Werkauftrag

Evaluation des Kommunikationstools ICE und testweise Integration in das Multiagentensystem VIENA

14. November 1996

Werkauftragnehmer:

Ivo Schnüll

unter Mitarbeit von:

Liutger Franzen

Inhaltsverzeichnis

1 Einleitung **2**

2 Die Kommunikation unter VIENA **2**

3 Vorstellung von ICE **4**

3.1 Die Grundlagen von ICE 5

3.2 Höhere Kommunikationsschichten 6

3.3 Die Funktionsweise der Kommunikation unter ICE 7

4 Ansätze zur Integration von ICE **8**

4.1 Möglichkeiten zur Integration 9

4.1.1 Die Funktion Komm_anmelden(...) 9

4.1.2 Die Funktion Komm_abmelden(...) 10

4.1.3 Die Funktion Komm_senden(...) 10

4.1.4 Die Funktion Komm_empfangen(...) 11

4.2 Konkrete Ansätze 11

4.2.1 Nachbildung des Komm_Servers unter ICE 11

4.2.2 Realisierung einer dezentralen Struktur 14

4.3 Nicht absehbare Fehlerquellen 17

5 Performance-Messungen **18**

6 Schlußbetrachtung **21**

6.1 Theoretische Überlegungen 21

6.2 Praktische Erfahrungen 22

1 Einleitung

Ziel dieser Arbeit ist es, die Möglichkeiten und Fähigkeiten des Kommunikationswerkzeuges ICE[1] zu testen. Dabei soll zum einen überprüft werden, ob der bestehende Kommunikationsmechanismus von VIENA mit Hilfe von ICE ersetzt werden kann, und zum anderen, inwieweit sich ICE für den Einsatz in zukünftigen Projekten eignet.

Dazu werden zunächst das bestehende Kommunikationssystem von VIENA und das Tool ICE vorgestellt. Im Anschluß daran wird ein Vergleich zwischen beiden Systemen durchgeführt; dieser wird deutlich machen, wo die Probleme bei der Integration von ICE liegen. In diesem Zusammenhang werden verschiedene Möglichkeiten zur Lösung dieser Probleme erläutert, bevor dann Performance-Messungen unter ICE durchgeführt und ein Ausblick auf mögliche Erweiterungen des Kommunikationssystems geben werden.

Alle Angaben und Informationen sind, soweit nicht anders angegeben, aus der Dokumentation zu ICE und aus der Diplomarbeit von Norbert Siekmann ("Spezifikation und Implementierung eines Kommunikations-Subsystems für Mehragentensysteme") entnommen.

Die aktuellen Quelltexte und die Dokumentation für unsere Implementierung finden sich im Projektverzeichnis von VIENA.

2 Die Kommunikation unter VIENA

Im bisher bestehenden VIENA-System werden die Daten über einen Kommunikationsserver transportiert. In der dem Server zugrundeliegenden Struktur wird dies mit Hilfe von RPC[2] realisiert. Die potentiellen Teilnehmer einer Kommunikation verfügen dabei über die folgenden vier Funktionen, mit Hilfe derer sie mit dem Server in Kontakt treten können:

- **Komm_anmelden(...)**; diese Funktion wird vom Agenten aufgerufen, um sich beim Server anzumelden. Er übergibt dabei als Parameter unter anderem den Namen der Agentur, für die er sich anmelden möchte.

[1] **I**NTARC **C**ommunication **E**nvironment

[2] **R**emote **P**rocedure **C**alls

Der Server liefert im Falle einer erfolgreichen Anmeldung den Agenturnamen, versehen mit einer Nummer, zurück. Dies ist der Name, unter dem der Agent später angesprochen werden kann. Der Server vergibt diesen Namen einmalig; auch wenn der Agent sich später abmeldet, wird dieser Name nicht mehr vergeben. Der erste Agent einer Agentur, der sich anmeldet, bekommt die Nummer "1" zugewiesen. Diese Nummer wird danach automatisch vom Server hochgezählt, d.h. die Nummern sind aufsteigend und aufeinanderfolgend.

- **Komm_abmelden(...)**; diese Funktion ruft der Agent auf, sobald er sich vom Server abmelden möchte. Er kann dies tun, wenn beispielsweise seine Arbeit beendet ist, oder wenn er vom Server die Aufforderung erhalten hat, sich abzumelden (z.B. weil VIENA beendet werden soll).

- **Komm_senden(...)** wird von den Agenten zum Übertragen von Nachrichten benutzt. Der Agent kann dabei als Empfänger einen einzelnen Agenten, eine Agentur oder alle am System angemeldeten Agenten angeben.

- **Komm_empfangen(...)** dient zum Empfangen von Nachrichten. Der Agent muß diese Funktion explizit aufrufen, um Nachrichten zu empfangen; es wird also kein Event ausgelöst, welches diese Funktion automatisch ausführt. Ein Agent muß ständig prüfen, ob neue Nachrichten für ihn vorhanden sind. Sollte keine Nachricht für den Agenten vorhanden sein, ist dies eine Aufgabe, die das System unnötig belastet, da jedesmal ein RPC ausgeführt wird. Um die Belastung des Systems dennoch möglichst niedrig zu halten, werden, wenn keine Nachrichten vorliegen, die Zeitabstände, in welchen der Agent nach neuen Nachrichten fragt, dynamisch bis zu einer maximalen Zeitspanne erhöht.

Die Kommunikation findet dabei asynchron statt, d.h. sie ist nicht blockierend. Ein Agent kann allerdings, wenn er z.B. mit der Durchführung der ihm gestellten Aufgabe beschäftigt ist, das System dadurch blockieren, indem er seine Nachrichten nicht abholt. Es gibt also keine definierte Zeit, in der ein Agent seine Nachrichten in jedem Fall abholen soll. Dies ist allerdings ein

Problem der Kooperationsschicht von VIENA und nicht der Kommunikationsschicht. Es kann also durch eine andere Form der Kommunikation nicht gelöst werden.

