren möglichen Werten 27 verschiedene Kombinationen zusammenstellen. Nicht alle davon sind sinnvoll, z.B. ist die Lebensdauer eines Agenten mit Sicherheit hoch, wenn die Lebensdauer an den Orten und die Anzahl der Migrationen hoch ist. Im folgenden wird daher nicht für jede mögliche Kombination (auch nicht für jede sinnvolle) eine Anwendung angegeben, sondern nur für einige, die nachher für die Bewertung des Agent-Locators sinnvoll sind.

3.1 Anwendung 1: Active Mail

Unter Active Mail versteht man ein System für elektronische Nachrichten. Diese sind nicht wie bei herkömmlichen E-Mail Systemen passiv, sondern können bestimmte Handlungen „aktiv“ beim Empfänger ausführen. Dies kann dazu dienen, einen Dialog mit dem Empfänger zu führen, Daten auszuwerten oder dynamische Informationen anzuzeigen. Zur Realisierung eines Active Mail Systems kann ein

Agentensystem verwendet werden. Als Nachrichten werden dann Agenten verschickt, die beim Empfänger durch das Agentensystem ausgeführt werden. Active Mail benötigt dabei nur einen Teil der Funktionalität des Agentensystems.

Eine typische Eigenschaft dieser Anwendung ist, daß die Agenten nur selten migrieren und unter Umständen lange an einem Ort bleiben. Im Extremfall wird ein Agent nur einmal verschickt und bleibt dann beim Empfänger, bis dieser die Nachricht löscht. Da es in einem solchen System möglich sein sollte, Nachrichten aufzubewahren, kann es vorkommen, daß der Agent sehr lange beim Empfänger verbleibt.

Ein Agent-Locator könnte folgende Aufgaben in einem solchen Active Mail System übernehmen:

- Er könnte überprüfen, ob eine Mail (ein Agent) schon beim Empfänger angekommen ist oder nicht.
- Er könnte die Existenz von Mail-Adressen überprüfen, falls das Active Mail System auch die Mailboxen der Benutzer durch Agenten implementieren würde. In diesem Fall hat jeder Benutzer einen eigenen immobilen Agenten, an den die Nachrichten (ebenfalls Agenten) für den Benutzer gesendet werden.

Eigenschaft	Bewertung
Anzahl der Migrationen	niedrig (u.U. nur 1x)
Aufenthaltsdauer an einem Ort	hoch
Lebensdauer eines Agenten	hoch

3.2 Anwendung 2: Informationsrecherche

In dieser Anwendung wird das Agentensystem zum Abrufen von Informationen verwendet. Informationen sollen dabei von einem oder mehreren (Datenbank-) Server im Netzwerk abrufbar sein. Zur Abfrage von Informationen benötigt ein Benutzer ein Programm auf seinem Computer, welches seine Anfrage entgegennimmt. Das Programm führt eine semantische Interpretation dieser Anfrage durch und fragt bei Bedarf beim Benutzer nach, um Unklarheiten zu beseitigen. Die daraus entstehende veränderte Anfrage wird dann durch einen Agenten zu einem oder mehreren Informations-Servern gebracht. Diese werten die Anfrage aus und übergeben dem Agenten die gewünschten Informationen. Dieser migriert wieder „nach Hause“ und übergibt dem Benutzer die gesammelten Informationen. Einer der Vorteile bei diesem Verfahren ist die Möglichkeit des Agenten, die erhaltenen Informationen an Ort und Stelle (beim Server) filtern zu können. Dies wird durch die semantische Analyse der Anfrage möglich. Damit wird eine kleinere Informationsmenge über das Netzwerk versandt und der Benutzer von nicht benötigten Informationen verschont.

In einem solchen System hätte ein Agent-Locator folgende Aufgaben:

- Er würde die Möglichkeit schaffen, Agenten, die gerade unterwegs sind, Nachrichten zukommen zu lassen. Damit könnte man den Agenten veranlassen sofort zurückzukommen oder zu terminieren.
- Ebenso kann mit dem Agent-Locator überprüft werden, ob ein Agent (und damit eine Anfrage) noch existiert.
- Wenn die Informationsserver durch Agenten im System vertreten sind, läßt sich das Vorhandensein eines Informationsdienstes an einem Ort überprüfen.

Eigenschaft	Bewertung
Anzahl der Migrationen	mittel
Aufenthaltsdauer an einem Ort	hoch
Lebensdauer eines Agenten	mittel

3.3 Anwendung 3: Der virtuelle Marktplatz

In dieser Anwendung wird von einer Nutzung des Agentensystems in einem kommerziellen Umfeld ausgegangen. Agenten, die möglicherweise digitales Geld mit sich führen, migrieren zu virtuellen Läden, um dort für ihre Eigentümer etwas zu kaufen oder einen Dienst in Anspruch zu nehmen. Die Agenten erfüllen hier weitgehend autonom einen Benutzerauftrag. So könnte ein Agent, der den Auftrag hat, bestimmte Informationen zu kaufen, Mithilfe eines Vermittlungsdienstes mögliche Anbieter herausfinden, zu jedem migrieren, um dabei Verfügbarkeit und Preis der gesuchten Information zu ermitteln und dann das günstigste Angebot wahrzunehmen. Durch das Agentensystem wird hier ein virtueller Marktplatz (oder ein Kaufhaus, je nach Geschmack) realisiert, in welchem sich die Agenten als Vertretung ihrer Besitzer bewegen können.

Ein Agent-Locator hätte in dieser Anwendung folgende Aufgaben:

- Er würde die Möglichkeit schaffen, dem Agenten nach dessen Ortsbestimmung, eine Nachricht zukommen lassen zu können, z.B. um ihm neue Informationen zu geben, seinen Auftrag abzuändern, ihn zu terminieren oder ähnliches.
- Er könnte verwendet werden um zu testen, ob ein bestimmter Agent noch existiert. Dies ist besonders wichtig, wenn Agenten elektronisches Geld oder sensible Informationen (z.B. Kreditkartendaten) mit sich führen.
- Wenn Anbieter von Diensten ebenfalls durch Agenten im System vertreten sind, kann deren Existenz überprüft werden.

Eigenschaft	Bewertung
Anzahl der Migrationen	mittel
Aufenthaltsdauer an einem Ort	mittel
Lebensdauer eines Agenten	mittel

3.4 Anwendung 4: Netzwerkmanagement

Ein Agentensystem könnte auch im Netzwerkmanagement eingesetzt werden. Eine Möglichkeit in diesem Bereich wäre, durch das Agentensystem wichtige Daten über die am Netzwerk angeschlossene Hardware zu sammeln. Dafür könnten Agenten in periodischen Abständen entsprechende Punkte im Netzwerk besuchen, die gewünschten Daten lesen und vielleicht sogar schon auswerten, um dann die gesammelten Daten an einer zentralen Stelle abzuliefern, wo sie z.B. für den Netzwerk-Administrator visualisiert werden könnten.

Ein Agent-Locator könnte hier bei der internen Fehlersuche des Management-Tools zum Einsatz kommen, um z.B. herauszufinden, ob ein Agent verloren gegangen oder noch unterwegs ist.

Diese Anwendung ist zugegebenermaßen etwas konstruiert, hat aber dafür die Eigenschaft, daß die Agenten nur sehr kurz an einem Ort bleiben und häufig migrieren. Ein Fall, der für den Agent-Locator eine besondere Herausforderung darstellt, da er mit dieser „Geschwindigkeit“ mithalten muß.

Eigenschaft	Bewertung
Anzahl der Migrationen	hoch
Aufenthaltsdauer an einem Ort	niedrig
Lebensdauer eines Agenten	niedrig

4 Bewertungskriterien

Die Beurteilung der im nächsten Kapitel beschriebenen Verfahren zur Realisierung eines Agent-Locators, erfolgt aufgrund der nun folgenden Bewertungskriterien.

4.1 Netzpartitionierung

Der Agent-Locator sollte auch bei einer Partitionierung des Netzes nicht völlig versagen, sondern in den jetzt getrennten Teilnetzen weiter funktionieren. Dies ist in großen Agentensystemen besonders wichtig, da Partitionierungen des zugrundeliegenden Netzwerkes häufiger auftreten können, je größer das Netzwerk ist. Natürlich können vom Agent-Locator keine Agenten gefunden werden, die in einer anderen Partition des Netzwerkes sind, jedoch sollte dies in der eigenen Partition sehr wohl weiter funktionieren. Optimal wäre, wenn eine Möglichkeit existieren würde, die Partitionierung des Netzes zu erkennen und herauszufinden, ob der Agent vor der Trennung in eine andere Partition migriert ist oder nicht. Damit könnte man bei einer Anfrage eine differenziertere Antwort geben als nur „Agent not found“. Das Verfahren soll sich auch dann nicht aus dem Konzept bringen lassen, wenn während einer Anfrage das Agentensystem partitioniert oder wieder verbunden wird. Verfahren, die bei Netzpartitionierung völlig versagen, sind keine guten Anwärter für die Realisierung des Agent-Locators. Die Funktion des Verfahrens bei Partitionierungen ist daher ein wichtiges Bewertungskriterium.

Die Möglichkeit, mobile Clients effizient zu unterstützen, ist einer der größten Vorteile des Agentenparadigmas. Mobile Clients sind Rechner, die ihren Standort wechseln können und daher meist nur temporär mit dem Netzwerk verbunden sind. Denkbar wären hier Notebooks, die bei Bedarf mit einem Funkmodem den Kontakt zum Netzwerk herstellen. Ist der mobile Client gerade nicht am Netzwerk angemeldet, so kann dies als Netzpartitionierung aufgefaßt werden. Der Agent-Locator sollte in der Lage sein, mit mobilen Clients zurechtzukommen. Natürlich kann ein Agent nicht gefunden werden, wenn er sich auf einem mobilen Client befindet, der gerade nicht mit dem Agentensystem verbunden ist. Ist der mobile Client jedoch mit dem Netzwerk verbunden, sollte der Agent-Locator auch für ihn normal funktionieren.

4.2 Korrektheit des Ergebnisses

Bei einigen Verfahren ist nicht immer sichergestellt, daß das Ergebnis einer Anfrage an den Agent-Locator auch wirklich den gesuchten Aufenthaltsort als Resultat liefert. So kann z.B. bei einem sehr oft migrierenden Agenten der Aufenthaltsort, der in der Antwort genannt wird, veraltet sein. Der vom Agent-Locator genannte Aufenthaltsort kann daher richtig oder falsch sein. Der Anteil der richtigen Antworten ist die „Trefferwahrscheinlichkeit“ des Verfahrens. Eine Trefferwahrscheinlichkeit von 100% ist natürlich optimal aber für manche Anwen-

dungen vielleicht gar nicht notwendig und vom Agent-Locator unter Umständen gar nicht leistbar. Der Aufwand der Implementierung, die Kosten einer Anfrage sowie die Antwortzeit können hier in Konkurrenz zur Trefferwahrscheinlichkeit stehen, so daß ein sinnvoller Kompromiß zwischen diesen Aspekten gefunden werden muß.

Am Ende der Bewertung dieses Kriteriums für das jeweilige Verfahren schließt sich eine Tabelle an, in der die möglichen Antworten, die der Agent-Locator zurückliefern kann, sowie ihre Bedeutung beschrieben sind.

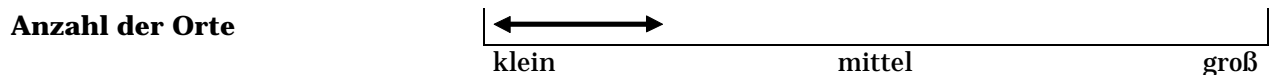
4.3 Skalierbarkeit

Da ein im Internet eingesetztes Agentensystem wie MOLE aus einer potentiell hohen Anzahl von Orten und Agenten bestehen kann, kommt der Skalierbarkeit des zu verwendenden Verfahrens eine große Bedeutung zu. Schlecht skalierbar sind Verfahren,

- bei denen die Kosten bei einem Anstieg der Zahl der Agenten, der Orte oder der Anzahl der Anfragen an den Agent-Locator überproportional steigen.
- bei denen es Obergrenzen für die Anzahl von Agenten, Orten oder Anfragen gibt.

Für jedes Verfahren gibt es ein Schaubild, welches die Eignung eines Verfahrens für Agentensysteme verschiedener Größe und unterschiedlicher Anzahl der Anfragen an den Agent-Locator veranschaulicht.

Für jede Eigenschaft gibt es eine Skala wie diese:



Die Linie gibt dabei die Eignung des Verfahrens in Bezug auf die jeweilige Eigenschaft an. Je weiter nach rechts die Linie geht, desto besser ist das Verfahren.

4.4 Antwortzeit (Kosten einer Anfrage)

Natürlich muß die Antwort auch in einer akzeptablen Geschwindigkeit erfolgen. Es bringt dem Anwender meist wenig, wenn er weiß wo sein Agent vor fünf Minuten war. Daher gilt: Je kürzer die Antwortzeit desto besser.

Die Antwortzeit hängt von der Anzahl und Art der Operationen ab, die bei einer Anfrage ausgeführt werden müssen. Eine besonders zeitaufwendige Operation ist in diesem Zusammenhang der Versand einer Nachricht. Andere Operationen (wie z.B. das Lesen oder Schreiben von Variablen oder Datenstrukturen) fallen dagegen weniger ins Gewicht, so daß die Antwortzeit hauptsächlich von den, zur Beantwortung einer Anfrage nötigen, Nachrichtenaustauschoperationen abhängt.

Falls die Antwort für einen menschlichen Benutzer bestimmt ist, kann auch die Visualisierung der Antwort (z.B. via X-Windows in einem Fenster) eine zeitauf-

wendige Angelegenheit werden. Doch ist dies weniger ein Problem des Verfahrens, das zur Realisierung des Agent-Locators verwendet wird, sondern eher der Implementierung der Benutzerschnittstelle.

Bei der Bewertung wird unterschieden, ob eine Anfrage erfolgreich war oder nicht. Bei einer erfolgreichen Anfrage wird der Agent gefunden, bei einer erfolglosen dagegen nicht. Je nach Verfahren, kann die Antwortzeit bei Erfolg oder Mißerfolg unterschiedlich sein. Da die Antwortzeit je nach Aufenthaltsort des gesuchten Agenten ebenfalls unterschiedlich ist, wird hier immer von der „worst case“ Zeit ausgegangen, also der längsten möglichen Antwortzeit.

4.5 Kosten zur Laufzeit

Unter Kosten verstehe ich in diesem Zusammenhang den Aufwand an Ressourcen, wie z.B. Speicher und Netzwerkbandbreite, die zur Ausführung des Agent-Locators benötigt werden. Die Frage lautet: „Wieviel kostet es, damit das Agentensystem weiß, wo einzelne Agenten sind“.

Ein wesentlicher Bestandteil der Kosten zur Laufzeit ist bei allen Verfahren, der durch ihren Einsatz im Agentensystem verursachte Kommunikationsaufwand. Der Kommunikationsaufwand wird im wesentlichen durch drei Faktoren bestimmt:

1. die Anzahl der notwendigen Nachrichtenaustauschoperationen.
2. der Verzögerung einer Nachricht bis sie den Empfänger erreicht.
3. die für jede Nachricht benötigte Bandbreite.

Da bei einem Agent-Locator keine großen Datenmengen befördert werden müssen, sondern nur relativ kurze Nachrichten benötigt werden, ist der 3. Punkt hier weniger relevant. Der 2. Punkt ist ebenfalls schwierig zu bestimmen. Da die Entfernung, die bei einem Nachrichtenaustausch überbrückt werden muß, bei allen Verfahren zwischen 0 (Sender und Empfänger an einem Ort) und der maximalen Entfernung zweier Orte im Agentensystem liegen kann, kann hier immer nur mit einem Mittelwert gerechnet werden. Dieser unterscheidet sich bei den einzelnen Verfahren jedoch nicht. Zur Bestimmung des Kommunikationsaufwandes wird im folgenden daher nur die Anzahl der Nachrichtenaustauschoperationen herangezogen. Auch wird angenommen, daß alle Agenten des Agentensystems den Agent-Locator in Anspruch nehmen wollen.

Eine besondere Art des Nachrichtenaustausches ist die Migration eines Agenten. Diese besteht im wesentlichen aus drei Schritten:

- Persistent machen des Agenten
- Versenden des Agenten über das Netzwerk
- Wiedereinbringen des Agenten in das Zielsystem

Der zweite Schritt ist hierbei ein Nachrichtenversand. Im Vergleich zu den beiden Schritten 1 und 3 ist dies der zeitlich aufwendigste. Die anderen Schritte sind im Vergleich dazu vernachlässigbar. Eine Migration ist daher zwar aufwendiger als das alleinige Senden einer Nachricht, doch wird der Zeitaufwand primär durch den Nachrichtenversand determiniert. Im Folgenden wird daher der Aufwand für eine Migration dem eines Nachrichtenversandes gleichgesetzt.

4.6 Kosten der Implementierung

Mit diesem Kriterium soll der Implementierungsaufwand für ein Verfahren bewertet werden. Bei Verfahren mit ähnlichen Eigenschaften ist dasjenige vorzuziehen, welches einfacher zu realisieren ist. Dabei ist zu berücksichtigen inwieweit bestehende Dienste und vorhandener Programmcode zur Realisierung verwendet werden kann. Die Kosten sind auch im Vergleich zum entstehenden Nutzen zu sehen. Ist für eine Anwendung der Agent-Locator wichtig, dann ist auch ein höherer Implementierungsaufwand gerechtfertigt.

Der Implementierungsaufwand kann natürlich nicht für jedes Verfahren exakt berechnet werden, jedoch kann man jeweils grob angeben, ob der Aufwand niedrig oder hoch ist.

4.7 Sicherheit/Datenschutz

Die Implementierung eines Agent-Locators darf keine negativen Auswirkungen auf die Sicherheit des Agentensystems haben. Agentensysteme müssen sicherstellen, daß Agenten das System nicht beschädigen und umgekehrt auch nicht vom System beeinträchtigt werden können. Natürlich muß ebenfalls verhindert werden, daß Agenten sich gegenseitig manipulieren oder schädigen können. Ein Agent-Locator darf vorhandene Sicherheitsmaßnahmen nicht unterlaufen oder neue Sicherheitslücken schaffen. Ebenso darf der Datenschutz für die Anwender des Systems nicht beeinträchtigt werden.

Zur Realisierung eines Agent-Locators kann es nötig sein, Informationen über den Weg, den ein Agent im System zurücklegt, zu sammeln. Diese Daten müssen entsprechend gesichert sein, da sich aus dem Wissen über die Aufenthaltsorte eines Agenten Rückschlüsse über dessen Auftrag ziehen lassen. Kennt man die Aufträge mehrerer Agenten eines Benutzers, so lassen sich daraus leicht Informationen z.B. über das Kaufverhalten eines Benutzers ableiten.

Es wird in dieser Arbeit vorausgesetzt, daß das Agentensystem an sich sicher ist. Es wird nur untersucht, ob durch den Agent-Locator neue Risiken entstehen. Bei einem sicheren Agentensystem ist es z.B. für einen Agenten nicht möglich, den Programmcode eines anderen zu lesen oder gar zu manipulieren.

Allgemein stellt sich bei einem Agent-Locator die Frage, ob jeder Benutzer nach jedem Agenten suchen darf oder ob sich daraus Datenschutzprobleme ergeben. Datenschutzprobleme können sich dann ergeben, wenn das Wissen über den Aufenthaltsort eines Agenten es zuläßt, daraus Informationen über seinen Besitzer oder andere Benutzer des Agentensystems zu erlangen. Um dies zu umgehen, kann man die Menge der Agenten, nach denen ein Benutzer suchen darf, einschränken. So kann nur erlaubt sein, nach Agenten zu suchen, die der Benutzer besitzt. Eine andere Möglichkeit ist die Einteilung der Agenten in bestimmte Zugriffsklassen z.B. in öffentliche, private und geheime Agenten. Je nachdem, welcher Klasse ein Agent angehört, ist eine Suche nach ihm für jeden, nur für den Besitzer oder für niemanden möglich. Ein weiterer Ansatz ist, nur bestimmten Benutzern den Gebrauch des Agent-Locators zu gestatten. Dafür ist dann eine Authentifizierung des Benutzers beim Agentensystem oder beim Agent-Locator nötig. Dieser Ansatz ist in einem offenen System jedoch nur schwer zu realisieren. Beide Möglichkeiten können auch kombiniert werden.

Da diese Problematik allgemein für einen Agent-Locator zutrifft und nicht auf ein spezielles Verfahren, wird in der Bewertung der Verfahren darauf nicht mehr eingegangen. Bewertet werden nur Sicherheits- und Datenschutzprobleme die darüber hinausgehen.

Man kann verschiedene Annahmen darüber machen, wer in einem Agentensystem vertrauenswürdig ist und wer nicht. Als nicht vertrauenswürdig sind in erster Linie Agenten einzustufen, da sie in der Regel von jedem Benutzer erzeugt werden können. Das Agentensystem muß daher auch dafür sorgen, daß Orte und andere Programme, die auf den Computern des Agentensystems ausgeführt werden, nicht durch Agenten beeinträchtigt werden können.

In manchen Fällen kann auch ein Ort als nicht vertrauenswürdig angesehen werden. In einem großen Agentensystem gibt es viele Orte, und niemand kann garantieren, daß jeder Betreiber eines Ortes vertrauenswürdig ist. Daher ist auch denkbar, daß der Programmcode eines Ortes manipuliert wird, um an Informationen zu gelangen.

Bei Überlegungen, die die Sicherheit des Agentensystems betreffen, müssen deshalb immer beide Möglichkeiten in Betracht gezogen werden.

4.8 Eignung für die Beispiel-Anwendungen

Als letztes Kriterium wird das Verfahren daraufhin bewertet, wie gut es für die oben beschriebenen Anwendungen geeignet ist. Damit wird vor allem bewertet wie gut das Verfahren für verschiedene Werte der Eigenschaften Migrationsanzahl, Aufenthaltsdauer pro Ort und Lebensdauer verwendbar ist.

Je besser ein Verfahren mit hoher Migrationsanzahl, niedriger Aufenthaltsdauer pro Ort und niedriger Lebensdauer der Agenten zurechtkommt, desto besser ist es. Für jedes Verfahren gibt es daher ein Schaubild, in der die Eignung des Verfahrens für die einzelnen Eigenschaften veranschaulicht wird.

5 Allgemeine Probleme

Es gibt einige Probleme, mit denen alle oder viele Verfahren zur Realisierung eines Agent-Locators zu kämpfen haben. Diese werden daher in diesem Kapitel gesondert behandelt.

5.1 Systemvoraussetzungen

Jedes Verfahren stellt bestimmte Anforderungen an das Agentensystem. Vor allem werden je nach Verfahren verschiedene Möglichkeiten zur Kommunikation benötigt. Manche Verfahren benötigen hier eine Kommunikationsmöglichkeit zwischen verschiedenen Orten, andere eine zwischen entfernten Agenten.

Bei einem Agentensystem kann auf jeden Fall vorausgesetzt werden, daß zwei Agenten, die sich am selben Ort befinden, miteinander kommunizieren können. Ebenfalls muß eine Möglichkeit gegeben sein, mit einem Benutzer außerhalb des Agentensystems zu kommunizieren. Ob sich Orte oder Agenten, die sich an unterschiedlichen Orten befinden, untereinander Nachrichten zukommen lassen können, kommt auf das jeweilige Agentensystem an.

In dieser Arbeit wird vorausgesetzt, daß die für ein Verfahren jeweils notwendigen Kommunikationsmöglichkeiten im Agentensystem vorhanden sind.

5.2 Migration des Agenten während einer Anfrage

Ein Problem ist die Möglichkeit, daß der Agent, während er gesucht wird, gerade migriert. Hier müssen Vorkehrungen getroffen werden, damit der Agent bei einer Anfrage nicht „übersehen“ wird. Dieses Problem tritt bei einigen Verfahren auf, bei anderen kann es dagegen nicht auftreten. Es gibt natürlich die Möglichkeit, die Anfrage einfach zu wiederholen, bis der Agent die Migration abgeschlossen hat, doch ist dies nicht sehr effizient. Es muß daher für jedes Verfahren eine individuelle Lösung gefunden werden.

5.3 Migration des Agenten direkt nach einer Anfrage

Unter der Voraussetzung, daß sich nach der Ortsbestimmung eines Agenten weitere Aktionen, wie z.B. das Senden von Nachrichten an den Agenten anschließen, ist auch das Migrieren oder die Terminierung eines Agenten kurz nach einer Anfrage problematisch, da dann die folgende Aktion ins Leere läuft. Für diesen Fall kann man jedoch besondere Maßnahmen treffen, wie z.B. das Blockieren des Agenten für eine kurze Zeitspanne. Diese sollte länger sein, als die Zeit, die bis zum Eintreffen der Nachricht an den Agenten benötigt wird. Blockiert bedeutet hier, daß der Agent nicht weiter migrieren oder sich beenden darf. Andere Aktionen sollte er noch durchführen können, so daß er seine „Arbeit“ am Ort weiter erledigen kann.

