(19) **Bundesrepublik Deutschland**
Deutsches Patent- und Markenamt

(10) **DE 699 08 349 T2** 2004.04.01

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 076 851 B1**
(21) Deutsches Aktenzeichen: **699 08 349.4**
(86) PCT-Aktenzeichen: **PCT/GB99/01302**
(96) Europäisches Aktenzeichen: **99 919 387.3**
(87) PCT-Veröffentlichungs-Nr.: **WO 99/057637**
(86) PCT-Anmeldetag: **27.04.1999**
(87) Veröffentlichungstag der PCT-Anmeldung: **11.11.1999**
(97) Erstveröffentlichung durch das EPA: **21.02.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **28.05.2003**
(47) Veröffentlichungstag im Patentblatt: **01.04.2004**

(51) Int Cl.[7]: **G06F 9/46**

(30) Unionspriorität:
9809512 01.05.1998 GB

(84) Benannte Vertragsstaaten:
DE, FR, GB, NL, SE

(73) Patentinhaber:
BRITISH TELECOMMUNICATIONS public limited company, London, GB

(74) Vertreter:
Beetz & Partner, 80538 München

(72) Erfinder:
LEBRE, Anne, Caroline, Ipswich, Suffolk IP1 3NB, GB; TITMUS, John, Richard, Ipswich, Suffolk IP4 3AN, GB

(54) Bezeichnung: **VERTEILTE DATENVERARBEITUNG, DIE CLIENT- UND SERVER-OBJEKTE VERWENDET**

Beschreibung

Gebiet der Erfindung

[0001] Diese Erfindung bezieht sich auf ein Verfahren zum Verarbeiten von Daten in einer verteilten Rechenumgebung.

Hintergrund

[0002] Eine Datenverarbeitung kann in einer verteilten Rechenumgebung ausgeführt werden, in der Client-Software mit Server-Software, die in einem Netz verbunden sind, interagiert. Ein Server kann als ein Betriebsmittel aufweisend betrachtend werden, das von mehreren Clients gemeinsam benutzt wird, die daran Interesse haben. Der Server wartet auf vom Client initiierte Anfragen und antwortet auf diese individuell mit Informationen, die aus dem vom Client abgefragten Betriebsmittel hergeleitet werden.

[0003] Gewöhnlich ist die Client-Software bei festen Arbeitsstationen angeordnet, die im Netz angeschlossen sind und mit Servern an festen Orten interagieren. Vor kurzem wurde eine mobile Agentensoftware entwickelt, die der Client-Software erlaubt, sich zu einem Ort in der Nähe eines Servers zu bewegen, um eine bessere Nutzung der Einrichtungen des Servers zu erreichen. Wenn z. B. ein verarbeitendes Unternehmen Fabriken an zwei verschiedenen Orten besitzt, die ihre eigenen lokalen Computernetze aufweisen, kann ein Operator am ersten Ort wünschen, Datenbanken von Servern an beiden Orten abzufragen, um z. B. die Verfügbarkeit bestimmter Lagergegenstände zu ermitteln, die in den Lagerhäusern an den zwei Orten gehalten werden können. In dieser Situation ist es für die Client-Datenabfragesoftware günstig, sich vom ersten Ort zum zweiten Ort zu verlagern, um näher am Server am zweiten Ort zu sein, um eine effiziente Abfrage der zugehörigen Datenbanken zu ermöglichen. Die mobile Client-Software ist als mobiler Agent bekannt.

[0004] Es wurden mehrere verschiedene Systeme entwickelt, die mobile Agenten zur Verfügung stellen: MuBot von Crystaliz, Inc.; Agent Tcl von Dartmouth College; Aglets von IBM; MOA von der Open Group, Inc.; GMAF/Magna von GMD Fokus; und Odyssey von General Magic Inc.

[0005] In "Agent-Based Configuration Management of Distributed Applications", Berghoff u. a., Proceedings of the third international conference on Configurable Distributed Systems (Annapolis USA, 6.-8. Mai 1996), S. 52–59, wird das Konfigurationsmanagement von verteilten Anwendungen beschrieben. Genauer wird die Eignung mobiler Agenten in einem solchen Konfigurationsmanagement beschrieben und behandelt. Ein Beispiel für Konfigurationsmanagement, das keine mobilen Agenten verwendet, ist als Hintergrund angegeben. In diesem Beispiel wird angenommen, daß eine Funktionseinheit M in jeder Komponente eines Satzes von unterteilten Server-Komponenten ausgetauscht werden muß. Um dies mit einer minimalen Störung der laufenden Anwendung zu bewerkstelligen, werden die unterteilten Server-Komponenten der Reihe nach weiter betrieben. Die folgenden Schritte werden ausgeführt, um die Einheit M in der ersten der unterteilten Server-Komponenten A auszutauschen.

– Anweisen des Vermittlers (der ankommende Anfragen zwischen Servern verteilt, um einen Lastausgleich zu erreichen), keine weiteren Anfragen an die Server-Komponente A weiterzuleiten.
– Anweisen der Server-Komponente A, ankommende Anfragen zurückzuweisen.
– Überwachen des aktuellen Verkehrs auf der Server-Komponente A.
– Entscheiden, zu welchem Zeitpunkt die Server-Komponente ausgetauscht wird, in Abhängigkeit vom aktuellen Verkehr.
– Wenn der Zeitpunkt gekommen ist, Aussetzen aller aktuellen Bindungen zur Server-Komponente A durch Passivieren der entsprechenden Schnittstellen auf der Client-Seite.
– Nachdem die Aktivitäten der Server-Komponente A gestoppt worden sind (aufgrund der Passivierung der Client-Schnittstellen), wird die Einheit M ausgetauscht.
– Wiederherstellen der Bindungen zur Server-Komponente A durch Aktivieren der entsprechenden Client-Schnittstellen.
– Anweisen der Server-Komponente A, ankommende Anfragen anzunehmen.
– Anweisen des Vermittlers, daß zukünftige Anfragen an die Server-Komponente A weitergeleitet werden können.

[0006] Das Papier beschreibt eine Management-Architektur, die auf einer globalen Management-Komponente (GMC) beruht, die mit den Anwendungskomponenten interagiert, um Managementoperationen durchzuführen. Diese zentralisierte Architektur gilt als nachteilig, wenn sie auf große und komplexe verteilte Anwendungen angewendet wird. Alle Nachrichten, die Management-Interaktionen betreffen, müssen von einer GMC behandelt werden. Der resultierende Verkehr tendiert dazu, die Management-Komponenten zu überlasten, und führt zu schlechten Antwortzeiten. Wenn die Anwendung über einen weiten Bereich verteilt ist, kann der Kommunikationsaufwand dramatisch anwachsen. Um diese Beschränkungen zu beseitigen, haben Berghoff u. a. ein dezentralisiertes Managementsystem untersucht, das unterteilte globale Management-Komponenten verwendet. Es hat sich jedoch herausgestellt, daß dieses System aufgrund des Synchronisationsaufwands zwischen den dezentralisierten Management-Komponenten immer noch an geringer Skalierbarkeit leidet.