Die Datenpakete, die unter VIENA übertragen werden, bestehen aus sogenannten Kommlisten. Diese bestehen ihrerseits aus verketteten Zeichenfolgen. Für die Übertragung werden diese in einen einzigen String konvertiert, welcher beim Empfänger automatisch zurückkonvertiert wird.

Eine Aufgabe, die der Kommunikationsserver übernimmt, ist die Nummerierung der Nachrichten. Alle verschickten Nachrichten werden mit einer Kennung versehen. Diese Kennung besteht aus einem Teil, der den Auftrag spezifiziert, auf welchen sich die Nachrichten beziehen, und aus einem weiteren Teil, der eine systemweite eindeutige Nachrichtenummer enthält. Der zweite Teil der Kennung wird vom Server vergeben. Zu diesem Zweck ist im Server ein Zähler implementiert, der automatisch bei jeder neuen Nachricht hochgezählt wird. Diese Kennung ist also, genau wie die Vergabe der Nummern für einzelne Agenten, aufsteigend und aufeinanderfolgend.

Der bestehende Kommunikationsserver soll ersetzt werden, weil er sich aufgrund des steigenden Nachrichtenaufkommens, das seinen Ursprung in den multimedialen Erweiterungen von VIENA hat, immer mehr als Nadelöhr (zentralistische Struktur) erweist.

Ferner bietet das Laufzeitverhalten des Kommunikationsservers Anlaß zur Kritik, da durch die benutzten Polling-Mechanismen und die Verwendung von RPC rechenzeitintensive Methoden Verwendung finden.

3 Vorstellung von ICE

ICE ist im Rahmen des Verbmobil-Projektes an der Universität Hamburg entwickelt worden. Ziel der Entwickler war es, eine einfach zu handhabende, effiziente Kommunikationsumgebung zu schaffen, in der es möglich ist, verteilte Systeme der künstlichen Intelligenz zu entwickeln und zu testen.

3.1 Die Grundlagen von ICE

Als Grundlage für die Implementierung von ICE wählten die Entwickler PVM[3]. Mit Hilfe von PVM ist es möglich, mehrere Maschinen einer heterogenen Netzwerkumgebung zu einer einzigen Maschine zusammenzufassen. Auf jedem Computer, der Teil dieser großen Maschine sein soll, muß ein Daemon gestartet werden, welcher unter anderem dazu dient, Nachrichten zu sammeln. Diese Funktionalität bildet die Grundlage des Kommunikationsmodells von PVM, welches die folgenden Merkmale aufweist:

- Die Kommunikation ist nachrichtenbezogen und arbeitet asynchron und nicht blockierend, d.h., Nachrichten können verschickt werden, ohne daß auf eine Abnahme oder eine Bestätigung des Empfängers gewartet werden muß.

- Die Kommunikation ist reihenfolge erhaltend, d.h., daß PVM dafür sorgt, daß die Nachrichten den Empfänger in der Reihenfolge erreichen, in der sie verschickt wurden.

- Um Daten zwischen den verschiedenen Rechnerarchitekturen austauschen zu können, wird eine automatische En- und Dekodierung vorgenommen. Diese kann ausgeschaltet werden, wenn die Kommunikation zwischen Rechnern der gleichen Architektur stattfindet. Auf diese Weise kann der Transport von Daten beschleunigt werden.

- PVM besitzt die Fähigkeit, für die jeweilige Architektur die beste Kommunikationsform auszuwählen. In der Regel werden die Daten via TCP/IP über das lokale Netz versendet. Befinden sich die Teilnehmer auf einer Maschine, so werden die Daten über gemeinsam genutzte Speicherbereiche (shared memory) übertragen. Prozesse, die auf einer Maschine laufen, können so erheblich schneller Daten austauschen.

Basierend auf PVM, stellt ICE eine Schnittstelle zur Verfügung, mit der höhere Schichten der Kommunikation verfügbar gemacht werden. Die physikalische Übertragung der Daten wird also über PVM realisiert. Dies ist

[3](**P**arallel **V**irtual **M**achine)

insofern wichtig, da mit PVM automatisch ein weiteres externes Tool mit ICE integriert wird.

3.2 Höhere Kommunikationsschichten

ICE stellt die folgenden erweiterten Möglichkeiten der Kommunikation zur Verfügung:

- Einrichten einer **synchronen Datenübertragung**, d.h. der Sender einer Nachricht wartet, bis diese vom Empfänger abgenommen wurde.
- Kommunikation über **mehrere Kanäle**. Mit deren Hilfe ist es möglich, daß die Teilnehmer beispielsweise zwischen Kontroll- und Datenströmen unterscheiden können.
- Mit Hilfe einer **Sendekontrolle** kann sichergestellt werden, daß nur ein Partner der Kommunikation die Sendeerlaubnis hat. Somit wird verhindert, daß beide nur Daten senden und keiner der beiden seine Daten empfängt.
- Nachrichten können mit Hilfe eines **Broadcasts** an alle Teilnehmer gesendet werden.
- Zentrale Anmeldestelle für alle Teilnehmer bildet der ILS[4]. Dieser verwaltet die Teilnehmer der Kommunikation. Er stellt zum einen sicher, daß Teilnehmernamen nicht doppelt vergeben werden, zum anderen koordiniert er die Kommunikation.
- Funktionen zum Transport komplexer Datentypen (**I**ntarc **D**DL **L**ayer). Mit Hilfe dieser Funktion soll es möglich sein, komplexe Datentypen, die mit Hilfe der **D**ata **D**escription **L**anguage beschrieben werden, zu transportieren. Diese Funktion ist allerdings aufgrund der Schwierigkeiten bei der Realisierung der **DDL** für verschiedene Plattformen und Programmiersprachen bisher nicht implementiert.

