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

(10) **DE 60 2005 005 727 T2** 2009.04.16

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 562 348 B1**
(21) Deutsches Aktenzeichen: **60 2005 005 727.7**
(96) Europäisches Aktenzeichen: **05 250 636.7**
(96) Europäischer Anmeldetag: **04.02.2005**
(97) Erstveröffentlichung durch das EPA: **10.08.2005**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **02.04.2008**
(47) Veröffentlichungstag im Patentblatt: **16.04.2009**

(51) Int Cl.[8]: ***H04L 29/06*** (2006.01)

(30) Unionspriorität:
2004007827 06.02.2004 KR

(84) Benannte Vertragsstaaten:
AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LI, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR

(73) Patentinhaber:
Samsung Electronics Co., Ltd., Kyonggi, KR

(74) Vertreter:
Grünecker, Kinkeldey, Stockmair & Schwanhäusser, 80802 München

(72) Erfinder:
Kim, Pyung-soo, Seoul, KR; Kim, Sun-Woo, Gwonseon-gu Suwon-si Gyeonggi-do, KR

(54) Bezeichnung: **Verfahren und Vorrichtung zur Verbindung von Knoten mit heterogenen Kommunikationsprotokollen**

Beschreibung

[0001] Ausführungsformen der vorliegenden Erfindung beziehen sich auf ein Verfahren, ein Medium und eine Vorrichtung zum Verbinden von Knoten, die heterogene Protokolle verwenden, und im Besonderen auf ein Verfahren und eine Vorrichtung zum Verbinden von Knoten der Internet-Protokollversion 4 (IPv4) mit Knoten der Internet-Protokollversion 6 (IPv6) sowie zum Verbinden nicht-mobiler Knoten mit mobilen Knoten.

[0002] Knoten, die in einem Netzwerk vorhanden sind, das die Internet-Protokollversion 4 (IPv4) sowie die Internet-Protokollversion 6 (IPv6) enthält, sind möglicherweise nicht mit einer IPv4/IPv6-Umwandlungsfunktion in einer IP-Schicht des Netzwerkes ausgestattet. In diesem Fall können IPv4-Knoten und IPv6-Knoten nach dem Stand der Technik nicht miteinander verbunden werden. Darüber hinaus sind möglicherweise IP-Knoten, die in einem mobilen Netzwerk vorhanden sind, in IP-Schichten des Netzwerkes nicht mit einer mobilen IP-Funktion ausgestattet. Demgemäß können nicht-mobile Knoten, die keine mobile IP-Funktion besitzen, nicht mit mobilen Knoten verbunden werden, die eine mobile IP-Funktion besitzen. Da darüber hinaus eine solche IP-Schicht auf der Kernel-Ebene enthalten ist, welche Benutzer oder Anbieter von Endgeräten nicht verwalten können, ist es für Benutzer oder Anbieter von Endgeräten schwierig, die oben genannten Probleme zu lösen.

[0003] Das Patent EP 0.667.693 offenbart ein Verfahren zum Betreiben einer Maschine zur Glasherstellung, die eine Datenbank enthält, welche auf Basis eines Protokolls funktioniert, sowie einen Arbeitsplatzrechner, der auf Basis eines zweiten Protokolls funktioniert.

[0004] Das Patent US 2002/015941 offenbart eine Übersetzungsvorrichtung zum Verbinden eines IPv6-Netzwerkes mit einem IPv4-Netzwerk.

[0005] Das Patent US 2002/0199019 offenbart ein Kommunikationssystem, das die Funktionen einer Socket-Abstraktionsschicht (socket abstraction layer) zwischen Vorrichtungen verteilt.

[0006] Das Patent US 2001/0021183 offenbart die Einführung eines Pipeline-Moduls unterhalb einer Anwendungsschicht.

[0007] Ausführungsformen der vorliegenden Erfindung stellen ein Verfahren, ein Medium sowie eine Vorrichtung zum Verbinden von Knoten bereit, die heterogene Protokolle verwenden, und im Besonderen ein Verfahren, ein Medium sowie eine Vorrichtung, die ein Benutzer oder ein Anbieter von Endgeräten leicht implementieren kann.

[0008] Gemäß der vorliegenden Erfindung werden eine Vorrichtung und ein Verfahren bereitgestellt, wie in den beigefügten Ansprüchen dargelegt. Bevorzugte Eigenschaften der Erfindung sind aus den abhängigen Ansprüchen und der folgenden Beschreibung ersichtlich.

[0009] Zusätzliche Aspekte und/oder Vorteile der Erfindung werden zum Teil in der nachfolgenden Beschreibung dargelegt und sind zum Teil aus der Beschreibung ersichtlich oder können durch Anwenden der Erfindung erlernt werden.

[0010] Alle im Folgenden beschriebenen Merkmale können mit jedem der oben genannten Aspekte in jedweder Kombination kombiniert werden.

[0011] Für ein besseres Verständnis der Erfindung und für eine Darstellung der Verwirklichung der Ausführungsformen derselben wird im Folgenden beispielhaft auf die beigefügten schematischen Zeichnungen Bezug genommen, in denen:

[0012] Fig. 1 ein Konfigurationsdiagramm einer Netzwerkumgebung gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung ist;

[0013] Fig. 2 ist ein Konfigurationsdiagramm einer Netzwerkumgebung gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung;

[0014] Fig. 3 ist ein Konfigurationsdiagramm einer anderen Netzwerkumgebung gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung;

[0015] Fig. 4 ist ein Konfigurationsdiagramm noch einer anderen Netzwerkumgebung gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung;

[0016] Fig. 5 ist ein Konfigurationsdiagramm noch einer anderen Netzwerkumgebung gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung;

[0017] Fig. 6 ist ein Ablaufdiagramm, das ein Verfahren zum Verbinden von Knoten heterogener Protokolle gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung darstellt;

[0018] Fig. 7 ist ein Ablaufdiagramm, das ein anderes Verfahren zum Verbinden von Knoten heterogener Protokolle gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung darstellt;

[0019] Fig. 8 ist ein Ablaufdiagramm, das noch ein anderes Verfahren zum Verbinden von Knoten heterogener Protokolle gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung darstellt; und

[0020] Fig. 9 ist ein Ablaufdiagramm, das noch ein anderes Verfahren zum Verbinden von Knoten heterogener Protokolle gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung darstellt.

[0021] Im Folgenden wird ausführlich Bezug auf die Ausführungsformen der vorliegenden Erfindung genommen, und es werden Beispiele davon in den begleitenden Zeichnungen dargestellt, wobei gleiche Referenznummern sich durchgängig auf gleiche Elemente beziehen. Die Ausführungsformen werden im Folgenden beschrieben, um die vorliegende Erfindung durch Bezugnahme auf die Figuren zu erläutern.

[0022] Fig. 1 ist ein Konfigurationsdiagramm einer Netzwerkumgebung gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung.

[0023] In Fig. 1 kann eine Netzwerkumgebung die Knoten **1**, **2** und **3** enthalten. Gemäß dieser Ausführungsform der vorliegenden Erfindung kann der Knoten **1** mit einem Doppelstapel (dual stack) ausgerüstet sein, der beide Protokollstapel enthält, wobei ein erstes Protokoll und ein zweites Protokoll verwendet werden, wobei der Knoten **2** das erste Protokoll verwendet und der Knoten **3** das zweite Protokoll verwendet.

[0024] In Fig. 1 umfasst der Doppelstapel eine Anwendungsschicht **100**, ein erstes Socket **200**, ein zweites Socket **300**, ein erstes Protokoll **400**, ein zweites Protokoll **500** sowie eine untere Schicht (lower layer) **600** gemäß der Ausführungsform der vorliegenden Erfindung. In dem Doppelstapel umfasst der rechte Stapel das erste Protokoll **400** und der linke Stapel umfasst das zweite Protokoll **500**. Die Anwendungsschicht **100** und die untere Schicht **600** des Doppelstapels können gemeinsame Schichten (common layers) sein.

[0025] Der Knoten **1** erzeugt das erste Socket **200**, welches eine Anwendungsprogramm-Schnittstelle (application program interface – API) ist, die für eine Kommunikation auf Basis des ersten Protokolls zu verwenden ist, sowie ein zweites Socket **300**, welches eine API ist, die für die Kommunikation auf Basis des zweiten Protokolls zu verwenden ist, indem Funktionen zum Verbinden mit speziellen Unterroutinen (subroutines) in der Anwendungsschicht **100** aufgerufen werden. Hier betrifft die spezielle Unterroutine das Erzeugen von Sockets.

[0026] Der Knoten **1** kommuniziert mit dem ersten Knoten **2**, der das erste Protokoll **400** verwendet, über das erste Socket **200**, welches erzeugt wird, indem Funktionen zum Verbinden mit speziellen Unterroutinen in der Anwendungsschicht aufgerufen werden, und mit dem zweiten Knoten **3**, der das zweite Protokoll **500** über das zweite Socket **300** verwendet, das erzeugt wird, indem Funktionen zum Verbinden mit speziellen Unterroutinen in der Anwendungsschicht aufgerufen werden. Hier beziehen sich spezielle Unterroutinen auf die Kommunikation über Sockets. Mit anderen Worten, die von dem Knoten **2** gesendeten Daten, der das erste Protokoll **400** verwendet, werden über ein Netzwerk an den Knoten **1** übertragen. Die von dem Knoten **1** empfangenen Daten passieren die untere Schicht **600**, das erste Protokoll **400** und das erste Socket **200** und kommen anschließend bei der Anwendungsschicht **100** an. Die bei der Anwendungsschicht **100** ankommenden Daten passieren das zweite Socket **300**, das zweite Protokoll **500** und die untere Schicht **600**. Der Knoten **3** empfängt die Daten, die die untere Schicht **600** passiert haben, über ein Netzwerk. Auf die gleiche Weise kann ein umgekehrter Datenstrom aufgebaut werden.

[0027] Wie oben beschrieben, kommuniziert der Knoten **1** über das erste Socket **200** mit dem ersten Knoten **2**, der das erste Protokoll **400** verwendet, und über das zweite Socket **300** mit dem zweiten Knoten **3**, das zweite Protokoll **500** verwendet. Demgemäß implementiert der Knoten **1** die Kommunikation mit dem Knoten **2** und dem Knoten **3** selbst dann, wenn Schichten der Kernel-Ebene, die Benutzer oder Anbieter von Endgeräten nicht verwalten können, das heißt, das erste Protokoll **400** und das zweite Protokoll **500**, keinen Umwandlungsmechanismus zwischen dem ersten Protokoll **400** und dem zweiten Protokoll **500** enthalten. Hier kann das erste Protokoll **400** IPv6 sein und das zweite Protokoll **500** kann IPv4 sein oder umgekehrt. Darüber hinaus kann das erste Protokoll **400** eine mobile IP sein und das zweite Protokoll **500** kann eine nicht-mobile IP sein oder umgekehrt. Jede der oben genannten Versionen wird nun im Folgenden spezieller beschrieben.

[0028] Fig. 2 ist ein Konfigurationsdiagramm einer ersten Netzwerkumgebung gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung.

[0029] In Bezug auf Fig. 2 kann die Netzwerkumgebung gemäß dieser Ausführungsform der vorliegenden Erfindung einen Knoten **11** einschließlich einer Vorrichtung zum Verbinden von Knoten heterogener Protokolle, einen IPv6-Knoten **21**, der einem Client entspricht, sowie einen IPv4-Knoten **31** enthalten, der einem Server entspricht. Der Knoten **11** spielt eine Rolle als ein Server entsprechend dem IPv6-Knoten **21** und gleichzeitig spielt er eine Rolle als ein Client entsprechend dem IPv4-Knoten **31**.

[0030] In Fig. 2 umfasst die Vorrichtung zum Verbinden von Knoten heterogener Protokolle gemäß dieser Ausführungsform der vorliegenden Erfindung eine Doppel-Socket-Erzeugungseinheit **101**, eine Doppel-Socket-Verbindungseinheit **102**, eine erste Socket-Kommunikationseinheit **103** sowie eine zwei-

te Socket-Kommunikationseinheit **104**. Wie in Fig. 2 dargestellt, ist die Vorrichtung zum Verbinden von Knoten heterogener Protokolle auf einer Anwendungsschicht **100** angeordnet.

