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

(10) **DE 600 36 503 T2** 2008.06.26

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 031 926 B1**
(21) Deutsches Aktenzeichen: **600 36 503.4**
(96) Europäisches Aktenzeichen: **00 400 461.0**
(96) Europäischer Anmeldetag: **21.02.2000**
(97) Erstveröffentlichung durch das EPA: **30.08.2000**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **26.09.2007**
(47) Veröffentlichungstag im Patentblatt: **26.06.2008**

(51) Int Cl.[8]: ***G06F 9/46*** (2006.01)

(30) Unionspriorität:
9902358 **25.02.1999** **FR**

(84) Benannte Vertragsstaaten:
AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LI, LU, MC, NL, PT, SE

(73) Patentinhaber:
Alcatel Lucent, Paris, FR

(74) Vertreter:
Knecht, U., Dipl.-Ing.(Univ.), Pat.-Anw., 70435 Stuttgart

(72) Erfinder:
Pietre, Armel, 75015 Paris, FR; Banctel, Fabrice, 91190 Gif-sur-Yvette, FR; Feray, Alexandre, 31000 Toulouse, FR

(54) Bezeichnung: **Verfahren zur Kommunikation zwischen Fernobjekten**

Anmerkung: Innerhalb von neun Monaten nach der Bekanntmachung des Hinweises auf die Erteilung des europäischen Patents kann jedermann beim Europäischen Patentamt gegen das erteilte europäische Patent Einspruch einlegen. Der Einspruch ist schriftlich einzureichen und zu begründen. Er gilt erst als eingelegt, wenn die Einspruchsgebühr entrichtet worden ist (Art. 99 (1) Europäisches Patentübereinkommen).

Die Übersetzung ist gemäß Artikel II § 3 Abs. 1 IntPatÜG 1991 vom Patentinhaber eingereicht worden. Sie wurde vom Deutschen Patent- und Markenamt inhaltlich nicht geprüft.

Beschreibung

[0001] Die vorliegende Erfindung bezieht sich auf ein Verfahren zur Kommunikation zwischen Fernobjekten, basierend auf einem Manager für verteilte Objekte ORB, was das angelsächsische Akronym für Object Request Broker ist.

[0002] Als besonders bekannte verwendete ORB kann man CORBA nennen, was das englische Akronym für Common Object Request Broker Architecture ist, und DCOM für Distributed Component Object Model. CORBA wird beispielsweise in Kapitel 11 des Werks von Robert Orfali et al. „The Essential Distributed Objects Survival Guide", 1996, beschrieben.

[0003] Zahlreiche Anwendungen verwenden eine solche Umgebung. Als Beispiel lassen sich die Überwachungsanwendungen im Bereich Telekommunikation oder Verkehr nennen, wobei die Anwendungen ein intelligentes Netz bilden ...

[0004] In einer Umgebung mit verteilten Objekten kann eine Anwendung verschiedene Server nutzen, um Dienste an Clients zu liefern. Als Client-Prozess wird ein Programm bezeichnet, das Dienste nutzt, und als Server-Prozess ein Programm, das Clients Dienste anbietet. Ein Objekt des Client-Prozesses kann einen Dienst vom Server in Anspruch nehmen, indem es ihm eine entsprechende Meldung schickt. Ein entsprechendes Objekt des Servers führt den Dienst aus und sendet gegebenenfalls eine Antwort an das anfordernde Objekt. Die Kommunikation zwischen diesen beiden Fernobjekten erfolgt in der Praxis entsprechend einem Objekt-Objekt-Protokoll.

[0005] Gemäß diesem Protokoll, dessen Funktionsprinzip in Fig. 1 dargestellt wird, läuft die entsprechende Meldung, wenn ein Objekt ObjC eines Client-Prozesses PROCLIENT einen beliebigen Dienst bezogen auf ein bestimmtes Server-Objekt eines Server-Prozesses in Anspruch nimmt, über ein Repräsentantenelementepaar dieses Objekts. Dieses Paar wird die entsprechenden Interprozessaufrufe managen, und zwar transparent für die beiden Fernobjekte, bei denen es sich um das Client-Objekt und das Server-Objekt handelt.

[0006] Wenn es sich beispielsweise um ein Objekt ObjA des Server-Prozesses handelt, wird ein Repräsentantenelement Proxy(A) dieses Objekts ObjA im Client-Prozess erstellt und ein entsprechendes Repräsentantenelement Stub(A) wird im Server-Prozess erstellt. Alle Meldungen des Objekts ObjC des Client-Prozesses, die für das Objekt ObjA des Server-Prozesses bestimmt sind, und die etwaigen Antworten (Ergebnisse) auf diese Meldungen laufen über diese Schnittstelle Proxy/Stub des Objekt-Objekt-Protokolls.

[0007] Man hat somit ein Repräsentantenelementepaar Proxy/Stub für jedes Objekt eines Server-Prozesses, mit dem ein Client-Objekt kommunizieren will. Diese Proxy/Stub-Terminologie entspricht derjenigen der DCOM-Umgebung. Sie entspricht der Stub/Skeleton-Terminologie der CORBA-Umgebung. Im Folgenden werden wir die DCOM-Terminologie beibehalten, aber der Fachmann kann alles, was folgt, auf jedweden bekannten ORB ausweiten.

[0008] In Fig. 1 haben wir somit zwei Repräsentantenelementepaare Proxy(A)/Stub(A) und Proxy(B)/Stub(B) für den jeweiligen Zugang durch ein Client-Objekt ObjC zu verschiedenen Diensten, die von den Objekten ObjA und ObjB des Server-Prozesses geliefert werden.

[0009] Dieses Objekt-Objekt-Protokoll findet ebenso gut Anwendung auf Meldungen, die von Server-Objekten an Client-Objekte übermittelt werden. Ein von ORB unterstützter Benachrichtigungsdienst ermöglicht es nämlich einem Client-Objekt eines Client-Prozesses sich anzumelden, um Benachrichtigungen von Server-Prozessen zu empfangen. Ein typisches Beispiel für die Verwendung eines Benachrichtigungsdienstes betrifft die Überwachungsanwendungen, bei denen die Clients über Änderungen von Objekt-Eigenschaften informiert werden wollen. Gemäß dem Objekt-Objekt-Protokoll und wie schematisch in Fig. 2 dargestellt: wenn ein Client-Objekt sich eingetragen hat, um ein sammelndes Objekt SinkC von Benachrichtigungen eines Objekts ObjA eines Server-Prozesses PROSERV zu sein, werden die von diesem Server-Objekt an dieses Client-Objekt gesendeten Benachrichtigungen über ein Repräsentantenelementepaar des sammelnden Objekts übertragen. Dieses Paar beinhaltet ein Repräsentantenelement Proxy(C) des sammelnden Clients SinkC im Server-Prozess PROSERV und ein entsprechendes Repräsentantenelement Stub(C) im Client-Prozess PROCLIENT.

[0010] Eine Benachrichtigungsmeldung n(xa, ...) vom Server-Objekt ObjA an den sammelnden Client SinkC wird an das Repräsentantenelement Proxy(C) übermittelt, das sie auf das entsprechende Repräsentantenelement Stub(C) des sammelnden Clients SinkC sendet, welcher somit die Benachrichtigungsmeldung n(xa, ...) empfängt, die für ihn bestimmt ist, ohne dass die Objekte ObjA und SinkC ihre jeweilige Lokalisierung kennen müssen.

[0011] Der Vorteil eines solchen Objekt-Objekt-Protokolls liegt darin, dass die Objekte nicht wissen müssen, wo die Objekte lokalisiert sind, mit denen sie Meldungen austauschen. Mit diesem Protokoll verdeckt der ORB in gewisser Weise die Lokalisierung der Objekte, was den Zugang zu den Objekten unabhängig von dem Ort, wo sie sich befinden, erheblich vereinfacht. Diese Repräsentantenelementepaare,

die dem Fachmann wohl bekannt sind, werden später nicht näher detailliert. Eine derartige Architektur, die auf einem Repräsentantenelementepaar zur Durchführung der Kommunikation zwischen den Objekten eines ersten Prozesses und denjenigen eines zweiten Prozesses basiert, wird beispielsweise in der Patentanmeldung EP 0 860 776 beschrieben.