Es bleibt zu erwähnen, daß ICE bisher mit Hilfe der Programmiersprachen C, C++, Lisp, TCL und Prolog zu verwenden ist.

[4] **I**NTARC **L**icense **S**erver.

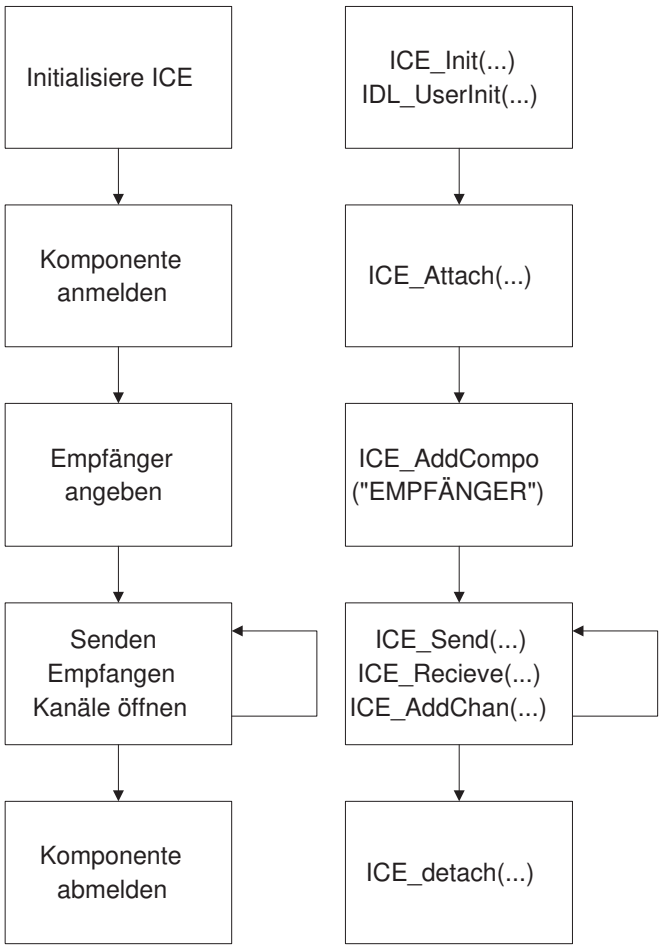

Abbildung 1: Schematischer Ablauf der Kommunikation unter ICE (links) und die hierfür nötigen Funktionen (rechts)

3.3 Die Funktionsweise der Kommunikation unter ICE

An dieser Stelle soll ein Überblick über den Ablauf der Kommunikation unter ICE gegeben werden (siehe Abbildung 1).

Die Teilnehmer einer Kommunikation unter ICE werden als Komponenten bezeichnet. Sobald eine Komponente sich an der Kommunikation beteiligen möchte, muß sie sich beim ILS mit ihrem Namen anmelden. Dabei sorgt der ILS dafür, daß Namen nicht doppelt vergeben werden. Nachdem die Anmeldung erfolgreich abgeschlossen ist, kann die Komponente an der Kommunikation teilnehmen. Um eine Nachricht an eine andere Komponente zu

senden, muß zu dieser ein sogenannter *base channel*, der mit Hilfe der Funktion `Ice_AddCompo` erstellt wird, aufgebaut werden. Über diesen können dann Nachrichten gesendet und empfangen werden. Zusätzlich können bei Bedarf weitere Kanäle zur Kommunikation geöffnet werden. Sobald eine Komponente nicht mehr an der Kommunikation teilnehmen möchte, muß sie sich beim ILS abmelden; dieser gibt daraufhin den Namen der Komponente frei, d.h. ein anderer Teilnehmer kann nun den Namen der abgemeldeten Komponente verwenden. Die Namen für Komponenten können also wieder benutzt werden, es können allerdings keine Komponenten mit gleichen Namen zur gleichen Zeit aktiv sein.

4 Ansätze zur Integration von ICE

Nachdem die Werkzeuge vorgestellt wurden, wird in diesem Kapitel auf die Möglichkeiten und die Schwierigkeiten bei der Integration von ICE in VIENA eingegangen. Dazu werden zunächst explizit die Unterschiede und Gemeinsamkeiten der beiden vorgestellten Systeme in der folgenden Tabelle herausgestellt.

	bestehender Server	**ICE**
Name des Agenten	vergibt der Server	vergibt Agent
Nachrichten IDs	vergibt der Server	nein
1:m Adressierung[5]	ja	nein
Struktur	zentral	dezentral
Sprachen	C	C, C++, Prolog, Lisp, TCL
Eventgesteuert	nein	bisher nur TCL/TK
Broadcast möglich	ja	ja
mehrere Kanäle	nein	ja
Kommunikation	asynchron	asynchron

Es wird deutlich, daß vor allem die ersten drei aufgeführten Punkte zu Schwierigkeiten bei der Implementation führen werden. Die Verfahrensweise der beiden Kommunikationswerkzeuge ist in diesen Punkten konträr. In den

[5] gemeint ist die Adressierung von Gruppen, beispielsweise "VIENA FARBE"

anderen Punkten gleichen sich die Werkzeuge, oder ICE erweitert die bestehenden Möglichkeiten. Hierdurch sollten also keine Schwierigkeiten hervorgerufen werden.