[0031] Die Doppel-Socket-Erzeugungseinheit **101** erzeugt ein erstes Socket **200**, das für die IPv6-Kommunikation von jedwedem Anwendungsprogramm zu verwenden ist, und das zweite Socket **300** ist für die IPv4-Kommunikation des Anwendungsprogramms zu verwenden. Ausführlicher erzeugt die Doppel-Socket-Erzeugungseinheit **101** als ein Server und Client das erste Socket **200**, indem eine Funktion Socket () aufgerufen wird, das heißt, eine Socket-Funktion in der Anwendungsschicht **100**, wobei die Funktion Socket () Informationen über IPv6 enthält. Die Funktion Socket () wird als eine Art von int socket (int family, int type, int protocol) definiert. So kann beispielsweise PF_INET in ein family-Feld eines Sockets () zum Erzeugen des ersten Sockets **200** geschrieben werden, um zu kennzeichnen, dass die Internet-Protokollfamilie verwendet wird, SOCK_STREAM kann in ein type-Feld geschrieben werden, um zu kennzeichnen, dass das TCP (transmission control protocol – Übertragungssteuerungsprotokoll) verbindungsorientierter Kommunikation verwendet wird, und IPv6 kann in ein protocol-Feld geschrieben werden, um die Verwendung von IPv6 anzuzeigen. Wenn das UDP (user datagram protocol – Benutzerdatagrammprotokoll) einer nicht verbindungsorientierten Kommunikation verwendet wird, kann anstelle von SOCK_STREAM SOCK_DGRAM in ein type-Feld geschrieben werden. Darüber hinaus erzeugt die Doppel-Socket-Erzeugungseinheit **101** das zweite Socket **300**, indem eine Funktion Socket () in der Anwendungsschicht **100** aufgerufen wird, wobei die Funktion Socket () Informationen über IPv4 enthält, mit der Ausnahme, dass IPv4 in ein protocol-Feld eines Socket () zum Kennzeichnen der Verwendung von IPv4 geschrieben wird, es wird notiert, dass eine Funktion Socket () zum Erzeugen des zweiten Socket **300** identisch mit der Funktion Socket () zum Erzeugen des ersten Socket **200** ist.

[0032] Darüber hinaus verbindet die Doppel-Socket-Erzeugungseinheit **101** als ein Server eine Adresse des Knotens **11** mit dem ersten Socket **200**, indem eine bind ()-Verbindungsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Verbindungsfunktion Informationen über das erste Socket **200** und eine Adresse des Knotens **11** enthält, der als Bestimmungsort des IPv6-Knotens **21** eingestellt ist. Die Verbindungsfunktion bind () wird als ein Typ von int bind (int sockfd, struct sockaddr *myaddr, int addrlen) definiert. Ein Socket-Deskriptor des ersten Socket **200** wird in ein sockfd-Feld einer Funktion bind () zum Verbinden einer Adresse mit dem ersten Socket **200** geschrieben, eine Adressstruktur, die eine IPv6-Adresse sowie eine Anschlussnummer (gort number) enthält, die von TCP/UDP **401** und IPv6 **402** bereitgestellt werden, wird in ein myaddr-Feld geschrieben und eine Gröle der Adressstruktur wird in ein addrlen-Feld geschrieben. Die Funktion bind () wird zum Verbinden der IPv6-Adresse und einer Anschlussnummer des Knotens **11**, der dem IPv6-Knoten **21** mit einem Socket-Deskriptor des ersten Socket **200** bekannt ist, verwendet, da der Socket-Deskriptor des ersten Socket **200** nur einem Anwendungsprogramm des Knotens **11** bekannt ist und von diesem verwendet wird.

[0033] Die Doppel-Socket-Verbindungseinheit **102** verbindet das erste Socket **200** und das zweite Socket **300**, die in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurden, mit dem IPv6-Knoten **21** beziehungsweise dem IPv4-Knoten **31**. Spezieller wartet die Doppel-Socket-Verbindungseinheit **112** als ein Server auf den Empfang einer Verbindungsanforderung, deren Bestimmungsort eine Adresse ist, die mit dem ersten Socket **200** verbunden ist, indem eine Funktion listen () einer Funktion wait in einer Anwendungsschicht **100** aufgerufen wird, wobei die Funktion listen () Informationen über das erste Socket **200** enthält. Die Funktion listen () wird als ein Typ von int listen (int sockfd, int backlog) definiert. Ein Socket-Deskriptor des ersten Socket **200** wird in ein sockfd-Feld einer Funktion listen () zum Empfangen einer Verbindungsanforderung, deren Bestimmungsort eine Adresse ist, die mit dem ersten Socket **200** verbunden ist, geschrieben und die Höchstanzahl von Verbindungsanforderungen, auf die gewartet werden kann, wird in ein backlog-Feld geschrieben.

[0034] Ferner lässt die Doppel-Socket-Verbindungseinheit **102** als ein Server die empfangene Verbindungsanforderung zu, indem eine Funktion accept () einer Akzeptanzfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die accept-Funktion Informationen über das erste Socket **200** und eine Adresse des IPv6-Knoten **21** enthält, der die Verbindungsanforderung gesendet hat. Hier wird ein neues Socket für eine Direktkommunikation (one to one communication) mit einem Prozess erzeugt, enthalten in einem Anwendungsprogramm, der in dem IPv6-Knoten **21** ausgeführt wird, der einem Client entspricht. Die Funktion accept () wird als ein Typ von int accept (int sockfd, struct sockaddr *clientaddr, int addrlen) definiert. Ein Socket-Deskriptor des ersten Socket **200** wird in ein sockfd-Feld einer Funktion accept () zum Zulassen einer Verbindungsanforderung von dem IPv6-Knoten **21**, der einem Client entspricht, geschrieben, eine Adressstruktur wird in ein clientaddr-Feld geschrieben, die Adressstruktur enthält eine IPv6-Adresse sowie eine Anschlussnummer des IPv6-Knotens **21**, und eine Größe der Adressstruktur wird in ein addrlen-Feld geschrieben.

[0035] Darüber hinaus fordert die Doppel-Socket-Verbindungseinheit **102** als ein Client eine Verbindung zu dem IPv4-Knoten **31** an, indem eine

Funktion connect () einer Verbindungsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Funktion connect () Informationen über das zweite Socket **300** und eine Adresse des IPv4-Knotens **31** enthält, der auf den Empfang einer Verbindungsanforderung wartet. Die Funktion connect () wird als ein Typ von int connect (int sockfd, struct sockaddr *serveraddr, int addrlen) definiert. Ein Socket-Deskriptor des zweiten Socket wird in ein sockfd-Feld der Funktion connect () zum Anfordern einer Verbindung zu dem IPv4-Knoten **31**, der als ein Server fungiert, geschrieben, eine Adressstruktur wird in ein serveraddr-Feld geschrieben, die Adressstruktur enthält eine IPv4-Adresse sowie eine Anschlussnummer des IPv4-Knotens **31**, und eine Größe der Adressstruktur wird in ein addrlen-Feld geschrieben.

[0036] Die erste Socket-Kommunikationseinheit **103** empfängt Daten, die von dem IPv6-Knoten **21** über das erste Socket **200**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **11** UDP verwendet, oder sie empfängt Daten, die von dem IPv6-Knoten **21** über das erste Socket **200** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit verbunden ist, wenn der Knoten **11** TCP verwendet. Dies liegt daran, dass eine Funktion listen () und eine Funktion accept () bei einer verbindungsorientierten Kommunikation wie beispielsweise TCP aufgerufen werden sollten, Daten können jedoch bei einer nicht verbindungsorientierten Kommunikation wie beispielsweise UDP auch direkt empfangen und gesendet werden, ohne dass die Funktionen listen () und accept () aufgerufen werden müssen.

[0037] Im Besonderen empfängt die erste Socket-Kommunikationseinheit **103** als ein Server Daten, die von dem IPv6-Knoten **21** über das erste Socket **200** übertragen wurden, indem eine Funktion recv () oder recvfrom () einer Empfangsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Empfangsfunktion Informationen über das erste Socket **200** und die von dem IPv6-Knoten **21** übertragenen Daten enthält. Die zweite Socket-Kommunikationseinheit **104** sendet als ein Client Daten, die von dem IPv6-Knoten **21** über das zweite Socket **300** übertragen wurden, indem eine Funktion send () oder sendto () einer Sendefunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Sendefunktion Informationen über das zweite Socket **300** und die von dem IPv6-Knoten **21** übertragenen Daten enthält. Die Funktionen recv () und send () werden in einer verbindungsorientierten Kommunikation wie beispielsweise TCP aufgerufen, oder die Funktionen recvfrom () und sendto () werden in einer nicht verbindungsorientierten Kommunikation wie beispielsweise UDP aufgerufen.

[0038] Die Funktion recv () wird als ein Typ von int recv (int sockfd, char buf, int buflen, int flags) definiert. Ein Socket-Deskriptor des ersten Socket **200** wird zum Empfangen der von dem IPv6-Knoten **21** über das erste Socket **200** übertragenen Daten in ein sockfd-Feld der Funktion recv () geschrieben, wobei der Zeiger (pointer) eines Zwischenspeichers (buffer) die empfangenen Daten in einem buf-Feld speichert, eine Größe des Zwischenspeichers wird in ein buflen-Feld geschrieben und ein Wert, der Bandüberschreitung (out of band) und dergleichen anzeigt, wird in ein flags-Feld geschrieben. Andererseits wird die Funktion recvfrom () als ein Typ von int recvfrom (int sockfd, char buf, int buflen, int flags, struct sockaddr *fromaddr, int addrlen) definiert. Das heißt, der mit der Funktion recv () identische Wert wird in ein sockfd-Feld, ein buf-Feld, ein buflen-Feld sowie ein flags-Feld der Funktion recvfrom () zum Empfangen von Daten über das erste Socket **200** geschrieben, eine Quellen-Adressstruktur wird in ein fromaddr-Feld geschrieben, wobei die Quellen-Adressstruktur eine IPv6-Adresse sowie eine Anschlussnummer des IPv6-Knoten **21** enthält, und eine Größe der Quellen-Adressstruktur wird in ein addrlen-Feld geschrieben.

[0039] Die Funktion send () wird als ein Typ von int send (int sockfd, char buf, int buflen, int flags) definiert. Ein Socket-Deskriptor des zweiten Socket **300** wird in ein sockfd-Feld der Funktion send () zum Senden der von dem IPv6-Knoten **21** übertragenen Daten über das zweite Socket **300** geschrieben; der Zeiger des Zwischenspeichers, der zu sendende Daten speichert, wird in ein buf-Feld geschrieben, wobei der Zeiger mit dem Wert der Funktion recv () identisch ist; und der Wert, der eine Bandüberschreitung und dergleichen anzeigt, wird in ein Hags-Feld geschrieben, wobei der Wert mit dem Wert der Funktion recv () identisch ist. Dieser Prozess ist auf den IPv6-zu-IPv4-Übergangsprozess in der Anwendungsschicht **100** anwendbar. Andererseits wird die Funktion sendto () als ein Typ von int sendto (int sockfd, char buf, int buflen, int flags, struct sockaddr *toaddr, int addrlen) definiert. Das heißt, die mit der Funktion send () identischen Werte werden in ein sockfd-Feld, ein buf-Feld, ein buflen-Feld und ein flags-Feld der Funktion sendto () zum Senden von Daten, die von dem IPv6-Knoten **21** über das zweite Socket **300** übertragen wurden, geschrieben, eine Quellen-Adressstruktur wird in ein toaddr-Feld geschrieben, die Quellen-Adressstruktur enthält eine IPv4-Adresse sowie eine Anschlussnummer des IPv4-Knotens **31**, und eine Größe der Adressstruktur wird in ein addrlen-Feld geschrieben.

[0040] Des Weiteren empfängt die zweite Socket-Kommunikationseinheit **104** Daten, die von dem IPv4-Knoten **31** über das zweite Socket **300**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **11** UDP verwendet, oder sie empfängt Daten, die von dem IPv4-Knoten **31** über das zweite Socket **300** übertra-

gen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **11** TCP verwendet. Die erste Socket-Kommunikationseinheit **103** sendet Daten, die von dem IPv4-Knoten **31** über das erste Socket **200**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **11** UDP verwendet, oder sie sendet Daten, die von dem IPv4-Knoten **31** über das erste Socket **200** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist.

[0041] Spezieller empfängt die zweite Socket-Kommunikationseinheit **104** als ein Server Daten, die von dem IPv4-Knoten **31** über das zweite Socket **300** übertragen wurden, indem eine Funktion recv () oder recvfrom () einer Empfangsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Empfangsfunktion Informationen über das zweite Socket **300** und die von dem IPv4-Knoten **31** übertragenen Daten enthält. Die erste Socket-Kommunikationseinheit **103** sendet als ein Client Daten, die von dem IPv4-Knoten **31** über das erste Socket **200** übertragen wurden, indem eine Funktion send () oder sendto () einer Sendefunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Sendefunktion Informationen über das erste Socket und die von dem IPv4-Knoten **31** übertragenen Daten enthält.

