Universität Stuttgart
Fakultät Informatik

Wintersemester 1996/97

Hauptseminar

Mobile Agenten

Thema:

Das Agentensystem ffMAIN

(Bearbeitet von: Sven Bäurle)

Vortragstermin: 11. Februar 1997
Betreuer: Dipl.-Inform. Fritz Hohl

Inhaltsverzeichnis

1 EINFÜHRUNG 2

2 DAS AGENTENSYSTEM FFMAIN 2

2.1 ENTSTEHUNG.. 2
2.2 ZIELSETZUNG.. 2
2.3 AGENTENBESTANDTEILE 3
2.3.1 Code 3
2.3.2 Zustand.. 3
2.3.3 Attribute 3
2.4 TRANSPORTPROTOKOLL 4

3 BESTANDTEILE DES AGENTENSYSTEMS 4

3.1 DER AGENTENSERVER 4
3.1.1 Die Aufgaben des Agentenservers 5
3.1.2 Der Informationsraum 5
3.1.3 Agentenverwaltung 6
3.1.4 Laufzeitumgebung.. 6
3.2 SPRACHERWEITERUNGEN.. 6
3.3 STARTPROGRAMM 7

4 FUNKTIONSWEISE.. 8

4.1 MIGRATION 8
4.2 KOMMUNIKATION.. 8
4.2.1 Agent mit Agentenserver 9
4.2.2 Agent mit Agent 9
4.2.3 Agent mit Eigentümer 11

5 SICHERHEITSASPEKTE... 11

5.1 AUSFÜHRUNG EINES AGENTEN 11
5.2 ÜBERTRAGUNG EINES AGENTEN... 11
5.3 STATUSABFRAGE DES EIGENTÜMERS 12
5.4 INFORMATIONSRAUM.. 12

6 ZUSAMMENFASSUNG 12

1 Einführung

Um nützlich zu sein, muß ein Agent mit seinem Herkunftssystem und mit anderen Agenten zusammenarbeiten. Er muß auf Informationen, die vom Rechner angeboten werden, zugreifen oder mit anderen Agenten über den Austausch von bestimmten Diensten verhandeln. Außerdem müssen Agenten in der Lage sein, sich innerhalb heterogener Netzwerke zu bewegen.
Dies ist nur möglich, wenn es eine standardisierte Infrastruktur für Agenten gibt. Diese Infrastruktur muß die Bewegungsfähigkeit und die Kommunikation von Agenten unterstützen. Außerdem muß sie den Rechner vor unkontrolliertem Zugriff fremder Agenten schützen und die Integrität der lokalen Agenten (so gut es geht) gewährleisten.
ffMAIN ist eine solche Infrastruktur.

2 Das Agentensystem ffMAIN

2.1 Entstehung

ffMAIN ist eine Abkürzung für die ausführliche Bezeichnung „**F**rank**f**urter **M**obile **A**genten **In**frastruktur". Mobile Agenten sind seit Anfang 1995 ein Forschungsgebiet der Abteilung Telematik an der Johann Wolfgang Goethe-Universität in Frankfurt am Main. Zu Forschungszwecken wurde eine *prototypische* Infrastruktur für Agenten ([1], [2]) implementiert, mit der verschiedene Experimente mit Agentensystemen durchgeführt werden konnten.
Das Agentensystem wird schrittweise erweitert. Zu diesem Zweck wurden Diplom- und Studienarbeiten vergeben, die noch nicht abgeschlossen sind.

2.2 Zielsetzung

Da es sich bei Agentensystemen noch um eine vergleichsweise neue Idee handelt, ist es sinnvoll, bei der Entwicklung einer Applikation großen Wert auf *Flexibilität* zu legen.

Diese Flexibilität wird durch folgende Eigenschaften von ffMAIN erreicht:

1. Es werden keine Annahmen über eine bestimmte Agentensprache gemacht.
2. Sprachunabhängige, einfache Kommunikationsmechanismen für Agenten.
3. Einfache Mechanismen, als Grundlage für spätere Erweiterungen.

Man war außerdem interessiert an der Frage, wie die minimale Unterstützung aussehen muß, um ein Agentensystem betreiben zu können. Diese Überlegungen sollten zu einer einfachen Basisstruktur führen, deren Konzepte deutlich zu erkennen sind. Nebenbei wird dadurch erreicht, daß sich die Fehleranfälligkeit in Grenzen hält.

2.3 Agentenbestandteile

Bei ffMAIN besteht ein Agent aus Code, Zustand und Attributen. Bei der Übertragung werden alle Teile des Agenten zusammen über das Netzwerk gesendet. Bei ffMAIN wird also eine *vollständige Migration* durchgeführt.