4.1 Möglichkeiten zur Integration

Das Ersetzen des bestehenden Kommunikationssystem sollte im Idealfall völlig so geschehen, daß Änderungen am Sourcecode der Agenten sollten nicht erforderlich sind. Das neue Kommunikationssystem sollte also durch erneutes Linken der Sourcen integriert werden. So kann sichergestellt werden, daß bei unvorhersehbaren Problemen das alte System mit wenig Aufwand wiederhergestellt werden kann.

Als möglicher Ansatzpunkt zur Integration erscheinen die im letzten Kapitel vorgestellten Kommunikationsfunktionen. Wir werden im Folgenden aufzeigen, wie diese Funktionen mit Hilfe von ICE nachgebildet werden können.

4.1.1 Die Funktion Komm_anmelden(...)

Die Funktion bekommt als Übergabeparameter den Agenturnamen, zu welcher der Agent gehören soll, und den Namen des Servers, an welchen er sich anmelden möchte. Als Rückgabe liefert der Server dem Agenten seinen Namen, unter welchem dieser im System zu erreichen ist. Die folgenden Probleme entstehen daher bei der Re-Implementierung mit ICE:

- Unter ICE existiert nur ein Server an dem man sich anmelden kann (die Angabe des Servernamens zur Anmeldung ist also überflüssig); dies führt allerdings zu keinem Problem, da in der bisherigen Version von VIENA immer nur ein Kommunikationsserver läuft. Die Einführung eines weiteren Kommunikationsservers würde zu einer Entlastung des bestehenden Servers führen. Da unter ICE der Server nur koordinierende Aufgaben übernimmt und die Daten nicht physikalisch transportiert, macht die Einführung eines zweiten Servers in diesem Fall keinen Sinn, da die Belastung durch einen erhöhten Koordinationsaufwand im Vergleich zu den höheren Datenmengen gering ist.

- Unter ICE benennt sich ein Agent bei seiner Anmeldung selbst. Das Problem, welches sich hierbei ergibt, ist daß ein Agent bei seiner Anmeldung kein Wissen über die Existenz von anderen Agenten hat und somit nicht in der Lage ist, sich selbst einen eindeutigen Namen, der den Konventionen entspricht, zu vergeben. Wir werden später zeigen, welche Möglichkeiten zur Lösung dieses Problems denkbar sind.

4.1.2 Die Funktion Komm_abmelden(...)

Um sich beim Server abzumelden, übergibt der Agent dem Server seinen Namen. Der Server gibt ihm eine Statusmeldung über den Erfolg dieser Aktion zurück. Bei der Re-Implementierung dieser Funktion sind keine Schwierigkeiten zu erwarten.

4.1.3 Die Funktion Komm_senden(...)

Diese Funktion benötigt als Übergabeparameter den Typ Meldung und liefert als Rückgabe ebenfalls einen Wert vom Typ Meldung. Der Typ Meldung enthält die folgenden Informationen: eine zweiteilige Kennung, den Absender, den Empfänger, den Typ, einen Bezug, einen Zeitstempel, die Lösungszeit und den Inhalt der Nachricht. Als problematisch erweisen sich dabei vor allem die folgenden Punkte:

- Die Kennung einer Nachricht. Wie schon beschrieben, generiert der Server einen Teil der Kennung. Jede Nachricht bekommt so eine systemweite eindeutige Nummer. ICE ist nicht in der Lage, Message-IDs zu generieren. Eine Komponente kann allerdings eine sogenannte Turn-ID definieren, die von der Empfängerkomponente als Bezug verwendet werden kann. Da allerdings mehrere Agenten die gleiche Nachricht beantworten können, reicht es nicht, diese Turn-ID einfach hochzuzählen, da dadurch Message-IDs doppelt vergeben werden könnten. Es wird später erläutert, welche anderen Ansätze zur Lösung dieses Problems denkbar sind.

- Der Server generiert einen Zeitstempel. Diese Aufgabe wird vom Server übernommen, weil somit nur eine Instanz für die Vergabe von Zeitstem-

peln zuständig ist. Es soll damit vermieden werden, daß Zeitstempel, die von lokalen Rechnern vergeben werden, aufgrund von falsch eingestellten Uhren Inkonsistenzen aufweisen. Für eben jene Problematik gibt es jedoch mit dem NTP[6] die Möglichkeit, eine hinreichend genaue Zeit (ca. 100ms) auf den lokalen Computern zu erhalten. Durch die Laufzeiten, die eine Nachricht durch das Netz benötigt, könnten in Extremfällen viel höhere Ungenauigkeiten enstehen.

- Unter ICE ist keine Gruppenadressierung möglich, auch hierfür wird später eine Lösung vorgestellt werden.

4.1.4 Die Funktion Komm_empfangen(...)

Für diese Funktion wird kein Übergabeparameter benötigt. Wenn auf dem Server eine Nachricht vorhanden ist, wird diese als Typ Meldung an den Agenten zurückgeliefert. Da unter ICE Nachrichten nach dem gleichen Prinzip empfangen werden, sind keine Probleme bei der Implementation zu erwarten. Mit Hilfe einer speziellen ICE-Funktion ist es sogar möglich abzufragen, ob eine Nachricht vorliegt. Dazu wird auf dem lokalen Rechner eine Anfrage an den lokalen PVM-Dämon gerichtet. Dies geht erheblich schneller als ein RPC über verteilte Rechner im jetzt bestehenden System.

4.2 Konkrete Ansätze

Nachdem im vorigen Kapitel die grundsätzlichen Probleme bei der Re-Implementierung aufgezeigt wurden, werden im Folgenden verschiedene Ideen zur Integration betrachtet.