5.4 Das „Moving Target“-Problem

Unter dem „Moving Target“-Problem versteht man die Schwierigkeit, ein sich schnell bewegendes Ziel zu „treffen“. In diesem Zusammenhang ist das Ziel ein Agent, der sehr schnell durch das Agentensystem migriert. Einen solchen Agenten, der sich an keinem Ort lange aufhält, nenne ich ab jetzt „Wanderer“.

Den Aufenthaltsort eines Wanderers im Agentensystem zu ermitteln, ist naturgemäß nicht einfach:

- Das Verfahren muß schnell auf Migrationen reagieren können. Verfahren, bei denen es lange dauert bis eine Migration bei einer Anfrage berücksichtigt wird, können keinen Wanderer finden.
- Der Wanderer verbringt einen Großteil seiner Lebenszeit bei der Migration. Die Wahrscheinlichkeit, daß der Wanderer bei einer Anfrage gerade migriert, ist daher hoch. Ein Verfahren, mit dem Wanderer gefunden werden sollen, muß daher mit Agenten, umgehen können, die bei einer Anfrage gerade migrieren.
- Bis die Information über den Aufenthaltsort des Wanderers beim Benutzer angekommen ist, kann der Wanderer schon weitermigriert sein. Der in der Antwort genannte Aufenthaltsort ist schon veraltet, wenn er dem Benutzer mitgeteilt wird.

In der Anwendung „Netzwerkmanagement“ (Abschnitt 3.4) kommen solche Wanderer vor. Ob ein Verfahren mit Wanderern zurechtkommt, kann daher bei der Bewertung unter anderem daran gemessen werden, wie gut das Verfahren für diese Anwendung geeignet ist. Auch beim Bewertungskriterium „Korrektheit des Ergebnisses“ (Abschnitt 4.2) wird auf die Verwendbarkeit des Verfahrens für das Auffinden von schnell migrierenden Agenten eingegangen.

5.5 Lokalisieren des Heimortes eines Agenten

Einige der folgenden Verfahren haben den Nachteil, daß man die „Heimat“ eines Agenten kennen muß, wenn man nach ihm suchen will. In vielen Anwendungen ist dies der Fall, da oft nach Agenten gesucht wird, deren Besitzer man selbst ist. Von seinen eigenen Agenten kennt man jedoch die „Heimatadresse“. Falls diese nicht bekannt ist, muß man sie herausfinden. Dazu gibt es im wesentlichen zwei verschiedene Möglichkeiten:

- Die erste Möglichkeit ist, die Adresse im Namen des Agenten zu kodieren. Falls diese Kodierung jedoch nur für den Agent-Locator erfolgt, schränkt dies die Möglichkeiten für die Benennung von Agenten unnötig ein. Falls die Namensvergabe so geregelt ist, daß jeder Ort einen bestimmten Teil des Namensraumes zu seiner Verfügung hat, kann man von der Instanz, die die Namensräume an die Orte vergibt, erfahren, welcher Ort den Namen für den gesuchten Agenten vergeben hat. Die Kodierung erfolgt hier also implizit von der namensgebenden Instanz.

- Die zweite Möglichkeit ist der Einsatz eines Name Service, bei dem die Heimatadresse des Agenten abgefragt werden kann. Voraussetzung ist allerdings, daß der Name Service auch im Falle einer Netzpartitionierung weiterhin funktioniert. Zwar kann man, wenn ein Name Service zur Verfügung steht, einfach gleich den Aufenthaltsort des Agenten beim Name Service speichern, doch bietet die Möglichkeit, nur den Heimatort dort abzulegen auch Vorteile. Die Information über den Heimatort des Agenten ändert sich während dessen Lebenszeit nicht, sein Aufenthaltsort dagegen schon. Der Name Service muß daher nicht so viele, schnell aufeinander folgende Änderungen verkraften können.

Bei der Bewertung der verschiedenen Verfahren zur Realisierung eines Agent-Locators gehe ich davon aus, daß der Heimatort eines Agenten bekannt ist. Dies ist insofern keine große Einschränkung, da auch vorausgesetzt wird, daß der Name des Agenten bekannt ist. Gibt es im Agentensystem jedoch eine Möglichkeit, die Namen von Agenten zu erfahren, die man nicht selber erzeugt hat, so kann man über den gleichen Weg auch das Heimatregister des Agenten erfahren.

6 Verfahren zur Realisierung

In diesem Kapitel werden sieben Verfahren, die zur Realisierung eines Agent-Locators verwendet werden können, beschrieben und anschließend bewertet.

6.1 Broadcast

6.1.1 Verfahren

Eine einfache Möglichkeit einen Agenten im System zu finden ist, bei einer Anfrage einfach alle erreichbaren Orte zu fragen, ob sich der gesuchte Agent gerade dort befindet. Falls ein solcher Ort existiert, sendet dieser eine entsprechende Nachricht zurück. Damit ist der Agent gefunden, und der Name des Ortes kann als Ergebnis an den Benutzer weitergegeben werden. Trifft nach einer gewissen Zeit (Timeout) keine solche Nachricht von einem Ort ein, so wird als Antwort weitergegeben, daß der Agent nicht gefunden wurde. Um eine solche Fehlermeldung zu vermeiden, wenn der gesuchte Agent gerade migriert, könnten die Orte die letzten Anfragen für eine kurze Zeit aufbewahren. Trifft einer der angefragten Agenten in dieser Zeit ein, kann die Rückmeldung erfolgen. Die Zeitspanne in der Anfragen bei den Orten gespeichert werden müssen, sollte so lange sein wie eine Migration dauern kann. Darauf muß natürlich auch die Zeit abgestimmt werden, die gewartet wird, bevor eine Fehlermeldung generiert wird.

Mit diesem Verfahren kann jeder Agent im Agentensystem gefunden werden. Es ist auch keine „Anmeldung“ des Agenten beim Agent-Locator nötig, um später nach ihm suchen zu können.

Um alle Orte fragen zu können, benötigt man einen Broadcastmechanismus im Agentensystem. Dies entspricht einem Multicast auf der Netzwerkebene. Zur Implementierung eines Broadcasts gibt es mehrere Möglichkeiten (Flooding, minimalspannender Baum...), diese sind in der Literatur ausführlich beschrieben [Boggs83], [Mullender90]. Für den Agent-Locator wäre es natürlich vorteilhaft, wenn bereits das Netzwerk die Möglichkeit eines Broadcasts bzw. Multicasts anbieten würde. Bei kleineren Netzwerken (LANs), ist dies häufig der Fall. Effizient ist der Broadcast hier besonders dann, wenn dem Netzwerk auch ein Broadcastmedium zugrundeliegt (z.B. Ethernet). Ist das Netzwerk jedoch aus mehreren Teilnetzen aufgebaut, so wird meist kein Broadcastmechanismus zur Verfügung gestellt. Im Internet (der Heimat von MOLE) gibt es keine Broadcastmöglichkeit und auch die Möglichkeit für einen Multicast sind eher beschränkt.

6.1.2 Bewertung

Netzpartitionierung

Das Verfahren funktioniert auch im Falle einer Netzpartitionierung, da ein Agent, der sich in der gleichen Partition wie der Benutzer befindet, durch den Broadcast

auf jeden Fall gefunden wird. Es kann jedoch nicht unterschieden werden, ob der Agent nicht mehr existiert oder ob eine Netzpartitionierung vorliegt und der Agent deswegen nicht gefunden werden konnte.

Das Verfahren hat keine Probleme mit mobilen Clients. Alle mobilen Clients, die gerade mit dem Agentensystem verbunden sind, werden vom Broadcast erreicht und können antworten.

Korrektheit des Ergebnisses

Das Ergebnis der Anfrage ist immer korrekt, da nur der Ort an dem sich der Agent zur Zeit befindet eine Nachricht zurückschickt. Das Ergebnis ist daher zum Zeitpunkt des Sendens der Nachricht immer aktuell. Bis diese jedoch den Empfänger erreicht, könnte der Agent weitermigriert sein. Für diesen Fall kann man jedoch besondere Vorkehrungen treffen wie bei Abschnitt 5.3: „Migration des Agenten direkt nach einer Anfrage“ bereits beschrieben.

Bei diesem Verfahren kann ein Benutzer folgende Antworten erhalten:

Antwort	Bedeutung
Agent gefunden	Ein Ort hat die Anfrage beantwortet.
Agent existiert nicht	Bis zum Timeout wurde die Anfrage nicht beantwortet

Antwortzeit

Erfolgreiche Suche: Die Antwortzeit wird hauptsächlich durch den Zeitaufwand für den Broadcast bestimmt. Dazu kommt noch die Zeit für die Nachricht, die der gesuchte Ort zurückmeldet.

Der Zeitaufwand für einen Broadcast hängt im wesentlichen von der Größe des Netzwerkes ab. Zwar kann schon der Nachbarort der Gesuchte sein, doch nehmen wir hier wegen der „worst case“ Bedingung an, daß der letzte Ort, der gefragt wird, der Gesuchte ist. Damit wird die Antwortzeit länger, je größer das Netzwerk ist. Trotzdem ist ein Broadcast auch bei großen Netzwerken noch effizient durchführbar.

Erfolglose Suche: Im Falle einer erfolglosen Suche ist die Antwortzeit der Timeout, der gewartet wird, bevor die Anfrage abgebrochen wird.

Kosten zur Laufzeit

Bei den Kosten zur Laufzeit sind die gleichen Überlegungen gültig, wie bei der Antwortzeit. Dies liegt daran, daß nichts getan werden muß wenn gerade keine Anfrage bearbeitet wird. Es entstehen daher nur Kosten während eine Anfrage abgearbeitet wird. Diese werden im wesentlichen durch die Kosten für den Broadcast bestimmt. Wird für jeden Ort o bei einem Broadcast genau eine Nachricht erzeugt, so ist der Kommunikationsaufwand N bei a Anfragen:

$$N = a \times (o-1)$$

Hierzu ist zu bemerken, daß in der Realität unter Umständen deutlich weniger Nachrichten erzeugt werden müssen, wenn Teilen des Netzwerkes oder dem

gesamten Netzwerk ein Broadcastmedium zugrunde liegt. Außerdem geschieht der Versand der Nachrichten nicht sequentiell, sondern hochgradig parallel. Der Zeitaufwand ist damit signifikant geringer als der Zeitaufwand für N sequentielle Nachrichten. Die Formel gibt daher eine obere Schranke für den Kommunikationsaufwand an.




Kosten zur Implementierung

Falls das Agentensystem bzw. das Netzwerk bereits einen Broadcastmechanismus besitzt, ist die Implementierung einfach. Man benötigt nur ein Programm, welches die Anfrage vom Benutzer entgegennimmt und den Broadcast initiiert, sowie an jedem Ort eine Prozedur, die beim Eintreffen eines Broadcasts überprüft, ob der entsprechende Agent da ist und eine Nachricht zurücksendet falls dies zutrifft. Damit ist dieses Verfahren einfach zu implementieren.

Skalierbarkeit

Leider ist das Verfahren nur schlecht skalierbar. Zwar steigen die Kosten eines Broadcasts mit der Anzahl der Orte, doch ist der Aufwand für einen Broadcast bei einem wachsenden System irgendwann so hoch, daß er nicht mehr vertretbar ist. Man stelle sich nur vor, daß in einem größeren Netzwerk gleichzeitig viele Agenten gesucht werden. Die daraus resultierende Nachrichtenflut würde einen nicht geringen Teil der Bandbreite des Netzwerkes verbrauchen, die damit für wichtigere Dinge nicht zur Verfügung stünde.

Das Verfahren eignet sich für Agentensysteme mit folgenden Eigenschaften:

Anzahl der Orte	
Anzahl der Agenten	
Anzahl der Anfragen	

Sicherheit/Datenschutz

Ein Problem des Verfahrens birgt die Möglichkeit für einen „böartigen“ Betreiber eines Ortes, durch Beantwortung der Broadcasts, die Existenz eines Agenten zu simulieren, den es nicht mehr gibt (aus welchem Grund auch immer). Eine solche Manipulation läßt sich allerdings nur schwer verhindern. Man könnte sie jedoch erschweren, wenn der Agent eine eindeutige Nummer mit sich führen würde, die der Ort bei einer Anfrage auslesen und mit der Antwortnachricht zurücksenden müßte. Damit wäre eine solche Manipulation nur dann möglich, wenn der Ort die entsprechende Nummer des Agenten kennt. Alle Orte, die der Agent bereits besucht hat, können die Nummer jedoch ausgelesen und gespeichert haben, so daß diese weiterhin in der Lage sind Broadcasts falsch zu beantworten.

Ein Ort kann auch durch wahlloses Beantworten der Broadcasts versuchen, die Funktion des Verfahrens zu stören. Auch das würde die oben geschilderte Methode erschweren.

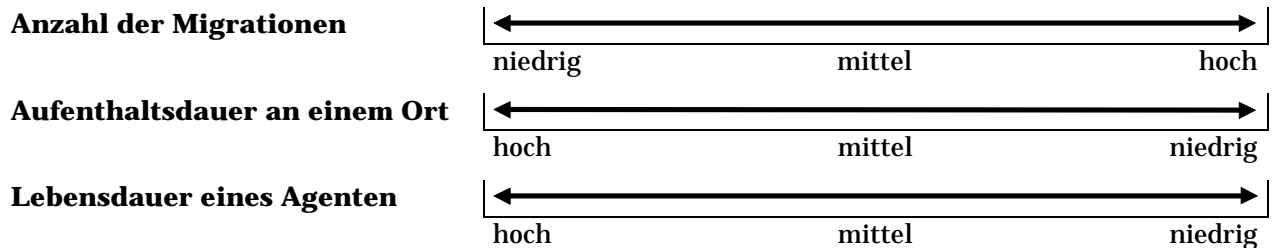
Diese Manipulationsmöglichkeiten zeigen, daß das Verfahren nur sicher ist, wenn davon ausgegangen werden kann, daß die Orte vertrauenswürdig sind. Da das Verfahren nur als Systemdienst implementiert werden kann und die Agenten nichts zur Funktion des Verfahrens beitragen, können „böartige“ Agenten das Verfahren nicht beeinträchtigen.

Eignung für die Beispiel-Anwendungen

Das Verfahren funktioniert selbst dann, wenn die Anzahl der Migrationen hoch, die Aufenthaltsdauer an einem Ort sowie die Lebensdauer der Agenten niedrig sind. Es ist unter diesem Aspekt für alle Anwendungen geeignet. Da der Aufwand für einen Broadcast bei großen Agentensystemen jedoch hoch werden kann, eignet sich das Verfahren besonders für Anwendungen, die den Agent-Locator selten benötigen. Außerdem sollte das Agentensystem nicht zu groß werden, um den Broadcast effizient durchführen zu können.

Die Anwendung Netzwerkmanagement wäre ein geeignetes Einsatzgebiet. Vorausgesetzt, es wird ein Local Area Network (LAN) und kein Metropolitan Area Network (MAN) oder Wide Area Network (WAN) verwaltet. Der Agent-Locator wird hier nur zur Fehlerbehandlung benötigt, und das Verfahren kann mit der Geschwindigkeit der Agenten mithalten.

Das Verfahren eignet sich für Anwendungen, bei denen die Agenten folgende Eigenschaften haben:



6.2 Periodische Broadcasts

6.2.1 Verfahren

Einen beliebigen Agenten zu finden wäre einfach, wenn jeder Ort die Aufenthaltsorte aller Agenten im Agentensystem besitzen würde. Dies kann dadurch erreicht werden, daß jeder Ort periodisch die Informationen, über die bei ihm gerade anwesenden Agenten, die den Agent-Locator verwenden wollen, an alle anderen Orte via Broadcast weitergibt:

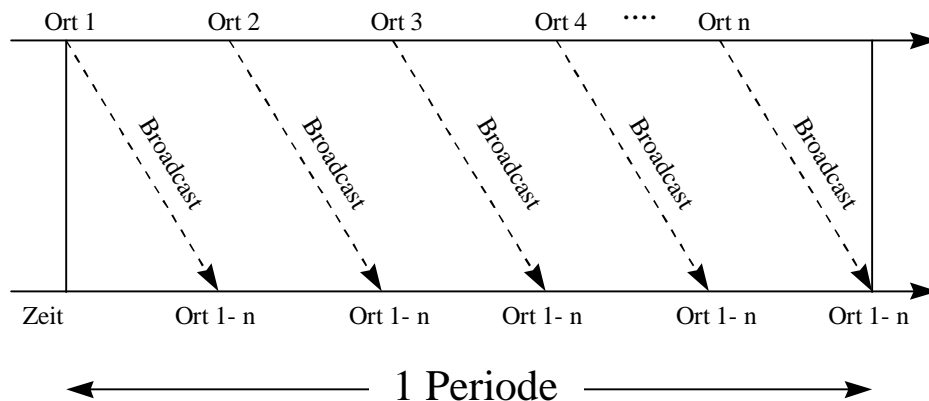


Abbildung 6-1: Periodischer Broadcast

Die Broadcasts müssen dabei nicht wie im Schaubild immer in gleichen Zeitabständen gesendet werden, sondern können auch gehäuft auftreten. Innerhalb einer Periode sendet jeder Ort genau einmal einen Broadcast. Bei dieser „Volkszählung“ weiß dann jeder Ort nach einiger Zeit über jeden Agenten Bescheid. Die Terminierung eines Agenten muß auf die gleiche Weise verbreitet werden, damit die Informationen über ihn an den Orten gelöscht werden kann. Die Informationen sind zwar nicht immer aktuell, doch kann durch eine Verkürzung der Zeitspanne zwischen den Broadcasts eine höhere Aktualität erreicht werden.

Eine Anfrage kann sich an jeden beliebigen Ort richten, da alle in etwa die gleichen Informationen besitzen (jeder Ort weiß natürlich über die bei ihm gerade anwesenden Agenten besser Bescheid als jeder andere). Damit die Orte nicht zuviel Speicherplatz für das Speichern der Aufenthaltsorte der Agenten verwenden müssen, kann dieser auch auf eine bestimmte Größe limitiert werden. Der Ort speichert dann in diesem Speicherbereich nur die jeweils letzten Informationen. Die älteren Informationen werden gelöscht. Da dann nicht mehr Informationen über alle Agenten an jedem Ort vorhanden sind, kann eine Anfrage unter Umständen nicht oder erst nach einer gewissen Zeitspanne (bis zum Eintreffen neuer Informationen über den gesuchten Agenten) beantwortet werden.

6.2.2 Bewertung

Netzpartitionierung

Ist der gesuchte Agent in der gleichen Partition wie der Benutzer, so funktioniert das Verfahren weiterhin. Ist der Agent allerdings in einer anderen Partition, so kann ein falscher Wert an den Benutzer gegeben werden, da die Informationen über Agenten in anderen Partitionen nicht aktualisiert werden können. Um dies zu vermeiden, kann man ein „Verfallsdatum“ für die Informationen an den Orten einführen. Die Informationen sollten ja sowieso innerhalb einer Periode aktualisiert werden. Geschieht dies nicht, kann man sie nach dieser Zeit auch löschen, da in diesem Fall höchstwahrscheinlich eine Partitionierung vorliegt.

Mobile Clients stellen kein Problem für das Verfahren dar.

Korrektheit des Ergebnisses

Das Ergebnis kann korrekt sein, muß aber nicht. In der Zeit zwischen den Broadcasts kann der Agent migrieren. Bis der neue Ort des Agenten den nächsten Broadcast sendet, ist die Information an den Orten über den Agenten falsch.

Auch bei einer Partitionierung können falsche Ergebnisse an den Benutzer gegeben werden. Dies geschieht, weil die Informationen über die Agenten in anderen Partitionen nicht mehr aktualisiert werden.

Bei diesem Verfahren kann ein Benutzer folgende Antworten erhalten::

Antwort	Bedeutung
Agent gefunden	Der Ort hat Informationen über den Agenten.
Agent existiert nicht	Der Ort hat keine Informationen über den Agenten.

Antwortzeit

Erfolgreiche Suche: Die Antwortzeit ist im Normalfall kurz. Da jeder Ort über jeden Agenten Informationen besitzt, kann der am nächsten liegende Ort gefragt werden. Dieser kann die Antwort sofort senden.

Wird, wie oben beschrieben, der Speicherplatz für die Speicherung der Aufenthaltsorte begrenzt, muß, falls keine Informationen über den Agenten vorliegen, gewartet werden, bis die nächsten Broadcasts aller Orte eingetroffen sind. Erst dann kann angenommen werden, daß der Agent nicht existiert.

Erfolglose Suche: Liegt an dem Ort, der die Anfrage erhält, keine Information über den Agenten vor, so wird angenommen, daß er nicht existiert. Die Antwortzeit ist daher gleich wie bei der erfolgreichen Suche. Wird der Speicherplatz für die Speicherung der Aufenthaltsorte begrenzt muß ebenfalls gewartet werden, bis die nächsten Broadcasts aller Orte eingetroffen sind.

Kosten zur Laufzeit

Der Verbrauch an Netzwerkbandbreite durch die vielen Broadcasts ist hoch. Dazu muß noch jeder Ort Speicherplatz für den Aufenthaltsort jedes Agenten vorhalten. Die Laufzeitkosten des Verfahrens sind damit deutlich höher, als die beim letzten Verfahren.

Kosten der Implementierung

Falls es einen Broadcastmechanismus im Agentensystem gibt, ist der Aufwand für die Implementierung gering. Muß ein solcher jedoch erst implementiert werden, ist die Implementierung aufwendig.

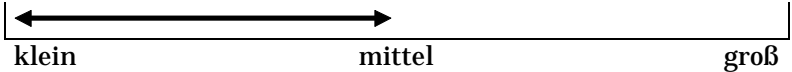


Skalierbarkeit

Das Verfahren ist schlecht skalierbar, da mit jedem neuen Ort die Kosten für die Broadcasts in einer Periode stark steigen:

Wenn der Aufwand für einen Broadcast b bei einem neuen Ort im System um den Faktor x ansteigt, so steigt der Aufwand innerhalb einer Periode um $(\text{Anzahl der bisherigen Broadcasts} + 1) \times x + b$ an. Da jeder Ort innerhalb einer Periode genau einmal einen Broadcast sendet, steigt der Aufwand bei o Orten um $(o+1) \times x + b$.

Außerdem sind die mit dem Broadcast gesendeten Informationen hier viel umfangreicher als beim letzten Verfahren, da nicht die Anfrage nach einem Agenten, sondern die Namen aller Agenten, die sich am jeweiligen Ort aufhalten, gesendet werden müssen.

Das Verfahren eignet sich für Agentensysteme mit folgenden Eigenschaften:

Anzahl der Orte	
Anzahl der Agenten	
Anzahl der Anfragen	

Sicherheit/Datenschutz

Der Datenschutz ist nur dann gewährleistet, wenn die Orte vertrauenswürdig sind. Ist dies nicht der Fall, kann ein Ort über alle Agenten im System Buch führen. Außerdem kann er durch falsche Broadcasts das ganze Verfahren torpedieren.

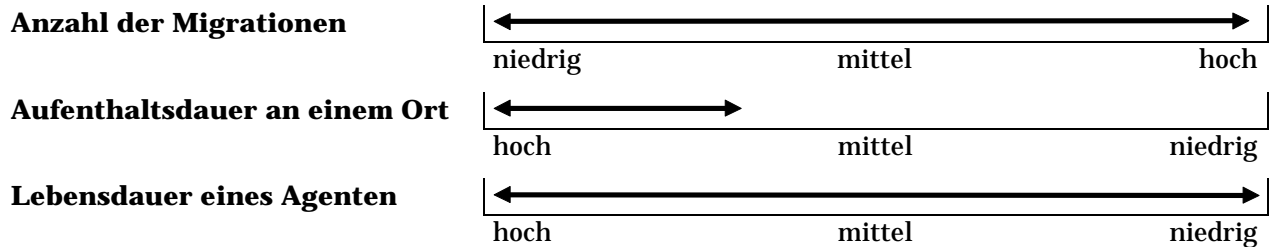
Eignung für die Beispiel-Anwendungen

Das Verfahren eignet sich am besten für Anwendungen, bei denen die Agenten wenig migrieren und lange an einem Ort bleiben. Active Mail ist die einzige Beispiel-Anwendung, die diese Eigenschaften hat. Jedoch ist es nicht unbedingt wünschenswert, daß alle Orte über sämtliche Mails im System Bescheid wissen (Datenschutz). Bei diesem Verfahren darf das Agentensystem auch nicht zu groß werden (Skalierbarkeit). Daher wäre auch nur ein Active Mail System denkbar,

welches wenige Teilnehmer hat, z.B. innerhalb einer Abteilung oder in einem kleinen Betrieb.

Das Verfahren ist daher für keine Anwendung wirklich gut geeignet.