[0012] Bei manchen Anwendungen können es sehr viele Meldungen sein, die gemäß diesem Protokoll in die eine oder andere Richtung ausgetauscht werden.

[0013] Man kann mehrere Beispiele wählen. Wenn man den Dienst betrachtet, der in der angelsächsischen Literatur als „life cycle service" bezeichnet wird, und der es einem Client-Objekt gestattet in Erfahrung zu bringen, ob ein Server-Objekt noch immer zugänglich ist, das heißt, dass es nicht vernichtet wurde, oder ob das Netz nicht defekt ist, werden von allen interessierten Client-Objekten zyklisch Meldungen bezogen auf die Server-Objekte gesendet, für die sie diese Information haben wollen. Nun ist es folgendermaßen: wenn der Server-Prozess nicht zugänglich ist, sind sämtliche Server-Objekte seines Prozesses ebenfalls nicht zugänglich.

[0014] Wenn man einen so genannten Identitäts-Dienst nimmt, mit Hilfe dessen ein Client-Objekt bei einem Server-Objekt die Rücksendung des Namens des Prozesses, in dem es sich befindet, anfordern kann, können alle Client-Objekte diesen Dienst bei Server-Objekten anfordern, die auf ein und demselben Prozess sein können.

[0015] Bei einem anderen Beispiel, das sich auf den Benachrichtigungsdienst bezieht, der insbesondere bei der Überwachung eingesetzt wird, kann es zahlreiche Benachrichtigungsmeldungen zu den Eigenschaften (Aktivität, Status) diverser Ausrüstungen eines Systems geben.

[0016] Ein anderes Beispiel ist das Beispiel der Meldungen über die Anmeldung bei einem Dienst, der von einem Server-Prozess ausgeführt wird, beispielsweise ein Benachrichtigungskanal. Wenn ein Client-Prozess erstellt wird, registriert er typischerweise die Objekte, die er bei einem (oder mehreren) Server-Prozess(en) enthält, ganz oder teilweise. So wird verständlich, dass die Anzahl der Anmeldungsmeldungen sehr groß sein kann.

[0017] Bei diesen Beispielen sieht man, dass die Anzahl der Interprozessaufrufe nicht optimiert ist. Es wäre wichtig, die Aufrufzahl zu senken.

[0018] Es lohnt sich also, die Meldungen zwischen zwei Prozessen zu gruppieren, um sie als Gruppe zu schicken, so dass die kostspieligen Interprozessaufrufe verringert werden, und die Kommunikation zu optimieren.

[0019] Aus diesen Gründen wurde für manche Dienste die Zentralisierung der Meldungen vorgesehen. Diese Zentralisierungsdienste empfangen also alle Meldungen der Prozesse, die Teilnehmer an dem entsprechenden Dienst sind, und bearbeiten diese Meldungen. Ein Zentralisierungsdienst kann beispielsweise die Meldungen abändern, filtern, sie in regelmäßigen Abständen an die Bestimmungsprozesse schicken, ...

[0020] Ein solches Kommunikationsverfahren für einen Benachrichtigungsdienst wird schematisch in Fig. 3 dargestellt. Mit einem solchen Verfahren muss ein Objekt ObjA eines Server-Prozesses seine Benachrichtigungsmeldung mit einem Parameter zur Identifikation des sammelnden Bestimmungsobjekts SinkD übergeben, typischerweise ein Pointer pSinkD. Das Objekt ObjA sendet also eine Meldung vom Typ n(pSinkD), xa, ...) und das sammelnde Objekt SinkD empfängt eine Meldung vom Typ n(xa, ...). Man sieht, dass mit einem solchen Dienst die Transparenz des Objekt-Objekt-Protokolls verloren geht: das sendende Objekt muss einen Pointer hinsichtlich des Bestimmungsobjekts liefern.

[0021] Dieser zentralisierte Dienst ist also ein weiteres Zwischenglied im Kommunikationsprotokoll, mit dem man den Vorteil der Transparenz des Objekt-Objekt-Protokolls verliert.

[0022] Ein Gegenstand der vorliegenden Erfindung ist ein Verfahren zur Kommunikation zwischen Fernobjekten, das diesen Nachteil nicht aufweist.

[0023] Gemäß der Erfindung nutzt man die Mechanismen des Objekt-Objekt-Protokolls des ORB, das heißt ohne in die internen Mechanismen des ORB hineinzugehen, um ein Kommunikationsprotokoll durchzuführen, das die Zentralisierung der Meldungen gestattet, und gleichzeitig die vom Objekt-Objekt-Protokoll gebotene Transparenz bewahrt.

[0024] Indem man über dem ORB bleibt, hat man ein vorteilhaft tragbares Kommunikationsverfahren, das heißt eines, das auf jedweden ORB anwendbar ist.

[0025] In einer ORB-Umgebung ist es nämlich möglich, die Repräsentantenelemente vom Typ Proxy/Stub des Objekt-Objekt-Protokolls gemäß bekannten Mechanismen, die vom ORB abhängig sind, zu modifizieren. Gemäß der Erfindung nutzt man diese Eigenschaft, um die Meldungen durch Zentraldienstmanager zu übermitteln, die jeweils in den Prozessen vorgesehen sind.

[0026] Gemäß der Erfindung erstellt man also ein Managerobjekt in jedem der Prozesse, um die Meldungen bezüglich eines bestimmten Dienstes zu zentralisieren.

[0027] Für diesen Dienst erstellt man einen rufenden Manager im rufenden Prozess und einen empfangenden Manager im Bestimmungsprozess.

[0028] Die Repräsentantenelementepaare werden abgeändert, um die Übermittlung der Meldungen betreffend den betrachteten Dienst zu ermöglichen, welche von den Elementen vom Typ Proxy empfangen wurden, und zwar von einem rufenden Prozess hin zu dem diesem Dienst entsprechenden rufenden Manager, der in diesem Prozess vorgesehen ist.

[0029] Dieser rufende Manager kann die Meldungen je nach Bedarf des Dienstes bearbeiten. Er kann sie beispielsweise filtern, die Parameter abändern, sie sortieren und gruppieren, um sie an entsprechende in den Bestimmungsprozessen vorgesehene empfangende Manager zu übermitteln ...

[0030] Jeder dieser empfangenden Manager der Bestimmungsprozesse kann dann die so empfangenen Meldungen an die Bestimmungsobjekte verteilen oder selbst den entsprechenden Dienst erweisen. Die Erfindung hat also ein Kommunikationsverfahren zum Gegenstand, das auf einem Manager für verteilte Objekte ORB basiert, wobei die Übermittlung von Meldungen von Objekten eines ersten Prozesses hin zu Objekten eines anderen Prozesses durch Repräsentantenelementepaare erfolgt, mit in jedem Paar einem ersten Repräsentantenelement im rufenden Prozess und einem zweiten Repräsentantenelement im anderen Prozess. Dieses Verfahren ist dadurch gekennzeichnet, dass es darin besteht, die ersten Repräsentantenelemente des ersten Prozesses abzuändern, damit sie die Meldungen bezüglich eines bestimmten Dienstes, die von Objekten des ersten Prozesses hin zu Objekten anderer Prozesse gesendet werden, an einen diesem Dienst entsprechenden rufenden Manager übermitteln, der im ersten Prozess vorgesehen ist, wobei dieser rufende Manager die so empfangenen Meldungen managt und einige dieser Meldungen auf einen entsprechenden empfangenden Manager überträgt, der jeweils in den Bestimmungsprozessen vorgesehen ist.