4.2.1 Nachbildung des Komm_Servers unter ICE

Eine Möglichkeit, ICE zu integrieren, besteht darin, den jetzigen Kommunikationsserver nachzubilden, was einer kompletten Neufassung desselben entspräche. Die Funktionsweise des Systems könnte man dann wie folgt beschreiben:

[6] **N**etwork **T**ime **P**rotokoll

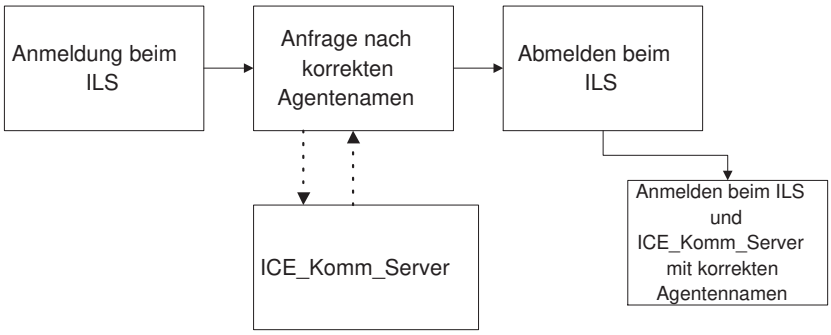

Abbildung 2: Anmeldeprozedur beim ICE_Komm_Server

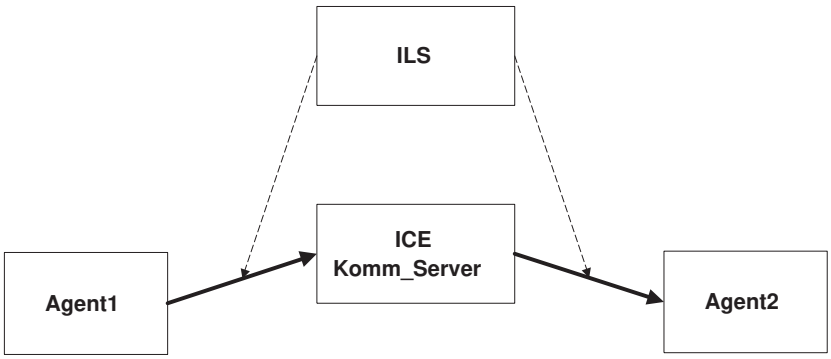

Abbildung 3: Nachrichtenübertragung mit dem ICE_Komm_Server

Der ICE_Komm_Server wäre der erste "Agent", der sich bei ICE anmeldet. Damit ein normaler Agent an der Kommunikation teilnehmen kann, muß er sich zunächst beim ILS unter einem freien Namen anmelden. Zu diesem Zweck wäre seine TID[7] denkbar, da diese PVM weit eindeutig ist. Nach der Anmeldung müßte er sich mit seinem Agenturnamen beim ICE_Komm_Server melden, damit dieser ihm seinen konkreten Agentennamen übermitteln könnte. Danach meldet sich der Agent beim ILS ab, um sich sofort wieder unter seinem neu erhaltenen Agentennamen sowohl beim ILS als auch beim ICE_Komm_Server anzumelden. Die Anmeldeprozedur ist in der Abbildung 2 noch einmal dargestellt.

Nachdem die Anmeldung ordnungsgemäß durchgeführt wurde, kann der

[7] **T**ask-**ID**

Agent an der Kommunikation teilnehmen. Sobald er eine Nachricht übermittelt, wird diese an den ICE_Komm_Server geleitet (siehe Abbildung 3). Dieser trägt die Message-ID in den Nachrichtenkopf ein und übernimmt die weitere Verteilung an die Empfänger. Da der Transport der Daten von ICE realisiert wird, ergeben sich Vorteile, wenn der Agent und der ICE_Komm_Server auf der gleichen Maschine laufen. In diesem Fall werden die Daten über shared-memory Bereiche ausgetauscht. Der ILS liefert also lediglich die Information, welche die effektivste Möglichkeit zur Übertragung ist.

Der Einsatz des hier beschriebenen Modells hat folgende Vorteile:

- Agentennamen können auf die gleiche Art wie bisher vergeben werden
- der Server kann Message-IDs wie bisher generieren
- in günstigen Konstellationen (Server und Agent laufen auf der gleichen Maschine) ergeben sich Geschwindigkeitsvorteile gegenüber der bisherigen Version.

Dem gegenüber stehen die folgenden Nachteile:

- zentralistische Struktur wird beibehalten
- alle Nachrichten werden wie bisher doppelt transportiert (vom Absender zum Server und von dort zum Empfänger)
- die Anmeldeprozedur ist sehr aufwendig
- der bestehende Server muß komplett neu implementiert werden
- eine Erweiterung der Kommunikation auf die neuen Möglichkeiten von ICE ließe sich nur schwer realisieren.

Zusammenfassend läßt sich also sagen, daß die hier vorgestellte Methode zur Integration von ICE nicht zu empfehlen ist. Im Grunde würde der Einsatz von ICE im Gegensatz zur bisherigen Implementierung keine wesentlichen Vorteile bringen. Sicherlich ist der Nachrichtentransport mit ICE effizienter

als bisher, dennoch wird die zentralistische Struktur früher oder später dazu führen, daß der Server sich als Flaschenhals beim Transport der Nachrichten erweist. Bedenkt man, daß eine Lösung, die den direkten Transport der Daten von Agent zu Agent ermöglicht, gegenüber der hier vorgestellten Lösung fast automatisch eine Halbierung der Transportzeit mit sich bringt, so ist eine Realisierung der Kommunikation mit Hilfe einer dezentralen Struktur angezeigt.

4.2.2 Realisierung einer dezentralen Struktur

Wenn die bestehende Kommunikation auf eine dezentrale Struktur umgestellt werden soll, so muß das Wissen, daß im Moment zentral im Server gespeichert ist, in einer dezentralen Form zugänglich gemacht werden. Die Informationen, die bisher vom Server verwaltet wurden, sind zum einen die Agentennamen und zum anderen die Message-IDs.