[0042] Ein Socket-Deskriptor des zweiten Socket **300** wird zum Empfangen der von dem IPv4-Knoten **31** über das zweite Socket **300** übertragenen Daten in ein sockfd-Feld der Funktion recv () geschrieben; ein Zwischenspeicher-Zeiger, der die empfangenen Daten speichert, wird in ein buf-Feld geschrieben; eine Größe des Zwischenspeichers wird in ein buflen-Feld geschrieben; und ein Wert, der Bandüberschreitung und dergleichen anzeigt, wird in ein flags-Feld geschrieben. Andererseits werden die Werte, die mit der Funktion recv () identisch sind, in ein sockfd-Feld, ein buf-Feld, ein buflen-Feld und ein flags-Feld der Funktion recvfrom () zum Empfangen von Daten geschrieben, die von dem IPv4-Knoten **31** über das zweite Socket **300** übertragen wurden; eine Quellen-Adressstruktur wird in ein fromaddr-Feld geschrieben, wobei die Quellen-Adressstruktur eine IPv4-Adresse sowie eine Anschlussnummer des IPv4-Knotens **31** enthält; und eine Größe der Adressstruktur wird in ein addrlen-Feld geschrieben.

[0043] Ein Socket-Deskriptor des ersten Socket **200** wird in ein Feld sockfd der Funktion send () zum Senden von Daten, die von dem IPv4-Knoten **31** über das erste Socket **200** übertragen wurden, geschrieben; ein Zeiger eines Zwischenspeichers, der zu sendende Daten speichert, wird in ein buf-Feld geschrieben, wobei der Zeiger mit dem Wert der Funktion recv () identisch ist; und ein Wert, der eine Bandüberschreitung und dergleichen anzeigt, wird in ein flags-Feld geschrieben, wobei der Wert mit dem Wert der Funktion recv () identisch ist. Dieser Prozess ist auf den IPv4-zu-IPv6-Übergangsprozess in der Anwendungsschicht **100** anwendbar. Andererseits wird der Wert, der mit der Funktion send () identisch ist, in ein sockfd-Feld, ein buf-Feld, ein buflen-Feld und ein flags-Feld einer Funktion sendto () geschrieben, zum Senden von Daten, die von dem IPv4-Knoten **31** über das erste Socket **200** übertragen wurden; eine Quellen-Adressstruktur wird in ein toaddr-Feld geschrieben, wobei die Quellen-Adressstruktur eine IPv6-Adresse sowie eine Anschlussnummer des IPv6-Knotens **21** enthält; und eine Größe der Adressstruktur wird in ein addrlen-Feld geschrieben.

[0044] Fig. 3 ist ein Konfigurationsdiagramm einer anderen Netzwerkumgebung gemäß einer anderen beispielhaften Ausführungsform der vorliegenden Erfindung.

[0045] In Fig. 3 umfasst die Netzwerkumgebung einen Knoten **12**, der mit einer Vorrichtung zum Verbinden von Knoten heterogener Protokolle, einem IPv4-Knoten **22**, der als ein Client fungiert, sowie mit dem IPv6-Knoten **32** ausgestattet ist, der als ein Server fungiert. Der Knoten **12** spielt in Bezug auf den IPv4-Knoten **22** eine Rolle eines Servers und spielt gleichzeitig in Bezug auf den IPv6-Knoten **32** auch eine Rolle eines Clients. Ausführungsformen der vorliegenden Erfindung werden nun spezieller beschrieben, wobei das Augenmerk auf die Unterschiede zwischen der Netzwerkumgebung aus Fig. 2 und der Netzwerkumgebung aus Fig. 3 gelegt wird, die Gemeinsamkeiten zwischen den beiden Netzwerkumgebungen werden nicht erneut beschrieben.

[0046] Wie in Fig. 3 dargestellt, erzeugt die Doppel-Socket-Erzeugungseinheit **101** das erste Socket **200**, das für die IPv4-Kommunikation von jedwedem Anwendungsprogramm zu verwenden ist, und das zweite Socket **300** ist für die IPv6-Kommunikation des Anwendungsprogramms zu verwenden. Die Doppel-Socket-Erzeugungseinheit **101** erzeugt als ein Server und Client das erste Socket **200**, indem eine Funktion Socket () einer Socket-Funktion in einer Anwendungsschicht **100** aufgerufen wird, wobei die Funktion Socket () Informationen über IPv4 enthält. IPv4 wird in ein protocol-Feld der Funktion Socket () zum Erzeugen des ersten Socket **200** geschrieben, um somit die Verwendung von IPv4 anzuzeigen. Die Doppel-Socket-Erzeugungseinheit **101** erzeugt das zweite Socket **300**, indem eine Funktion Socket () in der Anwendungsschicht aufgerufen wird, wobei die Funktion Socket () Informationen über IPv6 enthält. IPv6 wird in ein protocol-Feld der Funktion Socket () zum Erzeugen des zweiten Socket **300** geschrieben, um somit die Verwendung von IPv6 anzuzeigen. Darüber hinaus verbindet die Doppel-Socket-Erzeugungseinheit **101** als ein Server eine Adresse des Knotens **12** mit dem ersten Socket **200**, indem eine Funktion bind () einer Verbindungsfunkti-

on in der Anwendungsschicht **100** aufgerufen wird, wobei die Verbindungsfunktion Informationen über das erste Socket **200** und eine Adresse des Knotens **12** enthält, der als Bestimmungsort des IPv4-Knotens **22** eingestellt ist. Eine Adressstruktur wird zum Verbinden der Adresse mit dem ersten Socket **200** in ein myaddr-Feld der Funktion bind () geschrieben, wobei die Adressstruktur eine IPv4-Adresse sowie eine Anschlussnummer enthält, die von TCP/UDP **403** und IPv4 **404** bereitgestellt wurden.

[0047] Die Doppel-Socket-Verbindungseinheit **102** lässt als ein Server die empfangene Verbindungsanforderung zu, indem eine Funktion accept () einer Akzeptanzfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Funktion accept Informationen über das erste Socket **200** und die Adresse des IPv4-Knotens **22** enthält, der die Verbindungsanforderung gesendet hat. Eine Adressstruktur wird in ein clientaddr-Feld der Funktion accept () zum Zulassen der Verbindungsanforderung von dem IPv4-Knoten **22**, der als ein Client fungiert, geschrieben, die Adressstruktur enthält eine IPv4-Adresse sowie eine Anschlussnummer des IPv4-Knotens **22**; und eine Größe der Adressstruktur wird in ein addrlen-Feld geschrieben.

[0048] Darüber hinaus fordert die Doppel-Socket-Verbindungseinheit **102** als ein Client eine Verbindung zu dem IPv6-Knoten **32** an, indem eine Funktion connect () einer Verbindungsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Funktion connect () Informationen über das zweite Socket **300** und eine Adresse des IPv6-Knotens **32** enthält, der auf den Empfang einer Verbindungsanforderung wartet. Eine Adressstruktur wird in ein serveraddr-Feld einer Funktion connect () zum Anfordern einer Verbindung mit dem IPv6-Knoten **32**, der einem Server entspricht, geschrieben, wobei die Adressstruktur eine IPv6-Adresse sowie eine Anschlussnummer des IPv6-Knotens **32** enthält.

[0049] Die erste Socket-Kommunikationseinheit **103** empfängt Daten, die von dem IPv4-Knoten **22** über das erste Socket **200**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **12** UDP verwendet, oder sie empfängt Daten, die von dem IPv4-Knoten **22** über das erste Socket **200** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **12** TCP verwendet. Die zweite Socket-Kommunikationseinheit **104** empfängt Daten, die von dem IPv4-Knoten **22** über das zweite Socket **300**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **12** UDP verwendet, oder sie empfängt Daten, die von dem IPv4-Knoten **22** über das zweite Socket **300** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **12** TCP verwendet.

[0050] Die erste Socket-Kommunikationseinheit **103** empfängt als ein Server Daten, die von dem IPv4-Knoten **22** über das erste Socket **200** übertragen wurden, indem eine Funktion recv () oder recvfrom () einer Empfangsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Empfangsfunktion Informationen über das erste Socket **200** und die von dem IPv4-Knoten **22** übertragenen Daten enthält. Die zweite Socket-Kommunikationseinheit **104** sendet als ein Client Daten, die von dem IPv4-Knoten **22** über das zweite Socket **300** übertragen wurden, indem eine Funktion send () oder sendto () einer Sendefunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Sendefunktion Informationen über das zweite Socket und die von dem IPv4-Knoten **22** übertragenen Daten enthält.

[0051] Eine Quellen-Adressstruktur wird in ein fromaddr-Feld der Funktion recvfrom zum Empfangen von Daten geschrieben, wobei die Quellen-Adressstruktur eine IPv4-Adresse sowie eine Anschlussnummer des IPv4-Knotens **22** enthält, und eine Größe der Quellen-Adressstruktur wird in ein addrlen-Feld geschrieben.

[0052] Ein Socket-Deskriptor des zweiten Socket **300** wird in ein sockfd-Feld der Funktion send () zum Senden der von dem IPv4-Knoten **22** übertragenen Daten über das zweite Socket **300** geschrieben; ein Zeiger eines Zwischenspeichers, der zu sendende Daten speichert, wird in ein buf-Feld geschrieben, wobei der Zeiger mit dem Wert der Funktion recv () identisch ist; eine Größe des Zwischenspeichers wird in ein buflen-Feld geschrieben, wobei die Größe mit dem Wert der Funktion recv () identisch ist; und ein Wert, der eine Bandüberschreitung und dergleichen anzeigt, wird in ein flags-Feld geschrieben, wobei der Wert mit dem Wert der Funktion recv () identisch ist. Dieser Prozess ist auf den IPv4-zu-IPv6-Übergangsprozess in der Anwendungsschicht **100** anwendbar. Andererseits wird eine Quellen-Adressstruktur in ein toaddr-Feld eines toaddr-Feldes zum Senden von Daten, die von dem IPv4-Knoten **22** über das zweite Socket **300** übertragen wurden, geschrieben, die Quellen-Adressstruktur enthält eine IPv6-Adresse sowie eine Anschlussnummer des IPv6-Knotens **32**.

[0053] Des Weiteren empfängt die zweite Socket-Kommunikationseinheit **104** Daten, die von dem IPv6-Knoten **32** über das zweite Socket **300**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **12** UDP verwendet, oder sie empfängt Daten, die von dem IPv4-Knoten **22** über das zweite Socket **300** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **12** TCP verwendet. Die erste Socket-Kommunikationseinheit **103** sendet Daten, die von dem IPv6-Knoten **32** über das erste Socket **200**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertra-

gen wurden, wenn der Knoten **12** UDP verwendet, oder sie sendet Daten, die von dem IPv6-Knoten **32** über das erste Socket **200** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **12** TCP verwendet.

[0054] Spezieller empfängt die zweite Socket-Kommunikationseinheit **104** als ein Server Daten, die von dem IPv6-Knoten **32** über das zweite Socket **300** übertragen wurden, indem eine Funktion recv () oder recvfrom () einer Empfangsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Empfangsfunktion Informationen über das zweite Socket **300** und die von dem IPv6-Knoten **32** übertragenen Daten enthält. Die erste Socket-Kommunikationseinheit **103** sendet als ein Client Daten, die von dem IPv6-Knoten **32** über das erste Socket **200** übertragen wurden, indem eine Funktion send () oder sendto () einer Sendefunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Sendefunktion Informationen über das erste Socket und die von dem IPv6-Knoten **32** übertragenen Daten enthält.

[0055] Eine Quellen-Adressstruktur wird in ein fromaddr-Feld der Funktion recvfrom () zum Empfangen von Daten, die von dem IPv6-Knoten **32** über das zweite Socket **300** übertragen wurden, geschrieben, wobei die Quellen-Adressstruktur eine IPv6-Adresse sowie eine Anschlussnummer des IPv6-Knotens **32** enthält.

[0056] Ein Socket-Deskriptor des ersten Socket **200** wird in ein Feld sockfd der Funktion send () zum Senden von Daten, die von dem IPv6-Knoten **32** über das erste Socket **200** übertragen wurden, geschrieben; ein Zeiger eines Zwischenspeichers, der die zu sendenden Daten speichert, wird in ein buf-Feld geschrieben, wobei der Zeiger mit dem Wert der Funktion recv () identisch ist; eine Größe des Zwischenspeichers wird in ein buflen-Feld geschrieben, wobei die Größe mit dem Wert der Funktion recv () identisch ist; und ein Wert, der eine Bandüberschreitung und dergleichen anzeigt, wird in ein Hags-Feld geschrieben, wobei der Wert mit dem Wert der Funktion recv () identisch ist.