[0007] Als Alternative schlagen Berghoff u. a. eine Infrastruktur für mobile Agenten vor. Berghoff u. a. erläutern, daß der Vorteil von mobilen Agenten im Ver-

gleich zu herkömmlichen Lösungsansätzen zur verteilten Berechnung hauptsächlich in der Effizienz liegt. Viele Operationen erfordern einen langwierigen Dialog oder das Verschieben beträchtlicher Datenmengen. Wenn dies unter Verwendung herkömmlicher Mittel über ein Netz bewerkstelligt wird, kann dies zu einer erhöhten Netzbelastung und zu einem Leistungsverlust aufgrund der Latenzzeit und eines Mangels an Bandbreite führen. Mobile Agenten tragen dazu bei, die Notwendigkeit einer Netzkommunikation zu reduzieren, indem der Code, der die Interaktion handhabt, näher an die Datenquelle bewegt wird. Berghoff u. a. heben hervor, daß in einer Umgebung, die mobile Agenten unterstützt, nicht jeder Agent mobil sein muß – tatsächlich spielen stationäre Agenten eine wichtige Rolle, indem sie als Mittler oder Proxies zwischen der "Agentenwelt" und der normalen Betriebsumgebung auf einem Host arbeiten.

[0008] Sie haben eine Prototyp-Agenten-Infrastruktur implementiert, um Experimente auf dem Gebiet mobiler Agenten zu unterstützen. Dies bildet die Grundlage einer Ausführungsumgebung für mobile und stationäre Agenten. Die Agenten-Infrastruktur umfaßt einen Agenten-Server, der sich mit den Einzelheiten des Akzeptierens von Agenten, die laufen sollen, des Konstruierens einer geeigneten Laufzeitumgebung und des Einführens und Entfernens des Agenten der Reihe nach befaßt. Agenten müssen ferner miteinander und mit anderen Programmen kommunizieren. Die Prototyp-Agenten-Infrastruktur unterstützt dies über den Gedanken eines Informationsraums, den Agenten zum gemeinsamen Nutzen von Daten und zum Einleiten und Beantworten von Anfragen benutzen können. Dies war eine Einrichtung auf niedriger Ebene, die im allgemeinen ausreichte, um eine Vielfalt von Interaktionsarten zu erlauben, sowohl deklarative als auch RPC-ähnliche. Anschließend haben sie eine Erweiterung des bestehenden zentralisierten Managementsystems mit den folgenden Hauptanforderungen entwickelt. Anwendungskomponenten sollten nicht modifiziert werden müssen, um durch Agenten verwaltbar zu sein. Bestehende Management-Anwendungskomponenten sollten wiederverwendbar sein. Es sollte möglich sein, von einem rein zentralisierten Managementsystem zu einem solchen auf Agentenbasis inkrementell fortzuschreiten. Um den Übergang zu erleichtern und Betriebsmittelbeschränkungen des Netzes des Hosts zu berücksichtigen, sollte die Agenten-Infrastruktur nicht auf jedem Host vorhanden sein müssen, auf dem die verteilte Anwendung läuft. Die bestehende Management-Architektur wurde auf einem Host mit mobilen Agenten integriert, indem spezialisierte Agenten eingeführt wurden, die als Vermittler zwischen mobilen Agenten und dem Managementsystem arbeiteten. Diese Agenten waren stationär, d. h. sie wurden bestimmten Hosts zugewiesen, von denen sie sich nicht wegbewegten. Sie kommunizierten mit mobilen Agenten über den Informationsraum und führten Management-Anfragen an Anwendungskomponenten aus. Berghoff u. a. schließen ihr Papier, indem sie die Hauptvorteile ihres Lösungsansatzes für das Konfigurationsmanagement auf Agentenbasis identifizieren: Erhöhte Flexibilität, da mobile Agenten Code führen können, um die Funktionalität stationärer Agenten, Anwendungs- und Managementkomponenten zu erweitern oder zu ersetzen. Eine neue Managementfunktionalität oder Protokolle werden in einer einfachen Weise eingeführt.

[0009] Erhöhte Zuverlässigkeit, da mobile Agenten autonom in einem vorübergehend abgetrennten Teil des Netzes arbeiten können.

[0010] Erhöhte Leistungsfähigkeit und reduzierter Netzverkehr in großen Anwendungen, da sich die Agenten zu dem Bereich bewegen, in dem die Managementoperationen ausgeführt werden müssen. Während der Verarbeitung seiner Management-Aufgabe ist die Latenzzeit eines Management-Aufrufs (die nun Kurzstreckenaufrufe sind) verringert, wobei die Gesamtlatenzzeit ebenfalls abnimmt. Die Zwischeninformationen, die zum Erfüllen der Management-Aufgabe erforderlich sind, werden lokal verarbeitet und müssen nicht weit zu einer GMC übermittelt werden. Somit nimmt die Gesamtbelastung des GMC deutlich ab.

[0011] Berghoff u. a. schlagen nirgendwo vor, daß die mobilen Agenten eine Server-Funktionalität aufweisen sollen, noch daß es möglich ist oder in irgendeiner Weise wünschenswert wäre. Es wird nicht gelehrt, mobile Server vorzusehen.

Zusammenfassung der Erfindung

[0012] Gemäß der Erfindung wird angenommen, daß es Situationen gibt, in denen es vorteilhaft wäre, den Server innerhalb einer verteilten Rechenumgebung mobil zu machen.

[0013] Gemäß der Erfindung wird ein Verfahren zum Verarbeiten von Daten in einer verteilten Rechenumgebung geschaffen, in der ein Client und ein Server Daten verarbeiten, wobei das Verfahren das Senden des Servers von einem ersten Ort, an dem er mit dem Client kommuniziert, über die verteilte Rechenumgebung zu einem zweiten anderen Ort, damit er von hier eine Datenverarbeitung ausführt, umfaßt.

[0014] Das Verfahren kann das Einfrieren ankommender Aufrufe für eine Datenverarbeitung beim Server an dem ersten Ort, während er von dem ersten Ort zu dem zweiten Ort gesendet wird, und danach das Leiten der eingefrorenen Aufrufe zu dem zweiten Ort, damit sie durch den Server verarbeitet werden, wenn er an dem zweiten Ort arbeitsfähig geworden ist, umfassen. Dies hat den Vorteil, daß sichergestellt wird, daß Verbindungen für den Server nicht verloren gehen, während er sich vom ersten Ort zum zweiten Ort bewegt.