[0031] Dieses Kommunikationsverfahren findet ebenso auf Dienste Anwendung, für welche Client-Objekte Meldungen hin zu Server-Objekten übergeben oder Dienste, für welche Server-Objekte Benachrichtigungsmeldungen an Client-Objekte senden.

[0032] Ein weiterer Zweck der Erfindung ist die Reduzierung der Anzahl von Registrierungsmeldungen, die von den Objekten eines Client-Prozesses an die Adresse eines Server-Prozesses gesendet werden. Hierfür hat die Erfindung ein Verfahren zur Eintragung eines Client-Prozesses bei einem Server-Prozess zum Gegenstand, wobei die Objekte dieses Client-Prozesses ganz oder teilweise jeweils eine Eintragungsmeldung senden. Das Verfahren wird dadurch gekennzeichnet, dass die Eintragungsmeldungen dem Server-Prozess gemäß dem vorstehend beschriebenen Kommunikationsverfahren mitgeteilt werden.

[0033] Weitere Kennzeichen und Vorteile der Erfindung werden in der nachfolgenden Beschreibung beschrieben, die zur Information und keineswegs einschränkend unter Bezugnahme auf die beigefügten Zeichnungen erfolgt, wobei:

[0034] die bereits beschriebene Fig. 1 ein allgemeines Schema für die Ausführung eines Objekt-Objekt-Protokolls für die Übermittlung von Meldungen von einem Client-Prozess zu einem Server-Prozess ist;

[0035] die bereits beschriebene Fig. 2 ein allgemeines Schema für die Ausführung eines Objekt-Objekt-Protokolls für die Übermittlung der Benachrichtigungsmeldungen von einem Server-Prozess hin zu einem Client-Prozess ist;

[0036] die bereits beschriebene Fig. 3 schematisch das Managen der Meldungen zwischen einem Client-Prozess und einem Server-Prozess und umgekehrt durch einen zentralisierten Benachrichtigungsdienst gemäß dem Stand der Technik darstellt;

[0037] Fig. 4 schematisch das Management der Meldungen gemäß dem Kommunikationsverfahren laut der Erfindung zwischen einem Client-Prozess und einem Server-Prozess darstellt;

[0038] Fig. 5 das Managen der Meldungen gemäß dem Kommunikationsverfahren aus der der Erfindung angewandt auf einen Benachrichtigungsdienst darstellt; und

[0039] Fig. 6 schematisch einen Mechanismus zur Abänderung eines Parameters der Meldungen darstellt.

[0040] Fig. 4 entspricht einer ersten Ausführungsart der Erfindung angewandt auf einen bestimmten Dienst, den wir als SX schreiben, für die Übermittlung der diesem Dienst entsprechenden Meldungen von einem Client-Objekt ObjC eines Client-Prozesses $PROCLIENT_1$ hin zu Server-Objekten ObjA und ObjB eines Server-Prozesses $PROSERV_1$.

[0041] Der Client-Prozess kann verschiedene Client-Objekte einschließen. Jedem dieser verschiedenen Client-Objekte können Repräsentantenelemente des Typs Proxy/Stub entsprechen, die erstellt wurden als diese Client-Objekte sich verschiedenen Diensten an bezüglich dieser Objekte angeschlossen haben.

[0042] Bei dem in Fig. 4 dargestellten vereinfachten Beispiel wurde ein einziges Client-Objekt ObjC dargestellt, für das zwei Repräsentantenelementepaare Proxy/Stub erstellt wurden, Proxy(A)/Stub(A) und Proxy(B)/Stub(B), die den Zugang zu verschiedenen Diensten ermöglichen, die von Server-Objekten ObjA und ObjB geliefert werden, die bei diesem Beispiel zum gleichen Server-Prozess $PROSERV_1$ gehören.

[0043] Der Client-Prozess $PROCLIENT_1$ schließt außerdem einen rufenden Manager $GCLIENT_1$ für den Dienst SX ein. Dieser rufende Manager empfängt Meldungen betreffend diesen Dienst SX, die von den Repräsentantenelementen vom Typ Proxy des Client-Prozesses übertragen werden. Wenn sich die Meldung nicht auf den Dienst SX bezieht, kann das erste Repräsentantenelement Proxy Standardverhalten haben, das darin besteht, diese Meldungen auf das entsprechende zweite Repräsentantenelement Stub zu leiten, oder aber das besondere spezialisierte Verhalten gemäß der Erfindung, das darin besteht, diese Meldungen hin zum rufenden Manager eines entsprechenden Dienstes umzulenken. Der Server-Prozess $PROSERV_1$ enthält einen empfangenden Manager $GSERV_1$ des Dienstes SX, der Meldungen von entsprechenden rufenden Managern (das heißt des gleichen Dienstes SX), von verschiedenen rufenden Prozessen erhält und sie auf die Bestimmungs-Server-Objekte verteilt.

[0044] Die Standard-Repräsentanten-Elemente des Objekt-Objekt-Protokolls, die erstellt werden, wenn sich die Client-Objekte an Dienste auf Server-Objekten anschließen, werden abgeändert, um die In-Verbindung-Setzung der rufenden Manager und Server des Dienstes SX gemäß der Erfindung zu ermöglichen. Für jede Meldung, die er empfängt, muss der rufende Manager eines Dienstes in einem Client-Prozess nämlich nicht nur das Identifikationsparameter des Fern-Server-Objekts kennen, für das sie bestimmt ist, sondern auch des entsprechenden empfangenden Managers.

[0045] Nun ist es so, dass das Client-Objekt, das die Meldung sendet, nicht die Lokalisierung des Server-Objekts kennt, das es ruft. Hingegen kennt jedes Repräsentantenelement durch das Objekt, zu dem es zugehörig ist, die Adressen der Objekte, die im Prozess präsent sind, in dem es sich befindet, oder kann diese kennen. Somit werden die Identifikationsparameter des Bestimmungs-Server-Objekts und des entsprechenden empfangenden Managers vom Repräsentantenelement Stub gekannt (oder können gekannt werden), das im betreffenden Prozess befindlich ist, oder vom entsprechenden Server-Objekt.

[0046] Bei einem Beispiel der praktischen Umsetzung der Erfindung werden diese Repräsentantenelemente Stub der Server-Prozesse also abgeändert, damit jedes seine Informationen an das zugehörige Repräsentantenelement Proxy weitergibt. Wie in Fig. 4 durch einen gestrichelten Pfeil dargestellt gibt das Element Stub(A) des Paares Proxy(A)/Stub(A) die Identifikationsparameter pObjA und $pGSERV_1$ des Server-Objekts ObjA und des Empfangender Managers $GSERV_1$ des betrachteten Server-Prozesses an das Element Proxy(A) weiter. Diese Informationsweitergabe kann bei der Erstellung des Repräsentantenelementepaares oder danach erfolgen.

[0047] Bei diesem Beispiel für die Umsetzung des Verfahrens gemäß der Erfindung bedient man sich also des Gateways Proxy/Stub, um auf der Ebene der Proxy-Elemente die Informationen zurück zu führen, die erforderlich sind, damit rufende und empfangende Manager in Beziehung gebracht werden können. Man stellt fest, dass andere Mechanismen vom Fachmann eingesetzt werden können. So kann ein anderer Mechanismus darin bestehen direkt das Bestimmungs-Server-Objekt zu verwenden, um diese Informationen an die Proxy-Elemente zu liefern, indem beispielsweise gemeinsam benutzte Mittel verwendet werden.

[0048] Gemäß der Erfindung werden die Repräsentantenelemente Proxy der Client-Prozesse also abgeändert, um die Meldungen bezüglich des Dienstes SX nicht mehr an ihr entsprechendes Repräsentantenelement Stub zu übermitteln, sondern an den in ihren Prozessen vorgesehenen rufenden Manager dieses Dienstes, wobei in diesen Meldungen jeweils die Identifikationsparameter des Bestimmungsobjekts und des entsprechenden empfangenden Managers weitergegeben werden.