Das Verfahren eignet sich für Anwendungen, bei denen die Agenten folgende Eigenschaften haben:



6.3 Energiekonzept

6.3.1 Verfahren

Bei diesem Verfahren kann jedem Agenten von seinem Heimatort eine bestimmte Energiemenge zugeteilt werden. Ein Agent verbraucht bei jeder Aktion, die er ausführt, Energie. Ist sein Energievorrat erschöpft, wird er terminiert. Der Heimatort ist der einzige Ort, der in der Lage ist, dem Agenten einen neuen Energievorrat zu geben. Durch eine abnehmende Energiemenge wird ein Agent also gezwungen sich bei seinem Heimatort zu melden. Dies kann entweder durch eine Nachricht oder durch eine Rückkehr des Agenten zum Heimatort geschehen. Der Heimatort kann dem Agenten daraufhin durch eine Nachricht oder direkt eine neue Energiemenge zuteilen. Die Energiemenge, die der Agent erhält, kann von Mal zu Mal unterschiedlich sein. Der Heimatort hat damit die Möglichkeit zu steuern, wie lange der Agent sich vom Heimatort entfernen darf.

Anfragen müssen immer an den Heimatort des Agenten gerichtet werden. Meldet sich der Agent am Heimatort, so werden alle Anfragen, die seit seiner letzten Meldung am Heimatort eingegangen sind, beantwortet. Der Heimatort kann ungefähr abschätzen nach welcher Zeit der Energievorrat des Agenten auf jeden Fall aufgebraucht sein muß. Hat er sich nach dieser Zeit nicht beim Heimatort gemeldet, so wurde er terminiert (keine Energie mehr). Vorliegende Anfragen können nach dieser Zeit also mit der Information beantwortet werden, daß der Agent nicht mehr existiert.

Einer der Vorteile an diesem Konzept ist, daß es hier keine Rolle spielt, wie oft oder wie schnell ein Agent migriert. Es können damit also auch Wanderer gefunden werden. Da der Agent an seinem Aufenthaltsort warten muß, bis er neue Energie erhält, stellt es auch kein Problem dar, wenn sich auf eine Anfrage weitere Aktionen, wie z.B. das Senden einer Nachricht an den Agenten anschließen.

6.3.2 Bewertung

Netzpartitionierung

Agenten, die sich in der gleichen Partition befinden wie ihr Heimatort, können sich bei ihm melden, um sich „auftanken“ zu lassen. Das Problem ist, daß der Benutzer ebenfalls in dieser Partition sein muß, um eine Anfrage an den Heimatort zu richten. Wird ein Agent durch eine Partitionierung von seinem Heimatort getrennt, so kann er keine neue Energie erhalten. Dauert diese Trennung entsprechend lange an, geht dem Agenten die Energie aus, und er wird terminiert. Bei einer Netzpartitionierung können daher nur diejenigen Agenten gefunden werden, die in der gleichen Partition sind wie ihr Heimatort und der Benutzer. Sind Benutzer oder Heimatort in einer anderen Partition, so ist der Agent nicht auffindbar.

Korrektheit des Ergebnisses

Das Ergebnis einer Anfrage ist immer korrekt. Da der Agent immer warten muß bis er neue Energie bekommt, kann der Heimatort in gewissem Umfang auch Steuern, wann der Agent weitermigriert.

Bei diesem Verfahren kann ein Benutzer folgende Antworten erhalten:

Antwort	Bedeutung
Agent am Heimatort	Der Agent befindet sich gerade am Heimatort.
Agent ist terminiert	Der Agent wurde terminiert.

Antwortzeit

Erfolgreiche Suche: Die Antwortzeit hängt von der Energiemenge, die der Heimatort dem Agenten mitgegeben hat und dem Energieverbrauch des Agenten ab. Je größer die Energiemenge des Agenten und je geringer sein Energieverbrauch, desto länger ist die Antwortzeit.

Erfolglose Suche: Die Antwortzeit hängt hier ebenfalls von der Energiemenge und dem Energieverbrauch des Agenten ab. Der Heimatort muß die Zeitspanne abschätzen können, die der Agent mit der ihm zur Verfügung stehenden Energie höchstens überleben kann. Diese Zeitspanne plus einem „Sicherheitspolster“ ist die Antwortzeit bei einer erfolglosen Suche.

Die Antwortzeit ändert sich damit dynamisch, je nach Energieausstattung und Energieverbrauch des Agenten.

Kosten zur Laufzeit

Zur Realisierung dieses Verfahrens muß das Energiekonzept im Agentensystem implementiert werden. Einem Agenten muß bei jeder Aktion eine bestimmte Energiemenge abgezogen werden. Somit hat jede Aktion in einem solchen System auch ihren „Preis“. Damit kann dieses Konzept auch für Abrechnungszwecke oder zur Ressourcenkontrolle verwendet werden. Der Agent sollte vor einer Aktion über deren „Preis“ unterrichtet werden, so daß er sich abhängig von den Kosten

entscheiden kann, ob er die Aktion ausführt, eine billigere Alternative sucht oder sich erst von seinem Heimatort „aufzutanken“ läßt. Laufzeitkosten treten daher durch den „Zahlungsvorgang“, durch den „Tankvorgang“, für die Kalkulation der Kosten einer Aktion und durch die Kalkulationen auf, die ein Agent durchführen muß, um sich „richtig“ zu entscheiden.

Wird das Energiekonzept außer für den Agent-Locator auch zur Ressourcenkontrolle oder für Abrechnungszwecke verwendet, sind die Laufzeitkosten auf jeden Fall tragbar.

Kosten der Implementierung

Die Implementierung des Energiekonzeptes ist aufwendig. Besonders, weil Sicherheitsmechanismen vorhanden sein müssen, damit ein Agent nicht selber Energie herstellt. Das Energiekonzept kann auch nur schwer als Anwendung implementiert werden, sondern sollte vom Agentensystem realisiert werden.

Skalierbarkeit

Das Verfahren ist frei skalierbar.

Das Verfahren eignet sich für Agentensysteme mit folgenden Eigenschaften:



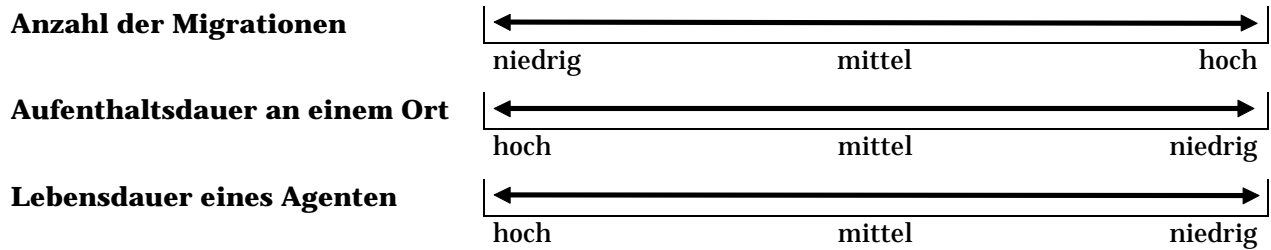
Sicherheit/Datenschutz

Da keine Informationen über Agenten gespeichert werden, gibt es keine Datenschutzprobleme. Für die Realisierung des Energiekonzeptes sind jedoch Sicherheitsvorkehrungen nötig, die verhindern, daß ein Agent selber Energie herstellen kann, anderen Agenten Energie entziehen kann oder von anderen Orten als seinem Heimatort Energie erhält. Dies ist insbesondere dann wichtig, wenn das Energiekonzept auch zur Ressourcenkontrolle oder für Abrechnungszwecke verwendet wird.

Eignung für die Beispiel-Anwendungen

Der Einsatz dieses Verfahrens ist in allen Anwendungen denkbar. So kann z.B. durch das regelmäßige Melden beim Heimatort die Existenz eines Agenten überprüft werden. Eine solche Existenzprüfung ist in allen Anwendungen eine der möglichen Aufgaben des Agent-Locators.

Das Verfahren eignet sich für Anwendungen, bei denen die Agenten folgende Eigenschaften haben:



6.4 Verfolgen der Migrationsroute

6.4.1 Verfahren

Jeder Ort merkt sich, welche Agenten bei ihm waren und wohin sie migriert sind. Wenn man nun den Ort kennt, an dem der gesuchte Agent gestartet ist (seine Heimat), so kann man die Route nachverfolgen, die der Agent durchs Netzwerk genommen hat. Dies kann durch die Sendung eines weiteren (Detektiv-) Agenten geschehen, der dem gesuchten Agenten hinterhereilt. Der Detektiv fragt an jedem Ort nach, wohin der gesuchte Agent migriert ist und reist ihm nach. Hat er ihn eingeholt, so sendet er eine Nachricht zurück, in der der Aufenthaltsort des Agenten genannt wird.

Das Verfahren funktioniert natürlich nur, wenn der Detektiv schneller ist als der gesuchte Agent. Dies sollte jedoch der Fall sein, da der Detektiv nur sehr kurz an den einzelnen Orten verweilen muß. Wenn allerdings der gesuchte Agent ebenfalls sehr schnell durch das Netzwerk migriert, kann es vorkommen, daß er nie eingeholt wird. In diesem Fall endet die Suche damit, daß der Detektiv den Ort erreicht, an dem der gesuchte Agent terminiert ist und dies zurückmeldet, bevor er sich selber beendet. Um die Chancen des Detektiven zu erhöhen, wäre es auch denkbar, ihm eine höhere Priorität an den Orten zu geben, so daß z.B. die Anfrage zum Migrationspfad des gesuchten Agenten vorrangig bearbeitet wird.

Damit sich ein Ort nicht Informationen über alle Agenten, die ihn besucht haben, merken muß, ist es sinnvoll, die Agenten, die den Dienst in Anspruch nehmen wollen, zu „markieren“. Ein Ort braucht dann nur Informationen über markierte Agenten zu speichern. Damit werden nur Informationen über diejenigen Agenten gesammelt, die den Dienst auch benötigen.

Ein Datensatz der für die Speicherung von Agenteninformation benötigt wird, ist zwar eher klein, doch stellt sich trotzdem die Frage, wie groß die Informationsmenge werden kann, die ein Ort zur Verfügung stellen muß. Dies hängt im wesentlichen auch davon ab, nach welcher Zeitspanne Informationen über einen Agenten wieder gelöscht werden können. Wenn die maximale Lebensdauer eines

Agenten berechnet werden kann, können die Informationen über einen Agenten nach dieser Zeitspanne gelöscht werden. Ist dies jedoch nicht der Fall oder ist die max. Lebensdauer zu lange, kommt es zu einem Konflikt zwischen der Speicherkapazität eines Ortes und dem Interesse des Agent-Locators die Informationen möglichst lange aufzubewahren. Hier muß ein praktikabler Kompromiß gefunden werden, der je nach Anwendungsgebiet des Agentensystems unterschiedlich sein kann (bestimmt durch die Anzahl der Agenten und Migrationen). Lösen ließe sich das Problem auch durch die Rückmeldung über die Terminierung eines Agenten, rückwärtsgehend entlang der gespeicherten Route. Jeder Ort, der eine solche Rückmeldung empfängt, löscht die Daten über den entsprechenden Agenten aus seinem Speicher. Dies spart Speicherplatz bei den Orten, kostet jedoch zusätzliche Netzbandbreite für die Rückmeldungen. Außerdem muß nicht nur gespeichert werden wohin ein Agent migriert, sondern auch woher er gekommen ist.

Eine andere Möglichkeit ist, daß der Agent vor seiner Terminierung immer an seinen Ausgangspunkt zurückkehren muß. Ist er dort angekommen und terminiert, kann der Ort eine Nachricht entlang der Route des Agenten senden, daß die Informationen über den Agenten gelöscht werden können. Allerdings bedeutet die Forderung, daß der Agent immer an seinen Ausgangspunkt zurückkehren muß, eine Einschränkung für die Mobilität der Agenten.

Die Orte sollten eventuell auftretende „Schleifen“ in der Migrationsroute eines Agenten erkennen und „abschneiden“. Gibt es für einen Agenten immer nur einen Eintrag bei einem Ort, so geschieht dies automatisch, da der Eintrag überschrieben wird, wenn der Agent den Ort erneut besucht. Es muß allerdings auf jeden Fall noch eine Nachricht zu den Orten entlang der Schleife geschickt werden, daß sie die Informationen über den jeweiligen Agenten löschen können. Die Informationen dürfen allerdings nur dann gelöscht werden, wenn sich kein Detektiv in der Schleife befindet, der den entsprechenden Agenten sucht.

Bei diesem Verfahren stellt es kein Problem dar, wenn der gesuchte Agent während der Anfrage gerade migriert. Kommt der Detektiv an einen Ort, an dem der gesuchte Agent noch nicht gewesen ist, so hat er ihn überholt und kann auf ihn warten, bevor er seine Erfolgsnachricht sendet.

Das Verfahren hat den Nachteil, daß es empfindlich auf den Ausfall von Orten im Agentensystem reagiert. Wenn ein Ort ausfällt, so sind alle Agenten, die ihn in der letzten Zeit besucht haben, nicht mehr auffindbar. Die „Spur“, die sie gelegt haben, ist unterbrochen. Lösen kann man das Problem, wenn an den Orten nicht nur gespeichert wird, wohin die Agenten migrieren, sondern auch woher sie kommen. Ein Ort muß dann, wenn er sich vom Agentensystem „abmeldet“, dem Ort, den der Agent vorher besucht hat, mitteilen wohin der Agent danach gegangen ist. Damit wird der Ort aus der Route des Agenten entfernt und diese somit abgekürzt.

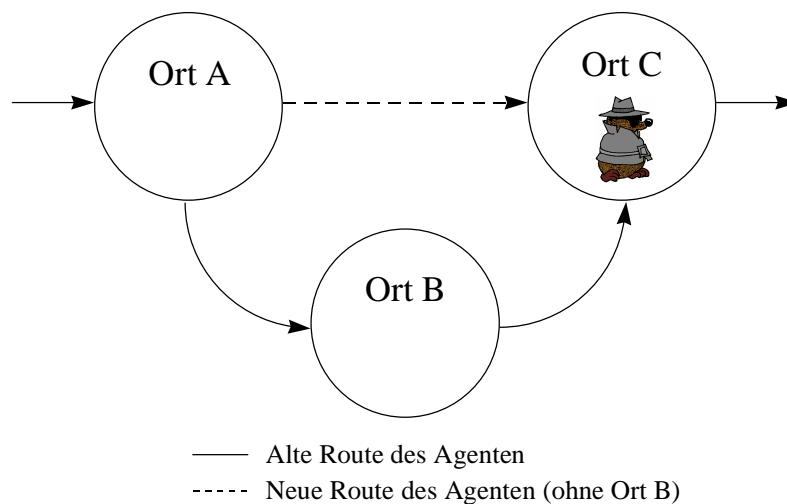


Abbildung 6-2: Verkürzen der Migrationsroute

Dieser Mechanismus ist auch eine Möglichkeit, um die Chancen der Detektive zu erhöhen. Durch die kürzere Route, wird der gesuchte Agent schneller eingeholt. Er löst das beschriebene Problem jedoch nicht, wenn die Orte nicht geregelt vom Agentensystem getrennt werden, sondern z.B. durch einen Rechnerabsturz ausfallen.

Anstatt einen Agenten loszuschicken, könnte der Agent alternativ auch durch das Versenden von Nachrichten gefunden werden, wobei hier die Orte die Aufgabe haben, die Nachrichten entlang der entsprechenden Route weiterzuleiten. Trifft die Nachricht an dem Ort ein, an dem sich der Agent aufhält, meldet der Ort dies zurück. Hier fällt also den Orten die gesamte Arbeit zu, während im vorherigen Fall der Ort „nur“ die Migrationsoperationen protokollieren und eine Schnittstelle zur Abfrage dieser Informationen anbieten muß. Der Agent-Locator ist in diesem Fall also tiefer im Agentensystem integriert. Diese Version hat den Vorteil, daß eine Nachricht weniger Zeit beansprucht, als eine Migration und damit der gesuchte Agent schneller eingeholt wird. Die erste Möglichkeit bleibt dafür im Agenten-Paradigma und ist daher konzeptionell schöner.

Nachteil bei diesem Verfahren ist, daß der „Startpunkt“ des gesuchten Agenten bekannt sein muß oder herausgefunden werden kann. Zwei Lösungen für dieses Problem wurden bereits im Abschnitt 5.5: „Lokalisieren des Heimortortes eines Agenten“ beschrieben. Theoretisch könnte zwar jeder Ort gefragt werden, der auf der Route des Agenten liegt. Dies würde schließlich nur einen anderen (besseren) Einstieg für den Detektiv bedeuten. Jedoch kennt auch der Benutzer die Route des Agenten meist nicht, und „Schüsse ins Blaue“ sind bei einem Agentensystem mit vielen Orten wenig erfolgversprechend.

6.4.2 Bewertung

Netzpartitionierung

Dieses Verfahren verträgt Netzpartitionierungen leider nicht besonders gut. Es gibt vor allem drei Fälle bei denen das Verfahren versagt:

- Agent und Benutzer sind in der gleichen Partition, die Heimat des Agenten jedoch in einer Anderen. Damit kann der Agent durch den Benutzer nicht gefunden werden.
- Agent, Benutzer und die Heimat des Agenten sind in der gleichen Partition, der Agent war jedoch vor der Partitionierung schon in einem Teil des Netzwerkes, das jetzt nicht mehr erreichbar ist. Damit ist die „Spur“, die der Agent gelegt hat unterbrochen. Auch hier kann der Agent nicht gefunden werden. Wenigstens kann hier die Partitionierung des Netzwerkes erkannt werden.
- Der Detektiv wird bei seiner Suche durch eine Netzpartitionierung vom gesuchten Agenten seiner Heimat und dem Benutzer getrennt. Er ist damit in einem Teil des Netzwerkes „gefangen“.

Positiv ist, daß die Partitionierung erkannt wird, wenn nur der Agent in einer anderen Partition ist. In diesem Fall müßte der Detektiv beim Verfolgen des Agenten an einen Ort migrieren, der im Moment nicht erreichbar ist. Damit kann die Partitionierung festgestellt und zurückgemeldet werden.

Mobile Clients können dann verwendet werden, wenn ein Mechanismus vorgesehen ist, der die Migrationsinformationen korrigiert, wenn sich ein Ort vom Agentensystem abmeldet. Eine Möglichkeit dazu wurde oben bereits für den Ausfall von Orten beschrieben.

Korrektheit des Ergebnisses

Das Ergebnis der Anfrage ist fast immer korrekt, da das Ergebnis zum Sendezeitpunkt der Nachricht immer aktuell ist. Bis diese jedoch den Empfänger erreicht, könnte der Agent weitermigrieren oder terminieren. Für diesen Fall kann man jedoch besondere Vorkehrungen treffen, wie im Abschnitt 5.3: „Migration des Agenten direkt nach einer Anfrage“ beschrieben.

Da der Agent immer einen Vorsprung vor dem Detektiv hat, kann es häufig vorkommen, daß der Detektiv nur noch melden kann, an welchem Ort der Agent terminiert ist. Dies tritt um so häufiger auf, je schneller die Agenten migrieren und je kürzer sie leben.

Bei diesem Verfahren kann ein Benutzer folgende Antworten erhalten:

Antwort	Bedeutung
Agent gefunden	Der Detektiv hat den Agenten eingeholt.
Agent terminiert	Der Detektiv konnte nur noch die Terminierung des Agenten feststellen
Keine Information	Der Agent existiert nicht, ist nicht beim Agent-Locator angemeldet oder es wurde der falsche Startpunkt für die Suche angegeben.
Partitionsfehler	Der Agent ist in einer anderen Partition des Agentensystems
Fehler	Der Detektiv konnte den Agenten nicht finden (z.B. Spur verloren).

Antwortzeit

Erfolgreiche Suche: Die Antwortzeit hängt von der Dauer ab, die der Detektiv zur Verfolgung des Agenten benötigt. Da dieser im schlimmsten Fall genauso schnell migriert wie der Detektiv, wird dieser ihn nie einholen. Die längste Antwortzeit ist daher die Lebensdauer des gesuchten Agenten. Diese kann unter Umständen jedoch sehr hoch sein, so daß man bei diesem Verfahren, wenn man Pech hat, sehr lange auf die Beantwortung einer Anfrage warten muß. Dieser Fall tritt jedoch bei den meisten Anwendungen nicht auf, da der Detektiv normalerweise schneller als der gesuchte Agent ist. Trotzdem ist die Antwortzeit im Vergleich zu anderen Verfahren sehr lang, da im Normalfall doch ein paar Migrationen nötig sein werden, um den gesuchten Agenten einzuholen.

Erfolglose Suche: Eine Suche endet hier erfolglos, wenn der Detektiv die Spur des gesuchten Agenten verliert oder aus anderen Gründen kein Ergebnis zurückmelden kann. Im schlimmsten Fall geht der Detektiv verloren. Um dann überhaupt noch antworten zu können, muß die Suche nach einem Timeout abgebrochen werden, der jedoch sehr lange sein muß, um dem Detektiv genügend Zeit zu lassen.

Die Antwortzeit des Verfahrens ist damit sehr lang.

Kosten zur Laufzeit

Die Kosten zur Laufzeit hängen von der mittleren Anzahl der Migrationen ab, die ein Detektiv braucht, bis er sein Ziel erreicht hat, sowie den Kosten zum Speichern der Migrationsinformationen an den Orten.

- Die mittlere Anzahl an Migrationen hängt wiederum davon ab, wie häufig die Agenten migrieren und wie lange sie leben. Wenn der gesuchte Agent schon eine Weltreise hinter sich hat, bevor der Detektiv losgeschickt wird, können die Kosten für eine Anfrage hoch sein.
- Der benötigte Speicherplatz an den Orten hängt von der Anzahl der Migrationen ab, die pro Zeiteinheit im Agentensystem vorkommen. Für jede Migration erfolgt ein Eintrag bei einem der Orte. Orte die häufiger von Agenten besucht werden, brauchen auch dementsprechend mehr Speicherplatz, als Orte die selten einen

Agenten „zu Gesicht“ bekommen. Bei Anwendungen, in denen es viele Agenten gibt, die auch noch viel migrieren, kann der Speicherplatzbedarf insgesamt hoch werden.

Das Verfahren ist unter diesem Aspekt nur für Anwendungen empfehlenswert, bei denen insgesamt wenig Migrationen vorkommen und es nicht viele Agenten gibt. Auch muß die Migrationsanzahl einigermaßen gleichmäßig auf alle Orte verteilt sein.

Kosten der Implementierung

Für die Implementierung müssen die Orte dahingehend erweitert werden, daß sie jede Migration protokollieren. Da ein Ort sowieso darüber informiert sein muß, wann ein Agent migriert, bleibt hier die Aufgabe diese Informationen in einer Art Tabelle zu speichern. Außerdem muß eine Schnittstelle zur Abfrage dieser Informationen für die Detektive vorhanden sein. Zusätzlich muß noch ein Programm implementiert werden, welches mit dem Benutzer kommuniziert (Benutzerschnittstelle) und die Detektivagenten erzeugen kann.

Der Implementierungsaufwand hält sich damit in einem vernünftigen Rahmen. Aufwendiger wird es, falls aus Sicherheitsgründen eine Authentisierung von Detektiven zur Abfrage der Migrationsinformationen nötig wird.

Skalierbarkeit

Das Verfahren ist schlecht skalierbar, da bei einer Vergrößerung der Agentenanzahl und der Anzahl der Migrationen der Speicherplatz, der im Agentensystem zur Speicherung der Migrationsrouten verwendet werden muß, stark zunimmt.

Das Verfahren eignet sich für Agentensysteme mit folgenden Eigenschaften:

Anzahl der Orte	
Anzahl der Agenten	
Anzahl der Anfragen	

Sicherheit/Datenschutz

Der kritische Punkt hierbei ist die Speicherung der Migrationsinformationen an den Orten:




- Zum einen muß die Schnittstelle zur Abfrage dieser Informationen so „sicher“ sein, daß sich ein „böswilliger“ Detektiv durch sie keinen Zugang zu anderen Informationen oder Funktionen des Ortes beschaffen kann.
- Zum anderen können die gespeicherten Daten den Datenschutz gefährden. Denn durch die Information, wo sich ein Agent aufgehalten hat, können Rückschlüsse auf seine Tätigkeit und seine Aufgabe gezogen werden. Damit lassen sich dann Informationen über den Besitzer des Agenten sammeln. Um dies zu verhindern

müßte eine Authentifizierung der Detektiv-Agenten erfolgen. Damit wäre ausgeschlossen, daß sich Agenten als Detektive ausgeben, um Informationen über andere Agenten zu erschleichen.

Eignung für die Beispiel-Anwendungen

Das Verfahren eignet sich für Anwendungen, bei denen die Agenten keine langen Wege gehen und lange an einem Ort bleiben. Active Mail und die Informationsrecherche haben diese Eigenschaften. Für beide wäre das Verfahren einsetzbar, wenn man damit leben kann, daß das Verfahren bei Netzpartitionierungen öfter einmal versagt. Es sollten jedoch auf jeden Fall Sicherheitsvorkehrungen getroffen werden, damit die gespeicherten Informationen über die Agenten nicht mißbraucht werden können.