[0057] Dieser Prozess ist auf den IPv6-zu-IPv4-Übergangsprozess in der Anwendungsschicht **100** anwendbar. Andererseits wird die Quellen-Adressstruktur in ein toaddr-Feld der Funktion sendto () zum Senden von Daten, die von dem IPv6-Knoten **32** über das erste Socket **200** übertragen wurden, geschrieben, wobei die Quellen-Adressstruktur eine IPv4-Adresse sowie eine Anschlussnummer des IPv4-Knotens **22** enthält.

[0058] Fig. 4 ist ein Konfigurationsdiagramm einer anderen Netzwerkumgebung gemäß noch einer anderen beispielhaften Ausführungsform der vorliegenden Erfindung.

[0059] In Fig. 4 umfasst die Netzwerkumgebung einen Knoten **13**, der mit einer Vorrichtung zum Verbinden von Knoten heterogener Protokolle, einem mobilen Knoten **23**, der als ein Client fungiert, sowie mit einem nicht-mobilen Knoten **33** ausgestattet ist, der als ein Server fungiert. Der Knoten **13** spielt bei dem mobilen Knoten **23** eine Rolle eines Servers und gleichzeitig spielt er bei dem nicht-mobilen Knoten **33** die Rolle eines Client. Ausführungsformen der vorliegenden Erfindung werden nun spezieller beschrieben, wobei das Augenmerk auf die Unterschiede zwischen der Netzwerkumgebung aus Fig. 2 und der Netzwerkumgebung aus Fig. 4 gelegt wird, die Gemeinsamkeiten zwischen den beiden Netzwerkumgebungen werden nicht erneut beschrieben.

[0060] Die Doppel-Socket-Erzeugungseinheit **101** erzeugt das zweite Socket **300**, das für die mobile Kommunikation von jedwedem Anwendungsprogramm zu verwenden ist, sowie das zweite Socket **300**, das für die nicht-mobile Kommunikation des Anwendungsprogramms zu verwenden ist. Die Doppel-Socket-Erzeugungseinheit **101** erzeugt als ein Server und ein Client das erste Socket **200**, indem eine Funktion Socket () einer Socket-Funktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Funktion Socket () Informationen über eine mobile IP enthält. Die mobile IP wird in ein protocol-Feld der Funktion Socket () zum Erzeugen des ersten Socket **200** geschrieben, um somit die Verwendung einer mobilen IP anzuzeigen. Die Doppel-Socket-Erzeugungseinheit **101** erzeugt das zweite Socket **300**, indem eine Funktion Socket () in der Anwendungsschicht aufgerufen wird, wobei die Funktion Socket () Informationen über die nicht-mobile IP enthält. Die nicht-mobile IP wird in ein protocol-Feld der Funktion Socket () zum Erzeugen des zweiten Socket **300** geschrieben, um somit die Verwendung einer Verwendung einer nicht-mobilen IP anzuzeigen. Darüber hinaus verbindet die Doppel-Socket-Erzeugungseinheit **101** als ein Server eine Adresse des Knotens **13** mit dem ersten Socket **200**, indem eine Funktion bind () einer Verbindungsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Verbindungsfunktion Informationen über das erste Socket **200** und eine Adresse des Knotens **13** enthält, der als Bestimmungsort des mobilen Knotens **23** eingestellt ist. Eine Adressstruktur wird in ein myaddr-Feld der Funktion bind () zum Verbinden der Adresse mit dem ersten Socket **200** geschrieben, wobei die Adressstruktur eine mobile IP-Adresse sowie eine Anschlussnummer enthält, die von TCP/UDP **405** und der mobilen IP **406** bereitgestellt wurden.

[0061] Ferner lässt die Doppel-Socket-Verbindungseinheit **122** als ein Server die empfangene Verbindungsanforderung zu, indem eine Funktion accept () einer Akzeptanzfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die accept-Funktion Informationen über das erste Socket **200** und eine Adres-

se des mobilen Knotens **23** enthält, der die Verbindungsanforderung gesendet hat. Eine Adressstruktur wird in ein clientaddr-Feld der Funktion accept () zum Zulassen der Verbindungsanforderung von dem mobilen Knoten **23**, der einem Client entspricht, geschrieben, wobei die Adressstruktur eine mobile IP-Adresse sowie eine Anschlussnummer des mobilen Knotens **23** enthält, und eine Größe der Adressstruktur wird in ein addrlen-Feld geschrieben.

[0062] Darüber hinaus fordert die Doppel-Socket-Verbindungseinheit **102** als ein Client eine Verbindung zu dem nicht-mobilen Knoten **33** an, indem eine Funktion connect () einer Verbindungsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Funktion connect () Informationen über das zweite Socket **300** und eine Adresse des nicht-mobilen Knotens **33** enthält, der auf den Empfang einer Verbindungsanforderung wartet. Eine Adressstruktur wird in ein serveraddr-Feld einer Funktion connect () zum Anfordern einer Verbindung mit dem nicht-mobilen Knoten **33**, der einem Server entspricht, geschrieben, wobei die Adressstruktur eine nicht-mobile IP-Adresse sowie eine Anschlussnummer des nicht-mobilen Knotens **33** enthält.

[0063] Die erste Socket-Kommunikationseinheit **103** empfängt Daten, die von dem mobilen Knoten **23** über das erste Socket **200**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **13** UDP verwendet, oder sie empfangt Daten, die von dem mobilen Knoten **23** über das erste Socket **200** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **13** TCP verwendet. Die zweite Socket-Kommunikationseinheit **104** empfängt Daten, die von dem mobilen Knoten **23** über das zweite Socket **300**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **13** UDP verwendet, oder sie empfangt Daten, die von dem mobilen Knoten **23** über das zweite Socket **300** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **13** TCP verwendet.

[0064] Die erste Socket-Kommunikationseinheit **103** empfängt als ein Server Daten, die von dem mobilen Knoten **23** über das erste Socket **200** übertragen wurden, indem eine Funktion recv () oder recvfrom () einer Empfangsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Empfangsfunktion Informationen über das erste Socket **200** und die von dem mobilen Knoten **23** übertragenen Daten enthält. Die zweite Socket-Kommunikationseinheit **104** sendet als ein Client Daten, die von dem mobilen Knoten **23** über das zweite Socket **300** übertragen wurden, indem eine Funktion send () oder sendto () einer Sendefunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Sendefunktion Informationen über das zweite Socket **300** und Informationen über die von dem mobilen Knoten **23** übertragenen Daten enthält.

[0065] Eine Quellen-Adressstruktur wird in ein fromaddr-Feld der Funktion recvfrom () zum Empfangen von Daten über das erste Socket **200** geschrieben, die Quellen-Adressstruktur enthält eine mobile IP-Adresse und eine Anschlussnummer des mobilen Knotens **23**; und eine Größe der Quellen-Adressstruktur wird in ein addrlen-Feld geschrieben.

[0066] Ein Socket-Deskriptor des zweiten Socket **300** wird in ein sockfd-Feld der Funktion send () zum Senden von Daten, die von dem mobilen Knoten **23** über das zweite Socket **300** übertragen wurden, geschrieben; ein Zeiger eines Zwischenspeichers, der zu sendende Daten speichert, wird in ein buf-Feld geschrieben, wobei der Zeiger mit dem Wert der Funktion recv () identisch ist; und eine Größe des Wertes, der mit einem Zwischenspeicher identisch ist, wird in ein buflen-Feld geschrieben; und der Wert, der eine Bandüberschreitung und dergleichen anzeigt, wird in ein flags-Feld geschrieben, wobei der Wert mit dem Wert der Funktion recv () identisch ist. Dieser Prozess ist auf den Übergangsprozess von der mobilen IP zu der nicht-mobilen IP in der Anwendungsschicht **100** anwendbar. Andererseits wird die Quellen-Adressstruktur in ein toaddr-Feld der Funktion sendto () zum Senden von Daten, die von dem mobilen Knoten **23** über das erste Socket **300** übertragen wurden, geschrieben, wobei die Quellen-Adressstruktur eine nicht-mobile Adresse und eine Anschlussnummer des nicht-mobilen Knotens **33** enthält.

[0067] Des Weiteren empfängt die zweite Socket-Kommunikationseinheit **104** Daten, die von dem nicht-mobilen Knoten **33** über das zweite Socket **300**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **13** UDP verwendet, oder sie empfangt Daten, die von dem nicht-mobilen Knoten **33** über das zweite Socket **300** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **13** TCP verwendet. Die erste Socket-Kommunikationseinheit **103** sendet Daten, die von dem nicht-mobilen Knoten **33** über das erste Socket **200**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **13** UDP verwendet, oder sie sendet Daten, die von dem nicht-mobilen Knoten **33** über das erste Socket **200** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **13** TCP verwendet.

[0068] Die zweite Socket-Kommunikationseinheit **104** empfängt als ein Server Daten, die von dem nicht-mobilen Knoten **33** über das zweite Socket **300** übertragen wurden, indem eine Funktion recv () oder recvfrom () einer Empfangsfunktion in der Anwen-

dungsschicht **100** aufgerufen wird, wobei die Empfangsfunktion Informationen über das zweite Socket **300** und Informationen über die von dem nicht-mobilen Knoten **33** übertragenen Daten enthält. Die erste Socket-Kommunikationseinheit **103** sendet als ein Client Daten, die von dem nicht-mobilen Knoten **33** über das erste Socket **200** übertragen wurden, indem eine Funktion send () oder sendto () einer Sendefunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Sendefunktion Informationen über das erste Socket und Informationen über die von dem nicht-mobilen Knoten **33** übertragenen Daten enthält.

[0069] Eine Quellen-Adressstruktur wird in ein fromaddr-Feld der Funktion recvfrom () zum Senden von Daten, die von dem nicht-mobilen Knoten **33** über das zweite Socket **300** übertragen wurden, geschrieben, wobei die Quellen-Adressstruktur eine nicht-mobile IP-Adresse und eine Anschlussnummer des nicht-mobilen Knotens **33** enthält.

[0070] Ein Socket-Deskriptor des ersten Socket **200** wird in ein sockfd-Feld der Funktion send () zum Senden der von dem nicht-mobilen Knoten **33** übertragenen Daten über das zweite Socket **300** geschrieben; ein Zeiger eines Zwischenspeichers, der zu sendende Daten speichert, wird in ein buf-Feld geschrieben, wobei der Zeiger mit dem Wert der Funktion recv () identisch ist; eine Größe des Zwischenspeichers wird in ein buflen-Feld geschrieben, wobei die Größe mit dem Wert der Funktion recv () identisch ist; der Wert, der eine Bandüberschreitung und dergleichen anzeigt, wird in ein Hags-Feld geschrieben, wobei der Wert mit dem Wert der Funktion recv () identisch ist. Dieser Prozess ist auf den Übergangsprozess von der nicht-mobilen IP zu der mobilen IP in der Anwendungsschicht **100** anwendbar. Andererseits wird eine Quellen-Adressstruktur in ein toaddr-Feld der Funktion sendto () zum Senden von Daten, die von dem nicht-mobilen Knoten **33** über das erste Socket **200** übertragen wurden, geschrieben, wobei die Quellen-Adressstruktur eine mobile IP-Adresse und eine Anschlussnummer des mobilen IP-Knotens **23** enthält.

[0071] Fig. 5 ist ein Konfigurationsdiagramm einer anderen Netzwerkumgebung gemäß noch einer anderen beispielhaften Ausführungsform der vorliegenden Erfindung.

[0072] In Fig. 5 umfasst die Netzwerkumgebung einen Knoten **14**, der mit einer Vorrichtung zum Verbinden von Knoten heterogener Protokolle, einem nicht-mobilen Knoten **24**, der als ein Client fungiert, sowie mit dem mobilen Knoten **34** ausgestattet ist, der als ein Server fungiert. Der Knoten **14** spielt in Bezug auf den nicht-mobilen Knoten **34** eine Rolle eines Servers und spielt gleichzeitig in Bezug auf den mobilen Knoten **34** auch eine Rolle eines Clients. Ausführungsformen der vorliegenden Erfindung werden nun spezieller beschrieben, wobei das Augenmerk auf die Unterschiede zwischen der Netzwerkumgebung aus Fig. 2 und der Netzwerkumgebung aus Fig. 5 gelegt wird, die Gemeinsamkeiten zwischen den beiden Netzwerkumgebungen werden nicht erneut beschrieben.