[0049] Wenn das Client-Objekt ObjC eine Meldung vom Typ a(x, y, ...) bezüglich des Dienstes SX an das Objekt ObjA des Server-Prozesses $PROSERV_1$ senden will, übermittelt das entsprechende Repräsentantenelement Proxy(A) eine Meldung des Typs a(pObjA, p $GSERV_1$, x, y ...) an den rufenden Manager $GCLIENT_1$ dieses Dienstes SX, der im Client-Prozess vorgesehen ist, anstatt die Meldung unmittelbar an das Repräsentantenelement Stub(A) zu schicken.

[0050] So empfängt der rufende Manager im Client-Prozess alle Meldungen bezüglich des Dienstes SX, die von Client-Objekten des Client-Prozesses herkommen und für ein beliebiges Server-Objekt eines beliebigen Server-Prozesses bestimmt sind.

[0051] Gemäß dem betrachteten Dienst SX kann er die Meldungen entsprechend dem Bedarf des Dienstes bearbeiten. Er kann beispielsweise diese Meldungen nach Server-Prozess filtern, sortieren und gruppieren, um an jeden der Server-Prozesse eine entsprechende Gruppe empfangener Meldungen zu schicken, gemäß vorgegebenen Meldungsgruppierungs- und Sendepolitiken.

[0052] So kann er die Meldungen paketweise asynchron senden: wenn der empfangende Manager das Meldungspaket empfangen hat, gibt er den rufenden Manager frei, der sich um das Managen anderer Meldungen kümmern kann. Wenn der rufende Manager die Meldungen in Paketen sendet, kann dieses Senden zyklisch erfolgen.

[0053] Wenn das Senden zyklisch ist, so kann es sich um einen jeweils vom rufenden Manager festgelegten Zyklus handeln oder um einen Zylus, der vom empfangenden Manager festgelegt wird.

[0054] Zu einem zyklischen Sendevorgang kann das Senden hinzukommen, das durch ein besonderes Ereignis ausgelöst wird, wie beispielsweise wenn eine bestimmte Anzahl von Meldungen empfangen wurde.

[0055] Er kann diese Meldungen auch filtern, um nur manche entsprechend vorgegebenen Kriterien zu übermitteln, wie wir anschließend anhand eines Beispiels für die Anwendung auf einen Identifikationsdienst sehen werden.

[0056] Er kann auch eine Meldung zu einem entsprechenden empfangenden Manager senden und ein Rückergebnis abwarten, entsprechend einem synchronen Übermittlungsverfahren.

[0057] Es gibt also zahlreiche Varianten für die praktische Umsetzung, die im Wesentlichen von den Diensten abhängen, auf die man die Erfindung anwendet.

[0058] In Fig. 4 haben wir somit eine Anwendung der Erfindung auf einen beliebigen Dienst SX dargestellt, bei dem Client-Objekte Meldungen auf Server-Objekte senden. Je nach betrachtetem Dienst ist ein Ergebnis an das Client-Objekt zurück zu senden oder nicht.

[0059] Falls bei dem betrachteten Dienst SX ein Ergebnis zurück zu senden ist, halten die Ergebnisse normalerweise den umgekehrten Übertragungsweg, bis zum rufenden Objekt ein: ObjA → $GSERV_1$ → $GCLIENT_1$ → Proxy(A) → ObjC. Sie werden also vom empfangenden Manager und vom rufenden Manager gemäß dem umgekehrten Prozess ungefähr wie die Meldungen behandelt: der empfangende Manager empfängt die Ergebnisse direkt von den Server-Objekten; er kann sie mit den Meldungen verbinden, denen sie entsprechen, sie sortieren und gruppieren. Er kann sie paketweise an die entsprechenden rufenden Manager senden, die ihrerseits diese Ergebnisse an die entsprechenden Client-Objekte verteilen werden. Das Senden kann zyklisch erfolgen, von einem bestimmten Ereignis ausgelöst werden, ... Alles was vorstehend bezüglich des Managens der Meldungen durch die Manager gesagt wurde, gilt ebenso für das Managen der entsprechenden Ergebnisse.

[0060] Zur Gewährleistung diese Meldungsmanagements gemäß der Erfindung und gegebenenfalls des Ergebnismanagements sieht man vor, dass die Manager geeignete Datenstrukturen (Tabelle, Stapel, ...) verwenden.

[0061] Auf Tabelle 4 hat man so eine Datenstruktur TAB1 in Form einer Tabelle dargestellt, die vom rufenden Manager $GCLIENT_1$ gemanagt wird. Diese Datenstruktur enthält zu einem bestimmten Zeitpunkt die Liste der empfangenen Meldungen pro Bestimmungsobjekt für jeden betreffenden empfangenden Manager. Bei dem Beispiel enthält der rufende Manager $GCLIENT_1$ zwei Meldungen für einen empfangenden Manager, der vom Pointer $pGSERV_1$ identifiziert wird und eine Meldung für einen anderen empfangenden Manager, der durch einen Pointer $pGSERV_2$ gekennzeichnet wird.

[0062] Der empfangende Manager $GSERV_1$ des Server-Prozesses kann selbst eine Datenstruktur zu managen haben, wie die Datenstruktur TAB2, die in Fig. 4 dargestellt ist, in der er die empfangenen Meldungen speichern wird. Dieses ermöglicht es ihm insbesondere, die von ihm empfangenen Nachrichten nach Bestimmungs-Server-Objekten zu sortieren und sie an diese Server-Objekte zu senden. Das ermöglicht es ihm auch die entsprechenden Ergebnisse zu verwalten, die er etwaig als Rückantwort erhält, um sie an die entsprechenden rufenden Manager zurück zu senden.

[0063] So managt jeder Manager eine (oder mehrere) Datenstruktur(en) gemäß seinem Bedarf. Er kann dort die Meldungen, die Ergebnisse und Informationen im Zusammenhang mit seinem eigenen Management speichern, wie beispielsweise eine Anzeige für Übermittlung der Meldung, eine für Warten auf Ergebnis,...

[0064] Außerdem können der rufende und der empfangende Manager Umformungen vom Typ Filtern, Komprimierung vornehmen oder bestimmte Parameter der Meldungen und/oder von entsprechenden Ergebnissen abändern, und zwar bei jedem Übertragungsschritt dieser Meldungen und/oder dieser Ergebnisse. Diese Möglichkeit wird nachstehend anhand eines Beispiels für die Anwendung der Erfindung auf den Benachrichtigungsdienst eingehender erläutert. Aber sie kann für alle Dienste eingesetzt werden, auf die man die Erfindung anwendet, für den Bedarf des Dienstes im Allgemeinen und zur Optimierung der Interprozessrufe.

[0065] Bei der Umsetzung eines Kommunikationsverfahrens gemäß der Erfindung kann vorgesehen werden, dass der empfangende Manager des Server-Prozesses selbst den entsprechenden Dienst er-

bringt. Zur Zentralisierung der Übermittlung der Meldungen (und gegebenenfalls der Ergebnisse) fügt man also die Zentralisierung des Dienstes an sich hinzu.

[0066] Diese Anwendungsvariante der Erfindung richtet sich typischerweise an Dienste wie den Identifikationsdienst der Server-Prozesse, die ein Client-Objekt nutzt, um zu erfahren auf welchem Prozess sich das Server-Objekt befindet, an das es sich richtet, oder der Fehlererkennungsdienst, den ein Client-Objekt nutzt, um in Erfahrung zu bringen, ob das Server-Objekt, an das es sich wendet, immer noch verfügbar ist (das heißt, ob es nicht verschwunden ist oder ob das Netzwerk nicht defekt ist).