Das Verfahren eignet sich für Anwendungen, bei denen die Agenten folgende Eigenschaften haben:

Anzahl der Migrationen		niedrig	mittel	hoch
Aufenthaltsdauer an einem Ort		hoch	mittel	niedrig
Lebensdauer eines Agenten		hoch	mittel	niedrig

6.5 Straßensperren

6.5.1 Verfahren

Falls man keine Informationen über die Route eines Agenten im Agentensystem sammeln will, kann man auch nach der folgenden Methode vorgehen:

Statt eines Detektivagenten, der den gesuchten Agenten verfolgt, kann man viele Detektive aussenden. Diese verteilen sich über das Agentensystem und warten an wichtigen Knotenpunkten, ob der gesuchte Agent vorbeikommt. Dies entspricht in der Realität einer Polizeifahndung mit Straßensperren. Wie dort ist es auch hier nicht sicher ob das gesuchte Objekt gefunden wird. Es ist daher sinnvoll, die Anfrage nach einem Timeout abzubrechen und die Detektive danach zu terminieren.

Anstatt an wichtigen Orten zu warten, können die Detektive sich auch „zufällig“ durchs Agentensystem bewegen, um so den gesuchten Agenten zu finden. Beide Möglichkeiten können auch zusammen verwendet werden: Ein Teil der Detektive wartet an „strategisch“ günstigen Orten, ein anderer Teil wandert durch das Agentensystem.

Die Detektive haben eine größere Chance wenn sie Anhaltspunkte haben an welchen Orten sich der gesuchte Agent aufhalten könnte. Da der Agent einen Auftrag seines Besitzers ausführt, ist es durchaus möglich, daß dieser entsprechende Informationen hat und sie den Detektiven „mitteilen“ kann. Dies könnte allerdings die Formulierung einer Anfrage erheblich erschweren.

Wenn die Detektive Informationen über den Auftrag des Agenten haben, könnten sie sich so verhalten, als wollten sie den Auftrag selber ausführen. Dadurch ist die Wahrscheinlichkeit hoch, daß sie an die gleichen Orte kommen wie der Agent.

Problematisch bei diesem Verfahren ist, das Verhalten der einzelnen Detektive aufeinander abzustimmen. Schließlich muß verhindert werden, daß alle Detektive in der selben „Gegend“ des Agentensystems suchen. Ein Extremfall wäre die Realisierung eines Broadcasts durch die Detektive, bei dem ein Detektiv für jeden Ort gebraucht wird.

6.5.2 Bewertung

Netzpartitionierung

Es gibt keine Probleme bei Netzpartitionierung, da die Detektive in der gleichen Partition suchen, in der der Benutzer sich aufhält.

Mobile Clients sind ebenfalls kein Problem.

Korrektheit des Ergebnisses

Falls eine Antwort erfolgt, ist sie korrekt. Falls die Detektive jedoch den Agenten bis zum Timeout nicht gefunden haben, heißt das nicht, daß er nicht existiert. Das Ergebnis ist daher im Erfolgsfall korrekt, bei einer erfolglosen Suche gibt es jedoch keine Aussage über den Verbleib des Agenten.

Bei diesem Verfahren kann ein Benutzer folgende Antworten erhalten:

Antwort	Bedeutung
Agent gefunden	Ein Detektiv hat den Agenten gefunden.
Agent nicht gefunden	keine Aussage über den Verbleib des Agenten.

Antwortzeit

Erfolgreiche Suche: Die Zeit bis der Agent gefunden wird ist hier variabel. Der Agent kann sofort gefunden werden oder erst kurz vor dem Timeout. Die längste Antwortzeit ist deshalb die Zeitspanne die für den Timeout angesetzt wurde.

Erfolglose Suche: Eine Suche ist erfolglos, wenn sie durch den Timeout abgebrochen wird. Da der Timeout lang genug gewählt werden muß, damit die Detektive eine Chance haben, ist auch die Antwortzeit lang.

Kosten zur Laufzeit

Es treten nur Kosten bei der Ausführung einer Anfrage auf. Diese hängen von der Anzahl der Detektive ab, die im Durchschnitt pro Anfrage erzeugt werden, sowie deren Migrationshäufigkeit.

Kosten der Implementierung




Man benötigt ein Programm, welches die Anfrage vom Benutzer entgegennimmt und die Detektive erzeugt. Ansonsten ist nur noch eine Schnittstelle an den Orten nötig, damit die Detektive Nachfragen können, ob der gesuchte Agent da ist. Eine solche Schnittstelle sollte jedoch sowieso an den Orten vorhanden sein.

Die Schwierigkeit liegt hier in der Programmierung der Detektive, da diese Strategien für ihre Suche benötigen und ihr Verhalten aufeinander abstimmen müssen.

Skalierbarkeit

Theoretisch beliebig skalierbar. Es ist jedoch sinnvoll mehr Detektive einzusetzen, wenn das Agentensystem wächst, da sonst die Erfolgchancen der Detektive sinken. Dieser Mehraufwand kann jedoch in der Größe so dimensioniert werden, daß dies keine überproportionale Belastung darstellt.

Das Verfahren eignet sich für Agentensysteme mit folgenden Eigenschaften:

Anzahl der Orte	
Anzahl der Agenten	
Anzahl der Anfragen	

Sicherheit/Datenschutz

Solange die Detektive keine Informationen über andere, als den gesuchten Agenten sammeln, gibt es keine Sicherheits- oder Datenschutzprobleme. Das Programm, welches die Detektive erzeugt, sollte daher vertrauenswürdig sein. Auch wäre es gut, wenn sich ein Detektiv gegenüber einem Ort als Detektiv „ausweisen“ könnte.

Eignung für die Beispiel-Anwendungen

Das Verfahren ist für keine Anwendung sonderlich gut geeignet. Das Verfahren ist nur anwendbar, wenn das Agentensystem sehr klein ist oder wenn die Anwendung so beschaffen ist, daß es einige Orte gibt, die oft besucht werden müssen.

schaulicht. Dabei ist der Aufenthaltsort des Agenten oben, der Eintrag im Heimatregister des Agenten unten angegeben (Raum-Achse). Die Zeit „läuft“ von links nach rechts.

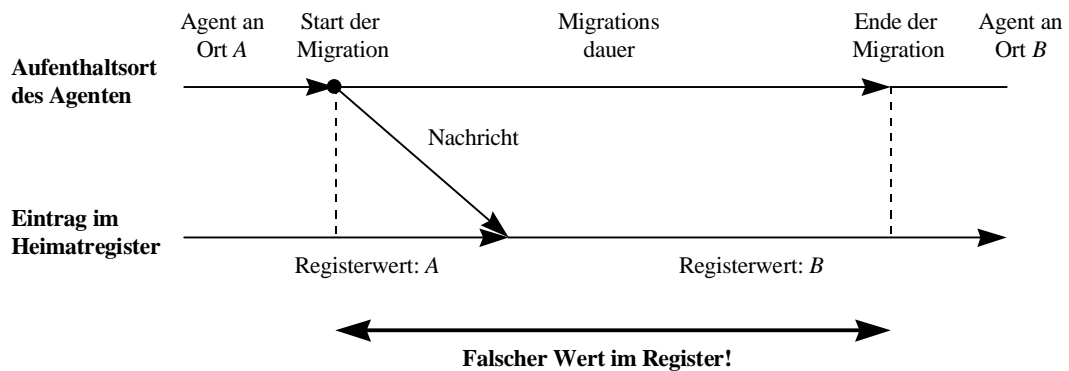


Abbildung 6-3: Aktualisierungsnachrichten 1. Version

- Die Nachricht vor einer Migration zu senden, hat den Vorteil, daß nur für die Zeitspanne der Migration ein falscher Wert im Heimatregister steht. Bis zum Eintreffen der Nachricht am Heimatort ist dies der alte Wert, bis zur Vollendung der Migration der neue (unter der Annahme, daß die Migration länger dauert als der Versand der Nachricht).

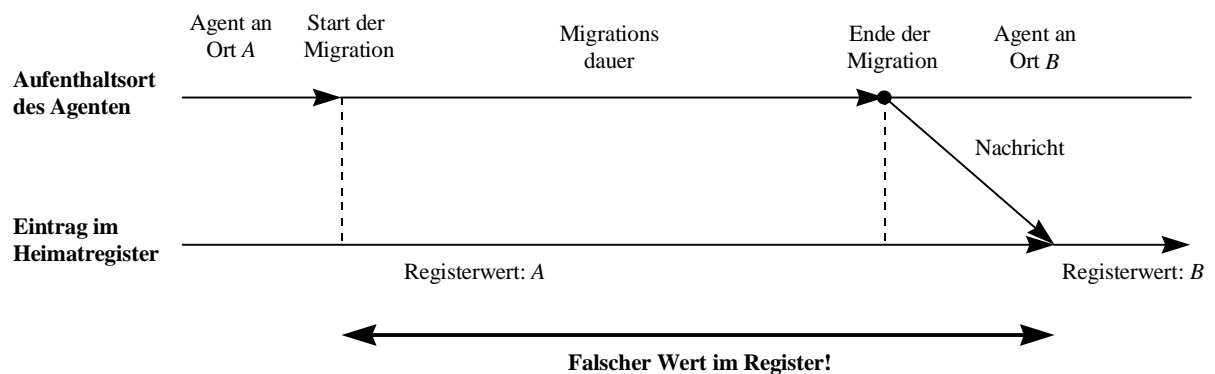


Abbildung 6-4: Aktualisierungsnachrichten 2. Version

- Das Senden einer Nachricht vor und nach der Migration verkürzt diese Zeitspanne auf die Zeit, die für den Versand zweier Nachrichten benötigt wird. Der alte Wert bleibt bis zum Eintreffen der 1. Nachricht erhalten. Diese ändert das Register auf einen Wert, der besagt, daß der Agent gerade migriert. Dieser Wert ist falsch von dem Zeitpunkt an, an dem die Migration abgeschlossen ist, bis zum Zeitpunkt des Eintreffens der 2. Nachricht. Die 2. Nachricht beinhaltet dann den neuen Aufenthaltsort des Agenten.

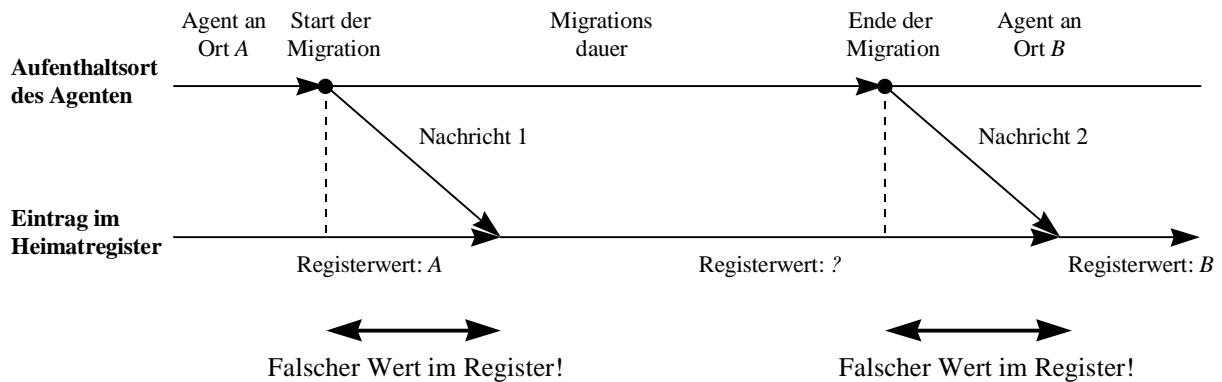


Abbildung 6-5: Aktualisierungsnachrichten 3. Version

Der Unterschied zwischen diesen Möglichkeiten wird immer deutlicher, je länger die Migration im Vergleich zum Senden einer Nachricht dauert. Dies ist natürlich auch von der Entfernung abhängig, die bei der Migration und beim Nachrichtenversand zu überwinden ist. Ich gehe hier davon aus, daß der Zeitbedarf für eine Migration höher ist, als der für das Senden einer Nachricht, da die Migration eines Agenten die wesentlich komplexere und aufwendigere Aufgabe ist.

Die letzte Möglichkeit bietet sich bei Anwendungen an, in denen Agenten häufig migrieren, da sonst die Zeit, in der ein falscher Wert im Heimatregister steht, über die Lebensdauer des Agenten gesehen, zu lang wird.

Das Senden der Aktualisierungsnachrichten kann sowohl vom Agenten selber, als auch automatisch von den Orten durchgeführt werden. Dies hängt davon ab, ob der Agent-Locator Dienst als Systemdienst oder als Anwendung implementiert werden soll. Die Verwaltung der Heimatregister der Agenten eines Ortes kann analog entweder vom Ort oder von einem (immobilen) Agenten geleistet werden.

Bei einer Anfrage an den Agent-Locator muß der Heimatort des gesuchten Agenten bekannt sein oder ermittelt werden können. An diesen Ort wird dann die Anfrage weitergeleitet. Ist der gesuchte Agent für den Agent-Locator Dienst registriert, so kann der Ort sofort den momentanen Aufenthaltsort des Agenten zurückmelden.

Fällt im Agentensystem ein Ort aus, so können alle Agenten, die dort ihr Heimatregister hatten, nicht mehr gefunden werden.

6.6.2 Bewertung

Netzpartitionierung

Hier sind wieder mehrere Fälle denkbar:

- Agent, Benutzer und Heimatregister sind in der gleichen Partition. In diesem Fall ist ein Auffinden des Agenten mit dem Agent-Locator problemlos möglich.

- Agent und Benutzer sind in verschiedenen Partitionen. Hier ist ein Auffinden des Agenten prinzipiell nicht möglich.
- Agent und Benutzer sind in der gleichen Partition, das Heimatregister in einer anderen. Da der Benutzer das Heimatregister nicht erreichen kann, kann auch der Aufenthaltsort des Agenten nicht ermittelt werden. Außerdem kann auch der Agent sein Heimatregister nicht mehr über Ortsänderungen informieren. Wenigstens wird die Partitionierung des Netzwerkes erkannt, da die entsprechenden Anfragen und Aktualisierungsnachrichten nicht zugestellt werden können.

Der letzte Punkt macht das Verfahren für Netzpartitionierungen anfällig. Da hier Benutzer und Agent in der gleichen Partition sind, sollte der Agent gefunden werden können.

Eine Verbesserungsmöglichkeit ist, statt eines Heimatregisters zwei oder drei pro Agent zu verwenden. Doch auch dies schützt nicht davor, daß es Trennungen geben kann, in der der Agent nicht gefunden wird, obwohl er in der gleichen Partition wie der Benutzer ist. Außerdem muß der Benutzer in diesem Fall nicht nur das Erste sondern auch die anderen Heimatregister kennen, damit er Anfragen an sie richten kann. Dies ist kein Problem, wenn der Benutzer des Agent-Locators der Besitzer des Agenten ist, da bei dessen Erzeugung die Positionen der Heimatregister an den Benutzer weitergegeben werden können, bzw. sogar von ihm festgelegt werden. Soll es jedoch möglich sein, nach beliebigen Agenten zu suchen, gibt es sowieso schon das Problem, wie man überhaupt das Heimatregister des gesuchten Agenten herausfindet. Eine Lösungsmöglichkeit ist, wie im Abschnitt 5.5 „Lokalisieren des Heimortes eines Agenten“ beschrieben, den Heimort im Namen des Agenten zu kodieren. Es ist jedoch nur möglich zwei oder drei Orte in einen Namen zu kodieren, wenn ein Agentennamen eine entsprechende Länge haben oder variabel lang sein darf. Ist dies nicht der Fall, wäre nur die Möglichkeit denkbar, die verschiedenen Heimatregister bei einem Name Service zu registrieren, bei dem sie bei einer Anfrage ermittelt werden können.

Insgesamt ist das Verfahren daher bei Netzpartitionierungen nur bedingt einsetzbar.

Die Verwendung von mobilen Clients ist möglich, da das Heimatregister eines Agenten frei wählbar ist. Damit können Agenten ihr Heimatregister an anderen Orten anlegen. Das Heimatregister auf einem mobilen Client zu haben, wäre nur dann sinnvoll, wenn dieser für die Lebenszeit des Agenten mit dem Netzwerk verbunden ist. Ansonsten sollte vermieden werden, Heimatregister auf mobilen Clients anzulegen, da diese, wenn sich der Client vom Agentensystem trennt, nicht mehr erreichbar sind und damit weder aktualisiert noch abgefragt werden können.

Korrektheit des Ergebnisses

Je nachdem, zu welchen Zeitpunkten Aktualisierungsnachrichten verschickt werden, sind im Heimatregister während einer Migration auch falsche Werte. Wird vor und nach der Migration eine Nachricht versendet, ist die Zeitspanne, in der falsche Werte im Register stehen, am geringsten (s.o.).

Kritisch ist jedoch, daß im Falle einer Netzpartitionierung, in der der Agent von Heimatregister und Benutzer abgetrennt wird, keine Aktualisierungsnachrichten mehr an das Heimatregister gesendet werden können. Die Partitionierung wird vom Heimatregister auch nicht bemerkt, so daß bei einer Anfrage ein falscher Wert an den Benutzer gegeben wird.

Abgesehen von diesem Fehlerfall, ist die Trefferwahrscheinlichkeit des Verfahrens gut.

Bei diesem Verfahren kann ein Benutzer folgende Antworten erhalten::

Antwort	Bedeutung
Agent gefunden	Das Heimatregister hat die Anfrage beantwortet.
HR nicht erreichbar	Das Heimatregister konnte nicht erreicht werden
HR nicht korrekt	Das angegebene Heimatregister ist nicht (mehr) vorhanden oder der Agent wurde nicht beim Agent-Locator angemeldet
Agent migriert gerade	Nur möglich, wenn pro Migration 2 Aktualisierungsnachrichten verschickt werden

Antwortzeit

Erfolgreiche Suche: Bei einer Anfrage müssen nur zwei Nachrichten gesendet werden. Eine, die die Anfrage an das Heimatregister meldet und eine, die die Antwort zurücksendet. Die Antwortzeit ist daher kurz.

Erfolglose Suche: Eine Suche ist hier erfolglos, wenn z.B. das Heimatregister nicht erreicht werden kann oder am angegebenen Ort kein Heimatregister für den Agenten existiert. Dies läßt sich jedoch schnell herausfinden.

Die Antwortzeit des Verfahrens ist insgesamt kurz.

Kosten zur Laufzeit

Für jede Migration im System muß mindestens eine Nachricht versendet werden. Falls, wie oben dargestellt, vor und nach einer Migration das Heimatregister benachrichtigt wird, sind es dementsprechend zwei Nachrichten pro Migration. Damit ergibt sich folgender Kommunikationsaufwand N im Zeitraum t mit m Migrationen und a Anfragen:

$$N_t = (2 \times a) + (2 \times m)$$

Außerdem muß jeder Ort für die Agenten, die ihr Heimatregister bei ihm haben, entsprechenden Speicherplatz reservieren. Beides sind vertretbare Aufwendungen.

Kosten der Implementierung

Folgende Programmteile müssen implementiert werden:

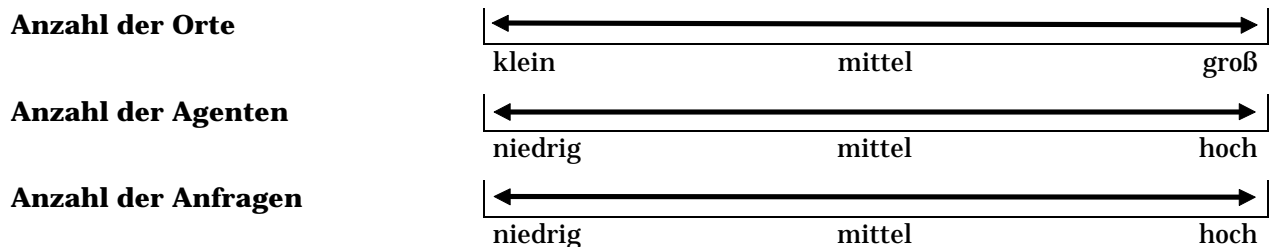
- Eine Erweiterung der Orte, die die Verwaltung der Heimatregister übernimmt, sowie die Aktualisierungsnachrichten versendet. Alternativ kann auch ein Agent programmiert werden, der die Heimatregister verwaltet und eine Erweiterung für Agenten, um die Aktualisierungsnachrichten zu versenden.
- Ein Programm, welches die Benutzeranfrage entgegennimmt, den Heimatort des gesuchten Agenten ermittelt, sofern er nicht bekannt ist und die Anfrage an ihn weiterleitet.

Der Implementierungsaufwand ist damit gering.

Skalierbarkeit

Das Verfahren ist gut skalierbar. Kommunikationsaufwand und Speicherbedarf steigen proportional mit der Anzahl der Agenten. Außerdem wird der Ressourcenverbrauch über das gesamte Agentensystem verteilt. Es muß nur verhindert werden, daß einzelne Orte zu stark belastet werden. Dies geschieht aber nur dann, wenn ein Ort von vielen Agenten als Platz für ihr Heimatregister gewählt wird. Durch das Setzen einer Obergrenze für die Anzahl der Heimatregister pro Ort läßt sich dieses Problem jedoch lösen. Diese Obergrenze könnte von verschiedenen Faktoren abhängen z.B. von der Größe, der dem Ort zur Verfügung stehenden Ressourcen oder der Anzahl der Agenten, die am Ort erzeugt werden.

Das Verfahren eignet sich für Agentensysteme mit folgenden Eigenschaften:



Sicherheit/Datenschutz

Wird das Verfahren so implementiert, daß nicht die Orte, sondern die Agenten die Aktualisierungsnachrichten an die Heimatregister senden, muß verhindert werden, daß ein Agent das Heimatregister eines anderen verändert. Ein „böser“ Agent könnte versuchen, durch eine falsche Aktualisierungsnachricht das Heimatregister eines anderen Agenten zu verfälschen. Durch eine Art Paßwort, das nur der zum Heimatregister „passende“ Agent kennt und welches bei jeder Aktualisierungsnachricht mit versendet werden muß, könnte dies verhindert werden.

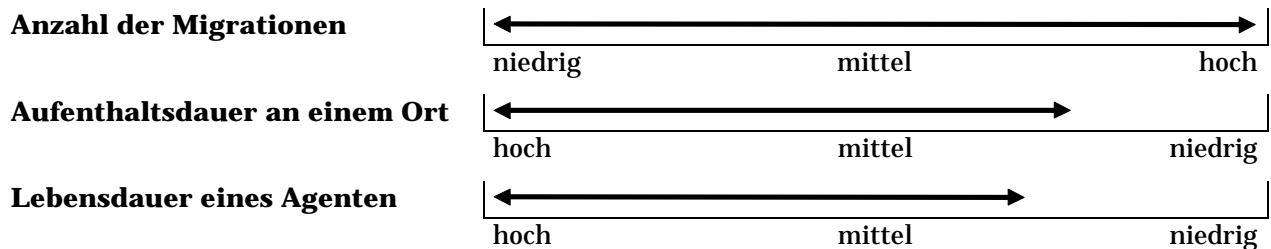
Auch die Abfrage des Heimatregisters sollte nur für dafür berechnigte Agenten oder Benutzer möglich sein.

Eignung für die Beispiel-Anwendungen

Das Verfahren ist für alle Anwendungen mit Ausnahme des Netzwerkmanagements geeignet. Beim Netzwerkmanagement migrieren die Agenten sehr schnell. Dies führt dazu, daß eine Aktualisierung des Heimatregisters schon veraltet sein kann, wenn sie dort eintrifft.

Man muß jedoch bei diesem Verfahren immer bedenken, daß das Heimatregister eines gesuchten Agenten bekannt sein muß oder ermittelt werden kann.

Das Verfahren eignet sich für Anwendungen, bei denen die Agenten folgende Eigenschaften haben:



6.7 Name Service

6.7.1 Verfahren

Wenn im Agentensystem ein globaler Name Service existiert, kann ein Agent-Locator einfach realisiert werden. Agenten könnten sich beim Name Service registrieren lassen und ihren Aufenthaltsort bei jeder Migration aktualisieren. Alternativ kann diese Aktualisierung auch automatisch von den Orten durchgeführt werden. In diesem Fall wäre wieder eine Markierung der Agenten sinnvoll, die den Agent-Locator Dienst in Anspruch nehmen wollen. Für den Zeitpunkt der Aktualisierung gibt es wieder die drei, beim Verfahren des Heimatregisters bereits vorgestellten, Möglichkeiten.

Der momentane Aufenthaltsort eines Agenten läßt sich bei diesem Verfahren einfach durch eine Anfrage an den Name Service ermitteln.

Bei diesem Verfahren existiert eine zentrale Stelle, die alle Informationen über den Aufenthaltsort der Agenten besitzt. Daß diese Stelle nicht wirklich zentral, sondern repliziert im Agentensystem vorhanden sein muß, ist Voraussetzung für die Sicherheit der Daten, sowie für die Funktion des Dienstes bei Netzpartitionierungen. Die Replikate sollten jeweils im Vollbesitz aller Informationen sein, da ein hierarchischer Aufbau Probleme bei einer Partitionierung des Agentensystems mit sich bringt und Agenten schwer hierarchisch zu ordnen sind. Alle mir bekannten Name Service Dienste sind jedoch in der einen oder anderen Form hierarchisch aufgebaut.