2.3.1 Code

Da bei ffMAIN großer Wert auf Flexibilität gelegt wurde, können die Agenten prinzipiell in unterschiedlichen Sprachen geschrieben sein. Momentan wird jedoch nur die Sprache *Tcl* optimal unterstützt. Für *Perl* existiert eine experimentelle Unterstützung, an der Unterstützung von *Java* wird z.Zt. gearbeitet. Nach der Migration des Agenten wird der Code auf dem neuen Rechner ausgeführt. Voraussetzung hierfür ist, wie bereits oben erwähnt, das Vorhandensein einer Infrastruktur für Agenten.

2.3.2 Zustand

Der Sinn von Agenten liegt u.a. in der Möglichkeit, Informationen im Namen des Eigentümers zu sammeln. Die gesammelte Information wird im Zustand des Agenten gespeichert und bei einer Migration mitgeführt, so daß die Informationen immer umfangreicher werden. Der Agent hat also vollen Zugriff auf seine Zustandsvariablen. Der Zustand kann jederzeit vom Eigentümer abgefragt werden.

2.3.3 Attribute

In den Attributen sind Informationen über den Agenten gespeichert. Der Agent kann diese Daten lesen, verändern darf er sie aber nicht. Es sind Daten über Eigentümer, Herkunftsadresse, Startdatum, Verfalldatum, Liste der bereits besuchten Rechnern, usw. Die Infrastruktur kann z.B. anhand dieser Daten, den Eigentümer über den Aufenthaltsort des Agenten informieren.

```
Owner: baeurle@trick.informatik.uni-stuttgart.de
Agent-ID: <a007.960211113000@trick.informatik.uni-stuttgart.de>
Home-URL: http://trick.informatik.uni-stuttgart.de:5055/home/a007/
Authentication-Cookie: 1234567890ab1234567890ab12345678
```

Beispiel 1: Die Attribute eines Agenten

2.4 Transportprotokoll

Die Basis des Agentensystems bildet das Transportprotokoll HTTP (*Hypertext Transfer Protocol* [3]).

Dieses Protokoll eignet sich für ein Agentensystem:

- Weit verbreitetes Protokoll (Basis des WWW[1])
- Agententransport mit vorhandenen HTTP-Methoden möglich
- Erweiterungen können definiert werden
- Nutzung von Ergebnissen aus der WWW-Forschung (Verschlüsselung, Authentifizierung, Elektronische Geldbörse)
- Gewohnte Umgebung für den Anwender, denn vorhandene WWW-Browser können als Benutungsschnittstelle für Agenten verwendet werden
- Integration von WWW- und Agentendiensten leicht möglich

3 Bestandteile des Agentensystems

ffMAIN besteht aus folgenden Komponenten:

- *Agentenserver*: Der zentrale Bestandteil von ffMAIN ist der Agentenserver.
- *Spracherweiterungen*: Für spezielle Agentenkommandos, welche die Kommunikation und die Migration ermöglichen, gibt es Spracherweiterungen für jede unterstützte Sprache.
- *Startprogramm*: Um einen Agenten erstmalig einem Agentenserver zur Ausführung zu übergeben, gibt es ein Startprogramm. Es wird vom Eigentümer des Agenten, unter Angabe der Attribute und des Programmcodes, aufgerufen.
- Zur Abfrage des Informationsraumes und der laufenden Agenten gibt es einen „Agenten-Visualisierer“ mit dem Namen „avis“. Er dient hauptsächlich zur Fehlersuche und ist wohl mehr für den Entwickler des Agentensystems gedacht. Deshalb wird er im Rahmen dieser Ausarbeitung nicht weiter behandelt, zumal es momentan auch keinerlei Dokumentation zu „avis“ gibt.

3.1 Der Agentenserver

Der Agentenserver bietet die geforderte Infrastruktur für Agenten und muß deshalb auf jedem Rechner laufen, der Agenten unterstützten will. Er ist in der Sprache *Perl* geschrieben, wobei der Quellcode ca. 1500 Zeilen umfaßt.

[1] World Wide Web

3.1.1 Die Aufgaben des Agentenservers

- Nimmt neue Agenten entgegen und führt deren Code in einer sicheren Umgebung aus. Der Heimserver des Agenten wird über den neuen Aufenthaltsort des Agenten informiert.
- Ermöglicht die Kommunikation der Agenten untereinander durch die Einführung eines *Informationsraumes*, auf den die Agenten durch spezielle Kommandos Zugriff haben.
- Ermöglicht Statusabfragen des Eigentümers an seinen Agenten.
- Ermöglicht den Agenten die Migration auf einen anderen Server.
- Versendet Mail im Auftrag des Agenten, z.B. um den Eigentümer zu informieren (asynchron).
- Überwacht die Ausführung von Agentencode.