Die Message-IDs werden in dem alten Kommunikationsserver durch einen einfachen Nachrichtenzähler generiert. Dieses Problem ließe sich relativ einfach lösen, indem man einen speziellen Agenten implementiert, der nur für die Vergabe von Nachrichtennummern zuständig wäre. Allerdings hätte man auch hier wieder den Nachteil, daß man eine zentrale Anlaufstelle geschaffen hätte. Für jede Nachricht, die ein Agent schicken möchte, müßte er zunächst diesen Nummernagenten kontaktieren.

Eine ideale Lösung wäre es allerdings, wenn der Agent selbst in der Lage wäre, eine Message-ID zu generieren. Die Hauptaufgabe der Message-ID ist es, eine Nachricht mit einem systemweiten eindeutigen Code zu belegen. Einen solchen Code kann der Agent z.B. aus seiner TID, welche unter allen PVM-Teilnehmern eindeutig ist, und der aktuellen Zeit ohne Probleme selbst erstellen.

Man verzichtet bei dieser Methode allerdings darauf, das die Message-IDs aufeinanderfolgend und aufsteigend sind. Es hat sich jedoch gezeigt, daß durch diesen Verzicht keinerlei Schwierigkeiten zu erwarten sind.

Der Vergabe der Agentennamen liegt die gleiche Problematik zugrunde. Die Agentennamen müssen ebenfalls eindeutig sein und dürfen systemweit nur einmal vergeben werden. Dabei bestehen die Agentennamen aus ihrem

Agenturnamen und einer angehängten Nummer. Diese beginnt bei "1" und wird vom bestehenden Server ebenfalls einfach hochgezählt.

Da die Vergabe der Agentennamen nur beim Anmelden des Agenten benutzt wird, scheint eine zentrale Lösung mit Hilfe eines Namengebers tolerierbar. Ein Agent, der sich beim System anmeldet, macht dies also zuerst unter einem allgemeinen Namen beim ILS. Danach fragt er beim Namengeberagenten nach seinem korrekten Namen. Sobald er diesen erhalten hat, meldet er sich zunächst ab und anschließend unter seinem korrekten Namen wieder an.

Es ist allerdings auch noch folgende Lösung denkbar (siehe Abbildung 4): Jeder Agent speichert für jede Agentur einen Nummernzähler. Wenn sich nun ein neuer Agent anmelden möchte, macht er dies zunächst unter einem allgemeinen Namen. Er sendet dann eine Broadcast-Nachricht an alle anderen Agenten mit der Bitte, ihm die nächste freie Nummer für seine Agentur mitzuteilen. Alle anderen Agenten senden ihm daraufhin die ihrer Meinung nach nächste freie Nummer. Der neue Agent meldet sich ab und unter seinem neuen Namen sofort wieder an. Dies teilt er wiederum allen Agenten mit, worauf diese ihre Agenturzähler anpassen.

Folgende Probleme sind dabei allerdings zu beachten:

- Damit Nummern nicht doppelt vergeben werden, darf sich immer nur ein Agent in der Anmeldephase befinden. Dies läßt sich beispielsweise durch ein Semaphor realisieren. Vor seinem Anmeldeversuch muß der Agent in einen geschützten Bereich eintreten und diesen mit einem Semaphor sichern. Versucht ein anderer Agent während der Anmeldephase sich beim VIENA-System anzumelden, so erhält er eine Absage von dem Agenten, der sich gerade in der Anmeldung befindet. Der geschützte Bereich wird durch die Bekanntgabe des Namens an das System verlassen.

- Agenten haben keine garantierte Antwortzeit, d.h. es kann während der Anmeldung eines Agenten dazu kommen, daß ein anderer Agent sein Wissen über die Agenturnummern nicht sendet.

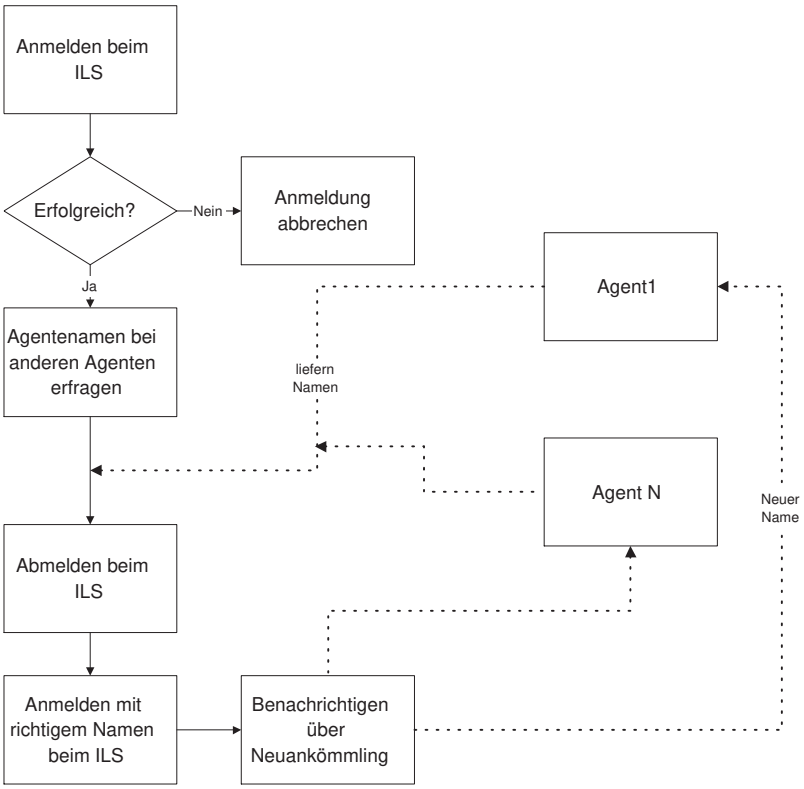

Abbildung 4: Anmeldung an einer dezentralen Struktur

- Da kein zentrales "Postamt" mehr existiert, müssen die Agenten selbst darüber buchführen, welche anderen Agenten im System angemeldet sind. Der Verwaltungsaufwand für diese Aufgabe ist nicht sonderlich hoch, da im bisher bestehenden System nur etwa zwanzig Agenten beteiligt sind.