[0073] Die Doppel-Socket-Erzeugungseinheit **101** erzeugt das erste Socket **200**, das für die nicht-mobile Kommunikation von jedwedem Anwendungsprogramm zu verwenden ist, sowie das zweite Socket **300**, das für die mobile Kommunikation des Anwendungsprogramms zu verwenden ist. Die Doppel-Socket-Erzeugungseinheit **101** erzeugt als ein Server und ein Client das erste Socket **200**, indem eine Funktion Socket () einer Socket-Funktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Funktion Socket () Informationen über die nicht-mobile IP enthält. Die nicht-mobile IP wird in ein protocol-Feld der Funktion Socket () zum Erzeugen des ersten Socket **200** geschrieben, um somit die Verwendung einer mobilen IP anzuzeigen. Die Doppel-Socket-Erzeugungseinheit **101** erzeugt das zweite Socket **300**, indem eine Funktion Socket () in der Anwendungsschicht aufgerufen wird, wobei das Socket () Informationen über die mobile IP enthält. Die mobile IP wird in ein protocol-Feld der Funktion Socket () zum Erzeugen des zweiten Socket **300** geschrieben, um somit die Verwendung einer mobilen IP anzuzeigen. Darüber hinaus verbindet die Doppel-Socket-Erzeugungseinheit **101** als ein Server eine Adresse des Knotens **14** mit dem ersten Socket **200**, indem eine Funktion bind () einer Verbindungsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Verbindungsfunktion Informationen über das erste Socket **200** und eine Adresse des Knotens **14** enthält, der als Bestimmungsort des nicht-mobilen Knotens **24** eingestellt ist. Eine Adressstruktur wird in ein myaddr-Feld der Funktion bind () zum Verbinden der Adresse mit dem ersten Socket **200** geschrieben, wobei die Adressstruktur eine nicht-mobile IP-Adresse sowie eine Anschlussnummer enthält, die von TCP/UDP **407** und der nicht-mobilen IP **408** bereitgestellt wurden.

[0074] Die Doppel-Socket-Verbindungseinheit **102** lässt als ein Server die empfangene Verbindungsanforderung zu, indem eine Funktion accept () einer Akzeptanzfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die accept-Funktion Informationen über das erste Socket **200** und eine Adresse des nicht-mobilen Knotens **24** enthält, der die Verbindungsanforderung gesendet hat. Eine Adressstruktur wird in ein clientaddr-Feld einer Funktion accept () zum Zulassen der Verbindungsanforderung von dem nicht-mobilen Knoten **24**, der als ein Client agiert, geschrieben, wobei die Adressstruktur eine nicht-mobile IP-Adresse sowie eine Anschlussnummer des nicht-mobilen Knotens **24** enthält, und eine Größe der Adressstruktur wird in ein addrlen-Feld geschrie-

ben.

[0075] Darüber hinaus fordert die Doppel-Socket-Verbindungseinheit **102** als ein Client eine Verbindung zu dem mobilen Knoten **34** an, indem eine Funktion connect () einer Verbindungsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Funktion connect () Informationen über das zweite Socket **300** und eine Adresse des mobilen Knotens **34** enthält, der auf den Empfang einer Verbindungsanforderung wartet. Eine Adressstruktur wird in ein serveraddr-Feld einer Funktion connect () zum Anfordern einer Verbindung mit dem mobilen Knoten **34**, der als ein Server agiert, geschrieben, wobei die Adressstruktur eine mobile IP-Adresse sowie eine Anschlussnummer des mobilen Knotens **34** enthält.

[0076] Die erste Socket-Kommunikationseinheit **103** empfängt Daten, die von dem nicht-mobilen Knoten **24** über das erste Socket **200**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **14** UDP verwendet, oder sie empfängt Daten, die von dem nicht-mobilen Knoten **24** über das erste Socket **200** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **14** TCP verwendet. Die zweite Socket-Kommunikationseinheit **104** sendet Daten, die von dem nicht-mobilen Knoten **24** über das zweite Socket **300**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **14** UDP verwendet, oder sie sendet Daten, die von dem nicht-mobilen Knoten **24** über das zweite Socket **300** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **14** TCP verwendet.

[0077] Die erste Socket-Kommunikationseinheit **103** empfängt als ein Server Daten, die von dem nicht-mobilen Knoten **24** über das erste Socket **200** übertragen wurden, indem eine Funktion recv () oder recvfrom () einer Empfangsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Empfangsfunktion Informationen über das erste Socket **200** und Informationen über die von dem nicht-mobilen Knoten **24** übertragenen Daten enthält. Die zweite Socket-Kommunikationseinheit **104** sendet als ein Client Daten, die von dem nicht-mobilen Knoten **24** über das zweite Socket **300** übertragen wurden, indem eine Funktion send () oder sendto () einer Sendefunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Sendefunktion Informationen über das zweite Socket **300** und Informationen über die von dem nicht-mobilen Knoten **24** übertragenen Daten enthält.

[0078] Eine Quellen-Adressstruktur wird in ein fromaddr-Feld der Funktion recvfrom () zum Empfangen von Daten über das erste Socket **200** geschrieben, die Quellen-Adressstruktur enthält eine nicht-mobile IP-Adresse und eine Anschlussnummer des nicht-mobilen Knotens **24**; und eine Größe der Quellen-Adressstruktur wird in ein addrlen-Feld geschrieben.

[0079] Ein Socket-Deskriptor des zweiten Socket **300** wird in ein sockfd-Feld einer Funktion send () zum Senden von Daten, die von dem nicht-mobilen Knoten **24** über das zweite Socket **300** übertragen wurden, geschrieben; ein Zeiger eines Zwischenspeichers, der zu sendende Daten speichert, wird in ein buf-Feld geschrieben, wobei der Zeiger mit dem Wert einer Funktion recv () identisch ist; und eine Größe des Zwischenspeichers wird in ein buflen-Feld geschrieben, wobei die Größe mit dem Wert einer Funktion recv () identisch ist; und ein Wert, der eine Bandüberschreitung und dergleichen anzeigt, wird in ein flags-Feld geschrieben, wobei der Wert mit dem Wert einer Funktion recv () identisch ist. Dieser Prozess ist auf den Übergangsprozess von der nicht-mobilen IP zu der mobilen IP in der Anwendungsschicht **100** anwendbar. Andererseits wird eine Quellen-Adressstruktur in ein toaddr-Feld der Funktion sendto () zum Senden von Daten, die von dem nicht-mobilen Knoten **24** über das zweite Socket **300** übertragen wurden, geschrieben, wobei die Quellen-Adressstruktur eine mobile IP-Adresse und eine Anschlussnummer des mobilen IP-Knotens **34** enthält.

[0080] Des Weiteren empfängt die zweite Socket-Kommunikationseinheit **104** Daten, die von dem mobilen Knoten **34** über das zweite Socket **300**, das in einer Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **14** UDP verwendet, oder sie empfängt Daten, die von dem mobilen Knoten **34** über das zweite Socket **300** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **14** TCP verwendet. Die erste Socket-Kommunikationseinheit **103** sendet Daten, die von dem mobilen Knoten **34** über das erste Socket **200**, das in der Doppel-Socket-Erzeugungseinheit **101** erzeugt wurde, übertragen wurden, wenn der Knoten **14** UDP verwendet, oder sie sendet Daten, die von dem mobilen Knoten **34** über das erste Socket **200** übertragen wurden, das mit der Doppel-Socket-Verbindungseinheit **102** verbunden ist, wenn der Knoten **14** TCP verwendet.

[0081] Die zweite Socket-Kommunikationseinheit **104** empfängt als ein Server Daten, die von dem mobilen Knoten **34** über das zweite Socket **300** übertragen wurden, indem eine Funktion recv () oder recvfrom () einer Empfangsfunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Empfangsfunktion Informationen über das zweite Socket **300** und Informationen über die von dem mobilen Knoten **34** übertragenen Daten enthält. Die erste Socket-Kommunikationseinheit **103** sendet als ein Cli-

ent Daten, die von dem mobilen Knoten **34** über das erste Socket **200** übertragen wurden, indem eine Funktion send () oder sendto () einer Sendefunktion in der Anwendungsschicht **100** aufgerufen wird, wobei die Sendefunktion Informationen über das erste Socket **200** und Informationen über die von dem mobilen Knoten **34** übertragenen Daten enthält.

[0082] Eine Quellen-Adressstruktur wird in ein fromaddr-Feld der Funktion recvfrom () zum Empfangen von Daten, die von dem mobilen Knoten **34** über das zweite Socket **300** übertragen wurden, geschrieben, wobei die Quellen-Adressstruktur eine mobile IP-Adresse und eine Anschlussnummer des mobilen Knotens **34** enthält.

[0083] Ein Socket-Deskriptor des ersten Socket **200** wird in ein Feld sockfd einer Funktion send () zum Senden von Daten, die von dem mobilen Knoten **34** über das erste Socket **200** übertragen wurden, geschrieben; ein Zeiger eines Zwischenspeichers, der zu sendende Daten speichert, wird in ein buf-Feld geschrieben, wobei der Zeiger mit dem Wert der Funktion recv () identisch ist; eine Größe des Zwischenspeichers wird in ein buflen-Feld geschrieben, wobei die Größe mit dem Wert der Funktion recv () identisch ist; und ein Wert, der eine Bandüberschreitung und dergleichen anzeigt, wird in ein flags-Feld geschrieben, wobei der Wert mit dem Wert der Funktion recv () identisch ist. Dieser Prozess ist auf den Übergangsprozess von der mobilen IP zu der nicht-mobilen IP in der Anwendungsschicht **100** anwendbar. Andererseits wird eine Quellen-Adressstruktur in ein toaddr-Feld einer Funktion sendto () zum Senden von Daten, die von dem mobilen Knoten **34** über das erste Socket **200** übertragen wurden, geschrieben, wobei die Quellen-Adressstruktur eine nicht-mobile IP-Adresse und eine Anschlussnummer des nicht-mobilen IP-Knotens **24** enthält.

[0084] Fig. 6 ist ein Ablaufdiagramm eines Verfahrens zum Verbinden von Knoten heterogener Protokolle gemäß einer beispielhaften Ausführungsform der vorliegenden Erfindung.

[0085] In Fig. 6 umfasst das Verfahren zum Verbinden von Knoten heterogener Protokolle Operationen wie nachstehend beschrieben. Das Verfahren zum Verbinden von Knoten heterogener Protokolle kann beispielsweise in einer Netzwerkumgebung implementiert werden, wie sie in Fig. 2 dargestellt wird.

[0086] Erstens erzeugt der Knoten **11** als ein Server und ein Client das erste Socket, indem eine Funktion Socket (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion Socket (IPv6) Informationen über IPv6 enthält, und ein zweites Socket wird erzeugt, indem eine Funktion Socket (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion Socket (IPv4) Informationen über IPv4 enthält (Operationen **601** und **602**). Gleichzeitig erzeugt der IPv6-Knoten **21** das dritte Socket, indem die Funktion Socket (IPv6) aufgerufen wird, das Informationen über IPv6 enthält, und der IPv4-Knoten **31** erzeugt ein viertes Socket, indem eine Funktion Socket (IPv4) aufgerufen wird, das Informationen über IPv4 enthält (Operationen **603** und **604**).

[0087] Anschließend verbindet der Knoten **11** als ein Server eine Adresse des Knotens **11** mit dem ersten Socket, indem eine Funktion bind (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion bind (IPv6) Informationen über das erste Socket und eine Adresse des Knotens **11** enthält, der durch den IPv6-Knoten **21** als ein Bestimmungsort eingestellt ist (Operation **605**). Anschließend wartet der Knoten **11** als ein Server den Empfang einer Verbindungsanforderung ab, deren Bestimmungsort eine Adresse ist, die mit dem ersten Socket verbunden ist, indem eine Funktion listen (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion listen (IPv6) Informationen über das erste Socket enthält (Operation **606**). Gleichzeitig verbindet der IPv4-Knoten **31** als ein Server eine Adresse des IPv4-Knotens mit dem vierten Socket, indem eine Funktion bind (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion bind (IPv4) Informationen über das vierte Socket enthält (Operation **607**). Anschließend wartet der IPv4-Knoten **31** als ein Server den Empfang einer Verbindungsanforderung ab, deren Bestimmungsort eine Adresse ist, die mit dem vierten Socket verbunden ist, indem eine Funktion listen (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion listen (IPv4) Informationen über das vierte Socket enthält (Operation **608**).