3.1.2 Der Informationsraum

Da prinzipiell Agenten in unterschiedlichen Sprachen unterstützt werden, muß ein allgemeiner Mechanismus zur Kommunikation bereitgestellt werden, der von jeder unterstützten Sprache genutzt werden kann. In ffMAIN wird dies durch die Einführung eines Informationsraumes erreicht. Dieser wird von dem Agentenserver erstellt und verwaltet. Er ist nur den *lokalen* Agenten zugänglich und dient hauptsächlich zur Kommunikation der Agenten untereinander. Agenten können dort Informationen ablegen, auf die andere Agenten dann zugreifen können.

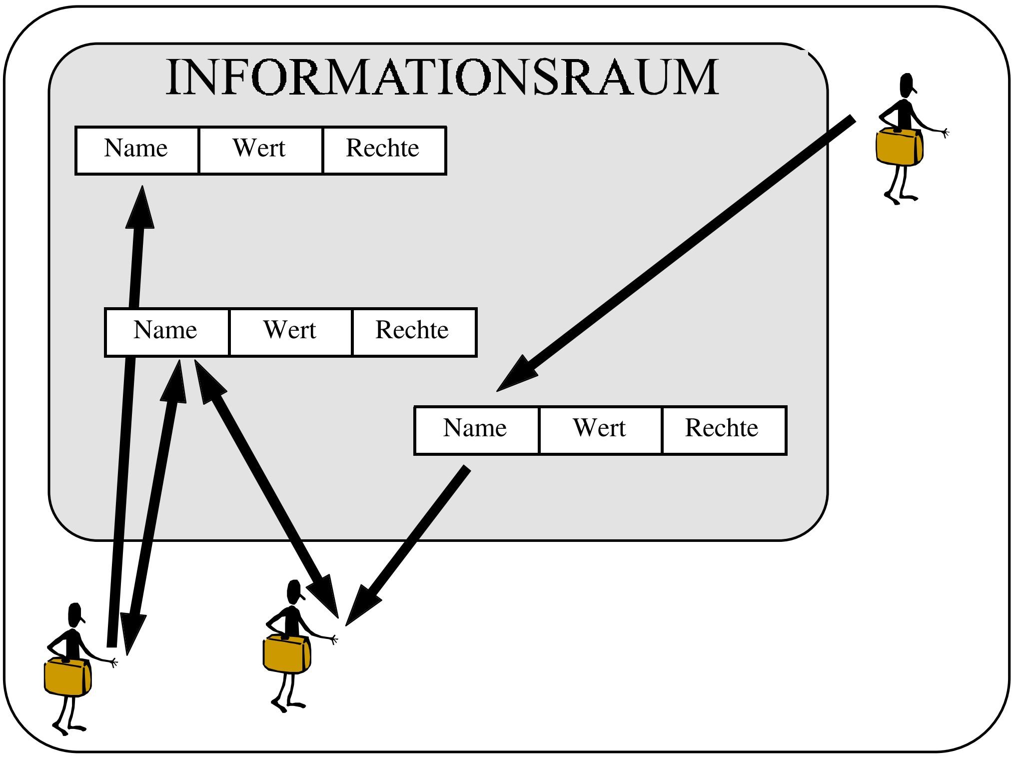

Abbildung 1: Agentenkommunikation über den Informationsraum

Ein Element des Informationsraums ist ein Tripel bestehend aus Name, Wert und Zugriffsrechte. Wenn ein Agent in diesem Informationsraum ein Element neu erzeugt, dann kann er dabei angeben, welche Agenten die Erlaubnis haben, diese Information zu lesen. Wird bei der Erstellung eines Elementes keine Angabe über die Zugriffsrechte gemacht, so bekommen nur Agenten des gleichen Typs Zugriff auf dieses Element.
Außer dem normalen Lesezugriff gibt es noch die Möglichkeit, eine Information zu lesen und anschließend aus dem Informationsraum zu entfernen. Dies wird *destruktives Lesen* genannt. Durch destruktives Lesen wird ein Element aus dem Informationsraum entfernt. Zur Änderung eines Elementes muß dieses destruktiv gelesen und dann wieder neu geschrieben werden. Ein Kommando für Änderungen im Informationsraum gibt es nicht.
Agenten, die sich für bestimmte Elemente im Informationsraum interessieren, können einen Namen oder Bereiche von Namen von dem Agentenserver überwachen lassen. Dieser wird dann den Agenten benachrichtigen, sobald sich Änderungen an den überwachten Elementen ergeben haben (siehe 4.2 Kommunikation).
Der Informationsraum dient aber nicht ausschließlich der Kommunikation, sondern er kann auch von dem Agentenserver benutzt werden, um Informationen zu veröffentlichen. So wäre es z.B. denkbar, daß der Agentenserver eine Liste mit den Adressen benachbarter Agentenserver veröffentlicht. Diese Information kann von den Agenten zur weiteren Wegewahl verwendet werden.

3.1.3 Agentenverwaltung

Jeder neue Agent bekommt eine eindeutige Identifikationsnummer (ID) zugewiesen. Die ID's der anwesenden Agenten werden im Informationsraum veröffentlicht, damit sie von anderen Agenten kontaktiert werden können.