Eine wichtige Eigenschaft des Name Service, in Bezug auf den Agent-Locator, ist vor allem die Möglichkeit, schnell viele Einträge hinzufügen, ändern und löschen zu können. Diese Operationen treten hier deutlich häufiger auf, als das Lesen eines Eintrages. Diese Anforderung ist eher untypisch für einen Name Service. In einem Name Service werden normalerweise nur Informationen abgelegt, die sich selten ändern, aber häufig ausgelesen werden. Bestehende Name Service Dienste sind daher auf einen lesenden Zugriff hin optimiert. Problematisch ist in diesem Zusammenhang auch das Caching der Daten bei Anfragen an den Name Service, wie es z.B. beim Domain Name Service (DNS) praktiziert wird. Hier werden die vom Name Service gelieferten Daten zwischengespeichert, um spätere Anfragen sofort beantworten zu können. Dies ist bei einem Einsatz des Name Service für den Agent-Locator äußerst unerwünscht, da die Information sich in der Zwischenzeit verändert haben können. Problematisch ist auch die Geschwindigkeit mit der Änderungen an die Replikate mitgeteilt werden. Im Falle einer Netzpartitionierung sollten alle Replikate möglichst auf dem gleichen (aktuellen) Stand sein. Da existierende Name Service Dienste (z.B. X500, DNS) keine entsprechende Funktionalität besitzen, scheint es notwendig, ein eigenes System für die Implementierung des Agent-Locator zu entwickeln. Da der Aufwand hierfür jedoch hoch ist, ist der Einsatz eines solchen Dienst nur dann sinnvoll, wenn er auch von anderen Anwendungen im Agentensystem verwendet wird. Allerdings sind dann die Anforderungen dieser Anwendungen an den Dienst ebenfalls zu beachten.

Da ein Name Service ein sehr universeller Dienst ist und für viele Anwendungen Vorteile bringt oder sogar von vielen Anwendungen benötigt wird, wäre es sinnvoll im Agentensystem einen entsprechenden Dienst zur Verfügung zu stellen.

6.7.2 Bewertung

Netzpartitionierung

Das Verfahren funktioniert bei Netzpartitionierung nur dann, wenn ein Replikat des Name Service in der gleichen Partition ist, wie der Benutzer. Die Funktion des Agent-Locators hängt hier also davon ab, wie oft die Informationen des Name Service im Agentensystem repliziert werden. Es läßt sich jedoch immer ein Fall konstruieren, in dem der Name Service von einer Partition aus nicht mehr zu erreichen ist (außer jeder Ort besitzt ein Replikat). Daher ist das Verfahren bei Partitionierungen ebenfalls nur bedingt einsetzbar.

Mobile Clients stellen für dieses Verfahren kein Problem dar. Replikate des Name Service sollten natürlich nicht auf mobilen Clients angelegt werden.

Korrektheit des Ergebnisses

Wie beim Heimatregister sind auch hier falsche Werte möglich. Das resultiert hier allerdings nicht nur aus dem Zeitpunkt, wann die Aktualisierungsnachrichten versendet werden, sondern auch daraus, wie lange der Name Service benötigt, um alle Replikate zu aktualisieren. Wird eine Anfrage an ein Replikat gestellt, welches

von der letzten Migration des Agenten noch nicht informiert ist, wird ein falscher Wert zurückgegeben.

Auch kann im Falle einer Netzpartitionierung, in der der Agent von Name Service und Benutzer abgetrennt wird, keine Aktualisierungsnachricht mehr an den Name Service versendet werden, so daß, bei einer Anfrage ein falscher Wert an den Benutzer gegeben wird.

Insgesamt ist die Trefferwahrscheinlichkeit des Verfahrens jedoch gut.

Bei diesem Verfahren kann ein Benutzer folgende Antworten erhalten::

Antwort	Bedeutung
Agent gefunden	Der Name Service konnte die Anfrage beantworten.
NS nicht erreichbar	Der Name Service konnte nicht erreicht werden.
Keine Information	Der Name Service hat keinen Eintrag für den Agenten.
Agent migriert gerade	Nur möglich, wenn pro Migration 2 Aktualisierungsnachrichten verschickt werden.

Antwortzeit

Erfolgreiche Suche: Bei einer Anfrage müssen nur zwei Nachrichten gesendet werden. Die Antwortzeit ist daher kurz. Dies gilt jedoch nur, wenn jedes Replikat des Name Service über sämtliche Informationen verfügt. Ist dies nicht der Fall, so kommt hier noch der Aufwand für Name Service hinzu, den entsprechenden Eintrag zu finden. Dazu können bei einem hierarchisch organisierten Name Service mehrere Nachrichten nötig sein.

Erfolglose Suche: Die Suche ist erfolglos, wenn der Name Service nicht erreichbar ist oder dort kein Eintrag über den Agenten vorhanden ist.

Die Antwortzeit des Verfahrens ist daher kurz.

Kosten zur Laufzeit

Für jede Migration im System muß eine bzw. zwei Nachricht(en) versendet werden. Zusätzlich kommt hier noch der Speicherbedarf des Name Service dazu. Ebenso der Kommunikationsaufwand, der nötig wird, um die Replikate, die der Name Service benötigt zu aktualisieren. Die Kosten zur Laufzeit hängen hier also ganz wesentlich mit davon ab, wie der Name Service implementiert ist. Ich gehe hier davon aus, daß bei jeder Migration, jedes Replikat benachrichtigt werden muß, und daher jedes Replikat bei einer Anfrage die gewünschte Information liefern kann. Damit ergibt sich folgender Kommunikationsaufwand N im Zeitraum t mit m Migrationen und a Anfragen:

$$N_t = (2 \times a) + (2 \times m) + (m \times (\text{Anzahl der Replikate}))$$

Dieser Kommunikationsaufwand ergibt sich, da für jede Anfrage zwei Nachrichten benötigt werden, pro Migration zwei Aktualisierungsnachrichten versendet werden und jedes Replikat bei einer Migration benachrichtigt werden muß. Der Kommunikationsaufwand ist also im Vergleich zum Heimatregister um so größer, je mehr

repliziert wird. Gibt es keine Replikate, so ist der Kommunikationsaufwand beider Verfahren gleich.

Der Speicherbedarf an den Orten, an denen Replikate des Name Services sind, ist hoch. Da dort Informationen über alle im System vorhandenen Agenten gespeichert werden müssen. Die Orte, die solche Replikate bekommen sollen, sollten daher über genügend Speicher und wegen der vielen Aktualisierungsnachrichten auch über einen Netzanschluß mit hoher Bandbreite verfügen.

Kosten der Implementierung

Der Implementierungsaufwand hängt wesentlich davon ab, ob ein Name Service mit den geforderten Eigenschaften bereits im Agentensystem existiert oder nicht. Ist ein solcher vorhanden, so ist der Implementierungsaufwand gering. Es muß nur ein Programm geschrieben werden, welches den Dialog mit dem Benutzer führt und den Name Service abfragt. Muß jedoch zusätzlich der Name Service entwickelt werden, ist der Implementierungsaufwand hoch.

Skalierbarkeit

Das Verfahren läßt sich gut skalieren. Bei einer Vergrößerung des Agentensystem müssen bestenfalls mehr Replikate angelegt werden.

Das Verfahren eignet sich für Agentensysteme mit folgenden Eigenschaften:

Anzahl der Orte	
Anzahl der Agenten	
Anzahl der Anfragen	

Sicherheit/Datenschutz

Die Abfrage des Name Service sollte aus Datenschutzgründen nur für das entsprechende Client-Programm möglich sein. Keinesfalls sollte die Möglichkeit bestehen, daß ein Agent Auskunft bezüglich des Aufenthaltsortes anderer Agenten vom Name Service erhält. Daher wäre eine Authentifikation des Client-Programmes gegenüber dem Name Service wünschenswert.

Für das Aktualisieren des Name-Service gilt das gleiche wie beim Heimatregister. Auch hier sollte verhindert werden, daß Agenten die Name Service Einträge anderer Agenten manipulieren können.

Eignung für die Beispiel-Anwendungen

Auch hier gilt das gleiche wie beim Heimatregister. Außer für das Netzwerkmanagement ist das Verfahren für alle Anwendungen geeignet.

Das Verfahren eignet sich für Anwendungen, bei denen die Agenten folgende Eigenschaften haben:



6.8 Sonstige Verfahren

Hier sollen noch kurz einige Möglichkeiten angesprochen werden, die zwar denkbar sind, jedoch für eine Realisierung des Agent-Locators nicht in Frage kommen, da sie die Funktion des Agentensystems zu stark beeinträchtigen oder andere gravierende Nachteile haben. Die Verfahren werden deshalb auch nicht ausführlich bewertet.

- Eine Möglichkeit wäre, daß alle Agenten vor ihrem Start ihre Route und die Zeit, die sie an jedem Ort bleiben, festlegen müssen. Der Heimatort des Agenten hat damit immer das exakte Wissen, wo sich der Agent aufhält. Die Route könnte bei Veränderungen auch vom Agenten aktualisiert werden. Für die Einhaltung der Aufenthaltsdauer an einem Ort, wären jedoch synchrone Uhren im Agentensystem nötig.
- Falls keine synchronen Uhren zur Verfügung stehen, könnte der Agent vor seinem Start auch nur die Reihenfolge festlegen, in der er die Orte besucht. Bei einer Suche können diese Orte dann nacheinander abgefragt werden, ob der Agent gerade dort ist. Die Ortsliste kann dabei vorwärts oder rückwärts abgearbeitet werden. Speichern die Orte Informationen, welche Agenten in der letzten Zeit bei ihnen waren, so könnte man auch mit einer „Binary Search“-Methode die Orte abfragen. Die Route könnte bei Änderungen vom Agenten auch aktualisiert werden.
- Denkbar wäre auch, daß die Agenten nicht frei im System migrieren dürfen, sondern nach jeder Migration wieder zum Heimatort zurückkehren müssen. Die Route des Agenten ist dann sternförmig mit dem Heimatort als Zentrum. Schreibt man nun noch eine Zeitspanne vor, die angibt, wie lange sich ein Agent höchstens vom Heimatort entfernen darf, so kann der Agent innerhalb dieser Zeitspanne gefunden werden. Die Antwort des Agent-Locators beinhaltet dann nicht den Ort des gesuchten Agenten, sondern nur die Information, daß er jetzt am Heimatort ist. Die Antwortzeit ist die Zeitspanne, die der Agent sich vom Heimatort entfernen darf.
- Man könnte auch zu einem bestimmten Zeitpunkt Informationen über alle Agenten und deren Aufenthaltsorte sammeln und diese Informationen im

Netzwerk verfügbar machen. Dies würde in der Realität der Herstellung eines Telefonbuches entsprechen, welches zum Zeitpunkt der Drucklegung aktuell ist, danach aber zunehmend falsche Daten enthält. Dummerweise ändern mobile Agenten häufiger ihre „Adresse“ als Menschen.

7 Zusammenfassende Bewertung

Die folgende Tabelle bietet einen Überblick über alle vorgestellten Verfahren und ihre Bewertungen. Die Bewertung erfolgt in den 4 Stufen + +, +, -, - -. Die beste Bewertung ist + +.

	Broad- cast	Period. BC	Energie- konzept	Migrat. route	Straßen- sperren	Heimat- register	Name Service
Partitionierung	+ +	+	-	- -	+ +	-	-
Korrektheit	+ +	-	+ +	+ +	+ +	+	+
Antwortzeit	+	+ +	variabel	-	- -	+ +	+ +
Skalierbarkeit	-	- -	+ +	-	-	+ +	+
Laufzeitkosten	-	- -	-	-	-	+ +	+
Impl. Kosten	+ + ¹⁾	+	- -	- -	+	+	+ + ²⁾
Datenschutz	+	- -	+ +	- -	+ +	+ +	+
Anwendungen	4	(1)	1, 2, 3, 4	1, 2	-	1, 2, 3	1, 2, 3

1) Unter der Bedingung, daß ein Broadcastmechanismus im Agentensystem existiert

2) Unter der Bedingung, daß ein Name Service mit den nötigen Eigenschaften vorhanden ist.

Hier noch Erläuterungen zu einigen Tabelleneinträgen:

- Broadcast und Straßensperren arbeiten bei Netzpartitionierung genauso gut oder schlecht wie sonst auch. Die Periodischen Broadcasts schneiden hier schlechter ab, da nach einer Partitionierung noch Informationen über nicht mehr erreichbare Agenten weitergegeben werden können.
- Heimatregister und Name Service sind bei der Partitionierung beide mit - bewertet. Der Name Service schneidet, durch die Möglichkeit die Informationen zu replizieren, natürlich besser ab als das Heimatregister. Er arbeitet bei Partitionierungen jedoch mit Sicherheit schlechter als der Periodische Broadcast. Daher „fehlt“ hier eine weitere Abstufung bei der Bewertung.
- Das Energiekonzept schneidet bei der Partitionierung genauso schlecht ab, wie das Heimatregister, da in beiden Fällen Benutzer, Agent und Heimatort bzw. Heimatregister in einer Partition sein müssen.
- Daß Heimatregister und Name Service bei der Korrektheit schlechter abschneiden, liegt an der Möglichkeit, daß je nach verwendeter Aktualisierungsmethode, ein falscher Wert im Register, bzw. im Name Service vorhanden ist. Durch das Aktualisieren dieses Wertes vor und nach einer Migration kann diese „Ausfallzeit“ jedoch minimiert werden. Der Name Service schneidet hier etwas schlechter ab, da hier nach einer Aktualisierungsnachricht erst noch alle Replikate aktualisiert werden müssen.
- Die Antwortzeit bei dem Verfahren Straßensperren ist je nachdem wann der Agent von den Detektiven gefunden wird sehr unterschiedlich. Da hier jedoch von einer „worst case“ Situation ausgegangen wird, ist er mit - - bewertet. Das

Verfahren Migrationsroute schneidet etwas besser ab, da man nicht mit einem Timeout arbeiten muß, um das Scheitern einer Anfrage zu erkennen.

- Die Antwortzeit des Energiekonzeptes ist je nach Energiemenge und Energieverbrauch für jeden Agenten unterschiedlich. Sie kann durch eine entsprechende Energiezuteilung für einen Agenten variabel gestaltet werden.
- Die Skalierbarkeit des Heimatregisters ist deshalb besser bewertet als die des Name Service, da dieser bei einem Anwachsen des Agentensystems unter Umständen umkonfiguriert werden muß (mehr Replikate, andere Orte für die Replikate). Das Heimatregisterverfahren skaliert sich „automatisch“.
- Der Broadcast schneidet in den Laufzeitkosten schlechter ab als das Heimatregister. Vergleicht man den Kommunikationswand N der beiden Verfahren, so ergibt sich, daß der Broadcast teurer wird, wenn die Anzahl der Orte größer wird als $3 + (2 \times m) / a$. Daraus folgt, je öfter der Agent-Locator verwendet wird, desto teurer wird der Broadcast im Vergleich zum Heimatregister. Wird er selten benutzt, kann der Broadcast aber auch billiger sein.
- Für die Laufzeitkosten des Name Service gilt das gleiche, jedoch schneidet er, im Vergleich zum Heimatregister, noch einmal um den Faktor $m \times (\text{Anzahl der Migrationen})$ schlechter ab.
- Der Name Service ist beim Datenschutz eine Stufe schlechter bewertet als das Heimatregister, da hier viele Informationen gesammelt vorliegen, die beim Heimatregister über das gesamte Agentensystem hinweg „versteckt“ sind. Ist der Name Service gegen unerlaubte Zugriffe durch entsprechende Verfahren abgesichert, schneidet er jedoch gleich gut oder sogar besser ab als das Heimatregister.

Aus dieser Tabelle wird die vielseitige Anwendbarkeit des Heimatregisters und des Name Service deutlich. Sie kommen mit allen Anwendungen zurecht, bei denen die Agenten nicht sehr schnell migrieren. Bei Anwendungen, bei welchen dies der Fall ist, gibt es aber ohnehin kein optimales Verfahren. Der Broadcast und das Energiekonzept sind hier die einzigen Möglichkeiten, um mit der Geschwindigkeit der Agenten mithalten zu können. Beide Verfahren haben jedoch andere Nachteile.

Der größte Nachteil des Heimatregisterverfahrens, ist die Tatsache, daß das Heimatregister eines Agenten bekannt sein muß. Zwei Lösungsvorschläge für dieses Problem wurden im Abschnitt 5.5: „Lokalisieren des Heimatortes eines Agenten“ erläutert. Wobei die Möglichkeit, das Heimatregister im Namen des Agenten zu kodieren, die elegantere Methode ist. Läßt die Architektur des Agentensystem es jedoch nicht zu, den Agenten entsprechende Namen zu geben und ist auch kein Name Service vorhanden, hat das Verfahren die Einschränkung, daß nur nach Agenten gesucht werden kann, von denen man weiß, an welchem Ort ihr Heimatregister ist. Dies sind vor allem diejenigen Agenten, die man selber besitzt.

Ein weiterer Nachteil ist die Anfälligkeit bei Netzpartitionierungen, die sich auch durch mehrere Heimatregister nicht völlig entschärfen läßt.

Der Name Service ist dagegen nur dann eine gute Option, wenn ein solcher Dienst entweder im Agentensystem verfügbar ist oder bereits für einen anderen Zweck entwickelt wurde und „nur“ noch ins Agentensystem zu integrieren ist. Die Entwicklung eines eigenen Dienstes, nur für den Agent-Locator, rentiert sich nur, wenn der Agent-Locator für die Anwendung wichtig ist.

Auch der Name Service hat den Nachteil anfällig auf Netzpartitionierungen zu reagieren. Durch eine gut durchdachte Platzierung der Replikate im Agentensystem, läßt sich das Problem jedoch abschwächen.

Der Broadcast ist eine Alternative, wenn das Agentensystem eher klein ist und der Agent-Locator selten verwendet wird. Bei Agentensystemen mit wenigen Orten, ist er auch für alle Anwendungen verwendbar. Der Broadcast hat den großen Vorteil, auch mit schnell migrierenden Agenten schritthalten zu können. Leider ist er nur schlecht skalierbar. Außerdem ist der Aufwand für eine Anfrage hoch.

8 Kombination der Verfahren

Durch die Kombination einzelner Verfahren lassen sich häufig Verbesserungen für den Agent-Locator erreichen. In diesem Kapitel werden einige solcher Kombinationen vorgestellt.

8.1 Heimatregister & Name Service

Diese Kombination wurde schon unter „Allgemeine Probleme“ erläutert. Der Name Service wird hier verwendet, um das (oder die) Heimatregister eines Agenten herausfinden zu können. Der größte Vorteil ist, daß damit nun auch ohne das Wissen über das Heimatregisters jeder Agent gefunden werden kann. Außerdem ist hier die Möglichkeit gegeben, mehrere Heimatregister für einen Agenten zu verwalten.

Gegenüber dem reinen Name Service hat die Kombination der Verfahren den Vorteil, daß die Informationen im Name Service seltener aktualisiert werden müssen.

Nachteil ist die höhere Anfälligkeit bei Partitionierungen. Der Benutzer darf jetzt weder vom Name Service, noch von den Heimatregistern des gesuchten Agenten abgeschnitten sein. Die Antwortzeit ist bei einer Kombination höher, da zuerst der Name Service über das Heimatregister des Agenten abgefragt werden muß und dann noch die Anfrage ans Heimatregister nötig ist. Die Laufzeitkosten sowie die Implementierungskosten sind ebenfalls höher.

Übersicht über die Vor- und Nachteile gegenüber dem Heimatregister:

Vorteile	Nachteile
<ul style="list-style-type: none"> – Heimatregister können gefunden werden – Mehrere Heimatregister möglich 	<ul style="list-style-type: none"> – Anfälliger bei Partitionierungen – Höhere Antwortzeit – Höhere Implementierungskosten – Höhere Laufzeitkosten

8.2 Heimatregister & Broadcast

Teilt man das Agentensystem in kleine Teile aus jeweils nur einigen Orten ein, so könnte innerhalb eines solchen Teilgebietes ein Agent durch einen Broadcast gefunden werden. Das Heimatregister gibt dann nicht mehr den Aufenthaltsort des Agenten an, sondern nur noch in welchem Teil des Agentensystems er sich befindet. Es muß nur dann aktualisiert werden, wenn der Agent von einem Teil des Agentensystems in einen anderen migriert. Bei einer Anfrage muß dann zuerst das Heimatregister abgefragt werden. Danach muß durch einen Broadcast in dem Teil des Agentensystems, in dem sich der Agent befindet, sein genauer Aufenthaltsort ermittelt werden.

Diese Kombination senkt bei einer selteneren Nutzung des Agent-Locators die Laufzeitkosten im Vergleich zum reinen Heimatregister Verfahren, da weniger Aktualisierungsnachrichten nötig sind. Dafür steigt der Aufwand bei einer Anfrage erheblich an. Wird der Agent-Locator häufig verwendet, wird dadurch die Einsparung bei den Aktualisierungsnachrichten schnell wieder kompensiert.

Übersicht über die Vor- und Nachteile gegenüber dem Heimatregister:

Vorteile	Nachteile
<ul style="list-style-type: none"> – Weniger Aktualisierungsnachrichten 	<ul style="list-style-type: none"> – Höhere Antwortzeit – Höhere Implementierungskosten – Höhere Laufzeitkosten

8.3 Name Service & Broadcast

Kann man bei einer Anfrage den Name Service nicht mehr erreichen, so liegt eine Partitionierung vor. Gibt es im Agentensystem viele Replikat des Name Services, so kann angenommen werden, daß die verbleibenden Partitionen relativ klein sind. In diesem Fall kann innerhalb einer Partition via Broadcast versucht werden, einen Agenten zu finden. Mit dieser Kombination kann man daher das Verhalten bei Partitionierungen, im Vergleich zum reinen Name Service, verbessern.

Übersicht über die Vor- und Nachteile gegenüber dem Name Service:

Vorteile	Nachteile
<ul style="list-style-type: none"> – Besser bei Partitionierungen 	<ul style="list-style-type: none"> – Höhere Implementierungskosten – Höhere Antwortzeit bei Partitionierungen – Höhere Laufzeitkosten bei Partitionierungen

8.4 Energiekonzept & Straßensperren

Bis die Energie eines Agenten ausgeht und er sich wieder an seinen Heimatort melden muß, kann je nach Energieladung des Agenten eine längere Zeitspanne vergehen. In dieser Zeit ist der Agent nicht aufzufinden. Möchte man trotzdem versuchen, ihn während seiner Abwesenheit vom Heimatort zu lokalisieren, so kann man nach dem Straßensperren Verfahren vorgehen. Dies ist jedoch nur dann sinnvoll, wenn die Detektive auch eine Chance haben, den Agenten zu finden, bevor dieser sich wieder an seinem Heimatort meldet.

Übersicht über die Vor- und Nachteile gegenüber dem Energiekonzept:

Vorteile	Nachteile
<ul style="list-style-type: none"> – Es besteht die Chance Agenten zu finden, die gerade nicht an ihren Heimatorten sind. 	<ul style="list-style-type: none"> – Höhere Implementierungskosten

8.5 Migrationsroute & Energiekonzept

Die Migrationsroute des Agenten wird an den Orten gespeichert. Der Agent besitzt darüber hinaus eine bestimmte Energiemenge, nach deren Verbrauch er terminiert wird. Um neue Energie zu erhalten, muß er an seinem Heimatort zurückkehren. Sobald er an seinem Heimatort angekommen ist, kann die bisherige Route des Agenten gelöscht werden. Der Vorteil des Verfahrens ist, daß die Länge der Migrationsroute beschränkt werden kann. Damit wird auch die Antwortzeit geringer.

Übersicht über die Vor- und Nachteile gegenüber dem Verfolgen der Migrationsroute:

Vorteile	Nachteile
<ul style="list-style-type: none"> – Kürzere Antwortzeiten – Es müssen weniger Routeninformationen gespeichert werden 	<ul style="list-style-type: none"> – Höhere Implementierungskosten

8.6 Migrationsroute & Straßensperren

Diese Kombination wäre möglich, wenn ein Detektiv die „Spur“ des gesuchten Agenten verliert. In diesem Fall könnte er Detektive erzeugen, um entweder die Route oder den Agenten wiederzufinden. Wird ein Ort gefunden, an dem der Agent gewesen ist und der noch nicht besucht wurde, kann die „Verfolgung“ von diesem Ort aus weitergehen.

Voraussetzung wäre, daß bei den Orten auch jeweils die Anzahl der Migrationen gespeichert wird, die der gesuchte Agent jeweils schon zurückgelegt hat. Damit kann verhindert werden, daß an einem bereits besuchten Ort noch einmal gesucht wird. Außerdem kann damit, falls mehrere Orte gefunden werden, derjenige als neuer Startpunkt bestimmt werden, an dem der gesuchte Agent zuletzt gewesen ist.