[0088] Anschließend fordert der IPv6-Knoten **21** als ein Client eine Verbindung mit dem Knoten **11** an, indem eine Funktion connect (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion connect (IPv6) Informationen über das dritte Socket und eine Adresse des Knotens **11** enthält, der den Empfang einer Verbindungsanforderung erwartet (Operation **609**). Anschließend lasst der Knoten **11** als ein Server die Verbindungsanforderung zu, indem eine Funktion accept (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion accept (IPv6) Informationen über das erste Socket und eine Adresse des IPv6-Knotens **21** enthält, der die Verbindungsanforderung gesendet hat (Operation **610**). Gleichzeitig fordert der Knoten **11** als ein Client eine Verbindung mit dem IPv4-Knoten **31** an, indem eine Funktion connect (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion connect (IPv4) Informationen über das zweite Socket und eine Adresse des IPv4-Knotens **31** enthält, der den Empfang einer Verbindungsanforderung erwartet (Operation **611**). Anschließend lässt der IPv4-Knoten **31** die empfangene Verbindungsanforderung zu, indem eine Funktion accept (IPv4) in der Anwendungs-

schicht aufgerufen wird, wobei die Funktion accept (IPv4) Informationen über das vierte Socket und eine Adresse des Knotens **11** enthält, der die Verbindungsanforderung gesendet hat (Operation **612**). Bei einer nicht-mobilen Kommunikation wie beispielsweise UDP können die Operationen des Aufrufens einer Funktion listen (), connect () und accept () ausgelassen werden.

[0089] Anschließend sendet der IPv6-Knoten **21** als ein Client Daten (von dem IPv6-Knoten **21**) über das dritte Socket, indem eine Funktion send (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (IPv6) Informationen über das dritte Socket und Informationen über von dem IPv6-Knoten **21** übertragene Daten enthält (Operation **613**). Anschließend empfängt der Knoten **11** als ein Server von dem IPv6-Knoten **21** übertragene Daten über das erste Socket, indem eine Funktion recv (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (IPv6) Informationen über das erste Socket und von dem IPv6-Knoten **21** übertragene Daten enthält (Operation **614**). Danach sendet der Knoten **11** von dem IPv6-Knoten **21** übertragene Daten über das zweite Socket, indem eine Funktion send (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Daten Informationen über das zweite Socket und von dem IPv6-Knoten **21** übertragene Daten **21** enthalten (Operation **615**). Dieser Prozess ist auf den IPv6-zu-IPv4-Übergangsprozess in der Anwendungsschicht anwendbar. Anschließend empfängt der IPv4-Knoten **31** als ein Server von dem IPv6-Knoten **21** übertragene Daten über das vierte Socket, indem eine Funktion recv (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (IPv4) Informationen über das vierte Socket und die von dem IPv6-Knoten **21** übertragenen Daten enthält (Operation **616**). Der IPv4-Knoten **31** verarbeitet die von dem IPv6-Knoten **21** übertragenen Daten.

[0090] Anschließend sendet der IPv4-Knoten **31** als ein Server von dem IPv4-Knoten **31** übertragene Daten über das vierte Socket, indem eine Funktion send (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (IPv4) Informationen über das vierte Socket und die von dem IPv4-Knoten **31** übertragenen Daten enthält (Operation **617**). Danach empfangt der Knoten **11** als ein Client von dem IPv4-Knoten **31** übertragene Daten über das zweite Socket, indem eine Funktion recv (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (IPv4) Informationen über das zweite Socket und von die dem IPv4-Knoten **31** übertragenen Daten enthält (Operation **618**). Anschließend sendet der Knoten **11** als ein Server von dem IPv4-Knoten **31** über das erste Socket übertragene Daten, indem eine Funktion send (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (IPv6) Informationen über das erste Socket und die von dem IPv4-Knoten **31** übertragenen Daten enthält (Operation **619**). Dieser Prozess ist auf den IPv4-zu-IPv6-Übergangsprozess in der Anwendungsschicht anwendbar. Danach empfängt der IPv6-Knoten **21** als ein Client von dem IPv4-Knoten **31** über das dritte Socket übertragene Daten, indem eine Funktion recv (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (IPv6) Informationen über das vierte Socket und die von dem IPv4-Knoten **31** übertragenen Daten enthält (Operation **620**). Der IPv6-Knoten **21** verarbeitet die von dem IPv4-Knoten **31** übertragenen Daten. In dem Fall einer nicht-mobilen Kommunikation wie beispielsweise UDP werden anstatt der Funktion send () und recv () die Funktionen sendto () beispielsweise recvfrom () aufgerufen.

[0091] Fig. 7 ist ein Ablaufdiagramm eines anderen Verfahrens zum Verbinden von Knoten heterogener Protokolle gemäß einer anderen beispielhaften Ausführungsform der vorliegenden Erfindung.

[0092] In Fig. 7 umfasst das Verfahren zum Verbinden von Knoten heterogener Protokolle Operationen wie nachstehend beschrieben. Das Verfahren zum Verbinden von Knoten heterogener Protokolle kann beispielsweise in der Netzwerkumgebung implementiert werden, wie sie in Fig. 3 dargestellt wird.

[0093] Erstens erzeugt der Knoten **12** als ein Server und ein Client das erste Socket, indem eine Funktion Socket (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion Socket (IPv4) Informationen über IPv4 enthält, und das zweite Socket wird erzeugt, indem eine Funktion Socket (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion Socket (IPv6) Informationen über IPv6 enthält (Operationen **701** und **702**). Gleichzeitig erzeugt der IPv4-Knoten **22** das dritte Socket, indem eine Funktion Socket (IPv4) aufgerufen wird, die Informationen über IPv4 enthält, und der IPv6-Knoten **32** erzeugt das vierte Socket, indem eine Funktion Socket (IPv6) aufgerufen wird, die Informationen über IPv6 enthält (Operationen **703** und **704**).

[0094] Anschließend verbindet der Knoten **12** als ein Server eine Adresse des Knotens **12** mit dem ersten Socket, indem eine Funktion bind (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion bind (IPv4) Informationen über das erste Socket und eine Adresse des Knotens **12** enthält, der als Bestimmungsort des IPv4-Knotens **22** eingestellt ist (Operation **705**). Danach wartet der Knoten **12** als ein Server den Empfang einer Verbindungsanforderung ab, deren Bestimmungsort eine Adresse ist, die mit dem ersten Socket verbunden ist, indem eine Funktion listen (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion listen (IPv4) Informationen über das erste Socket enthält (Operation **706**). Gleichzeitig verbindet der IPv6-Knoten **32** als

ein Server eine Adresse des IPv6-Knotens **32** mit dem vierten Socket, indem eine Funktion bind (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion bind (IPv6) Informationen über das vierte Socket und eine Adresse des Knotens **12** enthält, der durch den IPv6-Knoten **32** als ein Bestimmungsort eingestellt ist (Operation **707**). Danach wartet der IPv6-Knoten **32** als ein Server den Empfang einer Verbindungsanforderung ab, deren Bestimmungsort eine Adresse ist, die mit dem vierten Socket verbunden ist, indem eine Funktion listen (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion listen (IPv6) Informationen über das vierte Socket enthält (Operation **708**).

[0095] Danach fordert der IPv4-Knoten **22** als ein Client eine Verbindung mit dem Knoten **12** an, indem eine Funktion connect (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion connect (IPv4) Informationen über das dritte Socket und eine Adresse des Knotens **12** enthält, der den Empfang einer Verbindungsanforderung erwartet (Operation **709**). Anschließend lässt der Knoten **12** als ein Server die Verbindungsanforderung zu, indem eine Funktion accept (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion accept (IPv4) Informationen über das erste Socket und eine Adresse des IPv4-Knotens **22** enthält, der die Verbindungsanforderung gesendet hat (Operation **710**). Gleichzeitig fordert der Knoten **12** als ein Client eine Verbindung mit dem IPv6-Knoten **32** an, indem eine Funktion connect (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion connect (IPv6) Informationen über das zweite Socket und eine Adresse des IPv6-Knotens **32** enthält, der den Empfang einer Verbindungsanforderung erwartet (Operation **711**). Anschließend lässt der IPv6-Knoten **32** die empfangene Verbindungsanforderung zu, indem eine Funktion accept (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion accept (IPv6) Informationen über das vierte Socket und eine Adresse des Knotens **12** enthält, der die Verbindungsanforderung gesendet hat (Operation **712**). Bei einer nicht-mobilen Kommunikation wie beispielsweise UDP werden die Operationen des Aufrufens einer Funktion listen (), connect () und accept () ausgelassen.

[0096] Anschließend sendet der IPv4-Knoten **22** als ein Client von dem IPv4-Knoten **22** übertragene Daten über das dritte Socket, indem eine Funktion send (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (IPv4) Informationen über das dritte Socket und die von dem IPv4-Knoten **22** übertragenen Daten enthält (Operation **713**). Anschließend empfängt der Knoten **12** als ein Server von dem IPv4-Knoten **22** übertragene Daten über das erste Socket, indem eine Funktion recv (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (IPv4) Informationen über das erste Socket und von dem IPv4-Knoten **22** übertragene Daten enthält (Operation **714**). Danach sendet der Knoten **12** von dem IPv4-Knoten **22** übertragene Daten über das zweite Socket, indem eine Funktion send (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Daten Informationen über das zweite Socket und von dem IPv4-Knoten **22** übertragene Daten enthalten (Operation **715**). Dieser Prozess ist auf den IPv4-zu-IPv6-Übergangsprozess in der Anwendungsschicht anwendbar. Danach empfängt der IPv6-Knoten **32** als ein Server von dem IPv4-Knoten **22** übertragene Daten über das vierte Socket, indem eine Funktion recv (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (IPv6) Informationen über das vierte Socket und die von dem IPv4-Knoten **22** übertragenen Daten enthält (Operation **716**). Der IPv6-Knoten **32** verarbeitet die von dem IPv4-Knoten **22** übertragenen Daten.

[0097] Anschließend sendet der IPv6-Knoten **32** als ein Server von dem IPv6-Knoten **32** über das vierte Socket übertragene Daten, indem eine Funktion send (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (IPv6) Informationen über das vierte Socket und die von dem IPv6-Knoten **32** übertragenen Daten enthält (Operation **717**). Danach empfängt der Knoten **12** als ein Client von dem IPv6-Knoten **32** übertragene Daten über das zweite Socket, indem eine Funktion recv (IPv6) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (IPv6) Informationen über das zweite Socket und von dem IPv6-Knoten **32** übertragene Daten enthält (Operation **718**). Anschließend sendet der Knoten **12** als ein Server von dem IPv6-Knoten **32** übertragene Daten über das erste Socket, indem eine Funktion send (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (IPv4) Informationen über das erste Socket und die von dem IPv6-Knoten **32** übertragenen Daten enthält (Operation **719**). Dieser Prozess ist auf den IPv6-zu-IPv4-Übergangsprozess in der Anwendungsschicht anwendbar. Anschließend empfängt der IPv4-Knoten **22** als ein Client von dem IPv6-Knoten **32** übertragene Daten über das dritte Socket, indem eine Funktion recv (IPv4) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (IPv4) Informationen über das vierte Socket und die von dem IPv6-Knoten **32** übertragenen Daten enthält (Operation **720**). Der IPv4-Knoten **22** verarbeitet die von dem IPv6-Knoten **32** übertragenen Daten. In dem Fall einer nicht-mobilen Kommunikation wie beispielsweise UDP werden anstatt der Funktion send () und recv () die Funktionen sendto () beispielsweise recvfrom () aufgerufen.

[0098] Fig. 8 ist ein Ablaufdiagramm eines anderen Verfahrens zum Verbinden von Knoten heterogener Protokolle gemäß noch einer anderen beispielhaften Ausführungsform der vorliegenden Erfindung.

[0099] In Fig. 8 umfasst das Verfahren zum Verbin-

den von Knoten heterogener Protokolle Operationen wie nachstehend beschrieben. Das Verfahren zum Verbinden von Knoten heterogener Protokolle kann beispielsweise in der Netzwerkumgebung implementiert werden, wie sie in Fig. 4 dargestellt wird.

[0100] Zuerst erzeugt der Knoten **13** als ein Server und ein Client das erste Socket, indem eine Funktion Socket (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion Socket (mobile IP) Informationen über die mobile IP enthält, und das zweite Socket wird erzeugt, indem eine Funktion Socket (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion Socket (nicht-mobile IP) Informationen über die nicht-mobile IP enthält (Operationen **801** und **802**). Gleichzeitig erzeugt der mobile Knoten **23** das dritte Socket, indem eine Funktion Socket (mobile IP) aufgerufen wird, die Informationen über die mobile IP enthält, und der nicht-mobile Knoten **33** erzeugt das vierte Socket, indem eine Funktion Socket (nicht-mobile IP) aufgerufen wird, die Informationen über die nicht-mobile IP enthält (Operationen **803** und **804**).