3.1.4 Laufzeitumgebung

Zur Ausführung eines Agenten startet der Agentenserver den passenden Interpreter in einem separaten Prozeß. In dieser Laufzeitumgebung wird der Agentencode ausgeführt. Es stehen dem Agenten zusätzliche Kommandos für Migration und Kommunikation zur Verfügung. Hierfür wird beim Start des Interpreters eine spezielle Spracherweiterung eingebunden.

3.2 Spracherweiterungen

Bei den Spracherweiterungen handelt es sich um Laufzeitbibliotheken, die von dem entsprechenden Interpreter eingebunden werden. Die Spracherweiterungen haben die Aufgabe, die zusätzlich benötigten Kommandos zur Agentenmigration und Agentenkommunikation für die jeweilige Sprache bereitzustellen.
Eine Spracherweiterung existiert z.Zt. für die Sprachen *Tcl*, *Incr-Tcl* und *Perl*, wobei die Erweiterungen für *Incr-Tcl* und *Perl* nur experimentell implementiert und daher unvollständig sind.

3.2.1 Spracherweiterung für Tcl

Beim Start des Tcl-Interpreters wird das Kommando *agent* definiert. Es besitzt eine Vielzahl von Optionen, die für die Migration und Kommunikation erforderlich sind.

agent moveto <URL>	Migration durchführen
agent put/get/dget	Element im Informationsraum erzeugen/ lesen/ löschen
agent subscribe <bereich>	Überwachung bestimmter Elemente. Bei Änderungen bekommt der Agent einen Hinweis.
agent fetch <hinweis>	Liefert den Namen des geänderten Elements.
agent mail <text>	Email an den Eigentümer

Tabelle 1: Eine Auswahl der neuen Kommandos mit kurzer Beschreibung

3.3 Startprogramm

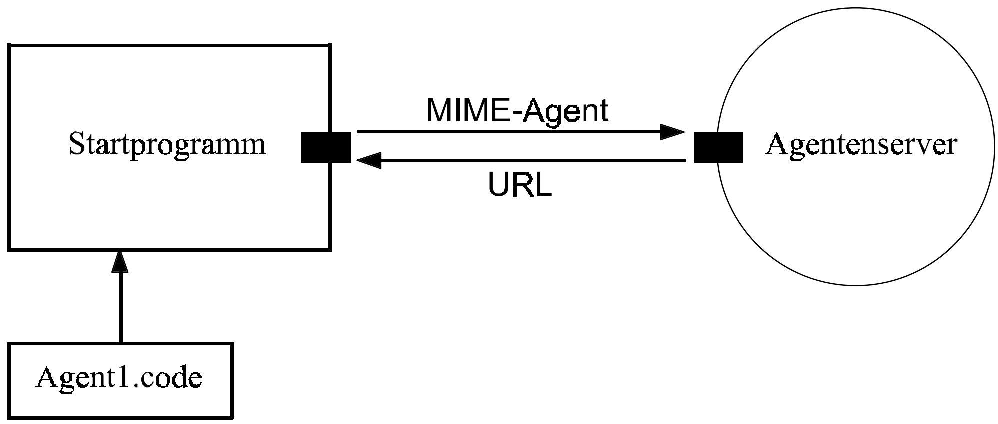

Abbildung 2: Ablaufschema zur Übergabe eines neuen Agenten

Mit dem Programm „*asubmit*“ wird dem Agentenserver ein neuer Agent zur Ausführung übergeben. Der Agent wird im MIME-Format mittels der HTTP-Methode POST an den lokalen Agentenserver gesendet. Dieser prüft die Attribute und erstellt eine Identifikationsnummer. Als Rückgabe erhält das Client-Programm „*asubmit*“ eine sogenannte Heimat-URL. Mit dieser Adresse kann der Eigentümer mit seinem Agenten kommunizieren. Die URL (*Uniform Ressource Locator* [4]) wird vom Agentenserver verwaltet. Wenn der Agent migriert, wird die URL ungültig. Deshalb gehört es zu den Aufgaben des Agentenservers für jeden ankommenden Agenten, dessen Herkunfts-Agentenserver über den neuen Aufenthaltsort zu informieren. Hierfür vergibt der neue Agentenserver eine temporäre Besucher-URL. Wenn nun der Eigentümer mittels der ihm übermittelten Heimat-URL seinen Agenten abfragen will, wird seine Anfrage von dem Agentenserver zu der temporären Besucher-URL umgeleitet.

4 Funktionsweise

4.1 Migration

In HTTP werden die Daten in einem besonderen Format, das auf MIME (*Multipurpose Internet Mail Extensions*,[5]) basiert, übertragen. Für den Transport der Agenten wird ein neuer Inhaltstyp (content type: application/agent), mit Subtypen für die einzelnen Bestandteile, eingeführt. Zusätzliche MIME-Attribute der Nachricht geben Auskunft über die Sprache und den Typ des Agenten.