Übersicht über die Vor- und Nachteile gegenüber dem Verfolgen der Migrationsroute:

Vorteile	Nachteile
<ul style="list-style-type: none"> – Besser bei Partitionierungen – Besser beim Ausfall von Orten 	<ul style="list-style-type: none"> – Höhere Implementierungskosten

8.7 Straßensperren & Broadcast

Für diese Kombination ist es notwendig, das Agentensystem in Teile aus jeweils nur wenigen Orten aufzuteilen. In jedem Teil muß die Möglichkeit des Broadcasts gegeben sein.

Das Straßensperrenverfahren kann dann dadurch verbessert werden, daß ein Detektiv, wenn er in einen anderen Teil des Agentensystems migriert, dort mit Hilfe eines Broadcasts nach dem Agenten suchen kann. Durch diese Kombination kann sich die Antwortzeit verkürzen, da der Detektiv nun mehrere Orte gleichzeitig abfragen kann.

Übersicht über die Vor- und Nachteile gegenüber den Straßensperren:

Vorteile	Nachteile
<ul style="list-style-type: none"> – Bessere Antwortzeit – Niedrigere Laufzeitkosten 	<ul style="list-style-type: none"> – Höhere Implementierungskosten

8.8 Heimatregister & Verfolgen der Migrationsroute

Anstatt das Heimatregister bei jeder Migration zu aktualisieren, könnte man dies auch erst nach mehreren Migrationen tun. Zwischen den Aktualisierungen wird die Migrationsroute des Agenten bei den Orten gespeichert. Bei einer Anfrage muß dann zuerst das Heimatregister befragt werden. Danach muß überprüft werden, ob der Agent vom dort gespeicherten Ort schon weiter migriert ist. Ist dies der Fall, muß die Route des Agenten bis zu dessen jetzigen Aufenthaltsort weiterverfolgt werden.

Dadurch kann die Antwortzeit im Vergleich zum Heimatregister erheblich ansteigen. Auch ist der Implementierungsaufwand hoch.

Vorteil der Kombination ist, daß jetzt weniger Aktualisierungsnachrichten ans Heimatregister gesendet werden müssen. Im Vergleich zu den entstehenden Nachteilen ist dies jedoch kein Gewinn.

Übersicht über die Vor- und Nachteile gegenüber dem Heimatregister:

Vorteile	Nachteile
<ul style="list-style-type: none"> – Weniger Aktualisierungsnachrichten 	<ul style="list-style-type: none"> – Anfälliger bei Partitionierungen – Höhere Antwortzeit – Höhere Implementierungskosten – Höhere Laufzeitkosten

9 Prototypische Implementierung

9.1 Kurze Einführung in MOLE

MOLE (engl.: Maulwurf, auch im Sinne von Spion) ist ein Projekt der Abteilung Verteilte Systeme des Instituts für Parallele und Verteilte Höchstleistungsrechner der Universität Stuttgart. Es baut auf der Diplomarbeit von Fritz Hohl [Hohl95] auf.



Abbildung 9-1:
MOLE Symbhol

Bei MOLE sind sowohl die Agenten, als auch das Agentensystem in der Programmiersprache JAVA von SUN Microsystems geschrieben [Gosling95], [Tutorial]. JAVA hat viele Eigenschaften, die es als Sprache für Agentensysteme interessant macht. Vor allem Sicherheitsfeatures machen JAVA attraktiv [SecurityStory]. JAVA ist objektorientiert, portabel und außerdem frei verfügbar. JAVA Quellcode wird durch einen Compiler in stackorientierten JAVA Bytecode übersetzt, der dann interpretiert wird. Dadurch, daß im Bytecode alle wesentlichen JAVA-Informationen, wie z.B. Typinformationen vorliegen, ist es möglich den Bytecode vor seiner Ausführung verschiedenen Prüfungen zu unterziehen.

„Locations“ sind die Orte im MOLE-System. Sie sind immer auf einer „Engine“ Zuhause. Eine Engine ist hier der JAVA-Interpreter, der in einem normalen Betriebssystemprozeß läuft. Eine Engine kann mehrere Locations haben. An jeder Location befindet sich ein stationärer Agent, der „Local Agent“, der die Location gegenüber Agenten auf anderen Locations vertritt.

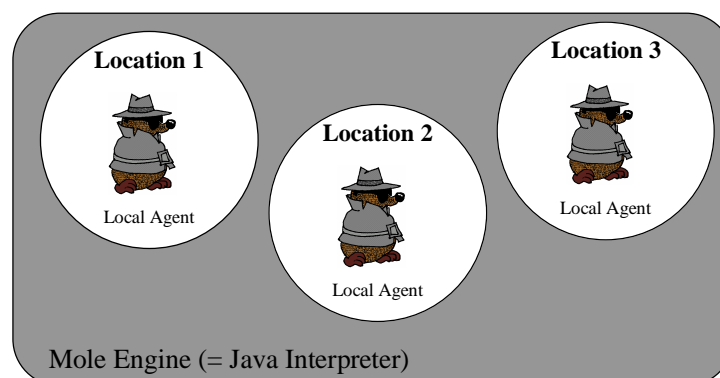


Abbildung 9-2: MOLE Engine

Kommunikation zwischen Agenten ist über Messages, und Remote Procedure Calls (RPC) möglich.

Agenten, die einen Dienst im Agentensystem anbieten wollen, können sich bei der Location, an der sie sich befinden, registrieren lassen, wobei sie einen Namen für ihren Dienst angeben müssen. Agenten, die einen bestimmten Dienst in Anspruch

nehmen wollen, können dessen Namen bei der Location erfragen und erhalten dann den Namen des Diensteanbieters. Jetzt können sie die Methoden des Anbieters, je nach Implementierung, durch RPC oder Nachrichten aufrufen.

Die Agentennamen sind in MOLE 8 Bytes lang. In der aktuellen Implementierung wird jeder Location bei ihrem Start ein bestimmter Namensraum zugewiesen. Diese Namensräume müssen natürlich disjunkt sein. Es gibt bisher noch keinen Mechanismus, der den Locations erlaubt Namensräume nachzufordern. Die Namen für die Agenten werden bisher sehr einfach vergeben: Jede Location besitzt einen Zähler, der mit der unteren Grenze des Namensraumes initialisiert und bei der Vergabe eines Agentennamens jeweils um eins erhöht wird.

Für diese Studienarbeit wurde MOLE in der Version Alpha 1.0 verwendet. Diese benötigt das Java Developers Kit (JDK) in der Version 1.02 sowie das JAVA Serialization Package.

9.2 Auswahl eines Verfahrens

Vor allem drei der beschriebenen Verfahren waren für die Implementierung eines Agent-Locators geeignet: Broadcast, Name Service und Heimatregister. Aus diesen drei wird nun eines für die Implementation ausgewählt:

Broadcast

Der Broadcast eignet sich vor allem für kleine Agentensysteme. Da MOLE auch im Internet und mit vielen Orten und Agenten einmal funktionieren soll, ist der Einsatz des Broadcast-Verfahrens nicht sinnvoll.

Für die jetzt bestehende Testumgebung im LAN, der Abteilung Verteilte Systeme am IPVR, wäre die Anwendung des Verfahrens jedoch möglich. Vor allem wenn der Agent-Locator primär als Instrument zur Fehlersuche verwendet werden soll. Leider besitzt MOLE selber keinen Broadcast Mechanismus, jedoch sollte es nicht allzu schwierig sein, für die Testumgebung einen zu implementieren.

Name Service

Leider gibt es in MOLE keinen Name Service; und einen mit den geforderten Eigenschaften zu entwickeln, würde den Rahmen dieser Studienarbeit sprengen. Ansonsten wäre der Name Service ein praktikables Verfahren für die Implementierung des Agent-Locators für MOLE.

Heimatregister

Das Heimatregisterverfahren ist vielseitig einsetzbar und gut skalierbar. Außerdem verursacht es nur geringe Kosten zur Laufzeit und ist relativ einfach zu implementieren.

Die Implementierung verwendet daher das Heimatregisterverfahren.

9.3 Entwurf

Für die Implementierung des Heimatregisterverfahrens benötigt man drei Teile:

1. Ein Programm, welches die Verwaltung der Heimatregister an einer Location übernimmt. Dies kann entweder von der Location selber oder einem immobilien Agenten geleistet werden. Der Agent-Locator soll jedoch laut Aufgabenstellung als Anwendung implementiert werden. Dies schließt tiefere Eingriffe in den MOLE-Systemkern aus. Auch das Verändern des Programmcodes einer Location muß möglichst vermieden werden. Das Heimatregister wird daher in Form eines immobilien Agenten verwirklicht. Diesen Agent nenne ich `RegistrationAgent`.
2. Das Senden der Aktualisierungsnachrichten kann entweder von den Locations oder von den Agenten selber initiiert werden. Da die Location möglichst nicht verändert werden soll, haben die Agenten die Verantwortung für das Senden der Aktualisierungsnachrichten. Man benötigt dann jedoch Methoden, die das Senden der Aktualisierungsnachrichten an das Heimatregister für die Agenten vereinfachen. Zwar könnte das Senden der Aktualisierungsnachrichten auch allein den Agenten überlassen werden, doch ist es nicht sinnvoll, wenn jeder Agent die gleichen Routinen implementieren muß. Die Methoden sollten daher irgendwo „zentral“ zur Verfügung gestellt werden. Dies kann geschehen, indem die Methoden in einer der Objektklassen eingefügt werden, von denen die Agenten abgeleitet werden. Dies bedeutet jedoch, daß jeder Agent selber entscheiden kann (und muß) wann er Aktualisierungsnachrichten versendet. Die Verantwortung dafür, ob überhaupt Nachrichten gesendet werden, wie häufig dies geschieht und, daß die richtigen Locations angegeben werden, liegt daher bei den einzelnen Agenten.
3. Der `RegistrationAgent` benötigt eine Schnittstelle zur Abfrage von Register-einträgen :

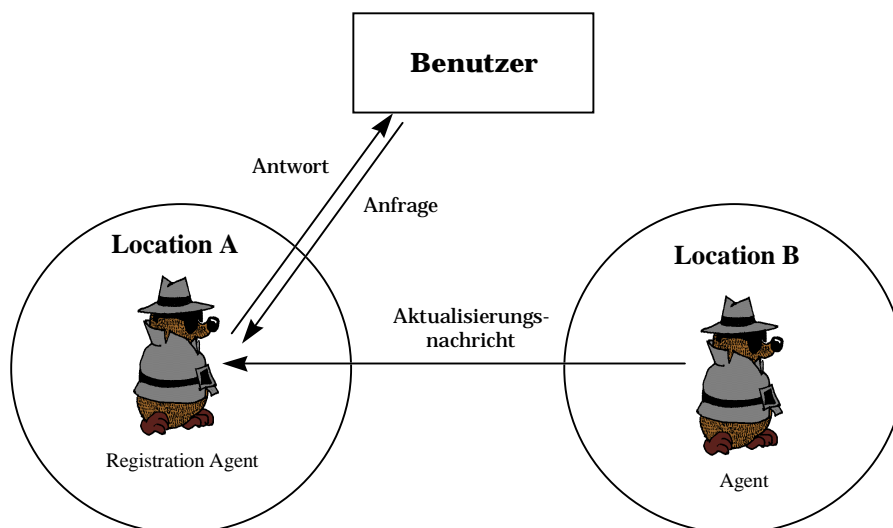


Abbildung 9-3: Agent-Locator API

Die Abbildung oben verdeutlicht, daß für die Realisierung des Agent-Locator zwei Schnittstellen definiert werden müssen:

- Die Schnittstelle zwischen Agent und `RegistrationAgent`.
- Die Schnittstelle zwischen Benutzer und `RegistrationAgent`.

Beide zusammen bilden die Agent-Locator API. Im folgenden wird zuerst die Agent-Locator API beschrieben und anschließend die Funktionalität der beiden benötigten Programmteile festgelegt.

9.3.1 Agent-Locator API

TEIL 1: SCHNITTSTELLE AGENT-REGISTRATIONAGENT

Ein Agent muß die Möglichkeit haben sich beim `RegistrationAgent` registrieren zu lassen, sein Registereintrag zu aktualisieren oder zu löschen.

Wenn sich ein Agent registrieren lassen will, kann er dies an der Location tun, an der er sich befindet (vorausgesetzt, es gibt dort einen `RegistrationAgent`). In diesem Fall ist die Kommunikation lokal. Es ist jedoch sinnvoll, daß sich der Agent auch bei einem `RegistrationAgent` an einer anderen Location registrieren lassen kann. Dies hat folgende Vorteile:

- Mobile Clients können unterstützt werden.
- Nicht an jeder Location muß ein `RegistrationAgent` vorhanden sein.
- Es können Obergrenzen für die Anzahl der Registereinträge pro `RegistrationAgent` vergeben werden. Damit kann eine gleichmäßigere Auslastung der `RegistrationAgents` erreicht werden.

Daher sollte auch für den Registrierungsprozess die Möglichkeit einer entfernten Kommunikation gegeben sein. Das Aktualisieren und Löschen eines Register-eintrages erfordert immer eine entfernte Kommunikation, da hierbei Agent und `RegistrationAgent` selten an derselben Location sind.

Für die lokale Kommunikation gibt es in MOLE zwei Möglichkeiten:

- Nachrichten (Messages)
- Procedure Calls

Ebenso gibt es zwei Möglichkeiten für die entfernte Kommunikation:

- Nachrichten
- Remote Procedure Calls (RPC)

Prozeduraufrufe haben den Nachteil, daß der Aufrufer der Prozedur für die Dauer der Prozedurausführung blockiert ist (es gibt keine asynchronen RPCs in MOLE). Für die Kommunikation zwischen Agent und `RegistrationAgent` werden daher Nachrichten eingesetzt.

Das Aktualisieren des Registereintrages soll möglichst schnell gehen, da die Aktualisierung pro Migration mindestens einmal durchgeführt werden muß und der Agent durch die Verwendung des Agent-Locators nicht unnötig „aufgehalten“ werden soll. Es wird daher auf das Senden von Bestätigungsnachrichten (Acknowledges) verzichtet. Dies bedeutet jedoch auch, daß das Register einen falschen Wert enthält, falls eine Nachricht verlorenght. Der Verlust einer Nachricht wird von keiner Seite bemerkt.

Entsprechend den Aufgaben des `RegistrationAgent` gibt es drei verschiedene Nachrichtentypen:

Die Register-Nachricht

Mit einer Register-Nachricht wird ein Agent beim `RegistrationAgent` erstmals registriert. Diese Nachricht muß daher gesendet werden, bevor eine Update- oder UnRegister-Nachricht (s.u.) gesendet werden darf.

Mit der Registration-Nachricht müssen zwei Informationen gesendet werden:

- Der Name des sendenden Agenten.
- Die Location, die als erstes im Register gespeichert werden soll. Dies kann entweder die Location sein, an der sich der Agent gerade aufhält, oder die Location an die er migrieren will.

Die Update-Nachricht

Die Update-Nachricht wird verwendet, um den Registereintrag eines Agenten zu aktualisieren. Dabei wird eine neue Location für den Agenten eingetragen. Die Update-Nachricht benötigt daher die gleichen Informationen wie die Register-Nachricht.

Die UnRegister-Nachricht

Die UnRegister-Nachricht wird verwendet, um den Eintrag eines Agenten aus dem Register zu löschen. Als Information muß der Name des Agenten mitgesendet werden.

TEIL 2 : SCHNITTSTELLE AGENT - BENUTZER

Ein Benutzer muß die Möglichkeit haben, den Registereintrag eines Agenten abfragen zu können. Der `RegistrationAgent` muß dafür eine entsprechende Schnittstelle anbieten.

Die Kommunikation zwischen Benutzer und `RegistrationAgent` kann durch Nachrichten oder Prozeduraufrufe erfolgen:

- Im ersten Fall sendet der Benutzer eine Nachricht an den `RegistrationAgent`. Diese Nachricht muß den Namen des Agenten enthalten, dessen Aufenthaltsort gesucht wird. Der `RegistrationAgent` sendet daraufhin den Aufenthaltsort des Agenten an den Sender zurück.

- Im zweiten Fall muß der `RegistrationAgent` eine Methode anbieten, die den Namen des gesuchten Agenten als Parameter erhält und als Resultat den Aufenthaltsort zurückgibt.

Der `RegistrationAgent` soll beide Möglichkeiten bieten und so dem Benutzer die Wahl lassen, ob er mit Nachrichten oder durch Methodenaufrufe mit dem `RegistrationAgent` kommunizieren will.

Ist der gesuchte Agent nicht beim `RegistrationAgent` registriert, muß eine entsprechende Fehlermeldung erfolgen.

Zu Testzwecken ist es sinnvoll, wenn es eine Möglichkeit gibt, alle Einträge eines `RegistrationAgents` abzufragen. Damit kann überprüft werden, welche Agenten registriert sind und wo sie sich gerade aufhalten. Diese Funktion läßt am besten mit Hilfe von Nachrichten implementieren, da dann auch entfernte `RegistrationAgents` abgefragt werden können.

9.3.2 Der `RegistrationAgent`

Durch den `RegistrationAgent` wird das eigentliche Heimatregister implementiert. Der `RegistrationAgent` ist ein immobiler Agent. An jeder Location, die den Agent-Locator Dienst anbieten will muß ein `Registration-Agent` vorhanden sein.

Der `RegistrationAgent` hat folgende Aufgaben:

1. Empfang und Auswertung der Aktualisierungsnachrichten der Agenten.
2. Eintragen, ändern und löschen der Registereinträge.
3. Dem Benutzer die Schnittstelle zur Abfrage von Registereinträgen zur Verfügung stellen.

Ein Register Eintrag besteht aus dem Namen eines Agenten und einem Locationnamen, der möglichst den aktuellen Aufenthaltsort des Agenten angeben sollte. Es könnten auch zusätzliche Daten über den Agenten im Register gespeichert werden, wie z.B. die Selbstbeschreibung des Agenten oder die Dienste, die ein Agent anbietet. Für die Implementierung wird jedoch auf die Speicherung zusätzlicher Informationen verzichtet.

Damit der `RegistrationAgent` von anderen Agenten überhaupt gefunden werden kann, muß er sich als Dienst registrieren lassen. Als Dienstname wurde „Agent-Locator“ gewählt.

Hier nun eine Beschreibung über das, was der `RegistrationAgent` beim Eintreffen einer bestimmten Nachricht tun muß:

- Beim Eintreffen einer Register-Nachricht muß ein neuer Eintrag ins Register vorgenommen werden. Ist der Agent bereits registriert, so soll der bisherige Registereintrag erst gelöscht und dann der Neue hinzugefügt werden.
- Beim Eintreffen einer Update-Nachricht muß der Eintrag des Agenten mit dem neuen Wert überschrieben werden. Ist noch kein Eintrag für den Agenten vorhanden, so wird ein solcher hinzugefügt.
- Beim Eintreffen einer UnRegister-Nachricht muß der entsprechende Eintrag aus dem Register gelöscht werden.
- Trifft eine Nachricht mit einer Anfrage ein, muß der `RegistrationAgent` den entsprechenden Eintrag ermitteln und die Antwort an den Sender zurückschicken.
- Beim Eintreffen einer Nachricht zum Ausgeben des gesamten Registers muß eine Nachricht an den Sender zurückgeschickt werden, die sämtliche Register-einträge in einer für den Empfänger sinnvollen Form enthält.

Es fällt auf, daß zwischen Update- und Register-Nachrichten kein funktioneller Unterschied besteht. In beiden Fällen wird ein neuer Wert ins Register eingetragen. Falls schon ein Wert für den Agenten existiert, wird dieser überschrieben. Ist noch keiner vorhanden, wird er neu eingefügt. Eine Unterscheidung zwischen diesen Nachrichtentypen wäre hier also gar nicht notwendig. Für den Agenten ist diese Unterscheidung jedoch sinnvoll, da beim erstmaligen Registrieren erst ein `RegistrationAgent` gefunden werden muß. Diese Unterscheidung wurde bei den Nachrichten beibehalten, um bei späteren Erweiterungen flexibler zu sein. Eine solche Erweiterung wäre, z.B. wenn zusätzliche Daten über den Agenten im Register gespeichert werden, die nicht bei jeder Migration aktualisiert werden müssen.

9.3.3 Zugriffsmethoden

Den Agenten werden Methoden zur Verfügung gestellt, die das Senden der Aktualisierungsnachrichten vereinfachen. Ein Agent braucht daher das Nachrichtenformat einer Aktualisierungsnachricht nicht zu kennen. Für jeden der drei Nachrichtentypen `Register`, `Update` und `UnRegister` müssen entsprechende Methoden vorhanden sein.

Die Methode `Register`

Durch den Aufruf dieser Methode kann sich ein Agent erstmalig bei einem `RegistrationAgent` registrieren.

Für diese Methode werden zwei Informationen benötigt:

- Die Location, an der der Agent registriert werden soll.
- Die Location, die zuerst ins Register eingetragen werden soll. Dies kann die Location sein, an der sich der Agent momentan befindet oder eine Location zu der er migrieren will.

Für das Registrieren eines Agenten sind zwei Schritte nötig:

- Herausfinden des Namens des `RegistrationAgent`s an der angegebenen Location. Falls dort kein `RegistrationAgent` existiert, muß eine Fehlermeldung zurückgegeben werden. Name und Ort des `RegistrationAgent`s müssen für spätere Update- und `UnRegister`-Nachrichten gespeichert werden.
- Senden der Register-Nachricht an den `RegistrationAgent`.

Die Methode Update

Diese Methode sollte von einem Agenten, der den Agent-Locator Service verwendet, vor oder nach einer Migration ausgeführt werden, um den neuen Aufenthaltsort ins Heimatregister einzutragen. Update erhält als Parameter einen Locationnamen, der ins Register eingetragen werden soll. Dieser Locationname wird in einer Update-Nachricht an den `RegistrationAgent` gesendet, der bei der Register Methode ermittelt wurde.

Die Methode UnRegister

`UnRegister` braucht keine Parameter und sendet nur eine `UnRegister`-Nachricht an den `RegistrationAgent`. Die Methode sollte von einem Agenten, der den Agent-Locator Service verwendet, ausgeführt werden bevor er terminiert oder wenn er den Agent-Locator Service nicht weiter verwenden will.

9.4 Implementierung

Die Implementierung erfolgt in JAVA, da Agenten in MOLE nur in JAVA programmiert werden können.

9.4.1 Der RegistrationAgent

Die Implementation des `RegistrationAgent` findet sich in der Datei `RegistrationAgent.java`.

Der `RegistrationAgent` benötigt eine Speichermöglichkeit für die Register-einträge. Ein Registereintrag besteht aus dem Zweiertupel Agentenname und Locationname. Der Agentenname ist dabei das Schlüsselement. Das heißt, es muß möglich sein Einträge durch Angabe des Agentennamens zu finden, zu verändern und zu löschen.

JAVA bietet mit der Klasse `java.util hashtable` eine ideale Möglichkeit ein solches Register zu implementieren. Eine Hashtable ist dabei ein Objekt, welches beliebige andere Objekte speichern kann. Jedem Objekt muß ein eindeutiger Schlüssel (key) zugeordnet sein. Schlüssel können ebenfalls beliebige Objekte sein, die die Methoden `hashCode()` und `equals()` implementieren. Anhand dieser Schlüssel können Objekte in der Hashtable gefunden, gelöscht und überschrieben werden. Eine Hashtable wächst dynamisch mit der Anzahl der zu speichernden Objekte. Als Schlüssel für die Registereinträge wird der in einen String umgewan-

delte Agentenname verwendet. In der Klasse `String` sind die beiden Methoden `hashCode()` und `equals()` implementiert. Durch die Umwandlung des Agentennamens in einen `String` müssen diese Methoden daher nicht für die Klasse `AgentName` implementiert werden.

Trifft bei einer `Location` eine Nachricht für einen Agenten ein, so wird (falls dieser sich an der `Location` aufhält) dessen Methode `receiveMessage(message m)` aufgerufen. Jeder Agent muß diese Methode implementieren. Beim Eintreffen einer Nachricht muß der `RegistrationAgent` unterscheiden, von welchem Typ die Nachricht ist:

- Ist der Inhalt der Nachricht (`message.content`) „register“ oder „update“, so wird der Name des sendenden Agenten und die `Location` des Senders ins Register eingetragen (`Hashtable.put()`).
- Ist der Inhalt der Nachricht das Wort „unregister“, so wird der Eintrag des sendenden Agenten aus dem Register gelöscht (`Hashtable.remove()`).
- Ist der Inhalt der Nachricht das Wort „getall“, so wird eine Nachricht an den Sender zurückgeschickt, die als Inhalt sämtliche Einträge des Registers in Form eines `String`s enthält (`Hashtable.toString()`). Ein solcher `String` wird vom `DisplayAgent` benötigt (s.u.).
- Ist der Inhalt der Nachricht der Name eines Agenten (also vom Typ `AgentName`), so wird eine Nachricht an den Empfänger zurückgesandt, die die für diesen Agenten im Register eingetragene `Location` enthält. Ist der Agent nicht im Register, so wird `NULL` zurückgeliefert.