[0015] In einem weiteren Aspekt enthält die Erfindung das Aufnehmen des vom ersten Ort gesendeten Servers, um eine Datenverarbeitung am zweiten

Ort durchzuführen.

[0016] Um den Server vom ersten Ort zum zweiten Ort zu senden, kann er von einer Arbeitskonfiguration am ersten Ort in eine Konfiguration umgewandelt werden, die für die Übertragung durch die verteilte Umgebung zu dem zweiten Ort geeignet ist. Die Umwandlung kann die Serialisierung des Servers umfassen.

[0017] Die Erfindung umfaßt ferner eine Software-Entität, die so betreibbar ist, daß sie einen Server für einen Client in einer verteilten Rechenumgebung bereitstellt, dadurch gekennzeichnet, daß die Software-Entität über die Umgebung wahlweise zu verschiedenen Orten verschoben werden kann.

[0018] In einem weiteren Aspekt umfaßt die Erfindung ein Signal für die Übertragung in einer verteilten Rechenumgebung, in der ein Client und ein Server Daten verarbeiten, wobei das Signal den Server umfaßt, der für die Übertragung über die verteilte Rechenumgebung zwischen einem ersten Ort, wo er mit dem Client kommuniziert, und einem zweiten anderen Ort serialisiert wird und eine Datenverarbeitung ausführt.

[0019] Die Übertragung des Servers vom ersten Ort zum zweiten Ort kann von einem Proxy kontrolliert werden, genauer umfaßt die Erfindung einen Proxy für die Verwendung in einer verteilten Rechenumgebung, in der ein Client und ein Server Daten verarbeiten, wobei der Proxy so betreibbar ist, daß er den Server von einem ersten Ort, wo er mit dem Client kommuniziert, über die verteilte Rechenumgebung zu einem zweiten anderen Ort sendet, damit er eine Datenverarbeitung ausführt.

Kurzbeschreibung der Zeichnungen

[0020] Damit die Erfindung vollständiger verstanden werden kann, wird im folgenden eine ihrer Ausführungsformen beispielhaft mit Bezug auf die beigefügten Zeichnungen beschrieben, in welchen:

[0021] **Fig.** 1 ein schematisches Blockschaltbild einer verteilten Rechenumgebung ist, die mobile Softwareagenten verwendet;

[0022] **Fig.** 2 ein genaueres Diagramm eines der in **Fig.** 1 gezeigten Hosts ist;

[0023] **Fig.** 3 schematisch die Bewegung eines mobilen Servers von einem ersten Ort zu einem zweiten Ort gemäß der Erfindung zeigt; und

[0024] **Fig.** 4 ein schematisches Zeitablaufdiagramm der Signalkommunikation zwischen dem ersten Ort und dem zweiten Ort bezüglich der Bewegung des Servers ist.

Genaue Beschreibung

[0025] In der folgenden Beschreibung wurde zur bequemen Erläuterung die Terminologie angewendet, die von der Object Management Group (OMG) für mobile Agenten angewendet wird. Die OMG hat einen gemeinsamen Standard für die Interoperabilität von Objekten zwischen verschiedenen Systemen unter einer gemeinsamen Objekt-Management-Architektur definiert, die ein Objektanfragevermittler, der im Handel als CORBA bekannt ist, bereitstellt, welcher eine Infrastruktur bietet, die Objekten erlaubt, unabhängig von den spezifischen Plattformen und Techniken, die zum Implementieren der Objekte verwendet werden, zu kommunizieren. Um die Interoperabilität mobiler Agenten zu bewältigen, hat die OMG ein Dokument "Mobile Agent Facility Specification", 1. September 1997, OMG TC Document orbos/97-09-20, erhältlich von der Object Management Group, 492 Old Conneticut Path, Framingham, MA 01707, USA, angefertigt. Die vollständige Spezifikation ist auch über die Internet-Seite von OMG (dessen URL www.omg.org ist) der Öffentlichkeit zugänglich. Dies wird im folgenden mit Bezug auf die **Fig.** 1 und 2 erläutert.

[0026] Für mobile Softwareagenten, die Clients sind, besteht die Welt aus Regionen, die Orte enthalten, zwischen denen sich der mobile Agent bewegen kann. Wie in **Fig.** 1 gezeigt ist, sind erste und zweite Host-Rechensysteme **1**, **2** über ein Netz **3** verbunden. Die ersten und zweiten Host-Systeme können irgendeiner geeigneten Form entsprechen, wie z. B. lokalen Netzen, einzelnen Computern und dergleichen, die mit ihrem eigenen Betriebssystem OS1, OS2 arbeiten können. In herkömmlicher Weise können die einzelnen Hosts **1**, **2** einen oder mehrere Computer oder Prozessoren enthalten, die jeweils einen Prozessor, einen flüchtigen Arbeitsspeicher und einen nichtflüchtigen Datenspeicher enthalten. Jeder Host ist mit einer Kommunikationsschnittstelle CI1, CI2 versehen, um eine Kommunikation zwischen diesen über das Netz **3** zu ermöglichen. Das Netz **3** kann irgendeiner geeigneten Form entsprechen, z. B. einem Weitverkehrsnetz, einem lokalen Netz, einem Intranet oder dem Internet.

[0027] Das Betriebssystem OS1 des Host **1** bietet eine Umgebung, in der Software arbeiten kann. Die Client-Software ist als mobiler Softwareagent **MA1** konfiguriert. In ähnlicher Weise weist der Host **2** ein Betriebssystem OS2 und einen mobilen Agent **MA2** auf. Ein weiterer mobiler Agent **MAn** ist im Host **1** gezeigt. Jeder mobile Agent MA ist an einem Ort P betriebsfähig. Bei Betrachtung des Host **1** ist somit der mobile Agent **MA1** am Ort P1 betriebsfähig, während der mobile Agent **MAn** an einem Ort Pn betriebsfähig ist. Der mobile Agent **MA2** befindet sich am Ort P2 im Host **2**. Die mobilen Agenten können sich von Ort zu Ort bewegen. Es ist klar, daß im Host **1** die Orte P individuelle Computer sein können, die in einem Netz verbunden sind, das den Host **1** oder irgendeine andere geeignete Hardwarekonfiguration umfaßt, die hier nicht genauer beschrieben wird. Das gleiche gilt für den Host **2**. Die OMG-Mobilagenten-Spezifikation ist so ausgelegt, daß sie eine Interoperabilität zwischen verschiedenen Betriebssystemen schafft, um den Transport von mobilen Client-Agenten von einem Host zu einem weiteren zu erlauben. In der Konfigu-

ration der **Fig.** 1 wird angenommen, daß verschiedene Betriebssysteme OS1, OS2 in Gebrauch sind, obwohl dies kein wesentliches Merkmal der Erfindung ist. Es wird angenommen, daß die OMG-Spezifikation CORBA nutzt, um eine Interoperabilität zwischen verschiedenen Hardware- und Software-Konfigurationen zu erlauben. Die Agenten, die innerhalb des Betriebssystems OS1 arbeiten, definieren ein Agentensystem AS1 im Host **1**. Ein ähnliches Agentensystem AS2 arbeitet in dem in **Fig.** 1 gezeigten Host **2**.