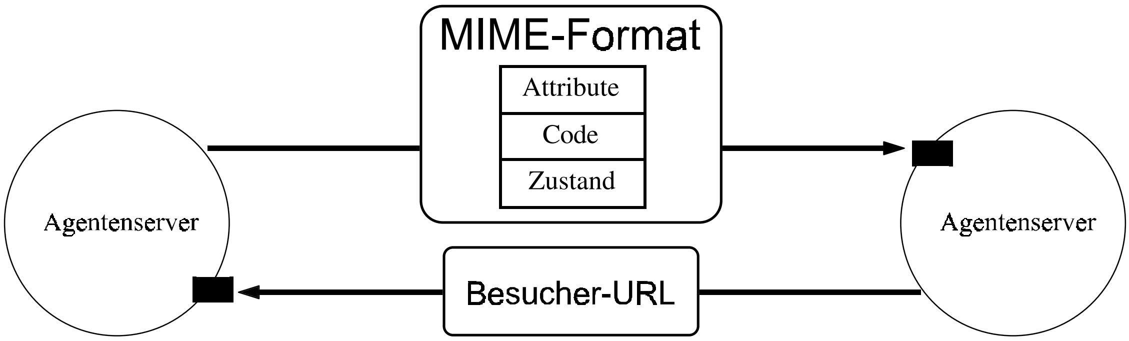

Abbildung 3: Agentenmigration mit HTTP

Die 3 Bestandteile des Agenten werden in das MIME-Format eingeschlossen und dann mit der HTTP-Methode POST an den neuen Agentenserver gesendet. Der neue Agentenserver verwendet ebenfalls die HTTP-Methode POST, um den Heim-Agentenserver die neue Besucher-URL des Agenten mitzuteilen. So wird gewährleistet, daß der Eigentümer den aktuellen Aufenthaltsort seines Agenten erfahren kann.
Damit der Agent auf dem entfernten Rechner direkt vom Agentensystem entgegengenommen werden kann, ist es notwendig, daß das Agentensystem auf einem bekannten Port zu erreichen ist. Dieser Port muß explizit angegeben werden, sonst wird standardmäßig von HTTP der Port 80 verwendet. Das Agentensystem ffMAIN verwendet üblicherweise den Port 5055.

4.2 Kommunikation

Die Fähigkeit, mit ihrer Umwelt zu kommunizieren, gehört zu den wichtigsten Eigenschaften mobiler Agenten. Es gibt unterschiedliche Situationen, in denen Kommunikation erforderlich ist.
Grundsätzlich ist der Agentenserver an jeder Kommunikation beteiligt, da er alle Agenten verwaltet.

4.2.1 Agent mit Agentenserver

Der Agentenserver wird immer über seine URL-Adresse und seinen Port angesprochen. Dabei werden die HTTP-Methoden POST oder GET verwendet. Sei *server* die URL-Adresse des Rechners und *port* der Port den der Agentenserver überwacht, dann ist *server:port/info?key* ein Zugriff auf ein Element *key* des Informationsraums. Mit POST wird ein neues Element erzeugt und mit GET ein bestehendes Element gelesen. Das HTTP wurde hier noch um die Methode DGET erweitert, mit der das destruktive Lesen ermöglicht wird.