Außerdem stellt der `RegistrationAgent` noch eine Methode `whereIs(AgentName Wanted)` zur Verfügung. Diese liefert als Resultat die `Location` des Agenten oder `NULL`, falls er nicht registriert ist.

9.4.2 Zugriffsmethoden

Für das Registrieren eines Agenten, sowie für das Aktualisieren des Registers und das Löschen des Registereintrages gibt es für die Agenten Methoden, die sie von der Arbeit, eigene Nachrichten senden zu müssen, befreit.

Die Wurzel des Ableitungsbaumes für einen Agenten ist die Klasse `Agent`. Von ihr werden die Klassen `UserAgent` und `SystemAgent` abgeleitet. Ein `Agent` ist immer eine Instanz einer dieser beiden Klassen. Will ein `UserAgent` mobil sein, muß er zusätzlich das Interface `MobileAgent` implementieren. Die Zugriffsmethoden befinden sich in der Klasse `RegisteredUserAgent`, die von `UserAgent` abgeleitet wird. Agenten, die den Agent-Locator Dienst verwenden wollen, müssen eine Instanz dieser Klasse sein.

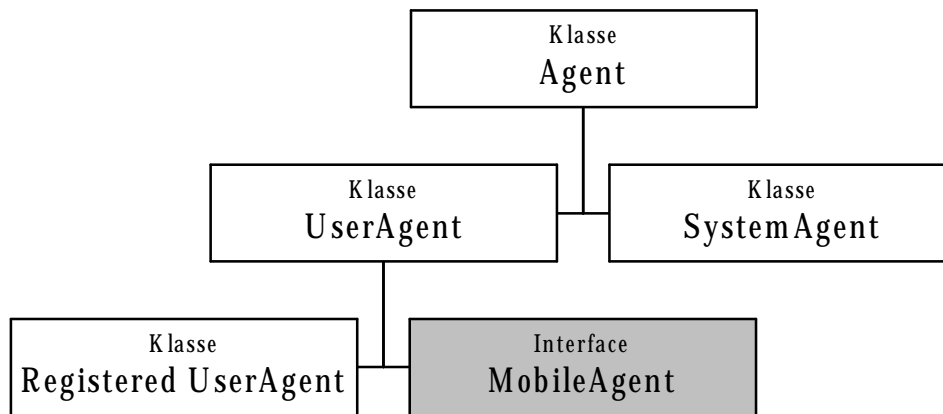


Abbildung 9-4: MOLE Klassenbaum für Agenten

Die register Methode

Zum erstmaligen Registrieren eines Agenten beim Agent-Locator gibt es die Methode `register`. Sie muß aufgerufen werden, bevor eine der Methoden `updateRegister` oder `unRegister` verwendet wird.

Die `register` Methode kann zwei Parameter haben:

- die Location, an der der Agent sich registrieren lassen will. An dieser Location muß ein `RegistrationAgent` vorhanden sein.
- die Location, die ins Register als Aufenthaltsort des Agenten eingetragen werden soll.

Beide Angaben können auch weggelassen werden. In diesem Fall wird der Agent an der Location registriert, an der er sich gerade befindet (vorausgesetzt es existiert dort ein `RegistrationAgent`). Als Aufenthaltsort wird ebenfalls die aktuelle Location eingetragen. Wird nur eine Location als Parameter übergeben, so wird diese als Aufenthaltsort für den Agenten bei der aktuellen Location registriert. Für jede dieser drei Varianten gibt es eine eigene Methode (method overloading). Bevor eine Register-Nachricht gesendet werden kann, muß erst der Name des `RegistrationAgent` an der Location bestimmt werden, an der der Agent sich registrieren lassen will. Der `RegistrationAgent` ist bei einer Location immer als Anbieter des Dienstes „Agent-Locator“ registriert. Somit kann der Name des Agenten durch einen Aufruf der Methode `serviceProvidersOf` der entsprechenden Location ermittelt werden. Falls die Location, an der der Agent registriert werden will, nicht die Location ist, an der er sich aufhält, so ist dafür ein RPC nötig. Die Implementierung von RPCs in MOLE ist für den Programmierer im Moment noch nicht besonders komfortabel, da der gesamte Ablauf eigenhändig programmiert werden muß. Damit der RPC funktioniert, muß an der aufgerufenen Location ein `LocalAgent` existieren, der in seiner `dispatch` Methode den RPC Aufruf für die Methode `serviceProvidersOf` vorsieht und den entsprechenden Rückgabewert zurückliefert. In Zukunft soll der RPC Mechanismus von MOLE durch das JAVA Packet „Remote Objects“ ersetzt werden, welches eine entsprechende Funktionalität bietet und in der nächsten JAVA-Version enthalten sein soll.

Der Name des `RegistrationAgents` wird in einer Klassen-Variable gespeichert, so daß die `updateRegister` und `unRegister` Methoden darauf zugreifen können.

Nachdem der Name des `RegistrationAgent` ermittelt wurde, wird eine Register-Nachricht an ihn gesandt. Eine Register-Nachricht enthält als Inhalt nur das Wort „register“. In die `Hashtable` des `RegistrationAgent` wird dann der Name und die `Location` des sendenden Agenten eingetragen (s.o.). Als `Senderlocation` wird daher nicht immer die `Location` angegeben, an der der Agent sich momentan befindet, sondern es wird die als Parameter angegebene `Location` verwendet.

Als Fehlercode wird bei einer erfolgreichen Ausführung 0 zurückgeliefert. War an der angegebenen `Location` kein `RegistrationAgent`, so wird 1 zurückgegeben. Gab es einen Fehler bei der Durchführung des RPC ist der Rückgabewert 2.

Ein Agent hat auch die Möglichkeit mehrere Heimatregister zu unterhalten. Er muß nur jede Migration Mithilfe der `register` Methode an alle seine Heimatregister melden. Die `updateRegister` Methode aktualisiert immer nur den letzten, bei einer `register` Methode verwendeten `RegistrationAgent`. Natürlich müssen bei einer Terminierung des Agenten alle seine Heimatregister informiert werden.

Die `updateRegister` Methode

Die `updateRegister` Methode sendet eine Update-Nachricht an den durch eine `register` Methode bestimmten `RegistrationAgent`. Als Parameter kann dabei ein `Locationname` angegeben werden, der als Aufenthaltsort ins Register eingetragen werden soll. Dieser Name dient als `Senderlocation` für die Update-Nachricht. Wird kein Parameter übergeben, wird der Name der `Location` verwendet, an der sich der Agent gerade befindet.

Der Rückgabewert ist 0 wenn kein Fehler auftrat oder 1 wenn noch kein `RegistrationAgent` bekannt ist. In diesem Fall muß zuerst die `register` Methode ausgeführt werden.

Die `unRegister` Methode

Die `unRegister` Methode sendet eine `UnRegister`-Nachricht an den durch eine `register` Methode bestimmten `RegistrationAgent`. Die Nachricht enthält als Inhalt nur das Wort „unregister“. Beim `RegistrationAgent` wird daraufhin der Eintrag des Agenten im Register gelöscht.

9.4.3 Der Display Agent

Der `DisplayAgent` wurde zum Testen des `RegistrationAgents` programmiert. Er kann auch bei der Administration einer `Location` hilfreich sein. Der `DisplayAgent` ist ein immobiler Agent, der in einem Fenster sämtliche Registerinträge eines `RegistrationAgents` darstellt. Dabei können auch entfernte `Registration-`

Agents abgefragt werden. Die Ausgabe erfolgt in einem AWT-Fenster. AWT bedeutet Abstract Window Toolkit und ist eine Klassenbibliothek zur Gestaltung grafischer Benutzeroberflächen in JAVA. Die Ausgabe erfolgt immer dort, wo sich der `DisplayAgent` aufhält.

Das Fenster für die Bildschirmausgabe ist in der Klasse `DisplayWindow.java` realisiert. Die Klasse `DisplayWindow` öffnet ein Fenster, in dem die Informationen dargestellt werden. Ein solches Fenster sieht unter Windows95 folgendermaßen aus:

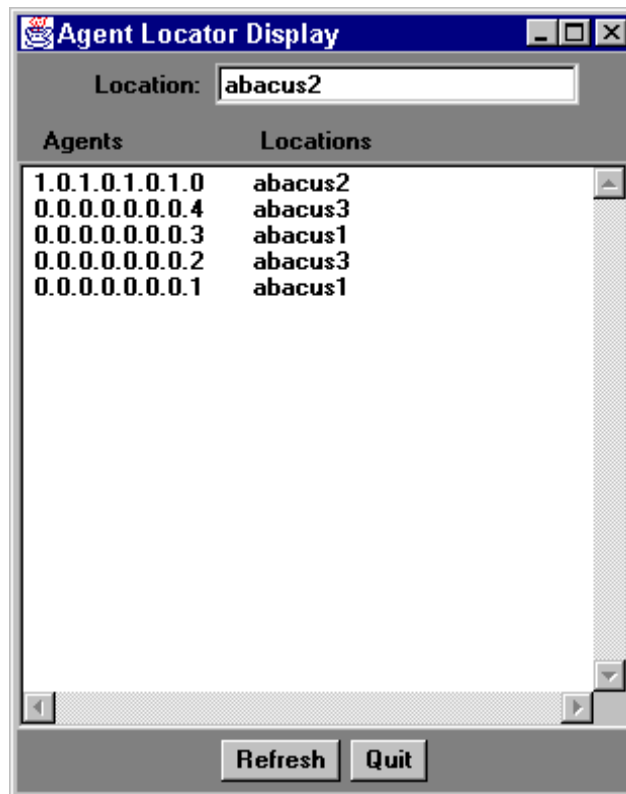


Abbildung 9-5: DisplayWindow Fenster

Im Fenster wird der Registerinhalt desjenigen `RegistrationAgents` angezeigt, dessen Location in der Eingabezeile angegeben wird. Der Refresh-Button aktualisiert die Anzeige. Der Quit-Button schließt das Fenster und terminiert den `DisplayAgent`. Der Agent wird auch dann terminiert, wenn das Fenster auf andere Weise geschlossen wird.

Bei der Eingabe einer Location wird die Methode `setTarget(LocationName ln)` des `DisplayAgent` aufgerufen. Diese erfragt bei der angegebenen Location den Namen des `RegistrationAgents` (wenn dort einer vorhanden ist) und sendet eine „getall“-Nachricht an diesen. Der `RegistrationAgent` antwortet daraufhin mit einer Nachricht, die sämtliche Registerinträge als String enthält. Dieser String hat die folgende Form:

```
{AgentName=Location,AgentName=Location...}
```

In diesem String werden einige Zeichen ausgetauscht, damit er direkt in der TextArea des Fensters dargestellt werden kann. Im einzelnen werden folgende Zeichenersetzungen durchgeführt:

- „{“ wird ersetzt durch „ “.
- „}“ wird ersetzt durch „ “.
- „,“ wird ersetzt durch LF.
- „=“ wird ersetzt durch TAB.

Anschließend wird der String als Parameter der Methode `refresh(String s)` an das `DisplayWindow` übergeben. Diese Methode überschreibt den bisherigen Inhalt der TextArea mit dem neuen String.

Beim Drücken des Refresh-Buttons geschieht dasselbe, jedoch ohne erneute Abfrage des Namens des `RegistrationAgents`, sofern sich die Eingabezeile nicht verändert hat.

9.4.4 Der Agent Walker

Zum Testen des Agent-Locators wurde der in MOLE bereits vorhandene Beispiel-Agent `Wanderer` modifiziert. Der modifizierte Agent wurde `Walker` genannt. Der `Walker` pendelt zwischen zwei Locations. Vor jeder Migration macht er eine Pause von 10ms. `Walker` ist eine Instanz der Klasse `RegisteredUserAgent` und verwendet den Agent-Locator Dienst. Bei seinem ersten Start registriert er sich bei einem `RegistrationAgent` und sendet vor jeder Migration eine Update-Nachricht. Wird er beendet sendet er eine `UnRegister`-Nachricht.

9.4.5 Debug Ausgabe

Wird der JAVA Interpreter mit dem Parameter `-debug` gestartet, werden Debug-Meldungen über die Abläufe im Agentensystem ausgegeben. Auch der Agent-Locator erzeugt solche Meldungen. Das allgemeine Format der Meldungen, die vom Agent-Locator erzeugt werden, sieht folgendermaßen aus:

```
( „Typ“ ) „Objektklasse.methode“: Meldungstext
```

Es gibt zwei Typen: I für eine Information, E für eine Fehlermeldung (Error). Das Eintreffen einer Nachricht beim `RegistrationAgent` erzeugt einen Meldungstext der folgenden Art:

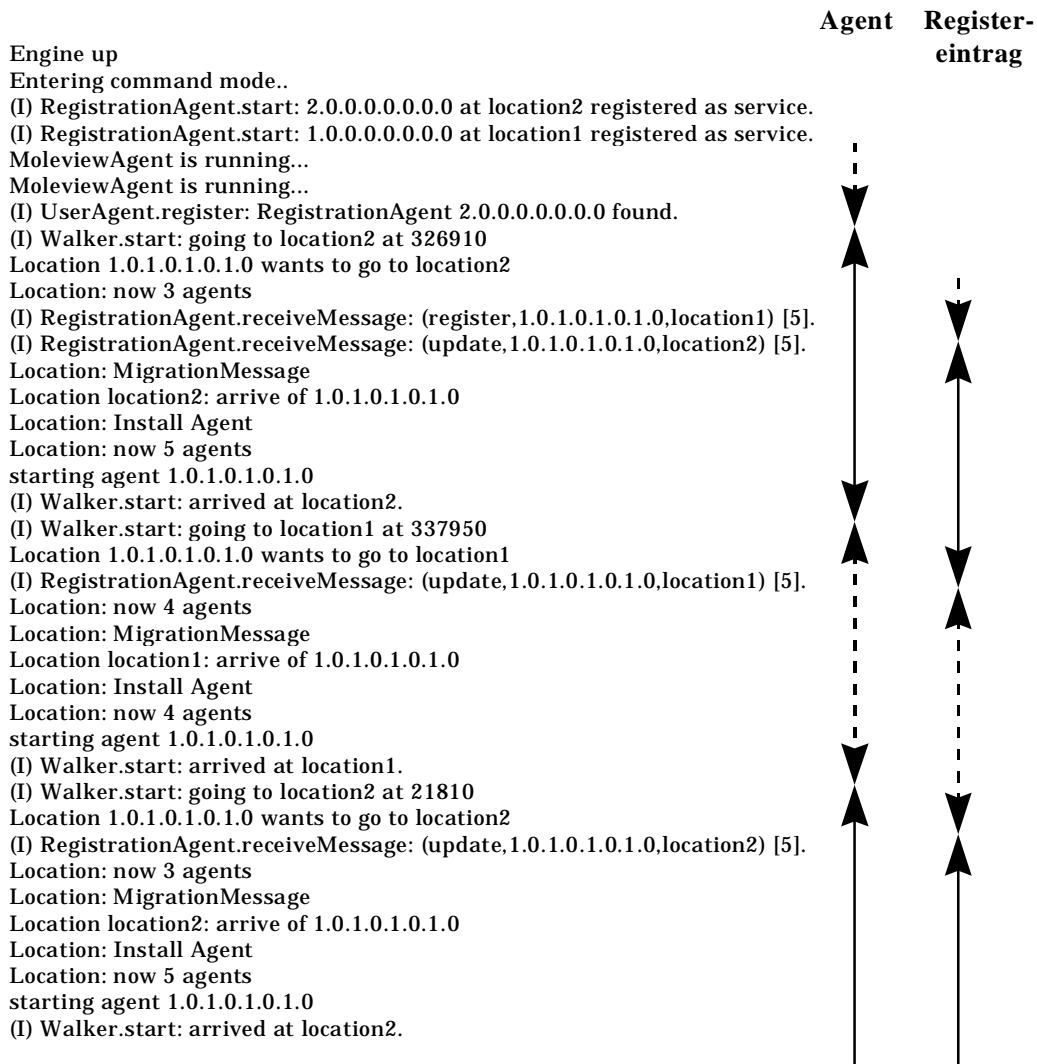
```
( „Typ“ , „Agentenname“ , „Location“ ) [ „Anzahl der Registereinträge“ ]
```

„Typ“ kann dabei folgende Werte annehmen: „register“, „update“, „unregister“, „getall“. Location ist der Locationname, der ins Register eingetragen wird.

Die folgende Ausgabe ist ein Auszug aus einem Programmlauf mit zwei Locations (`location1` und `location2`) auf einer Engine und einem „Walker“, der zwischen beiden Locations pendelt. „Walker“ wartet an immer 10 Sekunden, an einer Location, bevor er weitermigriert. Er ist beim `RegistrationAgent` von `location2` registriert und sendet vor jeder Migration eine Update-Nachricht.

Die Linien an der Seite geben an, wo der Agent sich zur Zeit befindet und welchen Wert das Register enthält. Eine gestrichelte Linie steht dabei für location1, eine durchgezogene Linie für location2.

Am Anfang sieht man die Meldungen, daß die `RegistrationAgents` der beiden Locations sich als Serviceanbieter registriert haben. Anschließend in Zeile 7 die Meldung, daß ein `RegistrationAgent` an location2 gefunden wurde. An diesen gehen daraufhin alle Update-Nachrichten.



Wie man sieht, wird das Register immer kurz nach dem Beginn der Migration, jedoch noch vor dem Eintreffen des Agenten an der neuen Location aktualisiert.

9.5 Test

Die Implementierung wurde unter Windows95, sowie unter Solaris getestet. In beiden Fällen wurden `walker` Agenten eingesetzt, die zwischen zwei Locations innerhalb einer Engine hin und her pendelten. Es wurden folgende Sachverhalte überprüft:

- Das Registrieren eines Agenten an seiner aktuellen Location (Win95, Solaris).

- Das Registrieren eines Agenten an einer anderen Location (Win95, Solaris).
- Der Versuch einen Agenten an einer Location zu Registrieren, an der kein `RegistrationAgent` vorhanden war (Win95).
- Der Versuch einen Agenten an einer nicht vorhandenen Location zu Registrieren (Win95, Solaris).
- Ein Scheitern eines Registrierungsversuchs an einer anderen Location durch einen Fehler bei einem RPC (Kein `LocationAgent` an der entsprechenden Location, keine Verarbeitung der `serviceProvidersOf` Methode durch den `LocalAgent`) (Win95).
- Das Aktualisieren des Registers vor einer Migration (Win95, Solaris)
- Das Aktualisieren des Registers nach einer Migration (Win95)
- Das Aktualisieren des Registers vor und nach einer Migration, wobei als Aufenthaltsort bei der ersten Nachricht die nicht existierende Location „Unterwegs“ eingetragen wurde (Win95).
- Das Löschen des Registereintrages (Win95, Solaris).
- Der Versuch eine `updateRegister` oder `unRegister` Methode auszuführen, bevor eine `register` Methode ausgeführt wurde (Win95).
- Das Verwenden von zwei Heimatregistern für einen Agenten, wobei hier vor einer Migration beide Register aktualisiert wurden. (Win95).
- Das Abfragen des Registers durch eine Nachricht (Win95, Solaris). Hier wurde auch der Fall getestet, daß der gesuchte Agent nicht registriert war.
- Das Abfragen des Registers durch die Methode `whereIs` (Win95, Solaris). Hier wurde auch der Fall getestet, daß der gesuchte Agent nicht registriert war.
- Die Funktion des `DisplayAgents` (Win95, Solaris).
- Eine „Überlastung“ eines `RegistrationAgents` durch Nachrichten mehrerer `Walker`, bei gleichzeitiger „Dauerabfrage“ des Registers durch einen `DisplayAgent`. Überprüft wurde dabei, ob alle Anfragen abgearbeitet werden.

Bei den Tests gab es keine Unterschiede zwischen den Versionen für Windows95 und Solaris. Alle Tests verliefen in der endgültigen Version des Agent-Locators erfolgreich.

Zusätzlich wurde ein Test unter Solaris durchgeführt, wo ein `Walker` zwischen zwei Locations auf verschiedenen Engines migrierte. Das Register wurde jeweils vor einer Migration aktualisiert. Der Test erbrachte keinen Unterschied zu den Tests, bei denen die Locations auf einer Engine waren. Da in beiden Fällen die gleichen Kommunikationsmechanismen verwendet werden, war dies auch nicht zu erwarten.

9.6 Nachteile und Erweiterungsmöglichkeiten

- Ein Nachteil des Heimatregisters ist, daß der Ort des Heimatregisters bei einer Anfrage bekannt sein muß. Eine Kodierung des Ortes im Namen des Agenten ist bei der bisherigen Methode zur Vergabe der Agentennamen nicht möglich. Auch ein Name Service steht nicht zur Verfügung. Es wird daher angenommen, daß ein Benutzer den Ort des Heimatregisters eines Agenten kennt, wenn er nach ihm sucht. Dies ist insofern keine große Einschränkung, als er auch den Namen des Agenten kennen muß. Dieser ist jedoch nur dem Besitzer des Agenten bekannt. Der Besitzer kennt jedoch auch das Heimatregister. Andere Benutzer müssen daher sowohl den Namen, als auch den Ort des Heimatregisters eines Agenten herausfinden. Wenn sie jedoch den Namen erfahren, können sie aus der gleichen Quelle auch den Ort des Heimatregisters herausfinden.
- Die Implementierung ist darauf angewiesen, daß die Agenten sich „korrekt“ verhalten. Da die Verantwortung für das Aktualisieren des Heimatregisters bei den Agenten liegt, kann auch ein falscher Wert im Register stehen, wenn ein Agent es unterläßt, sein Heimatregister von einer Migration zu unterrichten. Ein Agent kann auch einen falschen Aufenthaltsort in sein Heimatregister eintragen. Auch ist es nötig, daß ein Agent sein Heimatregister löscht, bevor er terminiert. Geschieht dies nicht, bleibt der jetzt unnötige Eintrag „ewig“ bestehen.
- Die Implementierung hat den Nachteil, daß ein Ausfall eines `RegistrationAgents` nicht bemerkt wird. Da bei einer Aktualisierung eines Registers keine Bestätigung erfolgt, wird der Ausfall vom Agenten nicht bemerkt. Dieser sendet weiterhin `updateRegister`-Nachrichten an einen nicht mehr existenten Agenten. Ein Agent hat jedoch die Möglichkeit, durch ein erneutes Ausführen der `register` Methode, die Existenz eines `RegistrationAgents` zu überprüfen.
- Die Implementierung bietet bis jetzt keine Möglichkeit eine Obergrenze für die Anzahl der Registerinträge eines `RegistrationAgents` anzugeben. Zwar könnte der `RegistrationAgent` bei einer Ablehnung einer Registrierung eine Nachricht an den entsprechenden Agenten senden, doch muß dieser dann in seiner `receiveMessage` Methode auf solche Nachrichten reagieren können. Ein einfacher Fehlercode durch die `register` Methode bei einer Ablehnung, ist bei der Implementierung durch Nachrichten nicht möglich.
- Es gibt keine Sicherheitsmaßnahmen in der Implementierung. Ein Agent ist damit beispielsweise in der Lage, das Heimatregister eines anderen Agenten zu manipulieren, indem er eine entsprechende Nachricht an dessen `RegistrationAgent` sendet (ohne eine der Zugriffsmethoden zu verwenden) und als Sendername den Namen des anderen Agenten angibt.
- Für einen menschlichen Benutzer des Agent-Locators wäre ein Programm mit grafischer Benutzeroberfläche wünschenswert, welches die Formulierung einer Anfrage ermöglicht und die Antwort visualisiert. Für das MOLE System wird

gerade ein grafischer Monitor implementiert, welcher die Agenten, Kommunikationsbeziehungen und andere Daten einer Location anzeigt [Beck96]. Eine Erweiterung von XMoleView, so der Name des Monitors, wäre eine einfache und elegante Möglichkeit, um einem menschlichen Benutzer die Formulierung einer Anfrage zu ermöglichen. Leider war XMoleView bis zum Abgabzeitpunkt dieser Arbeit weder fertiggestellt noch ins MOLE System integriert, so daß eine Implementierung dieser Funktionalität innerhalb dieser Studienarbeit nicht möglich war.

- Im Heimatregister eines Agenten könnten, außer seinem Aufenthaltsort, noch zusätzliche Informationen gespeichert werden. Eine Erweiterung diesbezüglich ist nicht allzu aufwendig. Es müßten natürlich dann auch Möglichkeiten geschaffen werden, die zusätzlichen Informationen über den Agenten auszulesen. Auch für den Agent-Locator selbst wäre es sinnvoll, zusätzliche Informationen im Register zu speichern. So könnte beispielsweise jeder Eintrag mit einer Zeitmarke versehen werden, die seine letzte Aktualisierung angibt. Einträge, die schon länger nicht mehr aktualisiert wurden, könnten dann automatisch gelöscht werden.

10 Zusammenfassung

Ein Agent-Locator ist ein sinnvoller Dienst in einem Agentensystem. Er bietet für viele Anwendungen Vorteile und ist ein hilfreiches Werkzeug für Tests und zur Fehlersuche innerhalb eines Agentensystems.