[0028] Der Softwareprozeß ist in einer Client-Server-Konfiguration angeordnet, wie im folgenden mit Bezug auf **Fig.** 2 erläutert wird. Günstigerweise, jedoch nicht notwendigerweise, kann die Software objektorientiert sein, so daß die mobilen Client-Agenten und die Server jeweils als Objekte betrachtet werden können. Wie in **Fig.** 1 gezeigt ist, ist die Server-Software **MS1** am Ort P1 gezeigt, der Aufrufe von mobilen Klienten bedienen kann, wie mit Bezug auf **Fig.** 1 beschrieben worden ist. Zum Beispiel ist der mobile Agent **MA1** ein Client am Ort P1 und kann Datenaufrufe an den Server **MS1** über den Pfad **4** stellen, um eine Datenverarbeitung auszuführen. Der Client und der Server müssen jedoch nicht am gleichen Ort P angeordnet sein. Im Beispiel der **Fig.** 2 kann somit der Server **MS1** Datenaufrufe vom mobilen Client-Agenten **MA2** am Ort **2** über den Kommunikationspfad **5** bedienen. Es ist klar, daß mehr als ein Server MS in der verteilten Rechenumgebung vorhanden sein kann.

[0029] Gemäß der Erfindung ist der Server **MS1** innerhalb der verteilten Rechenumgebung mobil. Um die Mobilität des mobilen Servers **MS1** zu managen, ist ihm ein Software-Proxy **pr1** zugeordnet, der an jedem Ort P verschieden ist. Der Proxy **pr1** wird dem CORBA mit der mobilen Server-Schnittstelle angezeigt, anstelle des mobilen Servers selbst. Alle Verarbeitungsaufrufe für den Server gehen zuerst an den Proxy und werden anschließend von diesem zum Server umgeleitet. Daher weiß der Proxy **pr1** zu jedem Zeitpunkt, wie viele Clients mit dem Server **MS1** verbunden sind, und wie viele Aufrufe durchgeführt werden.

[0030] Wie in **Fig.** 3 gezeigt ist, gibt es Situationen, in denen es günstig wäre, den Server-Agenten **MS1** vom Ort P1 über die Kommunikationsschnittstelle CI1, das Netz **3** und die Schnittstelle CI2 zum Ort P2 zu bewegen. Zum Beispiel könnte der Server **MS1** dann mit verbesserter Operabilität mit dem Client **MA2** arbeiten, der sich im Agentensystem AS2 im Host **2** befindet. Die Übertragung des Servers **MS1** vom Ort P1 zum Ort P2 wird im folgenden mit Bezug auf **Fig.** 4 genauer beschrieben.

[0031] Anfangs, wenn der mobile Server **MS1** entscheidet oder angewiesen wird, sich vom Ort P1 wegzubewegen, teilt er im Schritt S.0 seinem Proxy **pr1** den Ort mit, zu dem er bewegt werden soll. In diesem Fall soll der mobile Server **MS1** zum Ort P2 bewegt werden. Alternativ kann der Proxy **pr1** von einer bestimmten externen, dritten Partei angewiesen werden, den mobilen Server zu bewegen. Der Bewegungsprozeß beginnt anschließend.

[0032] Im Schritt S.1 friert der Proxy **pr1** alle ankommenden Aufrufe zur Datenverarbeitung an den mobilen Server **MS1** ein.

[0033] Im Schritt S.2 wartet der Proxy **pr1**, bis alle aktuellen Datenverarbeitungen, die vom mobilen Server **MS1** gehandhabt werden, beendet sind.

[0034] Anschließend teilt im Schritt S.3 der Proxy **pr1** dem mobilen Server **MS1** mit, daß er bewegt werden soll und daß er irgendeine Aufgabe ausführen muß, die abgeschlossen sein muß, bevor er den Ort P1 verläßt.

[0035] Anschließend veranlaßt der Proxy im Schritt S.4, daß der mobile Server **MS1** serialisiert wird, d. h. er wird von seinem Betriebszustand in einen Zustand umgewandelt, der für die Übertragung über das Netz **3** geeignet ist (**Fig.** 1).

[0036] Anschließend wird im Schritt S.5 der serialisierte mobile Server über die Kommunikationsschnittstelle CI1, das Netz **3** und die Kommunikationsschnittstelle CI2 zum Ort P2 des Host **2** gesendet.

[0037] Im Schritt S.6 wird ein neuer Proxy **pr1'** im Ort P2 für den mobilen Server MS 1 erzeugt, wenn er sich am Ort p2 befindet.

[0038] Im Schritt S.7 wird der mobile Server **MS1** am Ort P2 deserialisiert, wodurch er in einen Betriebszustand zurückversetzt wird.

[0039] Im Schritt S.8 sendet der neu erzeugte Proxy **pr1'** lokale Referenzdaten für den mobilen Server **MS1** zurück, um somit dem Proxy **pr1** die neue CORBA-Referenz des mobilen Servers **MS1** anzuzeigen.

[0040] Anschließend werden im Schritt S.9 die im Schritt S.1 eingefrorenen Aufrufe zum mobilen Server **MS1** über das Netz **3** mittels des Proxy **pr1** vom Ort P1 zum Ort P2 weitergeleitet.

[0041] Die mit Bezug auf **Fig.** 4 beschriebene Prozedur hat den Vorteil, daß eine Kommunikation mit dem mobilen Server **MS1** während des Übertragungsprozesses nicht verloren geht. Die Schritte stellen sicher, daß jede Datenverarbeitung, die am Ort P1 ausgeführt wird, abgeschlossen wird, bevor die Übertragung stattfindet, und daß, während die Übertragung stattfindet, ankommende Aufrufe eingefroren werden und anschließend zum neuen Ort übertragen werden.

[0042] Clients können den bewegten mobilen Server **MS1** finden, indem sie eine geeignete Anfrage stellen, wie für jedes andere CORBA-Objekt, und die Referenz seines Proxy empfangen. Der Proxy, der für den mobilen Agenten angezeigt wird, kann entweder der erste Proxy **pr1** sein, wobei in diesem Fall Aufrufe vom **pr1** zu **pr1'** geleitet werden, oder der Proxy **pr1'** selbst.