[0101] Anschließend verbindet der Knoten **13** als ein Server eine Adresse des Knotens **13** mit dem ersten Socket, indem eine Funktion bind (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion bind (mobile IP) Informationen über das erste Socket und eine Adresse des Knotens **13** enthält, der durch den mobilen Knoten **23** als Bestimmungsort eingestellt ist (Operation **805**). Danach wartet der Knoten **13** als ein Server den Empfang einer Verbindungsanforderung ab, deren Bestimmungsort eine Adresse ist, die mit dem ersten Socket verbunden ist, indem eine Funktion listen (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion listen (mobile IP) Informationen über das erste Socket enthält (Operation **806**). Gleichzeitig verbindet der nicht-mobile Knoten **33** als ein Server eine Adresse des nicht-mobilen Knotens **33** mit dem vierten Socket, indem eine Funktion bind (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion bind (nicht-mobile IP) Informationen über das vierte Socket und eine Adresse des nicht-mobilen Knotens **33** enthält, der durch den Knoten **13** als ein Bestimmungsort eingestellt ist (Operation **807**). Danach wartet der nicht-mobile Knoten **33** als ein Server den Empfang einer Verbindungsanforderung ab, deren Bestimmungsort eine Adresse ist, die mit dem vierten Socket verbunden ist, indem eine Funktion listen (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion listen (nicht-mobile IP) Informationen über das vierte Socket enthält (Operation **808**).

[0102] Danach fordert der mobile Knoten **23** als ein Client eine Verbindung mit dem Knoten **13** an, indem eine Funktion connect (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion connect (mobile IP) Informationen über das dritte Socket und eine Adresse des Knotens **13** enthält, der den Empfang einer Verbindungsanforderung erwartet (Operation **809**). Anschließend lässt der Knoten **13** als ein Server die Verbindungsanforderung zu, indem eine Funktion accept (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion accept (mobile IP) Informationen über das erste Socket und eine Adresse des mobilen Knotens **23** enthält, der die Verbindungsanforderung gesendet hat (Operation **810**). Gleichzeitig fordert der Knoten **13** als ein Client eine Verbindung zu dem nicht-mobilen Knoten **33** an, indem eine Funktion connect (nicht-mobile IP) einer Verbindungsfunktion in der Anwendungsschicht aufgerufen wird, wobei die Funktion connect (nicht-mobile IP) Informationen über das zweite Socket und eine Adresse des nicht-mobilen Knotens **33** enthält, der auf den Empfang einer Verbindungsanforderung wartet (Operation **811**). Anschließend lässt der nicht-mobile Knoten **33** die empfangene Verbindungsanforderung zu, indem eine Funktion accept (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion accept (nicht-mobile IP) Informationen über das vierte Socket und eine Adresse des Knotens **13** enthält, der die Verbindungsanforderung gesendet hat (Operation **812**). Bei einer nicht-mobilen Kommunikation wie beispielsweise UDP können die Operationen des Aufrufens einer Funktion listen (), connect () und accept () ausgelassen werden.

[0103] Danach sendet der mobile Knoten **23** als ein Client von dem mobilen Knoten **23** über das dritte Socket gesendete Daten, indem eine Funktion send (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (mobile IP) Informationen über das dritte Socket und Informationen über von dem mobilen Knoten **23** übertragene Daten enthält (Operation **813**). Anschließend empfängt der Knoten **13** als ein Server von dem mobilen Knoten **23** übertragene Daten über das erste Socket, indem eine Funktion recv (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (mobile IP) Informationen über das erste Socket und von dem mobilen Knoten **23** übertragene Daten enthält (Operation **814**). Danach sendet der Knoten **13** von dem mobilen Knoten **23** übertragene Daten über das zweite Socket, indem eine Funktion send (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Daten Informationen über das zweite Socket und von dem mobilen Knoten **23** übertragene Daten enthalten (Operation **815**). Dieser Prozess ist auf den Übergangsprozess von der mobilen IP zu der nicht-mobilen IP in der Anwendungsschicht anwendbar. Anschließend empfängt der nicht-mobile Knoten **33** als ein Server von dem mobilen Knoten **23** übertragene Daten über das vierte Socket, indem eine Funktion recv (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (nicht-mobile IP) Informationen über das vierte So-

cket und von dem mobilen Knoten **23** übertragene Daten enthält (Operation **816**). Der nicht-mobile Knoten **33** verarbeitet die von dem mobilen Knoten **23** übertragenen Daten.

[0104] Anschließend sendet der nicht-mobile Knoten **33** als ein Server von dem nicht-mobilen Knoten **33** übertragene Daten über das vierte Socket, indem eine Funktion send (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (nicht-mobile IP) Informationen über das vierte Socket und die von dem nicht-mobilen Knoten **33** übertragenen Daten enthält (Operation **817**). Danach empfängt der Knoten **13** als ein Client die von dem nicht-mobilen Knoten **33** übertragenen Daten über das zweite Socket, indem eine Funktion recv (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (nicht-mobile IP) Informationen über das zweite Socket und die von dem nicht-mobilen Knoten **33** übertragenen Daten enthält (Operation **818**). Anschließend sendet der Knoten **13** als ein Server die von dem nicht-mobilen Knoten **33** übertragenen Daten über das erste Socket, indem eine Funktion send (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (mobile IP) Informationen über das erste Socket sowie die von dem nicht-mobilen Knoten **33** übertragenen Daten enthält (Operation **819**). Dieser Prozess ist auf den Übergangsprozess von der nicht-mobilen IP zu der mobilen IP in der Anwendungsschicht anwendbar. Danach empfängt der mobile Knoten **23** als ein Client von dem nicht-mobilen Knoten **33** übertragene Daten über das dritte Socket, indem eine Funktion recv (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (mobile IP) Informationen über das vierte Socket sowie die von der nicht-mobilen IP **31** übertragenen Daten enthält (Operation **820**). Der mobile Knoten **23** verarbeitet die von dem nicht-mobilen Knoten **33** übertragenen Daten. In dem Fall einer nicht-mobilen Kommunikation wie beispielsweise UDP werden anstatt der Funktion send () und recv () die Funktionen sendto () beispielsweise recvfrom () aufgerufen.

[0105] Fig. 9 ist ein Ablaufdiagramm eines anderen Verfahrens zum Verbinden von Knoten heterogener Protokolle gemäß noch einer anderen beispielhaften Ausführungsform der vorliegenden Erfindung.

[0106] In Fig. 9 umfasst das Verfahren zum Verbinden von Knoten heterogener Protokolle Operationen wie nachstehend beschrieben. Das Verfahren zum Verbinden von Knoten heterogener Protokolle kann beispielsweise in der Netzwerkumgebung implementiert werden, wie sie in Fig. 5 dargestellt wird.

[0107] Zuerst erzeugt der Knoten **14** als ein Server und ein Client das erste Socket, indem eine Funktion Socket (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion Socket (nicht-mobile IP) Informationen über die nicht-mobile IP enthält, und das zweite Socket wird erzeugt, indem eine Funktion Socket (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion Socket (mobile IP) Informationen über die mobile IP enthält (Operationen **901** und **902**). Gleichzeitig erzeugt der nicht-mobile Knoten **24** das dritte Socket, indem eine Funktion Socket (nicht-mobile IP) aufgerufen wird, die Informationen über die nicht-mobile IP enthält, und der mobile Knoten **34** erzeugt das vierte Socket, indem eine Funktion Socket (mobile IP) aufgerufen wird, die Informationen über die mobile IP enthält (Operationen **903** und **904**).

[0108] Anschließend verbindet der Knoten **14** als ein Server eine Adresse des Knotens **14** mit dem ersten Socket, indem eine Funktion bind (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion bind (nicht-mobile IP) Informationen über das erste Socket und eine Adresse des Knotens **14** enthält, der durch den nicht-mobilen Knoten **24** als Bestimmungsort eingestellt ist (Operation **905**). Danach wartet der Knoten **14** als ein Server den Empfang einer Verbindungsanforderung ab, deren Bestimmungsort eine Adresse ist, die mit dem ersten Socket verbunden ist, indem eine Funktion listen (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion listen (nicht-mobile IP) Informationen über das erste Socket enthält (Operation **906**). Gleichzeitig verbindet der mobile Knoten **34** als ein Server eine Adresse des mobilen Knotens **34** mit dem vierten Socket, indem eine Funktion bind (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion bind (mobile IP) Informationen über das vierte Socket enthält (Operation **907**). Danach wartet der mobile Knoten **34** als ein Server den Empfang einer Verbindungsanforderung ab, deren Bestimmungsort eine Adresse ist, die mit dem vierten Socket verbunden ist, indem eine Funktion listen (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion listen (mobile IP) Informationen über das vierte Socket enthält (Operation **908**).

[0109] Danach fordert der nicht-mobile Knoten **24** als ein Client eine Verbindung mit dem Knoten **14** an, indem eine Funktion connect (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion connect (nicht-mobile IP) Informationen über das dritte Socket und eine Adresse des Knotens **14** enthält, der den Empfang einer Verbindungsanforderung erwartet (Operation **909**). Anschließend lässt der Knoten **14** als ein Server die Verbindungsanforderung zu, indem eine Funktion accept (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion accept (nicht-mobile IP) Informationen über das erste Socket und eine Adresse des nicht-mobilen Knotens **24** enthält, der die Verbindungsanforderung gesendet hat (Operation **910**). Gleichzeitig fordert der Knoten **14** als ein Client eine Verbindung zu dem mobilen Knoten **34** an, indem

eine Funktion connect (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion connect (mobile IP) Informationen über das zweite Socket und eine Adresse des mobilen Knotens **34** enthält, der auf den Empfang einer Verbindungsanforderung wartet (Operation **911**). Anschließend lässt der mobile Knoten **34** die empfangene Verbindungsanforderung zu, indem eine Funktion accept (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion accept (mobile IP) Informationen über das vierte Socket und eine Adresse des Knotens **14** enthält, der die Verbindungsanforderung gesendet hat (Operation **912**). Bei einer nicht-mobilen Kommunikation wie beispielsweise UDP werden die Operationen des Aufrufens einer Funktion listen (), connect () und accept () ausgelassen.

[0110] Danach sendet der nicht-mobile Knoten **24** als ein Client von dem nicht-mobilen Knoten **24** übertragene Daten über das dritte Socket, indem eine Funktion send (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (nicht-mobile IP) Informationen über das dritte Socket und Informationen über von dem nicht-mobilen Knoten **24** übertragene Daten enthält (Operation **913**). Anschließend empfängt der Knoten **14** als ein Server von dem nicht-mobilen Knoten **24** übertragene Daten über das erste Socket, indem eine Funktion recv (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (nicht-mobile IP) Informationen über das erste Socket und von dem nicht-mobilen Knoten **24** übertragene Daten enthält (Operation **914**). Danach sendet der Knoten **14** von dem nicht-mobilen Knoten **24** übertragene Daten über das zweite Socket, indem eine Funktion send (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Daten Informationen über das zweite Socket und von dem nicht-mobilen Knoten **24** übertragene Daten enthalten (Operation **915**). Dieser Prozess ist auf den Übergangsprozess von der nicht-mobilen IP zu der mobilen IP in der Anwendungsschicht anwendbar. Danach empfängt der mobile Knoten **34** als ein Server von dem nicht-mobilen Knoten **24** übertragene Daten über das vierte Socket, indem eine Funktion recv (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (mobile IP) Informationen über das vierte Socket und die von dem nicht-mobilen Knoten **24** übertragenen Daten enthält (Operation **916**). Der mobile Knoten **34** verarbeitet die von dem nicht-mobilen Knoten **24** übertragenen Daten.

[0111] Anschließend sendet der mobile Knoten **34** als ein Server von dem mobilen Knoten **34** übertragene Daten über das vierte Socket, indem eine Funktion send (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (mobile IP) Informationen über das vierte Socket und die von dem mobilen Knoten **34** übertragenen Daten enthält (Operation **917**). Danach empfängt der Knoten **14** als ein Client die von dem mobilen Knoten **34** übertragenen Daten über das zweite Socket, indem eine Funktion recv (mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (mobile IP) Informationen über das zweite Socket und die von dem mobilen Knoten **34** übertragenen Daten enthält (Operation **918**). Anschließend sendet der Knoten **14** als ein Server von dem mobilen Knoten **34** übertragene Daten über das erste Socket, indem eine Funktion send (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion send (nicht-mobile IP) Informationen über das erste Socket und von dem mobilen Knoten **34** übertragene Daten enthält (Operation **919**). Dieser Prozess ist auf den Übergangsprozess von der mobilen IP zu der nicht-mobilen IP in der Anwendungsschicht anwendbar. Anschließend empfängt der nicht-mobile Knoten **24** als ein Client von dem mobilen Knoten **34** übertragene Daten über das dritte Socket, indem eine Funktion recv (nicht-mobile IP) in der Anwendungsschicht aufgerufen wird, wobei die Funktion recv (nicht-mobile IP) Informationen über das vierte Socket und die von der mobilen IP **31** übertragenen Daten enthält (Operation **920**). Der nicht-mobile Knoten **24** verarbeitet die von dem mobilen Knoten **34** übertragenen Daten. In dem Fall einer nicht-mobilen Kommunikation wie beispielsweise UDP werden anstatt der Funktion send () und recv () die Funktionen sendto () beispielsweise recvfrom () aufgerufen.