[0067] Wenn man nämlich das Beispiel des Identifikationsdienstes nimmt, kann ein zentralisiertes Objekt sehr gut den Dienst anstelle jedes Server-Objekts erbringen. Bei der Erfindung sieht man also in dieser Variante vor, dass der Dienst vom empfangenden Manager selbst erbracht wird. Somit muss für diesen Dienst der entsprechende Programmcode nur in diesem Manager implementiert werden und nicht mehr in jedem der Server-Objekte. Das ist also sehr günstig. Außerdem erfolgt das transparent für die Client-Objekte.

[0068] Bei diesem Beispiel der Anwendung auf den Identifikationsdienst kann man auch einen Teil des Dienstes auf dem rufenden Manager zentralisieren, so dass die Interprozessrufe erheblich vermindert werden. Der rufende Manager kann nämlich in seiner Datenstruktur die Identität der Server-Prozesse nach und nach so wie er sie in Erfahrung bringt speichern. Wenn er bereits die Identität eines Prozesses besitzt, für den er eine Meldung mit Identitätsanforderung empfängt, sendet er somit selbst das Ergebnis, auf das gewartet wird, an das Client-Objekt zurück. Somit übermittelt der rufende Manager für jeden Server-Prozess, für den er in Anspruch genommen wird, nur ein einziges Mal eine entsprechende Identitätsanforderung an den empfangenden Manager, nämlich bei der ersten Anforderung, die er empfangt.

[0069] Bei diesem Anwendungsbeispiel wird man feststellen, dass man nicht unbedingt Meldungspakete zwischen den Manager übermittelt; das Kommunikationsverfahren kann also synchron sein, wobei jedes beteiligte sendende Element einer Meldung in Erwartung des Rückergebnisses bleibt: das Client-Objekt, das die Meldung sendet, der rufende Manager, der sie empfängt und sie an den empfangenden Manager übermittelt, falls er das Ergebnis nicht kennt.

[0070] In Fig. 5 wurde eine Anwendung der Erfindung auf ein anderes Dienstbeispiel, nämlich den Identifikationsdienst dargestellt. In diesem Fall sind die rufenden Manager in den Server-Prozessen und die empfangenden Manager in den Client-Prozessen. Jeder Serverbeziehungsweise Client-Prozess enthält einen einzigen rufenden beziehungsweise empfangenden Manager für diesen Benachrichtigungsdienst gemäß der Erfindung. Jeder Server-Prozess kann durch seinen rufenden Manager Benachrichtigungen an verschiedene Client-Prozesse senden. Jeder Client-Prozess kann durch seinen entsprechenden empfangenden Manager Benachrichtigungen von verschiedenen Server-Prozessen empfangen.

[0071] Bei dem in Fig. 5 dargestellten Beispiel hat man somit für diesen Benachrichtigungsdienst einen rufenden Manager $GNSERV_1$ im Server-Prozess $PROSERV_1$ und einen empfangenden Manager $GNCLIENT_1$ im Client-Prozess.

[0072] Der rufende Manager empfangt alle Benachrichtigungsmeldungen aller Server-Objekte dieses Prozesses, die diese Meldungen senden. Der empfangende Manager kann Benachrichtigungen von rufenden Managern verschiedener Prozesse empfangen. Er verteilt die empfangenen Benachrichtigungen auf die sammelnden Bestimmungsobjekte. Allgemeiner gesagt erzielt man diese Unabhängigkeit zwischen den beiden Prozessen, sobald die Meldungen (oder die Rückergebnisse) paketweise zumindest zwischen den beiden Managern gesendet werden. Je nach der Art des betreffenden Dienstes kann die Asynchronität nur auf der Ebene der Manager eingeführt werden oder aber auf allen Ebenen der Übertragungskette. Zur Einführung dieser Asynchronität kann der Fachmann die herkömmlichen Techniken benutzen (leichte Prozesse (threads in der angelsächsischen Literatur), Unterbrechungen, ...).

[0073] Der rufende Manager muss die verschiedenen Benachrichtigungsmeldungen, die er von den Server-Objekten seines Server-Prozesses empfängt, entsprechend dem Bestimmungs-Client-Prozess und für jeden Bestimmungs-Prozess gemäß dem sammelnden Bestimmungsobjekt sortieren, um sie an die rufenden Bestimmungsmanager zu senden, beispielsweise paketweise, wobei auf sie zuvor eine Umformung angewandt werden kann oder nicht. Hierfür verwendet er eine geeignete Datenstruktur, in Fig. 5 geschrieben als TAB3. Bei dem Beispiel enthält diese Datenstruktur 3 Meldungen: zwei Meldungen m1 und m2 für den Client-Prozess $PROCLIENT_1$, die eine, m1, kommt vom Server-Objekt ObjA für ein sammelndes Objekt SinkC her, die andere, m2, kommt von einem Server-Objekt ObjB für ein sammelndes Objekt SinkD her; und eine letzte Meldung, m3, kommt vom Server-Objekt ObjA her für ein sammelndes Objekt SinKO eines anderen nicht dargestellten Client-Prozesses $PROCLIENT_2$.

[0074] Vorzugsweise gruppiert der rufende Manager des Server-Prozesses die Meldungen um, die für

den entsprechenden empfangenden Manager ein und desselben Client-Prozesses bestimmt sind, um sie paketweise in einem einzigen Aufruf zu senden. Bei dieser Gelegenheit haben wir festgestellt, dass er auch Umformungen auf das von ihm gesendete Meldungspaket anwenden kann, wie zum Beispiel Filter- oder Komprimierungsmechanismen. Er kann auch manche Parameter abändern.

[0075] Er kann beispielsweise die Meldungen filtern, indem er nur diejenigen überträgt, die den letzten Abänderungen bestimmter Eigenschaften entsprechen. In diesem Fall unterdrückt er die früheren Meldungen. Er kann also einen Parameter der Meldung mit Anzeigefunktion abändern, der je nach Status anzeigt, ob eine Filterung zur Anwendung kam oder nicht.

[0076] Wie in Fig. 5 dargestellt kann der empfangende Manager eines Client-Prozesses auch eine Datenstruktur TAB4 managen, um die Paare Benachrichtigungsmeldung / sammelndes Objekt zu speichern, die er von rufenden Managern empfängt. Er kann auch selbst Bearbeitungen an den Meldungen vornehmen und/oder bestimmte Parameter dieser Meldungen abändern, bevor er sie an die betreffenden Objekte verteilt.

[0077] Weiter oben haben wir gesehen, dass der rufende oder empfangende Manager eines Prozesses die Möglichkeit hat, Parameter von Meldungen oder von entsprechenden Ergebnissen abzuändern, um seine Managementtätigkeit zu erleichtern und für den Bedarf des Dienstes im Allgemeinen. Ein Beispiel für einen solchen Parameter, der von einem Manager abgeändert werden kann, ist der Anzeigeparameter für die vorstehend betrachtete Filterung.

[0078] Ein anderes Beispiel betrifft die Parameter des Typs Schlüssel. Ein Element, das zahlreiche verschiedene Meldungen sendet, verwendet in der Tat üblicherweise einen Mechanismus für die Schlüsselzuweisung. Ein Schlüssel ist eine Kennung, die vom Client zum Zeitpunkt der Verbindung zugewiesen wird. Dieser Schlüssel wird in der Benachrichtigung übermittelt und der Client kann damit identifizieren welcher Verbindung die Benachrichtigung entspricht.

[0079] Da bei der Erfindung die Übermittlung der Meldungen durch Manager zentralisiert ist, empfangen diese Manager Meldungen, die von verschiedenen Elementen gesendet werden, die jeweils ihren eigenen Schlüsselzuweisungsmechanismus anwenden. So kann jeder Manager sich mit Meldungen wieder finden, die den gleichen Schlüssel verwenden.