Im Verlauf der Implementierung wurde diese Lösung noch leicht abgewandelt, in der Art, daß auch die Agentennummern von den Agenten selbst generiert werden. Durch die Verwendung von PVM ist die PVM-TID jedes Prozesses innerhalb des Systems eindeutig und erfüllt damit die Kriterien, die an die Agentennummern gestellt wurden. Diese Änderung verbessert vor allem das Startverhalten des Systems, da es die Anmeldezeiten der Agenten reduziert, und macht das System robuster, da die Eindeutigkeit der TIDs unter PVM schon implementiert und dementsprechend getestet ist.

Für dieses System sprechen die folgenden Punkte:

- komplett dezentrale Struktur
- Nachrichten werden nur einmal im System verschickt (bei einem Empfänger)
- der bestehende Kommunikationsserver wird nicht mehr benötigt
- die erweiterte Funktion von ICE läßt sich für zukünftige Agenten benutzen (siehe Kapitel 6)

Bei dieser Implementierung handelt es sich um eine komplette dezentrale Lösung. Das Wissen wird in einer verteilten Datenbank gespeichert. Die hier vorgestellte Lösung läßt sich durch eine Re-Implementierung der vier Kommunikationsfunktionen erreichen.

Da die Vorteile der zuletzt vorgestellten Lösung überwiegen und da eine komplett dezentrale Struktur des Gesamtsystems angestrebt wird, wurde die Implementierung der zuletzt vorgestellten Lösung favourisiert.

4.3 Nicht absehbare Fehlerquellen

Durch den Einsatz von ICE als Kommunikationswerkzeug wird ein komplett anderes Laufzeitverhalten innerhalb der Kommunikation eintreten. So

können zwei Agenten auf einer Maschine extrem viel schneller Daten übertragen, als das bisher möglich war. Außerdem funktioniert die Kommunikation in manchen Fällen, z.B. wenn je zwei Agenten auf verschiedenen Rechnern miteinander kommunizieren, quasi parallel. Eine Sequentialisierung, wie dies durch den Einsatz des bisherigen Servers gegeben war, findet nicht mehr statt. Das System ist zwar in der bisherigen Version nicht explizit auf die sequentielle Verarbeitung durch den Server angewiesen, jedoch kann nicht ausgeschlossen werden, daß es zu Spezialfällen kommt, in welchen sich das System anders verhält als bisher. Diese Probleme können allerdings erst nach der Fertigstellung des neuen Kommunikationssystems explizit aufgedeckt werden.

5 Performance-Messungen

Eine Zielsetzung bei der Integration eines neuen Kommunikationssystems besteht natürlich auch in der Verbesserung seines Laufzeitverhaltens. Im Idealfall sollte es deutlich schneller sein als das alte System; zumindest aber sollte ausgeschlossen werden, daß die Kommunikation subjektiv langsamer wird.

Die vorliegenden Messungen, die mit dem bestehenden Kommunikationsserver durchgeführt wurden, haben eine durchschnittliche Transferrate von 100 KB/s ergeben. In diesem Wert ist die Zeit, die für die Konvertierung der Datenstrukturen benötigt wurde mit eingerechnet. Außerdem liefen der Agent und der Kommunikationsserver auf verschiedenen Rechnern.

Bei den Versuchen, die Transferleistungen unter ICE zu ermitteln, ergaben sich weit gestreute Ergebnisse. Bei Prozessen, die auf verschiedenen Rechnern liefen, hatte die Netzwerkbelastung einen erheblichen Einfluß auf die Ergebnisse. In diesen Fällen alg jedoch die maximale Transferleistung bei bis zu 500KB/s. In diesem Wert ist jedoch der Aufwand zur Konvertierung der Daten nicht enthalten. Um trotzdem einen Vergleich zwischen den Leistungen der beiden Systeme durchführen zu können, wurden Messungen durchgeführt (siehe Tabelle 2), die Aufschluß über die zu erwartende Konvertierungszeit geben.

In Tabelle 1 ist die Transferleistung von zwei Agenten auf der gleichen

Tabelle 1: Versuchsergebnisse

Blocklänge [Byte]	**Anzahl**	**Sendezeit [sec]**	**Empfangszeit** [sec]
1	10000	8,459	10,193
		8,635	11,102
		7,588	9,866
10	10000	7,743	9,881
		8,594	9,483
		7,754	9,425
100	10000	8,407	9,103
		7,805	9,779
		8,248	9,181
1000	10000	10,777	10,239
		10,553	9,808
		10,391	9,778
10000	1000	5,147	4,467
		5,574	4,647
		5,858	5,011
10000	10000	47,0001	46,291
		44,535	43,641
		54,350	53,488

Maschine (Typ: SGI Indigo 2) wiedergegeben. Die maximale Transferrate beträgt dabei ca. 2MB. Hier zeigen sich die Vorteile von ICE besonders deutlich. Durch geschickte Verteilung der Agenten auf verschiedene Maschinen kann man das Laufzeitverhalten von VIENA erhöhen. Diese Daten beinhalten ebenfalls nicht die Zeiten zum Konvertieren der Daten.