[0043] Bei Abschluß des Bewegungsprozesses für den mobilen Server ist der Proxy pr1 nicht mehr erforderlich und wird gelöscht.

[0044] Es ist klar, daß Client-Agenten, wie z. B. der am Ort P1 in **Fig.** 3 gezeigte Agent MA2, in herkömmlicher Weise mobil sein können, entsprechend

der OMG-Spezifikation für mobile Agenten. Somit kann der Client-Agent MA2 in herkömmlicher Weise durch Serialisieren des Agenten, Übertragen desselben über das Netz **3** zu einem anderen Ort und Deserialisieren des Agenten am neuen Ort bewegt werden. Somit ist es gemäß der Erfindung möglich, eine gesamte Client-Server-Kombination von einem Ort zu einem weiteren oder verschiedenen Orten zu bewegen.

[0045] Es ist klar, daß der mobile Server **MS1**, wenn er sich an einem bestimmten Ort befindet, sich im Arbeitsspeicher eines bestimmten Computers innerhalb des Host befindet und bei Bedarf im nichtflüchtigen Speicher des dem Ort P zugeordneten Computers gespeichert werden kann, um eine Aufzeichnung desselben zu schaffen, wenn das Netz oder ein Teil desselben heruntergefahren wird. Der mobile Server kann ferner auf einem Speichermedium wie z. B. einer optischen oder magnetischen Platte zur Verfügung stehen, so daß er an einem bestimmten Ort P in einen Computer geladen werden kann und anschließend seine mobilen Aktivitäten im Netz aufnimmt.

[0046] Während die vorher beschriebenen Clients und Server in geeigneter Weise als Softwareobjekte in einer objektorientierten Umgebung konfiguriert sein können, ist dies nicht wesentlich, wobei sie als Stapel von herkömmlichem Code konfiguriert sein können. Während die Erfindung ferner mit Bezug auf eine CORBA-Objekt-Managementarchitektur beschrieben worden ist, können andere Management-Architekturen verwendet werden, wie z. B. OLE von Microsoft, die geeignet konfiguriert sind, um mobile Objekte zu behandeln.

[0047] Die Bewegung des Servers gemäß der Erfindung macht den Rechenprozeß viel flexibler. Wenn z. B. in einer Internet-Anwendung eine große Anzahl von Clients im Vereinigten Königreich einen Server aufruft, der sich an einem Ort in den USA befindet, müßte eine große Anzahl von Transatlantik-Aufrufen gemacht werden, was unwirtschaftlich ist. Gemäß der Erfindung kann das Server-Objekt von einem Ort in den USA zu einem Ort im Vereinigten Königreich verlagert werden, wodurch die Ausführung der individuellen Client/Server-Prozesse beschleunigt wird.

Patentansprüche

1. Verfahren zum Verarbeiten von Daten in einer verteilten Rechenumgebung, in der ein Client (**MA1**) und ein Server (**MS1**) Daten verarbeiten, **dadurch gekennzeichnet**, daß das Verfahren das Senden des Servers (**MS1**) von einem ersten Ort (P1), an dem er mit dem Client (**MA1**) kommuniziert, über die verteilte Rechenumgebung zu einem zweiten, anderen Ort (P2), damit er von hier eine Datenverarbeitung ausführt, umfaßt.

2. Verfahren nach Anspruch 1, das das Einfrieren ankommender Aufrufe für eine Datenverarbeitung beim Server (**MS1**) an dem ersten Ort (P1), während er von dem ersten Ort (P1) zu dem zweiten Ort (P2) gesendet wird, und danach das Leiten der eingefrorenen Aufrufe zu dem zweiten Ort (P2), damit sie durch den Server (**MS1**) verarbeitet werden, wenn er an dem zweiten Ort (P2) arbeitsfähig geworden ist, umfaßt.

3. Verfahren nach Anspruch 2, das das Warten auf den Server (**MS1**) umfaßt, damit er seine momentanen Verarbeitungsaufgaben abschließt, bevor er zu dem zweiten Ort (P2) gesendet wird.

4. Verfahren nach einem vorhergehenden Anspruch, das das Umwandeln des Servers (**MS1**) von einer Arbeitskonfiguration am ersten Ort (P1) in eine Konfiguration, die für die Übertragung durch die verteilte Umgebung zu dem zweiten Ort (P2) geeignet ist, umfaßt.

5. Verfahren nach Anspruch 4, bei dem die Umwandlung die Serialisierung des Servers (**MS1**) umfaßt.

6. Verfahren nach einem vorhergehenden Anspruch, das das Erzeugen eines Proxy (**pr1**) für den Server (**MS1**) an dem ersten Ort (P1), der das Senden des Servers (**MS1**) zu dem zweiten Ort (P2) steuert, umfaßt.

7. Verfahren nach einem vorhergehenden Anspruch, das das Senden des Client (**MA1**) an einen anderen Ort in der verteilten Rechenumgebung umfaßt.

8. Verfahren zum Verarbeiten von Daten in einer verteilten Rechenumgebung, bei dem ein Client (**MA1**) und ein Server (**MS1**) Daten verarbeiten, dadurch gekennzeichnet, daß das Verfahren das Empfangen des Servers (**MS1**), der von einem ersten Ort (P1), wo er mit dem Client (**MA1**) kommuniziert, über die verteilte Rechenumgebung gesendet wird, an einem zweiten, anderen Ort (P2) umfaßt, damit er an dem zweiten Ort (P2) eine Datenverarbeitung ausführt.

9. Verfahren nach Anspruch 8, bei dem der Server (**MS1**) an dem zweiten Ort (P2) in einer Form empfangen wird, die für die Übertragung über die verteilte Umgebung geeignet ist, und das das Umwandeln des empfangenen Servers (**MS1**) an dem zweiten Ort (P2) in eine Form, die für die Verarbeitung von Daten an dem zweiten Ort (P2) geeignet ist, umfaßt.

10. Verfahren nach Anspruch 9, bei dem die Umwandlung die Deserialisierung des Servers (**MS1**) umfaßt.

11. Verfahren nach Anspruch 8, 9 oder 10, das das Erzeugen eines Proxy (**pr1'**) für den empfange-

nen Server (**MS1**) an dem zweiten Ort (P2) umfaßt.

12. Verfahren nach einem der Ansprüche 8 bis 11, das das Empfangen von von dem ersten Ort (P1) zum zweiten Ort (P2) geleiteten Datenverarbeitungsaufrufen für den Server (**MS1**), nachdem der Server (**MS1**) an dem zweiten Ort (P2) arbeitsfähig geworden ist, umfaßt.