[0080] Gemäß der Erfindung sieht man also vor, dass jeder rufende oder empfangende Manager seinen eigenen Mechanismus für die Schlüsselzuweisung der empfangenen Meldungen verwendet. Bei dem in Fig. 6 dargestellten Beispiel empfängt ein empfangender Manager so eine Meldung von einem rufenden Manager $GCLIENT_1$ aus einem Client-Prozess für ein Server-Objekt ObjA. Diese Meldung ist vom Typ a(pObjA, X, Y, Schlüssel 1, ...), wo Schlüssel 1 der entsprechende Schlüssel ist, der vom rufenden Manager $GCLIENT_1$ zugewiesen wird.

[0081] Der empfangende Manager empfängt eine weitere Meldung von einem rufenden Manager $GCLIENT_2$ von einem anderen Client-Prozess, für das gleiche Server-Objekt ObjA. Diese Meldung ist vom Typ a(pObjA, X, Y, Schlüssel 1, ...), wo Schlüssel 1 der Schlüssel ist, der vom rufenden Manager $GCLIENT_2$ zugewiesen wird.

[0082] Wenn der empfangende Manager seine Meldungen empfängt, kann er einen neuen Schlüssel gemäß seinem eigenen Zuweisungsmechanismus (beispielsweise in der Reihenfolge des Eintreffens der Meldungen) zuweisen. In jeder Meldung ersetzt er also den empfangenen Schlüssel durch einen neuen von ihm zugewiesenen Schlüssel. Er kann die Entsprechung zwischen diesem neuen Schlüssel und dem ursprünglich empfangenen Schlüssel in der Datenstruktur TAB6 aufbewahren, um ein etwaiges entsprechendes Ergebnis an den rufenden Manager mit dem ursprünglichen Schlüssel zurücksenden zu können. Bei dem in Fig. 5 dargestellten Beispiel ist der neue Schlüssel der ersten Meldung somit $Schlüssel_{s1}$ während der Schlüssel der zweiten Meldung $Schlüssel_{s2}$ ist. Somit kann allgemein ein rufender oder empfangender Manager Parameter der Meldungen oder von entsprechenden Ergebnissen abändern.

[0083] Man wird feststellen, dass in bestimmten Fällen das Kommunikationsverfahren gemäß der Erfindung es ermöglicht die beiden Prozesse, d. h. den rufenden Prozess und den Bestimmungsprozess, unabhängig zu machen. Wenn man beim Beispiel des Benachrichtigungsdienstes die entsprechende in Fig. 5 abgebildete Übertragungskette betrachtet, sieht man dass wenn ein Objekt eine Benachrichtigungsmeldung sendet, der entsprechende rufende Manager im Server-Prozess diese empfängt, und letzterer gibt dann das Server-Objekt frei, das andere Dinge tun kann. Wenn der rufende Manager beschließt, ein Paket Benachrichtigungsmeldungen an einen empfangenden Manager zu senden, nimmt letzterer das Paket in Empfang und gibt sogleich den rufenden Manager frei, der sich um andere Pakete, andere rufende Manager kümmern kann. ... Und wenn der empfangende Manager schließlich Meldungen in Richtung Client-Objekt verteilt, gibt letzteres den empfangenden Manager ab Erhalt frei.

[0084] Mit einem Kommunikationsverfahren gemäß der Erfindung, von dem verschiedene Anwendungsbeispiele und zahlreiche Varianten gerade rein zur

Veranschaulichung beschrieben wurden, weiß ein rufendes Objekt nicht, dass es über einen rufenden Manager läuft und gegebenenfalls weiß ein Bestimmungsobjekt nicht, dass es von einem empfangenden Manager gerufen wird.

[0085] Man wird feststellen, dass man jederzeit zur normalen Kommunikation via Standard-Paar Proxy/Stub zurückkehren kann, beispielsweise um einen Einheitstest eines Server-Objekts zu ermöglichen. Im rufenden Prozess genügt es, das Proxy-Element erneut zu modifizieren, damit es erneut die Meldungen an das entsprechende Repräsentantenelement Stub übermittelt. Das Kommunikationsverfahren gemäß der Erfindung ist also umkehrbar.

[0086] Man kann feststellen, dass die Kommunikation zwischen einem rufenden Manager und einem empfangenden Manager dem Standardschema pro Paar Proxy/Stub des Objekt-Objekt-Protokolls folgen kann. Sie kann auch über gemeinsam benutzte Mittel R laufen, die auf dem ORB oder im Betriebssystem vorgesehen sind, wie in den Fig. 4 und Fig. 5 dargestellt. Bei diesen gemeinsam benutzten Mitteln kann es sich um einen Speicher, einen Netzanschluss, ... handeln.

[0087] Schließlich verwendet das Kommunikationsverfahren gemäß der Erfindung keinen internen Mechanismus des ORB, was es tragbar macht, das heißt mit jedem beliebigen ORB verwendbar.

[0088] Es gestattet es, die Anzahl der Interprozessrufe für jeden Dienst, auf den es angewandt wird, zu senken, mit Hilfe des Paketversandmechanismus für die Meldungen, mit oder ohne Umformung (Filterung, Komprimierung), und gegebenenfalls dank der Zentralisierung des Dienstes selbst, wobei gleichzeitig die Transparenz des Objekt-Objekt-Protokolls gewahrt wird.

[0089] In einigen Fällen, wie beispielsweise dem Fall des Benachrichtigungsdienstes, ermöglicht es vorteilhaft den rufenden Prozess und den Bestimmungsprozess unabhängig zu machen, denn in diesem Fall führt es eine Asynchronität zwischen diesen Prozessen ein.

[0090] In der Praxis lässt das Kommunikationsverfahren gemäß der Erfindung große Freiheit bei der Umsetzung, was eine Anpassung an die verschiedenen Anwendungen, die vorkommen können, ermöglicht.

[0091] Es ist insbesondere möglich, das Verfahren gemäß der Erfindung zu verwenden, um die Registrierung der Objekte eines Client-Prozesses bei einem Server-Prozess zu ermöglichen. Dieser Registrierungsschritt ermöglicht es dem Server-Prozess anschließend, Benachrichtigungsmeldungen wie vorstehend genannt an die Objekte zu übermitteln, die sich eingetragen haben. Typischerweise kann die Anzahl der Objekte, die sich bei ein und demselben Server-Prozess registrieren lassen wollen, in einem Client-Prozess sehr hoch sein. Es empfiehlt sich also, die Registrierungsmeldungen zu reduzieren, um die darunter liegenden Kommunikationsmittel nicht zu überlasten (insbesondere das Rechnernetz).

[0092] Es ist möglich hierfür das Verfahren gemäß der Erfindung zu verwenden: jede Registrierungsmeldung wird dann an einen rufenden Manager durch ein zugehöriges Repräsentantenelement (Proxy(A)) übermittelt. Dieser rufende Manager kann dann sämtliche Daten bezüglich der Registrierungsanforderungen in einer einzigen Registrierungsmeldung beim Server-Prozess übermitteln.

Patentansprüche

1. Kommunikationsverfahren, das auf einem Manager für verteilte Objekte ORB basiert, wobei die Übermittlung von Meldungen von Objekten eines ersten Prozesses hin zu Objekten eines anderen Prozesses durch Repräsentantenelementepaare erfolgt, mit in jedem Paar einem ersten Repräsentantenelement im rufenden Prozess und einem zweiten Repräsentantenelement im anderen Prozess, **dadurch gekennzeichnet**, dass es darin besteht, die ersten Repräsentantenelemente (Proxy(A)) des ersten Prozesses abzuändern, damit sie die Meldungen bezüglich eines bestimmten Dienstes (SX), die von Objekten (ObjC) des ersten Prozesses ($GCLIENT_1$) hin zu Objekten (ObjA) anderer Prozesse ($P_{ROSERV1}$) gesendet werden, an einen diesem Dienst entsprechenden rufenden Manager ($GCLIENT_1$) übermitteln, der im ersten Prozess vorgesehen ist, wobei dieser rufende Manager die so empfangenen Meldungen managt und einige dieser Meldungen auf einen entsprechenden empfangenden Manager ($GSERV_1$) überträgt, der jeweils in den Bestimmungsprozessen vorgesehen ist.