Aufgrund der gemessenen Transferraten läßt sich eine Steigerung der Kommunikationsgeschwindigkeit erwarten. Es bleibt jedoch abzuwarten, inwieweit sich diese Schätzungen im Laufzeitverhalten von VIENA wiederspiegeln. Da neben der Kommunikation auch andere Aufgaben erledigt werden, besteht die Gefahr, daß die Daten zwar schneller zur Verfügung stehen, diese aber trotzdem nicht schneller bearbeitet werden, da die entsprechenden Agenten noch mit beschäftigt sind.

Tabelle 2: Diese Messungen machen deutlich, wieviel Zeit für die Konvertierung der Kommlisten zu veranschlagen ist. Es wurden jeweils tausend Konvertierungen durchgeführt. Der durchschnittliche Konvertierungsaufwand liegt bei etwa 1 MB/s.

Stringlänge in[Byte]	**Anzahl Strings**	**gesamt versendet[Byte]**	**Zeit zum kodieren[ms]**	**Zeit zum dekodieren[ms]**
5	10	121	90	170
	100	661	210	690
	1000	6061	1920	6090
10	10	171	100	180
	100	1161	310	860
	1000	11061	2880	8230
20	10	271	110	230
	100	2161	490	1300
	1000	21061	4990	12680
100	100	10161	2100	4670
1000	10	10071	2010	4250

Klar abzusehen ist jedoch, daß die Einführung von ICE positive Auswirkungen auf die Kommunikation haben wird. Betrachtet man die Transferleistung über das Netz, so stellt man fest, daß diese nahe an der maximal möglichen Übertragungsrate liegt.

6 Schlußbetrachtung

6.1 Theoretische Überlegungen

Mit der Einführung von ICE ergibt sich die Möglichkeit die bestehende Kommunikation zu erweitern. So kann ein Transfer über mehrere Kanäle relativ einfach nachgerüstet werden, indem eine neue Kommunikationsfunktion realisiert würde. Durch eine einfache Anfrage könnte ein Agent mitteilen, ob er über die erweiterten Möglichkeiten der Kommunikation verfügt und diese gegebenenfalls nutzen. In diesem Fall bliebe es allerdings nicht aus, diesen Agenten entsprechend zu programmieren.

Um den Einsatz von ICE in einem anderen Projekt zu beurteilen, muß man die konkreten Vorraussetzungen für dieses Projekt kennen. Hilfreich mag die Betrachtung der Frage "Was wäre gewesen, wenn ICE zum Beginn des Projektes VIENA zur Verfügung gestanden hätte?" sein.

Anlaß zur Kritik hätte sicherlich die fehlende Gruppenadressierung gegeben. Diese läßt sich allerdings relativ einfach durch einen Agenturagenten realisieren. Dieser hätte einfach alle Nachrichten an seine beteiligten Agenten weiterleiten müssen. Nach außen hätte sich dieser Agenturagent wie ein normaler Agent verhalten. Bedenkt man, daß ein solcher Agent eventuell schon eine Selektion der Nachrichten vornehmen könnte oder andere Vorarbeiten erledigen könnte, so erwächst aus diesem anfänglichen Nachteil schnell ein Vorteil. Außerdem könnte man mit dieser Methode tiefere Hierarchien von Agenten und Agenturen erstellen.

Auch andere Probleme, die bei der Integration von ICE zu lösen waren, lassen sich eher auf die frühen Design-Entscheidungen von VIENA zurückführen. Hätte ICE seinerzeit zur Verfügung gestanden, so wären sicherlich einige Entscheidungen, wie die zentrale Vergabe von Message-IDs und Agentennamen, anders ausgefallen.

Durch die Implementierung von ICE auf Basis von PVM ist eine Rechnerarchitektur übergreifende Kommunikation möglich. So ist es beispielsweise denkbar, bisher getrennt entwickelte Projekte, die als Kommunikationswerkzeug ICE benutzen, zusammenzuführen.

Abschließend bleibt zu sagen, daß die Arbeit mit ICE einen durchaus positiven Eindruck dieses Werkzeuges hinterlassen hat. Sollte in naher Zukunft auch die DDL zur Verfügung stehen, so wäre dies sicher ein weiterer Grund für die Einführung von ICE sowohl unter VIENA als auch in neuen Projekten. Allerdings muß gesagt werden, daß die Alltagstauglichkeit von ICE erst im Praxiseinsatz gezeigt werden kann. Auch die Kommunikation mit ICE wird ihre Grenzen aufzeigen und somit früher oder später Ideen für ein neues Kommunikationsmodell hervorbringen.

6.2 Praktische Erfahrungen

Im Einsatz wurde vor allem die Flexibilität von ICE erkannt, die sicher zu einer Verbesserung der Architektur beitragen wird. Ein Beispiel hierfür ist der neue Anpasser-Agent, der die Möglichkeiten von ICE nutzt, um zusätzlich mit dem Inventor-Viewer zu kommunizieren. Auch die zusätzlichen Kanäle werden sehr hilfreich sein, um das System zu erweitern.

Die dezentrale Struktur von ICE, die im neuen Kommunikationssystem ebenfalls ihren Eingang gefunden hat, macht kommunikationsintensive nebenläufige Verarbeitungen erst sinnvoll und zum Teil sogar überhaupt erst möglich. Obwohl hier bewußt der sanfte Umstieg von der Client-Server-Architektur mittels RPC zur dezentralen Architektur mit ICE über PVM gewählt wurde, wurde dem System trotzdem die ganze Palette neuer Möglichkeiten gelassen, da auch weiterhin in Abstufungen Änderungen in der Architektur gemacht werden können, die jetzt wesentlich einfacher vonstatten gehen müßten, da die Basis gleichsam 'verbreitert' wurde.