13. Software-Entität, die so betreibbar ist, daß sie einen Server (**MS1**) für einen Client (**MA1**) in einer verteilten Rechenumgebung bereitstellt, dadurch gekennzeichnet, daß die Software-Entität über die Umgebung wahlweise zu verschiedenen Orten verschoben werden kann.

14. Entität nach Anspruch 13, die so betreibbar ist, daß sie als der Server (**MS1**) an einem ersten Ort (P1) in der Umgebung arbeitet und dann verschoben wird und als der Server (**MS1**) an dem zweiten Ort (P2) in der Umgebung arbeitet.

15. Entität nach Anspruch 13 oder 14, die so betreibbar ist, daß Datenaufrufe an sie von einem Client (**MA1**) während der Verschiebung eingefroren werden.

16. Entität nach einem der Ansprüche 13 bis 15, die so betreibbar ist, daß sie einen Proxy (**pr1**) bereitstellt, der so arbeitet, daß er den Server (**MS1**) über die Umgebung sendet, um die Verschiebung zu erzielen.

17. Entität nach Anspruch 16, bei der der Proxy (**pr1**) so arbeitet, daß er auf den Server (**MS1**) wartet, damit er seine momentanen Verarbeitungsaufgaben abschließt, bevor die Verschiebung begonnen wird.

18. Entität nach Anspruch 16 oder 17, bei der der Proxy (**pr1**) so betreibbar ist, daß er den Server (**MS1**) aus seiner Arbeitskonfiguration in eine Konfiguration serialisiert, die für die Übertragung über die verteilte Umgebung geeignet ist, um so die Verschiebung zu erzielen.

19. Software-Entität nach einem der Ansprüche 13 bis 18, die in einem Speichermedium gespeichert ist.

20. Signal für die Übertragung in einer verteilten Rechenumgebung, in der ein Client (**MA1**) und ein Server (**MS1**) Daten verarbeiten, dadurch gekennzeichnet, daß das Signal den Server (**MS1**) umfaßt, der für die Übertragung über die verteilte Rechenumgebung zwischen einem ersten Ort (P1), wo er mit dem Client (**MA1**) kommuniziert, und einem zweiten, anderen Ort (P2) serialisiert wird und eine Datenverarbeitung ausführt.

21. Proxy (**pr1**) für die Verwendung in einer verteilten Rechenumgebung, in der ein Client (**MA1**) und ein Server (**MS1**) Daten verarbeiten, dadurch gekennzeichnet, daß der Proxy (**pr1**) so betreibbar ist, daß er den Server (**MS1**) von einem ersten Ort (P1), wo er mit dem Client (**MA1**) kommuniziert, über die verteilte Rechenumgebung zu einem zweiten, anderen Ort (P2) sendet, damit er eine Datenverarbeitung ausführt.

22. Proxy (**pr1**) nach Anspruch 21, der so betreibbar ist, daß er ankommende Datenverarbeitungsaufrufe für den Server (**MS1**) an dem ersten Ort (P1) einfriert, während dieser von dem ersten Ort (P1) zu dem zweiten Ort (P2) gesendet wird, und danach die eingefrorenen Aufrufe zu dem zweiten Ort (P2) leitet, damit sie von dem Server (**MS1**) verarbeitet werden, wenn er an dem zweiten Ort (P2) arbeitsfähig geworden ist.

23. Proxy (**pr1**) nach Anspruch 21 oder 22, der so betreibbar ist, daß er auf den Server (**MS1**) wartet, damit dieser seine momentanen Verarbeitungsaufgaben abschließen kann, bevor er ihn zu dem zweiten Ort (P2) sendet.

24. Proxy (**pr1**) nach Anspruch 21, 22 oder 23, der so betreibbar ist, daß er den Server (**MS1**) aus einer Arbeitskonfiguration an dem ersten Ort (P1) in eine Konfiguration serialisiert, die für die Übertragung über die verteilte Umgebung zu den zweiten Ort (P2) geeignet ist.

25. Host (**1** oder **2**), der mit Client-Objekten (**MA1**) und Server-Objekten (**MS1**) für die Verarbeitung von Daten in einer objektorientierten verteilten Verarbeitungsumgebung versehen ist, dadurch gekennzeichnet, daß das Server-Objekt (**MS1**) wahlweise an andere Orte in der Umgebung verschiebbar ist.

26. Host (**1** oder **2**) nach Anspruch 25, bei dem das mobile Server-Objekt (**MS1**) so betreibbar ist, daß Datenaufrufe für das Server-Objekt (**MS1**) während der Verschiebung eingefroren werden.

27. Host (**1** oder **2**) nach Anspruch 25, bei dem der Server (**MS1**) mit einem Proxy (**pr1**) versehen ist, der mit einer CORBA- oder OLE-Architektur kompatibel ist.

28. Server-Objekt (**MS1**) für die Verarbeitung von Daten in einer objektorientierten verteilten Verarbeitungsumgebung, dadurch gekennzeichnet, daß das Server-Objekt (**MS1**) für eine Operation an unterschiedlichen Orten verschiebbar ist und im Betrieb mit einem Proxy (**pr1**) versehen ist, der Datenaufrufe für das Server-Objekt (**MS1**) während der Verschiebung einfriert und anschließend diese Datenaufrufe zu dem bewegten Server-Objekt (**MS1**) weiterleitet.