2. Kommunikationsverfahren gemäß Anspruch 1, dadurch gekennzeichnet, dass jedes der abgeänderten ersten Repräsentantenelemente (Proxy(A)) die Meldungen bezüglich dieses Dienstes an den entsprechenden rufenden Manager ($GCLIENT_1$) mit einem Parameter zur Identifikation ($pGSERV_1$) des entsprechenden empfangenden Managers ($GSERV_1$) im Bestimmungsprozess ($PROSERV_1$) und einem Parameter zur Identifikation (pObjA) des Bestimmungsobjekts (ObjA) übermittelt.

3. Kommunikationsverfahren gemäß Anspruch 2, dadurch gekennzeichnet, dass diese Parameter dem ersten Repräsentantenelement (Proxy(A)) vom zweiten zugehörigen Repräsentantenelement (Stub(A)) oder dem Bestimmungsobjekt (ObjA) geliefert werden.

4. Kommunikationsverfahren gemäß einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, dass der empfangende Manager eines Bestimmungsprozesses ($PROSERV_1$) die von rufenden Manager empfangenen Meldungen an die betreffenden Objekte dieses Prozesses übermittelt.

5. Kommunikationsverfahren gemäß Anspruch 4, dadurch gekennzeichnet, dass der empfangende Manager im Gegenzug zu den Meldungen Ergebnisse empfängt, die er an die betreffenden rufenden Manager weitergibt.

6. Verfahren gemäß einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, dass der empfangende Manager ($GSERV_1$) einer Meldung selbst den entsprechenden Dienst erbringt, wobei er gegebenenfalls ein entsprechendes Ergebnis an den betreffenden rufenden Manager zurücksendet.

7. Kommunikationsverfahren gemäß einem der Ansprüche 1 bis 6, dadurch gekennzeichnet, dass die Übermittlung der Meldungen und/oder der entsprechenden Ergebnisse zwischen einem rufenden Manager und einem empfangenden Manager vermittels eines Repräsentantenelementepaares erfolgt.

8. Kommunikationsverfahren gemäß einem der Ansprüche 1 bis 6, dadurch gekennzeichnet, dass die Übermittlung der Meldungen zwischen einem rufenden Manager und einem empfangenden Manager vermittels gemeinsam benutzter Mittel (R) erfolgt.

9. Kommunikationsverfahren gemäß einem der vorstehenden Ansprüche, dadurch gekennzeichnet, dass ein rufender Manager oder ein empfangender Manager Parameter der Meldungen und/oder entsprechender Ergebnisse abändert.

10. Kommunikationsverfahren gemäß einem der vorstehenden Ansprüche, dadurch gekennzeichnet, dass die Meldungen und/oder entsprechende Ergebnisse mit oder ohne Anwendung einer vorherigen Umformung paketweise zwischen den rufenden und den empfangenden Managern übermittelt werden.

11. Kommunikationsverfahren gemäß Anspruch 10, dadurch gekennzeichnet, dass die Übermittlung der Pakete zyklisch entsprechend einem vom empfangenden Manager oder vom rufenden Manager definierten Zyklus erfolgt.

12. Kommunikationsverfahren gemäß Anspruch 11, dadurch gekennzeichnet, dass die Übermittlung der Pakete außerdem durch das Eintreten eines bestimmten Ereignisses ausgelöst wird.

13. Kommunikationsverfahren gemäß einem der vorstehenden Ansprüche, dadurch gekennzeichnet, dass es auf mindestens einen Dienst angewandt wird, gemäß dem die rufenden Prozesse Client-Prozesse sind und die Bestimmungsprozesse Server-Prozesse sind.

14. Kommunikationsverfahren gemäß einem der Ansprüche 1 bis 12, dadurch gekennzeichnet, dass es auf mindestens einen Dienst angewandt wird, gemäß dem die rufenden Prozesse Server-Prozesse sind und die Bestimmungsprozesse Client-Prozesse sind.

15. Kommunikationsverfahren gemäß einem der vorstehenden Ansprüche, dadurch gekennzeichnet, dass die ORB-Umgebung eine CORBA-Umgebung ist.

16. Kommunikationsverfahren gemäß einem der Ansprache 1 bis 14, dadurch gekennzeichnet, dass die ORB-Umgebung eine DCOM-Umgebung ist.

17. Verfahren zur Eintragung eines Client-Prozesses bei einem Server-Prozess, wobei dieser Client-Prozess einen Komplex von Objekten enthält und jedes dieser Objekte eine Eintragungsmeldung sendet, dadurch gekennzeichnet, dass diese Eintragungsmeldungen gemäß einem der vorstehenden Ansprüche diesem Server-Prozess mitgeteilt werden.

Es folgen 4 Blatt Zeichnungen

Anhängende Zeichnungen

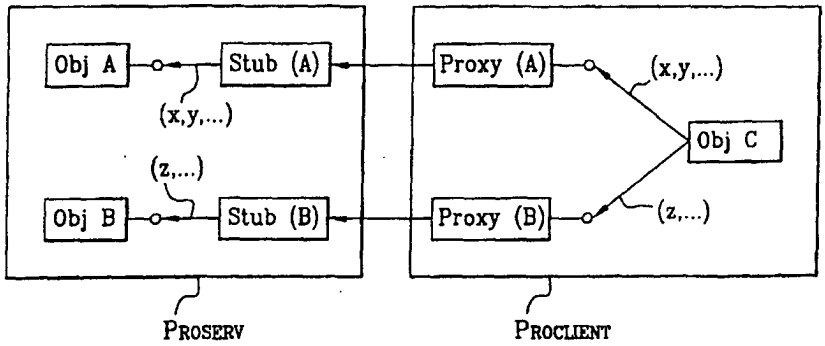

FIG.1

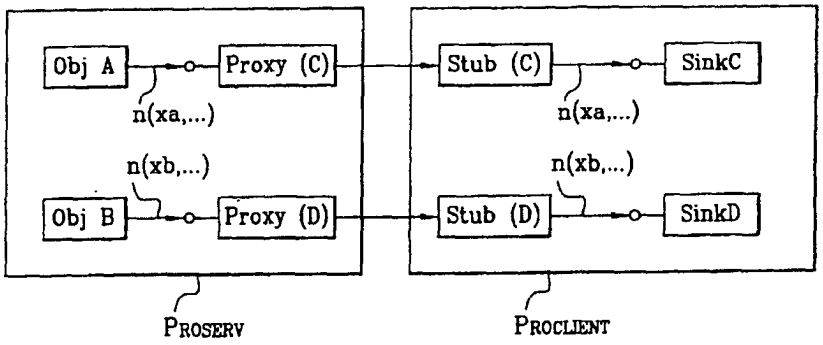

FIG.2

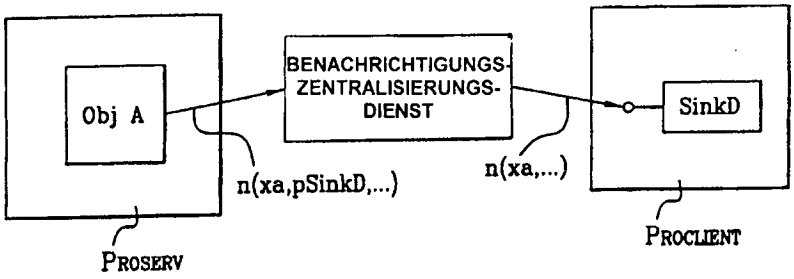
Obj A
BENACHRICHTIGUNGS-
ZENTRALISIERUNGS-
DIENST
SinkD
n(xa,pSinkD,...)
n(xa,...)
PROSERV
PROCLIENT