Zur Realisierung eines Agent-Locators kommen von den untersuchten Verfahren vor allem drei in Frage:

- Das Broadcast-Verfahren, in welchem der Aufenthaltsort eines Agenten über einen Broadcast ermittelt wird.
- Das Heimatregisterverfahren, bei dem ein Ort immer weiß, wo ein Agent sich aufhält.
- Die Verwendung eines Name Service, welcher die Aufenthaltsorte der Agenten speichert.

Besonders das Heimatregisterverfahren ist in diesem Zusammenhang interessant, da es nur geringe Anforderungen an die Infrastruktur des Agentensystems stellt und relativ einfach zu implementieren ist. Die anderen vorgestellten Verfahren eignen sich nur für einzelne Anwendungen mit spezifischen Anforderungen. So ist das Verfahren Straßensperren beispielsweise den Anforderungen eines universell einsetzbaren Agent-Locators nicht gewachsen, wäre aber innerhalb eines Multi User Dungeons verwendbar.

Zur Implementierung eines Agent-Locators für MOLE wurde das Heimatregisterverfahren ausgewählt. Die Implementierung zeigt die Funktionsfähigkeit des Verfahrens und läßt sich beim Entstehen höherer Anforderungen an den Agent-Locator leicht erweitern.

11 Ausblick

Die Unterstützung von mobilen Einheiten ist nicht nur ein Problem auf der Ebene eines Agentensystems, sondern auch schon auf der Netzwerkebene. Viele Anwender besitzen tragbare Computer, wie z.B. Notebooks, Palmtops oder Personal Digital Assistants (PDA) und wollen die Möglichkeit haben, sich an verschiedenen Orten mit dem Netzwerk zu verbinden. Gerade ein Netzwerk wie das Internet sollte deshalb einen Weg bieten, um solche mobilen Endgeräte zu unterstützen. Bisher war dies aufgrund des IP-Adressierungsschemas unmöglich. Durch die Entwicklung von Mobile IP wird nun jedoch eine entsprechende Möglichkeit geschaffen ([Tanenbaum96], S432ff).

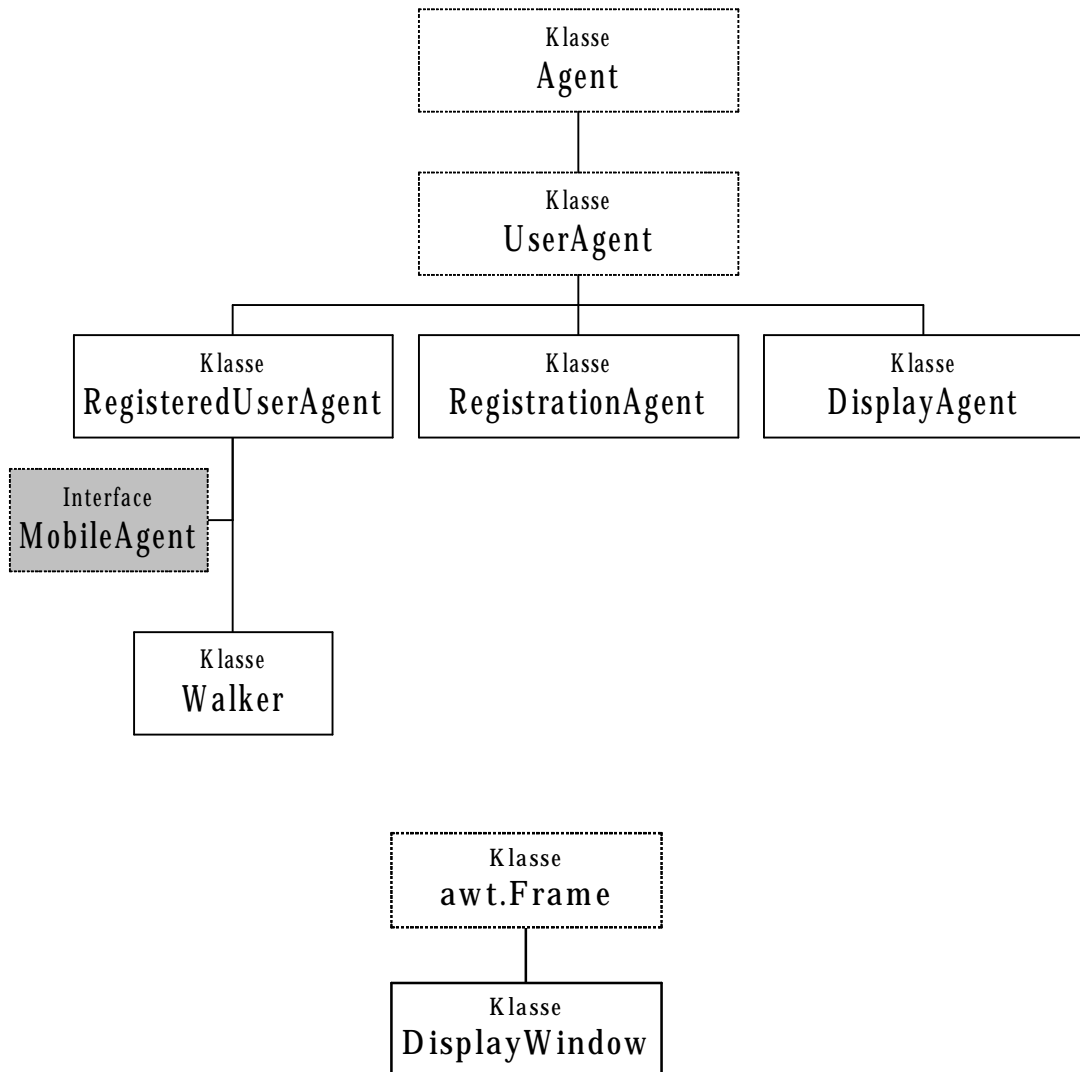
Mobile IP verwendet dabei ein dem Heimatregister ähnliches Verfahren. Um seinen Standort wechseln zu können, braucht ein Computer einen „home agent“. Wechselt er nun den Standort, so muß er sich an seinem neuen Standort bei einem „foreign agent“ registrieren lassen. Dieser informiert den entsprechenden „home agent“ über den Aufenthaltsort des Benutzers. Nachrichten an den Benutzer werden vom „home agent“ des Benutzers empfangen und über den „foreign agent“ an den Benutzer weitergeleitet. Außerdem wird dem Sender der Nachricht die Netzwerkadresse des „foreign agent“ übermittelt, so daß nachfolgende Nachrichten nicht mehr den Umweg über den „home agent“ nehmen müssen, sondern direkt an den „foreign agent“ gesendet werden können.

Es wäre denkbar Mobile IP auch auf der Ebene des Agentensystems einzusetzen. Anstatt für mobile Computer wird das Verfahren dann für mobile Agenten verwendet. Da mobile Agenten im Normalfall wohl häufiger ihren Aufenthaltsort wechseln als mobile Computer, muß allerdings überprüft werden, ob Mobile IP mit einer höheren „Migrationsgeschwindigkeit“ zurechtkommt. In der neuen IP Version 6 ist Mobile IP enthalten.

Eine andere Neuerung in IP Ver. 6 ist die erweiterte Möglichkeit für Multicasts. Da ein Multicast auf Netzwerkebene ein Broadcast auf Agentenebene sein kann, ist auch das Broadcast-Verfahren auf seine Verwendbarkeit mit IP Ver. 6 zu untersuchen.

12 Anhang

A Übersicht über die neuen Klassen



- Neue Klassen
- Bereits vorhandene Klassen

B Die Methoden der neuen Klassen

Klasse: RegistrationAgent

Der RegistrationAgent verwaltet die Heimatregister aller Agenten die sich an seiner Location registrieren.

Konstruktoren

```
public RegistrationAgent ()

public RegistrationAgent (String sd, AgentName aName)
    Der Standard Konstruktor für diese Klasse.
```

Methoden

```
public void start ()
    Wird aufgerufen, wenn der Agent gestartet wird. Registriert den Agenten
    bei der Location als Anbieter des Dienstes „Agent-Locator“.

public void stop ()
    Wird aufgerufen, wenn der Agent für eine Migration gestoppt wird.

public void end ()
    Wird aufgerufen, wenn der Agent terminiert wird.

public synchronized void receiveMessage (Message m)
    Wird aufgerufen, wenn Nachrichten für den Agenten eintreffen. Der
    RegistrationAgent „versteh“ Nachrichten mit folgendem Inhalt:
    „register“:          Trägt den Sender ins Register ein
    „UpdateRegister“:   Aktualisiert den Aufenthaltsort des Senders
    „UnRegister“:       Löscht den Sender aus dem Register
    ein Agentenname     Sendet eine Nachricht mit dem Aufenthaltsort des
                        Agenten zurück
    „getall“:           Sendet eine Nachricht mit sämtlichen Register-
                        einträge als String an den Sender zurück
    Als Aufenthaltsort wird jeweils die Senderlocation verwendet.

public synchronized LocationName whereIs (AgentName Wanted)
    Liefert den aktuellen Aufenthaltsort des Agenten „wanted“ zurück.
```

Klasse: RegisteredUserAgent

RegisteredUserAgent stellt Methoden für die Agenten zur Verfügung, um die Kommunikation mit einem RegistrationAgent zu vereinfachen.

Konstruktoren

```
public RegisteredUserAgent ()

public RegisteredUserAgent (String einString, AgentName aName)
    Der Standard Konstruktor für diese Klasse.
```

Methoden

```
public void start ()
    Wird aufgerufen, wenn der Agent gestartet wird.

public void stop ()
    Wird aufgerufen, wenn der Agent für eine Migration gestoppt wird.

public void end ()
    Wird aufgerufen, wenn der Agent terminiert wird.
```

```

public void receiveMessage (Message m)
    Wird aufgerufen, wenn Nachrichten für den Agenten eintreffen.

public int register ()
    Registriert den Agenten bei der aktuellen Location. Als Aufenthaltsort
    wird ebenfalls die aktuelle location eingetragen.
    Ergebnis: 0 Alles in Ordnung.
              1 Kein RegistrationAgent gefunden.

public int register (LocationName location)
    Registriert den Agenten bei der aktuellen Location. Als Aufenthaltsort
    wird „location“ eingetragen.
    Ergebnis: 0 Alles in Ordnung.
              1 Kein RegistrationAgent gefunden.

public int register (LocationName register, LocationName location)
    Registriert den Agenten bei der Location „register“. Als Aufenthaltsort
    wird „location“ eingetragen.
    Ergebnis: 0 Alles in Ordnung.
              1 Kein RegistrationAgent gefunden.
              2 Fehler beim Ausführen des RPC

public int updateRegister ()
    Ändert den Aufenthaltsort des Agenten im Register auf den Namen der
    aktuellen Location.
    Ergebnis: 0 Alles in Ordnung.
              1 Kein RegistrationAgent gefunden.

public int updateRegister (LocationName location)
    Ändert den Aufenthaltsort des Agenten im Register auf „location“.
    Ergebnis: 0 Alles in Ordnung.
              1 Kein RegistrationAgent gefunden.

public int unRegister ()
    Löscht den Registereintrag des Agenten.
    Ergebnis: 0 Alles in Ordnung.
              1 Kein RegistrationAgent gefunden.

```

Klasse: DisplayAgent

Der DisplayAgent stellt sämtliche Registereinträge eines RegistrationAgents in einem AWT-Fenster dar. Er benötigt die Klasse DisplayWindow.

Konstruktoren

```

public DisplayAgent ()

public DisplayAgent (AgentName aName)
    Der Standard Konstruktor für diese Klasse.

```

Methoden

```

public void start ()
    Wird aufgerufen, wenn der Agent gestartet wird.

public void stop ()
    Wird aufgerufen, wenn der Agent für eine Migration gestoppt wird.

public void end ()
    Wird aufgerufen, wenn der Agent terminiert wird.

public int setTarget (LocationName location)
    Überprüft, ob an der Location „location“ ein RegistrationAgent vorhanden
    ist und bestimmt dessen Namen.
    Ergebnis: 0 Alles in Ordnung.

```

```

1 Kein RegistrationAgent gefunden.
2 Fehler beim Ausführen des RPC

```

```

public void receiveMessage (Message m)
    Wird aufgerufen, wenn Nachrichten für den Agenten eintreffen. Der
    RegistrationAgent „versteh“ Nachrichten, die der Registration Agent als
    Antwort auf eine „getall“ Nachricht sendet.

public void askForEntrys ()
    Sendet eine „getall“ Nachricht an den RegistrationAgent.

public void exit ()
    Terminiert den Agenten.

```

Klasse: DisplayWindow

Definition des Fensters, welches für den DisplayAgent benötigt wird.

Konstruktoren

```

public DisplayWindow (Object obj)
    Der Standard Konstruktor dieser Klasse. Als Parameter muß ein
    Verweis auf den aufrufenden DisplayAgent übergeben werden.

```

Methoden

```

public void refresh (String s)
    Gibt „s“ auf dem Anzeigebereich des Fensters aus.

public boolean action (Event event, Object arg)
    Verwaltet die Ereignisse, die beim Drücken von Buttons erzeugt werden.

public boolean handleEvent (Event event)
    Verwaltet das Ereignis, welches beim Schließen des Fensters erzeugt wird.

```

Klasse: Walker

Beispielagent zum Testen des Agent-Locators. Pendelt zwischen zwei Locations.

Konstruktoren

```

public Walker ()

public Walker (String einString, AgentName aName,
              LocationName h, LocationName t)

public Walker (String einString, AgentName aName,
              LocationName h, LocationName t, Long t1, Long t2)

```

Methoden

```

public void start ()
    Wird aufgerufen, wenn der Agent gestartet wird. Beim ersten Start
    registriert sich der Agent bei einem RegistrationAgent.
    Nach einer Pause vom 10 ms migriert der Agent an die jeweils andere
    Location.

public void stop ()
    Wird aufgerufen, wenn der Agent für eine Migration persistent gemacht
    wird. Es wird das Register des Agenten aktualisiert.

public void end ()
    Wird aufgerufen, wenn der Agent terminiert wird. Der Registereintrag des
    Agenten wird gelöscht.

```

C Glossar

Die Begriffe sind hier in der Bedeutung erklärt, in der sie in dieser Arbeit verwendet werden. Für manche Begriffe bestehen darüber hinaus noch andere Bedeutungen [Duden93]. Ein → ist ein Hinweis darauf, daß für das darauffolgende Wort auch eine Erklärung vorhanden ist.

Agent

In dieser Arbeit sind mit dem Begriff Agent immer mobile Agenten gemeint. Also nicht Agenten aus dem Bereich der Künstlichen Intelligenz.

Ein Agent ist hier ein Programm, welches einem Benutzer helfen soll, eine Aufgabe zu erfüllen. Dazu haben sie die Möglichkeit, sich von einem Computer im Netzwerk zu einem anderen zu bewegen (→Migration).

Agentensystem

Laufzeitumgebung für Mobile Agenten. Das Agentensystem stellt die notwendige Infrastruktur für die Agenten zur Verfügung und schirmt das Computersystem vor den Agenten ab. →MOLE ist ein Agentensystem.

Agent-Locator

Ein Systemdienst oder eine Anwendung, die dazu dient, in einem →Agentensystem den aktuellen Aufenthaltsort eines Agenten zu ermitteln.

Anfrage

Eine Anfrage ist ein Aufruf des →Agent-Locator, der damit beauftragt wird, nach einem bestimmten Agenten zu suchen.

Anwendung

Ein Programm, welches das →Agentensystem (und damit die Agenten) verwendet, um bestimmte Aufgaben zu erfüllen.

Benutzer

Ein Programm, →Agent oder eine Person, die eine →Anfrage an den →Agent-Locator stellt.

Broadcast

Eine Nachricht, die von einem Computer im →Netzwerk an alle anderen Computer gesendet wird.

Client

Arbeitsstation oder Programm, welches die Dienste eines →Servers in Anspruch nimmt.

Detektiv

Ein \rightarrow Agent, dessen Aufgabe es ist, einen anderen Agenten zu suchen. Detektive werden in den Verfahren „Verfolgen der Migrationsroute“ und „Straßensperren“ eingesetzt.

Engine

Eine Engine ist in \rightarrow MOLE ein \rightarrow Interpreter, der in einem Betriebssystemprozeß läuft. Eine Engine kann daher mehrere \rightarrow Locations verwalten.

Heimatort

Der \rightarrow Ort, an dem ein \rightarrow Agent erzeugt wird. Dies ist damit auch der Startpunkt für eventuelle \rightarrow Migrationen.

Heimatregister

Im GSM-Standard für Mobilfunknetze hat jeder Teilnehmer ein Heimatregister. Dort ist neben anderen Daten der aktuelle Aufenthaltsort des Teilnehmers vermerkt. Es muß daher von Ortsänderungen des Teilnehmers informiert werden.

In dieser Arbeit wird auch ein Verfahren zur Realisierung des Agent-Locators als Heimatregister bezeichnet, da es nach dem gleichen Prinzip arbeitet.

Internet

Weltweites \rightarrow Netzwerk, welches früher hauptsächlich zu wissenschaftlichen Zwecken verwendet wurde, heute jedoch zunehmend auch kommerziell genutzt wird. Das Internet stellt seinen Benutzern viele Dienste zur Verfügung, wie z.B. Nachrichtenversand (E-Mail) und Informationsrecherche (WWW, Gopher). Das Internet basiert auf den Netzwerkprotokollen IP und TCP.

Interpreter

Ein Programm, welches ein Programm einer anderen Programmiersprache nach den notwendigen syntaktischen Überprüfungen sofort ausführt. Der Interpreter analysiert nacheinander jede Anweisung und Deklaration des Quellprogramms und führt diese unmittelbar aus. Einige Programmiersprachen benötigen einen Interpreter [Duden93].

JAVA

Eine architekturunabhängige, objektorientierte Programmiersprache, die von SUN Microsystem entwickelt wurde. JAVA Quellcode wird in einen stackorientierten Bytecode übersetzt, der interpretiert wird. Die Sprache besitzt einige Eigenschaften, die sie interessant für die Implementierung eines \rightarrow Agentensystems macht. Dies sind insbesondere die Sicherheitsfeatures von JAVA.

Lebensdauer

Die Zeitdauer, die ein \rightarrow Agent existiert. Dies ist die Zeit zwischen seiner Erzeugung und seiner Terminierung.

Local Area Network (LAN)

Ein \rightarrow Netzwerk mit nur geringer Ausdehnung, z.B. innerhalb eines Gebäudes.

Location

Ein \rightarrow Ort in \rightarrow MOLE.

Metropolitan Area Network (MAN)

Ein \rightarrow Netzwerk, welches sich über eine Stadt oder Region erstreckt und eine große Bandbreite zur Verfügung stellt.

Migration

Migration bezeichnet den Ortswechsel eines \rightarrow Agenten in einem \rightarrow Agentensystem. Agenten haben die Möglichkeit an beliebiger Stelle ihre Ausführung zu unterbrechen und an einen anderen \rightarrow Ort zu wechseln, an dem sie weiter ausgeführt werden.

Migrationsanzahl

- Die Anzahl an \rightarrow Migrationsen, die ein \rightarrow Agent während seiner \rightarrow Lebenszeit durchführt.
- Die Anzahl an Migrationen, die an einem \rightarrow Ort in einer gegebenen Zeitspanne ablaufen.

Mobiler Client

Ein \rightarrow Client, der nicht immer am selben Ort ist und daher auch nicht immer mit dem \rightarrow Netzwerk verbunden ist.

MOLE

MOLE ist ein \rightarrow Agentensystem, welches in der Abteilung Verteilte System des Instituts für Parallele und Verteilte Höchstleistungsrechner der Universität Stuttgart entwickelt wird. MOLE ist in \rightarrow JAVA implementiert.

Name Service

Ein Dienst der einem symbolischen Namen eine Adresse zuweist.

Netzwerk,

Ein System mehrerer untereinander verbundener Computer, welches den Austausch von Nachrichten und die gemeinsame Nutzung von Dienstleistungen erlaubt.

Ort

Eine Organisationseinheit eines →Agentensystems. →Agenten können sich nur an einem Ort aufhalten oder zwischen ihnen →migrieren.

Remote Procedure Call (RPC)

Ein RPC ermöglicht das entfernte Ausführen von Prozeduren. Dabei ruft ein →Client eine Prozedur eines →Servers auf. Der Client ruft dazu lokal einen Stellvertreter der entfernten Prozedur auf. Ein solcher Stellvertreter wird Stub genannt. Der Stub sendet Aufruf und Parameter an einen Partner-Stub auf dem →Server. Dieser führt die Prozedur mit den gegebenen Parametern aus und sendet die Ergebnisse auf dem gleichen Weg zurück (siehe auch [Birrel84]).

Replikat

Eine originalgetreue Kopie eines Originals.

Ressourcen

Hilfsmittel, die für die Ausführung eines Vorgangs oder einer Aufgabe nötig sind. In der Informatik sind Ressourcen beispielsweise Speicherplatz oder Rechenzeit.

Server

Netzwerkstation oder Programm, welches für →Clients →Dienste anbietet oder Ressourcen zur Verfügung stellt.

Skalierbarkeit

Die Möglichkeit ein bestehendes System in der Größe zu verändern, ohne das dadurch gravierende Nachteile in Bezug auf Performanz oder Stabilität entstehen.

Systemdienst

Ein →Dienst der nicht von einer →Anwendung, sondern vom →Agentensystem selber angeboten wird.

Terminieren

Unter der Terminierung eines →Agenten versteht man das Entfernen des Agenten aus dem →Agentensystem. Der Agent terminiert sich entweder selber oder die Terminierung wird vom →Ort ausgelöst, an dem sich der Agent befindet.

Wide Area Network (WAN)

Ein →Netzwerk, welches sich über ein großes Gebiet (bis zu weltweit) ausdehnt.

13 Literaturverzeichnis

- [Beck95] Beck B.: *Beispiele für Agentensysteme*. Vortrag im Rahmen des Hauptseminars „Kooperation in verteilten Systemen“, IPVR, Universität Stuttgart, Sommersemester 1995
- [Beck96] Beck, B.: *Konzeption und Implementierung eines graphischen Monitors für ein Mobile-Agenten-System*. Studienarbeit Nr. 1523, Institut für Parallele und Verteilte Höchstleistungsrechner, Universität Stuttgart, 1996.
- [Biala95] Biala, Jacek; *Mobilfunk und Intelligente Netze*. 2. Auflage, Vieweg Verlag, ISBN 3-528-15302-4, 1995
- [Birrel84] Birrel, A. D., Nelson, B. J.: *Implementing Remote Procedure Calls*. ACM Transactions on Computer Systems 39-59, Februar 1984
- [Boggs83] Boggs, R. D.: *Internet broadcasting*. Technical Report, Xerox Corporation, Palo Alto Research Center, 1983
- [Duden93] Claus, V.; Schwill, A.: *Duden Informatik*. BI-Wiss.-Verlag, 2 Auflage, ISBN 3-411-05232-5, 1993
- [Fünf95] Fünfroeken, Stefan: *DAFID: Agentenforschung in Deutschland*. 1995
[//www.informatik.th-darmstadt.de/~fuenf/work/agenten/agenten.html](http://www.informatik.th-darmstadt.de/~fuenf/work/agenten/agenten.html)
- [Gosling95] Gosling J., Mc Gilton H.: *The Java Language Environment- A white paper*. Sun Microsystems Computer Company, Mai 1995
- [Harrison94] Harrison, C.G.; Chess, D.M; Kershenbaum, A.: *Mobile Agents: Are they a good idea?*, IBM Research Report, RC 19887, Oktober 1994,
<http://www.research.ibm.com/xw-d953-mobag.ps>
- [Hohl95] Hohl, Fritz: *Konzeption eines einfachen Agentensystems und Implementation eines Prototyps*. Diplomarbeit Nr. 1267, Institut für Parallele und Verteilte Höchstleistungsrechner, Universität Stuttgart, 1995.
- [Mauly92] Mauly, M; Paulet, M.: *The GSM System for mobile Communication*. Europe Media Publication S.A., 1992
- [Mullender90] Mullender S.: *Distributed Systems*. Addison Wesley. ISBN 0-201-41660-3, 1990
- [SecurityStory] SUN Microsystems: Hotjava™: *The Security Story*.
[Http://java.sun.com/1.0beta/doc/security/security.html](http://java.sun.com/1.0beta/doc/security/security.html)
- [Tanenbaum96] Tanenbaum, A.: *Computer Networks*, 3. Auflage, Prentice Hall, ISBN 0-13-349945-6, 1996
- [Tutorial] The Java Tutorial online. <http://httpjava.sun.com/documents/tutorial>
- [White94] White, James E.: *Telescript Technology: The Foundation for the Electronic Marketplace*. General Magic White Paper, 1994

**Ich versichere, daß ich diese Arbeit
selbständig verfaßt und nur die
angegebenen Hilfsmittel verwendet habe.**

Stuttgart, den 31. Juli 1996