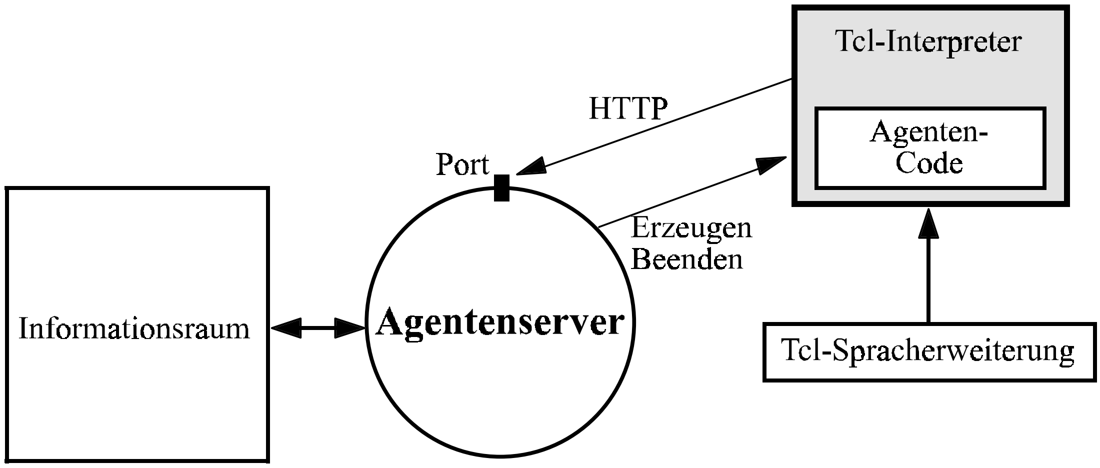

Abbildung 4: Kommunikation über den Agentenserver

4.2.2 Agent mit Agent

In *Tcl* ist es mit dem Kommando *send* möglich, mit einem anderen *Tcl*-Interpreter Nachrichten auszutauschen. Da ffMAIN aber unterschiedliche Programmiersprachen unterstützt, wird diese Möglichkeit nicht für die Agentenkommunikation genutzt. Zu diesem Zweck wird ein neutraler Informationsraum (siehe 3.1.2) vom Agentenserver verwaltet, über den die Agenten miteinander kommunizieren können. Hierfür stehen den Agenten verschiedenen Kommandos zur Verfügung. Sie können damit Elemente erzeugen, lesen, löschen oder ändern.

Jeder Agent, der auf Anfragen anderer Agenten wartet, überwacht mindestens einen Namensraum, nämlich alle Elemente, deren Namen vom Muster „*ID:req-**“ sind. Wenn ein Agent mit der Identifikationsnummer ID1 mit dem Agent ID2 kommunizieren will, so erzeugt er im Informationsraum ein Element mit dem Namen „*ID2:req-ID1*“. Da der Agent ID2 diesen Namensraum vom Agentenserver überwachen läßt, wird er über diese Änderung im Informationsraum benachrichtigt und kann entsprechend reagieren. Agent ID2 kennt nun seinen Kommunikationspartner. Der anfragende Agent ID1 erwartet eine Antwort von ID2 in dem Element „*ID2:repl-ID1*“. Um das Eintreffen der Antwort zu registrieren, muß er den dieses Element vom Agentenserver überwachen lassen.

Zur Verdeutlichung hier als Beispiel das *Tcl*-Programm des Agenten ID1. Agent ID1 sendet an ID2 die Nachricht *xxx* und wartet auf eine Antwort von ID2.

`set ID1 [agent attrib Local-ID]`	Eigene ID aus Attributen lesen
`set f_id [agent subscribe ID2:`**`repl`**`-ID1]`	Überwachung eines Elements im IR
`agent put ID2:`**`req`**`-ID1` *`xxx`*	Erzeugen einer Anfrage
`set Elementname [agent fetch $f_id]`	Warten auf Antwort
`set Antwort [agent dget $Elementname]`	Antwort destruktiv lesen
`agent unsubscribe $f_id`	Überwachung beenden

Beispiel 2: Programm für Agent ID1 in *Tcl*

Es ist geplant, die Kommunikation durch ein neues Kommando zu vereinfachen. Dann genügt das Kommando „*`agent request ID2 xxx`*" um mit Agent ID2 zu kommunizieren.

Agent ID2 ist ein Übersetzer von Englisch in Deutsch. Das Kommando „*`agent request ID2`* ***`recipient`***" liefert als Rückgabewert „Empfänger".

Beispiel 3: Agent ID2 als Wörterbuch

Die angebotenen Dienste sind noch nicht standardisiert. Damit ein Agent einen bestimmten Dienst nutzen kann, muß er vorher den Namen des Dienstes wissen. Man könnte sich Diensttypen vorstellen, wie z.B. Uhrzeit, Dateisuche, Buchsuche, Wörterbuch/Englisch. Dann könnte der Agent nach dem gewünschten Dienst in der Agentenliste suchen und dann an den entsprechenden Agenten eine Anfrage stellen.
Für die Suche nach einem bestimmten Dienst, muß in der vorliegenden Version des Agentensystems noch die Agentenliste im Informationsraum durchsucht werden. Auch hier ist ein vereinfachendes Kommando geplant. Das Kommando „*`agent find info`*" gibt die ID des Agenten zurück der den Dienst *info* anbietet.

4.2.3 Agent mit Eigentümer

Ein Agent kann seinem Eigentümer Nachrichten als Email senden. Dies hat den Vorteil, daß der Eigentümer nicht anwesend sein muß. Zur Kommunikation muß der Eigentümer den Agenten über seine Heimat-URL ansprechen. Der entfernte Agentenserver übermittelt dem Eigentümer, nach Überprüfung der Berechtigung, den aktuellen Zustand und die Attribute des Agenten. Zur komfortablen Anzeige der Daten, kann der Agent dann synchron eine HTML-Seite an den Eigentümer senden. Dies kann z.B. ein Formular sein, das direkt an den Agenten zurückgeschickt wird.

5 Sicherheitsaspekte

Die verwendeten Sicherheitsmechanismen in ffMAIN sind relativ einfach gehalten. Für die meisten Fälle von Mißbrauch sind sie aber ausreichend.