FIG.3

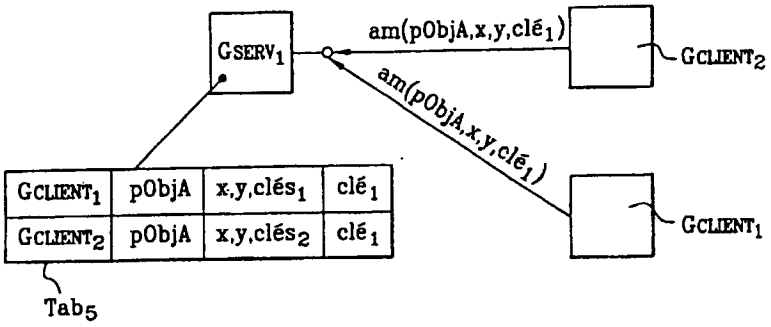
GSERV1
am(pObjA,x,y,clé1)
GCLIENT2
am(pObjA,x,y,clé1)
GCLIENT1
GCLIENT1 pObjA x,y,clés1 clé1
GCLIENT2 pObjA x,y,clés2 clé1
Tab5

FIG.6

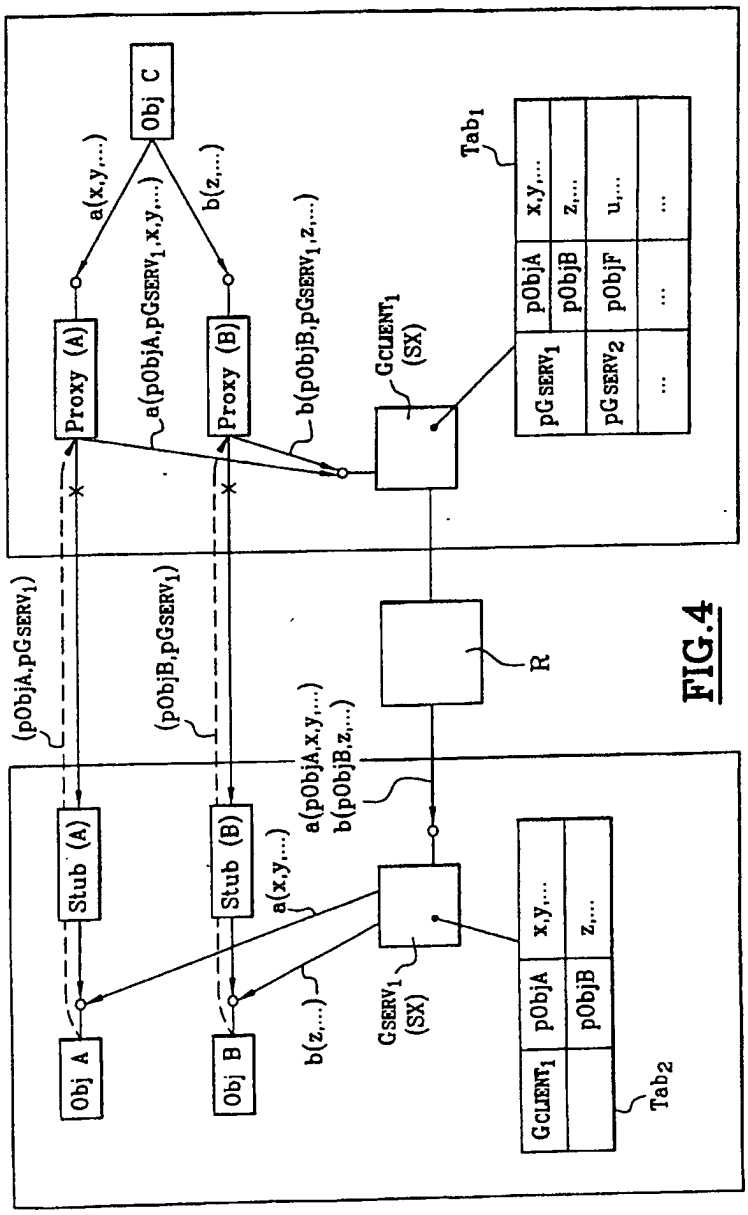
Obj C
a(x,y,...)
b(z,...)
Proxy (A)
Proxy (B)
a(pObjA,pGSERV1,x,y,...)
b(pObjB,pGSERV1,z,...)
GCLIENT1 (SX)
Tab1
pGSERV1
pGSERV2
pObjA
pObjB
pObjF
x,y,...
z,...
u,...
(pObjA,pGSERV1)
(pObjB,pGSERV1)
R
Stub (A)
Stub (B)
a(pObjA,x,y,...)
b(pObjB,z,...)
a(x,y,...)
b(z,...)
Obj A
Obj B
GSERV1 (SX)
GCLIENT1
Tab2

FIG.4

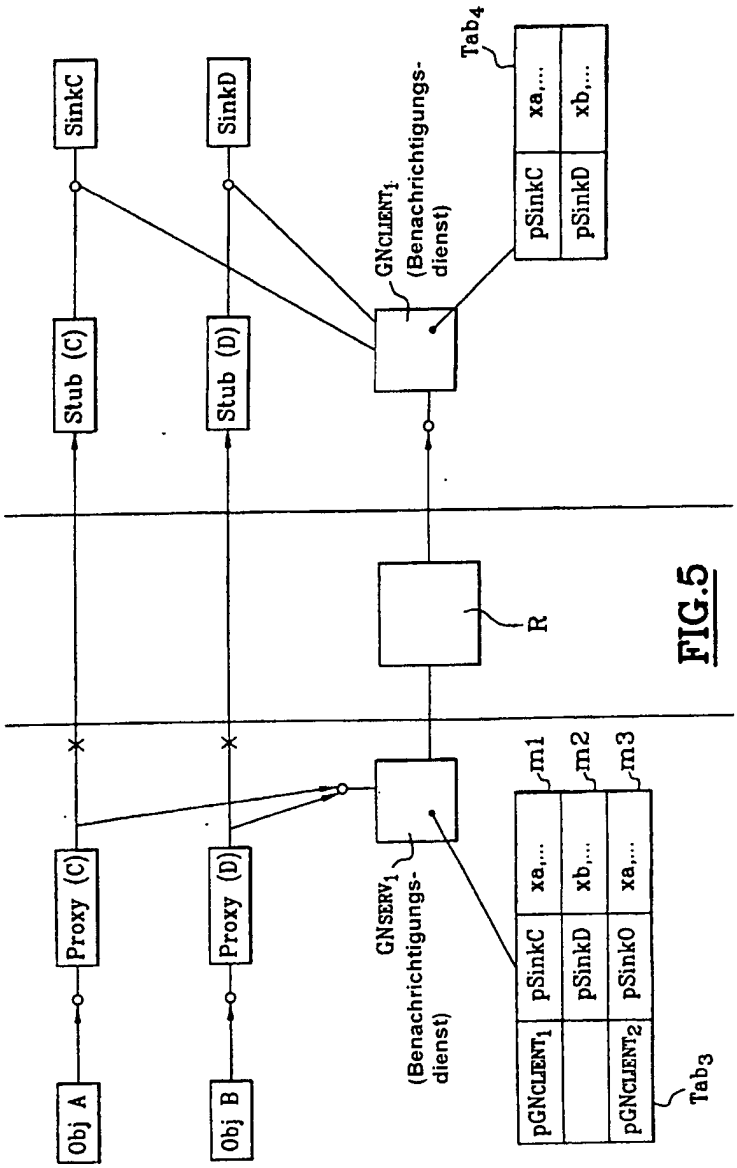
SinkC
SinkD
Tab4
xa,...
xb,...
pSinkC
pSinkD
GNCLIENT1 (Benachrichtigungs-dienst)
Stub (C)
Stub (D)
R
Proxy (C)
Proxy (D)
GNSERV1 (Benachrichtigungs-dienst)
m1
m2
m3
pSinkC
pSinkD
pSink0
pGNCLIENT1
pGNCLIENT2
Tab3
Obj A
Obj B

FIG.5