Es folgen 3 Blatt Zeichnungen

Anhängende Zeichnungen

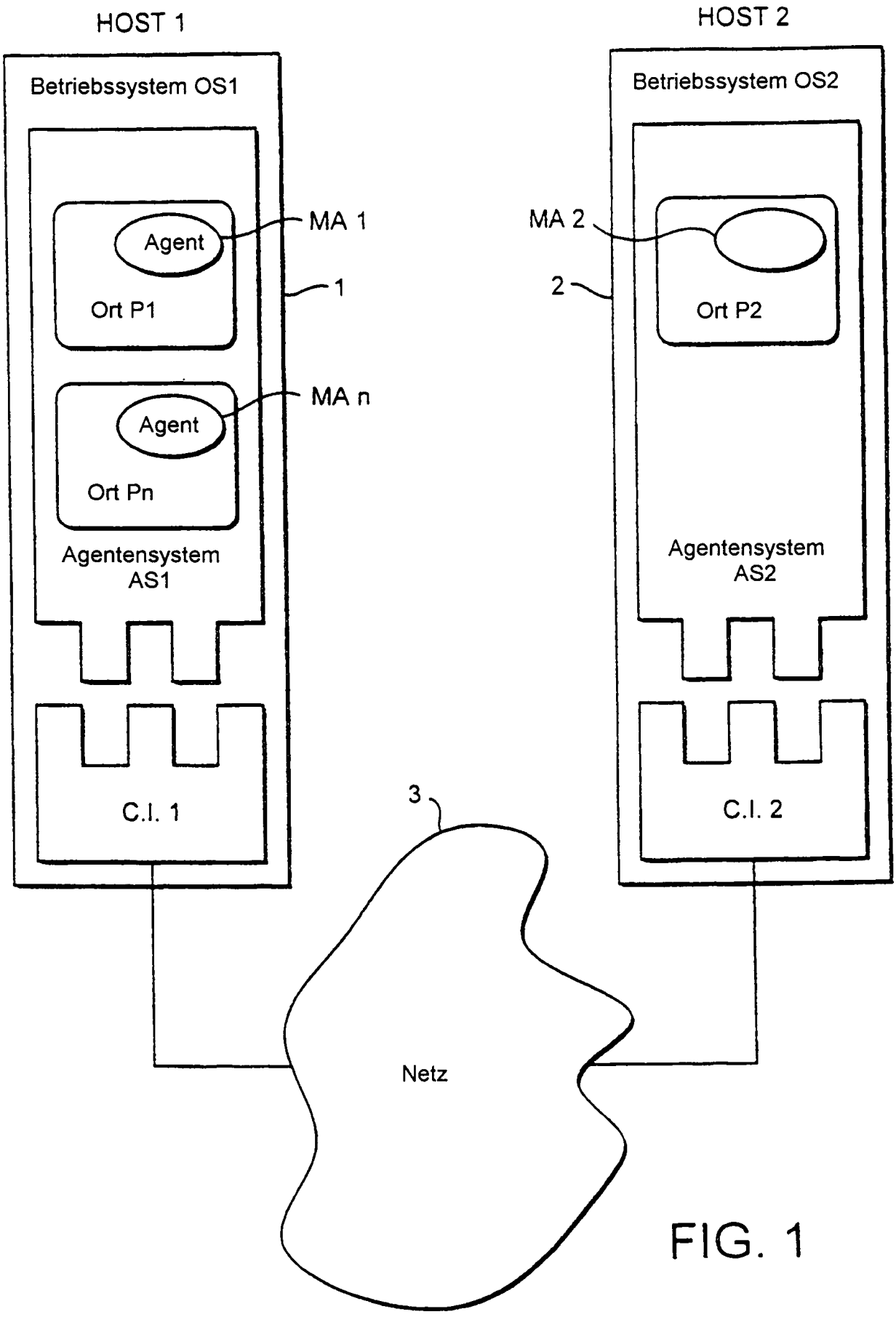

FIG. 1

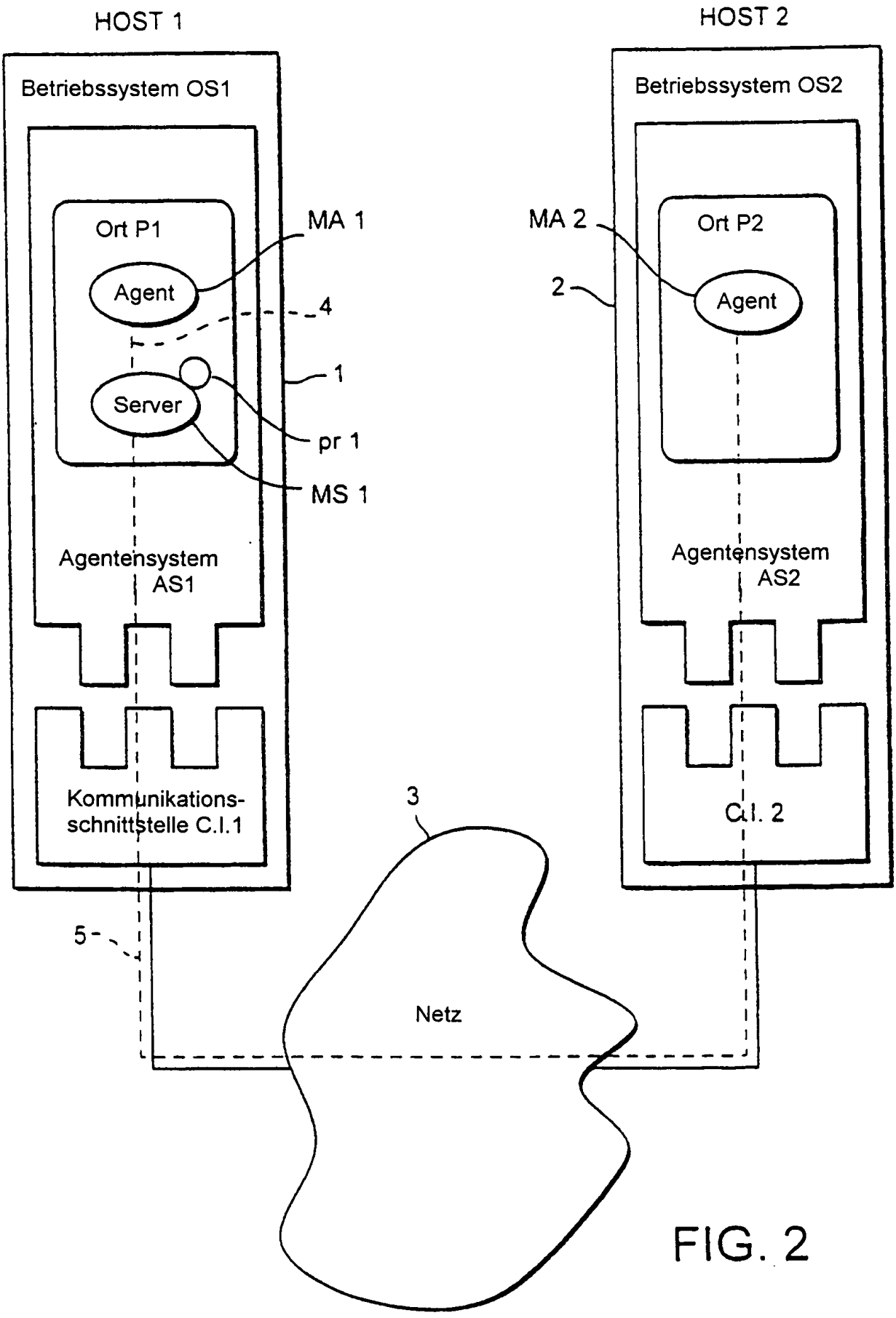
HOST 1
Betriebssystem OS1
Ort P1
MA 1
Agent
4
1
Server
pr 1
MS 1
Agentensystem
AS1
Kommunikations-
schnittstelle C.I.1
HOST 2
Betriebssystem OS2
MA 2
Ort P2
2
Agent
Agentensystem
AS2
C.I. 2
3
5
Netz