5.1 Ausführung eines Agenten

Bei der Ausführung eines Agenten ist es notwendig, den Rechner vor unerlaubten Zugriffen zu schützen. Es darf dem Agenten z.B. nicht möglich sein, die Festplatte zu formatieren oder eine Paßwortdatei einzulesen. Um dies zu verhindern müssen solche *unsicheren* Befehle verboten werden. Auch die Auswertung von Parametern bestimmter Befehle ist eine Möglichkeit, Mißbrauch zu verhindern.
In der Sprache *Tcl* gibt es die Möglichkeit mit dem Kommando *„interp“* einen untergeordneten Interpreter aufzurufen. Mit Anwendung der Option *-safe* wird ein sicherer Interpreter erstellt, der *unsichere* Befehle nicht mehr erlaubt, oder dem übergeordneten Prozeß zu Auswertung übergibt. Dies wurde bei ffMAIN realisiert. Bei anderen Sprachen gibt es diese Möglichkeit nicht. Hier muß die Ausführung vom Agentenserver überwacht werden; so realisiert bei der (noch experimentellen) Sprachunterstützung für *Perl*. Zu diesem Zweck läuft im Hintergrund ein Prozeß, der den Agentenserver sofort informiert, wenn ein Ereignis wie z.B. ein Dateizugriff erfolgt. Der Agentenserver kann dann über die Ausführung der entsprechenden Operation entscheiden.

5.2 Übertragung eines Agenten

Die Sicherheit bei der Übertragung von Agenten auf einen entfernten Rechner, ist abhängig von der Sicherheit des verwendeten Transportprotokolls. In ffMAIN werden Agenten mit HTTP übertragen. Die Sicherheit von HTTP ist nur bei einem sicheren Übertragungskanal gegeben, den es im Internet aber nicht gibt. Die Entwickler von ffMAIN verweisen auf die intensive Forschung und Entwicklung auf diesem Gebiet. Die Anbieter von Dienstleistungen im Internet verlangen einen einheitlichen, sicheren Mechanismus zur Übertragung vertraulicher Daten, wie z.B. Kreditkarten- oder Kontonummern. Sobald die WWW-Gemeinschaft sich auf einen einheitlichen Sicherheitsstandard geeinigt hat, kann dieser von ffMAIN genutzt werden.

5.3 Statusabfrage des Eigentümers

Bei einer Statusabfrage des Eigentümers wird der Agentenserver kontaktiert. Jeder Agent hat in seinen Attributen einen 32 Zeichen langen Zufallsstring, der zur Authentifizierung vom Eigentümer präsentiert werden muß. Nur der Eigentümer und die beteiligten Agentensysteme kennen diesen String.

5.4 Informationsraum

Der Informationsraum eines Agentenservers ist nur den lokalen Agenten zugänglich. Auch hier wird ein Zufallsstring verwendet, der zwischen Agentenserver und Laufzeitumgebung beim Start eines Agenten vereinbart wird. Die Authentifizierung wird von der Laufzeitumgebung, für den Agenten unsichtbar, durchgeführt. Der Agent kennt diesen String also nicht.

6 Zusammenfassung

Mit dem Agentensystem ffMAIN wird auf dem Gebiet der mobilen Agenten experimentiert. Es soll als Grundlage für ein komfortables Agentensystem dienen. Man will keine neuen Standards für Agentensysteme erstellen, sondern bereits etablierte Standards (MIME, HTTP, HTML) verwenden, um damit eine Grundfunktionalität für Agentensysteme bereitzustellen. Durch die konsequente Einbindung vorhandener Standards, ist es möglich, Agentendienste in vorhandene Software zu integrieren. Speziell eine Integration in bestehende Web-Browser wäre denkbar. Agenten könnten dann auf einfache Weise über eine HTML-Seite gestartet werden. Hierfür wird es in einer neueren Version von ffMAIN eine Oberfläche geben, mit der neue Agenten entwickelt, gestartet und überwacht werden können.

Literaturverzeichnis

[1] Lingnau, A. and Drobnik, 0., *An Infrastructure for Mobile Agents: Requirements and Architecture*, Proc. 13th DIS Workshop, Orlando, Florida, September 1995. Internet Ressource: „ftp://ftp.tm.informatik.uni-frankfurt.de/pub/papers/agents/13dis-paper.ps.gz“

[2] Lingnau, A., Drobnik, O. and Dömel, P., *An HTTP-based Infrastructure for Mobile Agents*. Internet Ressource: „ftp://ftp.tm.informatik.uni-frankfurt.de/pub/papers/agents/www4-paper.ps.gz“

[3] Berners-Lee, T., R. T. Fielding, and H. Frystyk Nielsen, *Hypertext Transfer Protocol - HTTP/1.0*, Internet Draft draft-ietf-http-v10-spec-00, HTTP Working Group, März 1995.

[4] Berners-Lee, T., L. Masinter, and M. McCahill, *Uniform Resource Locators (URL)*, RFC 1738, Network Working Group, Dezember 1994.

[5] Borenstein, N. and N. Freed, *MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, RFC 1521, Network Working Group, September 1993.