FIG. 2

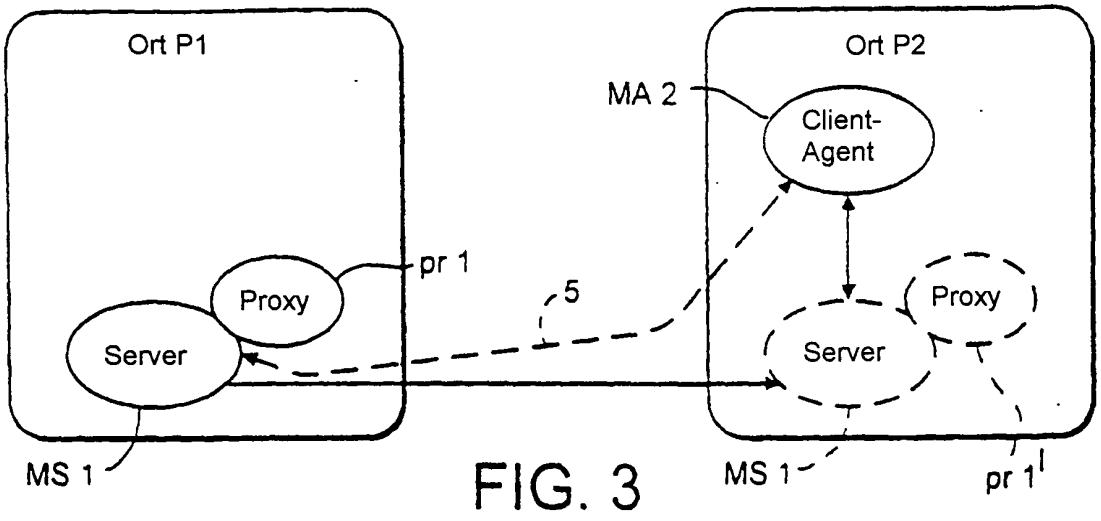
Ort P1
Ort P2
MA 2
Client-
Agent
pr 1
Proxy
5
Proxy
Server
Server
MS 1
MS 1
pr 1'

FIG. 3

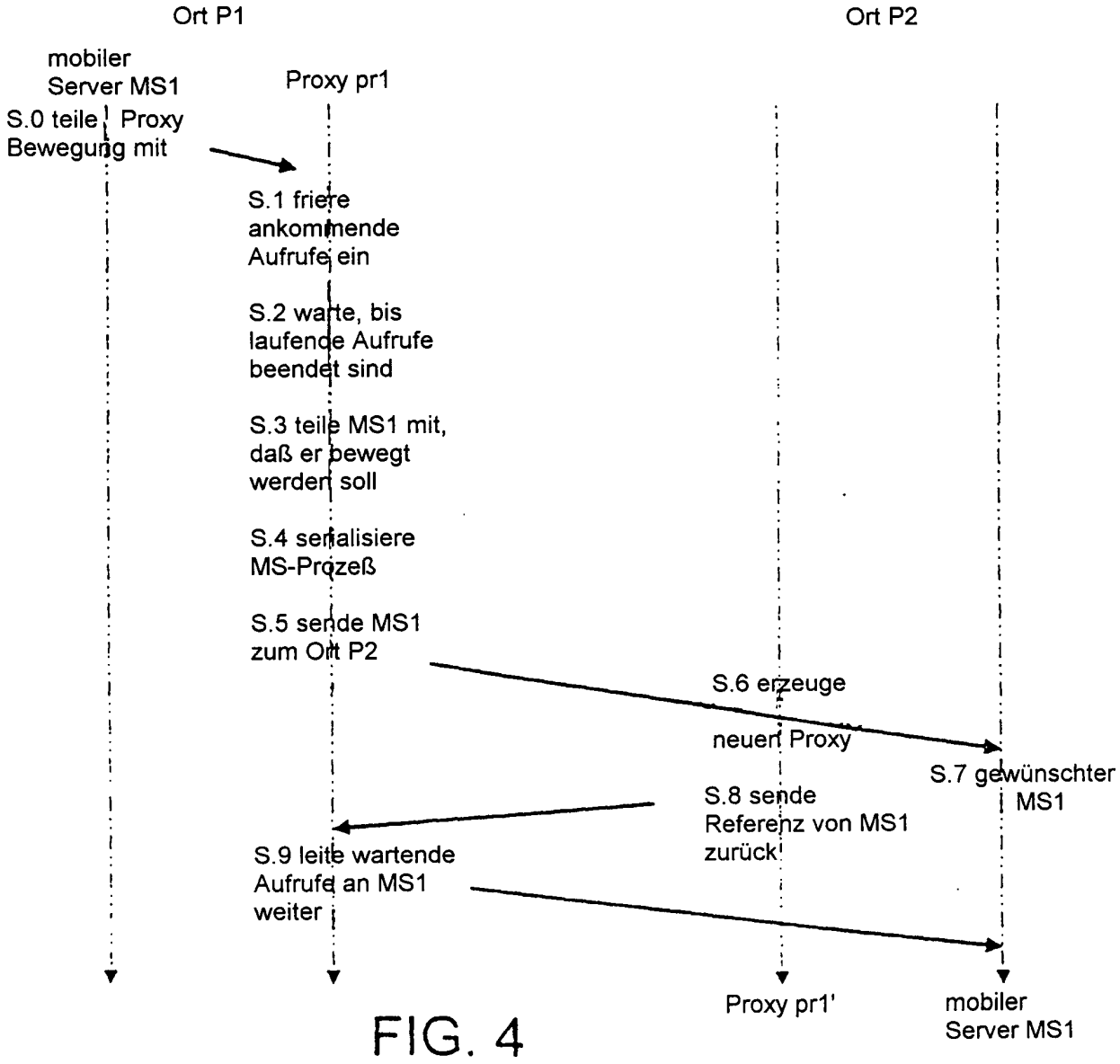
Ort P1
Ort P2
mobiler
Server MS1
Proxy pr1
S.0 teile Proxy
Bewegung mit
S.1 friere
ankommende
Aufrufe ein
S.2 warte, bis
laufende Aufrufe
beendet sind
S.3 teile MS1 mit,
daß er bewegt
werden soll
S.4 serialisiere
MS-Prozeß
S.5 sende MS1
zum Ort P2
S.6 erzeuge
neuen Proxy
S.7 gewünschter
MS1
S.8 sende
Referenz von MS1
zurück
S.9 leite wartende
Aufrufe an MS1
weiter
Proxy pr1'
mobiler
Server MS1

FIG. 4