Abbildungsverzeichnis

Abbildung 1: Agentenkommunikation über den Informationsraum *5*
Abbildung 2: Ablaufschema zur Übergabe eines neuen Agenten *7*
Abbildung 3: Agentenmigration mit HTTP *8*
Abbildung 4: Kommunikation über den Agentenserver *9*

Beispiel-Verzeichnis

Beispiel 1: Die Attribute eines Agenten *3*
Beispiel 2: Programm für Agent ID1 in Tcl *10*
Beispiel 3: Agent ID2 als Wörterbuch *10*

Anhang

Beispielagent:

Folgendes Agentenprogramm stellt einen Agent dar, der auf der Suche nach einer Datei 'hallo.html' ist. Zum Auffinden der Datei, benötigt er die Hilfe eines speziellen Dienstagenten Namens „DateiServerAgent". Das Programm des Dienstagenten befindet sich auf der nächsten Seite.

#DateiSuchAgent	
`set meineID [agent attrib Local-ID]`	Lokale ID aus den Attributen lesen.
`set AListe [agent get server:agents]`	Das reservierte Element *server:agents* aus dem IR enthält eine Liste der Agenten. Jeder Listen-Eintrag hat die Form: ID Typ Kontext Eigentümer Info. Im Info-Feld steht der evtl. angebotene Dienst.
`foreach Agent [split $AListe \n] {`	Für jeden Agent aus der Liste:
`  if {[string compare` `  [lindex $Agent 4] DateiServerAgent] == 0} {`	Wenn im Info-Feld der Dienst 'DateiSuchAgent' steht:
`    set DienstID [lindex $Agent 0]`	Die ID des Dienst-Agenten merken.
`    set Nummer` `        [agent subscribe $DienstID:repl-$meineID]`	Das reservierte Element für eine Antwort *DienstID:repl-KundeID* auf Änderungen überwachen lassen.
`    Agent put $DienstID:req-$meineID hallo.html`	Das reservierte Element für eine Anfrage *DienstID:req-KundeID* im IR erzeugen und mit dem Namen der gesuchten Datei ('hallo.html')belegen.
`    set Elementname [agent fetch $Nummer]`	Warten auf Änderung bei Antwort-Element. Bei Änderung gibt „fetch" den Namen des geänderten Elements zurück.
`    set DateiURL [agent dget $Elementname]`	Antwort-Element destruktiv lesen. Annahme: Enthält bei Vorhandensein der Datei deren URL, sonst leerer String.
`    if {[string compare $DateiURL NULL] == 0} {`	Wenn Datei nicht gefunden:
`      break`	Schleife beenden.
`    }`	
`    else {`	Wenn Datei gefunden:
`      agent mail $DateiURL`	URL der Datei an den Eigentümer mailen.
`    }`	
`  }`	
`}`	
`#MIGRIEREN`	
`if [llength $ASListe] {`	Wenn die Liste der Agentenserver nicht leer ist:
`  set NächsterAS [lindex $ServerListe 0]`	Erstes Element der Liste als nächstes Ziel auswählen.
`  set $ASListe [lreplace $ASListe 0 0]`	Erstes Element löschen.
`  agent moveto $NächsterAS`	Migriere zum nächsten Agentenserver.
`}`	
`else {`	Wenn Liste der Agentenserver leer ist:
`  agent quit`	Beende den Agenten ohne Ergebnis.
`}`	

#DateiServerAgent	
`set meineID [agent attrib Local-ID]`	Lokale ID aus den Attributen lesen.
`agent put server:info-$meineID DateiServerAgent`	Über das reservierte Element für Dienstangebote *server:info-ID* dem Agentenserver den neuen Dienst mitteilen.
`set Nummer [agent subscribe $meineID:req-*]`	Alle reservierten Elemente für eine Anfrage *DienstID:req-** auf Änderungen überwachen lassen.
`while 1 {`	Beginn der Endlosschleife
`  set Elementname [agent fetch $Nummer]`	Warten auf Änderung bei einem Anfrage-Element. Bei Änderung gibt „fetch“ den Namen des geänderten Elements zurück.
`  set Dateiname [agent dget $Elementname]`	Dateiname aus Anfrage-Element destruktiv lesen.

`  set KundeID`
`    [string range $Elementname`
`    [expr [string last - $Elementname] + 1] end]`
	Die ID des anfragenden Agenten muß aus dem Namen des Anfrage-Elements extrahiert werden.

`  set $DateiURL [SUCHE $Dateiname]`	Prozedur SUCHE aufrufen. Liefert bei Erfolg die URL-Adresse der gesuchten Datei. Sonst wird ein leerer String zurückgegeben..
`  agent put $meineID:repl-$KundeID $DateiURL`	Das reservierte Element für eine Antwort *ID1:repl-ID2* im IR erzeugen und mit der Variablen DateiURL belegen.
`}`	