[0112] Ausführungsformen der vorliegenden Erfindung können über computerlesbaren Code implementiert werden und sie können in digitalen Computern für allgemeine Zwecke implementiert werden, die den computerlesbaren Code mit einem Medium, beispielsweise mit computerlesbaren Aufzeichnungsmedien, ausführen. Beispiele der Medien enthalten beispielsweise magnetische Speichermedien (beispielsweise ROM, Disketten, Festplatten und dergleichen), optische Aufzeichnungsmedien (beispielsweise CD-ROMs oder DVDs) sowie Speichermedien wie beispielsweise Trägerwellen (beispielsweise Übermittlung über das Internet) ein.

[0113] Gemäß Ausführungsformen der vorliegenden Erfindung ist es möglich, Knoten miteinander zu verbinden, die heterogene Protokolle verwenden. Beispielsweise kann ein IPv4-Knoten mit einem IPv6-Knoten verbunden werden, und ein nicht-mobiler Knoten kann mit einem mobilen Knoten verbunden werden. Insbesondere können Benutzer oder Anbieter von Endgeräten die vorliegende Erfindung leicht durchführen, da die vorliegende Erfindung in der Anwendungsschicht durchgeführt werden kann, die von den Benutzern oder den Anbietern von Endgeräten verwaltet werden kann.

[0114] Obwohl einige bevorzugte Ausführungsformen gezeigt und beschrieben wurden, ist es für Personen mit gewöhnlicher Erfahrung auf dem Gebiet

der Technik erkennbar, dass verschiedene Änderungen und Modifikationen vorgenommen werden können, ohne von dem Umfang der Erfindung, wie in den beigefügten Ansprüchen definiert, abzuweichen.

[0115] Die beschriebenen Ausführungsformen dürfen nur im beschreibenden Sinn und nicht zum Zweck der Einschränkung berücksichtigt werden. Daher wird der Umfang der Erfindung nicht durch die ausführliche Beschreibung der Erfindung definiert, sondern durch die beigefügten Ansprüche, und alle Unterschiede innerhalb des Umfanges werden als in der vorliegenden Erfindung enthalten betrachtet.

Patentansprüche

1. Verfahren zum Verbinden von Knoten (**1**, **2**, **3**) heterogener Protokolle, wobei das Verfahren umfasst:
Empfangen erster Daten an einem Knoten (**1**), die von einem ersten Knoten (**2**) übertragen werden, der ein erstes Protokoll (**400**) verwendet, über ein erstes Socket (**200**) in dem Knoten (**1**); und
Senden der empfangenen ersten Daten zu einem zweiten Knoten (**3**), der ein zweites Protokoll (**500**) verwendet, über ein zweites Socket (**300**) in dem Knoten (**1**); wobei
beim Empfangen der von dem ersten Knoten (**2**) übertragenen ersten Daten die ersten Daten über das erste Socket (**200**) empfangen werden, indem eine Empfangsfunktion in einer Anwendungsschicht (**100**) des Knotens (**1**) aufgerufen wird, und die Empfangsfunktion Informationen über das erste Socket (**200**) und die ersten Daten enthält, und
beim Senden der empfangenen ersten Daten von dem Knoten (**1**) zu dem zweiten Knoten (**3**) die empfangenen ersten Daten über das zweite Socket (**300**) gesendet werden, indem eine Sendefunktion in der Anwendungsschicht (**100**) aufgerufen wird, und die Sendefunktion Informationen über das zweite Socket (**300**) und die Informationen über die ersten Daten enthält.

2. Verfahren nach Anspruch 1, wobei das erste Protokoll (**400**) IPv6 ist und das zweite Protokoll (**500**) IPv4 ist oder das zweite Protokoll (**500**) IPv6 ist und das erste Protokoll (**400**) IPv4 ist.

3. Verfahren nach Anspruch 1 oder Anspruch 2, das des Weiteren Erzeugen des ersten Socket (**200**) und des zweiten Socket (**300**) umfasst, die bei Kommunikation auf Basis des ersten Protokolls (**400**) bzw. des zweiten Protokolls (**500**) zu verwenden sind; und
Verbinden des erzeugten ersten Socket (**200**) mit dem ersten Knoten (**2**) und Verbinden des erzeugten zweiten Socket (**300**) mit dem zweiten Knoten (**3**),
wobei beim Empfangen der von dem ersten Knoten (**2**) übertragenen ersten Daten die ersten Daten über das erste Socket (**200**) empfangen werden, das beim Erzeugen des ersten und des zweiten Socket (**200**, **300**) verbunden wird, und
beim Senden der empfangenen ersten Daten zu dem zweiten Knoten (**3**) die empfangenen ersten Daten über das zweite Socket (**300**) gesendet werden, das beim Erzeugen des ersten und des zweiten Socket (**200**, **300**) verbunden wird.

4. Verfahren nach einem der Ansprüche 1 bis 3, wobei beim Erzeugen des ersten und des zweiten Socket (**200**, **300**) das erste Socket (**200**) und das zweite Socket (**300**), die Anwendungsprogramm-Schnittstellen sind, erzeugt werden, indem vorgegebene Funktionen in einer Anwendungsschicht (**100**) aufgerufen werden.

5. Verfahren nach einem der vorangehenden Ansprüche, wobei beim Kommunizieren Kommunikation über das erste Socket (**200**) und das zweite Socket (**300**) durchgeführt wird, die Anwendungsprogramm-Schnittstellen sind, indem vorgegebene Funktionen in einer Anwendungsschicht (**100**) aufgerufen werden.

6. Vorrichtung zum Verbinden von Knoten (**1**, **2**, **3**) heterogener Protokolle, wobei die Vorrichtung umfasst:
eine erste Socket-Kommunikationseinheit (**103**) in einem Knoten (**1**) zum Empfangen erster Daten über ein erstes Socket (**200**), wobei die ersten Daten von einem ersten Knoten (**2**) unter Verwendung eines ersten Protokolls (**400**) übertragen werden; und
eine zweite Socket-Kommunikationseinheit (**104**) in dem Knoten (**1**) zum Senden der durch die erste Socket-Kommunikationseinheit (**103**) empfangenen ersten Daten zu einem zweiten Knoten (**3**) über ein zweites Socket (**300**), wobei
die erste Socket-Kommunikationseinheit (**103**) so betrieben werden kann, dass sie Daten über das erste Socket (**200**) empfängt, indem eine Empfangsfunktion in der Anwendungsschicht (**100**) in dem Knoten (**1**) aufgerufen wird, und die Empfangsfunktion Informationen über das erste Socket (**200**) und die empfangenen Daten enthält, und
die zweite Socket-Kommunikationseinheit (**104**) so betrieben werden kann, dass sie die empfangenen Daten über das zweite Socket (**300**) sendet, indem eine Sendefunktion in der Anwendungsschicht (**100**) aufgerufen wird, und die Sendefunktion Informationen über das zweite Socket (**300**) und die Informationen über die empfangenen Daten enthält.

7. Vorrichtung nach Anspruch 6, wobei das erste Protokoll (**400**) IPv6 ist und das zweite Protokoll (**500**) IPv4 ist oder das zweite Protokoll (**500**) IPv6 ist und das erste Protokoll (**400**) IPv4 ist.

8. Vorrichtung nach Anspruch 6 oder Anspruch 7, die des Weiteren eine Doppel-Socket-Erzeugungseinheit (**100**) zum Erzeugen des ersten Socket (**200**) und des zweiten Socket (**300**), die bei Kommunikati-

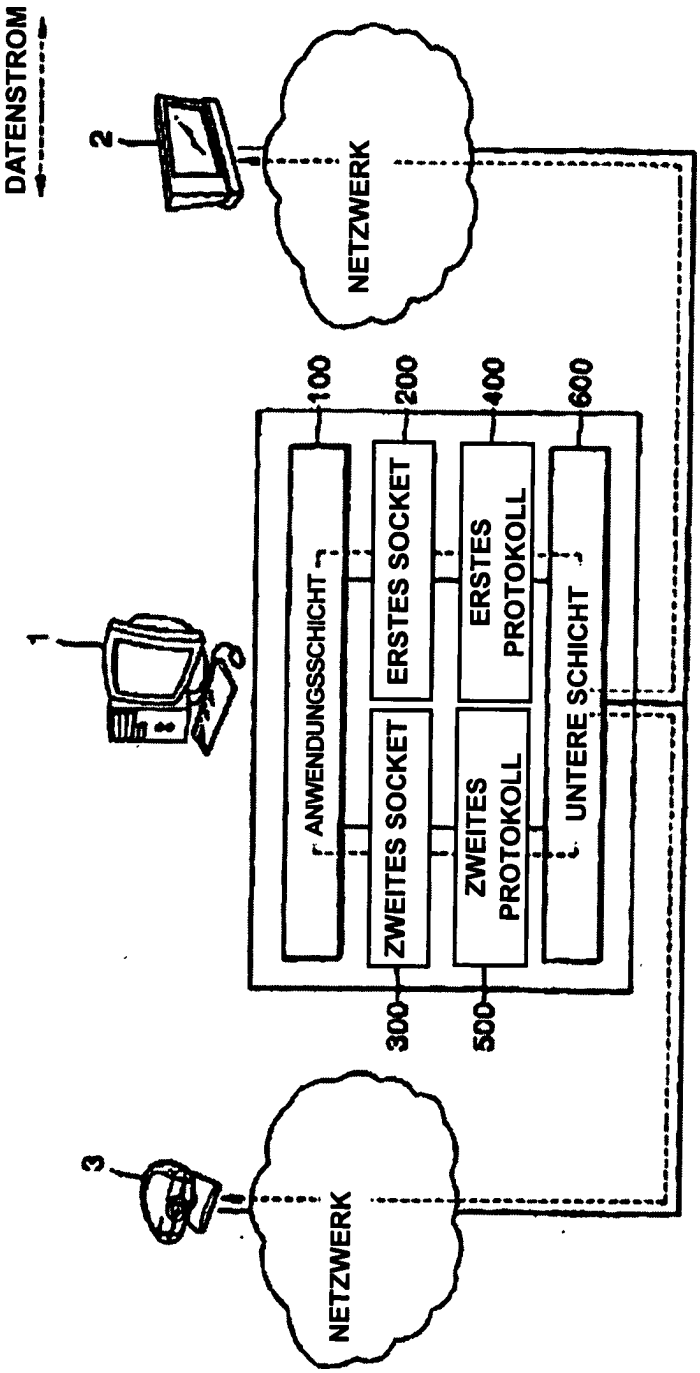
on auf Basis des ersten Protokolls (**400**) bzw. der Kommunikation auf Basis des zweiten Protokolls (**500**) zu verwenden sind; und
eine Doppel-Socket-Verbindungseinheit (**102**) zum Verbinden des ersten Socket (**200**) mit dem ersten Knoten (**2**) und des zweiten Socket (**300**) mit dem zweiten Knoten (**3**) umfasst, wobei das erste und das zweite Socket (**200**, **300**) durch die Doppel-Socket-Erzeugungseinheit (**101**) erzeugt werden,
wobei die erste Socket-Kommunikationseinheit (**103**) so betrieben werden kann, dass sie Daten über das erste Socket (**200**) empfängt, das durch die Doppel-Socket-Verbindungseinheit (**102**) verbunden ist, und die zweite Socket-Kommunikationseinheit (**104**) so betrieben werden kann, dass sie Daten über das zweite Socket (**300**) sendet, das durch die Doppel-Socket-Verbindungseinheit (**102**) verbunden ist.

9. Medium, das computerlesbaren Code umfasst, der, wenn er ausgeführt wird, die Verbindung von Knoten (**1**, **2**, **3**) heterogener Protokolle implementiert, wobei es umfasst
Empfangen erster Daten an einem Knoten (**1**), die von einem ersten Knoten (**2**) übertragen werden, der ein erstes Protokoll (**400**) verwendet, über ein erstes Socket (**200**) des Knotens (**1**); und
Senden der empfangenen ersten Daten von dem Knoten (**1**) zu einem zweiten Knoten (**3**), der ein zweites Protokoll (**500**) verwendet, über ein zweites Socket (**300**) des Knotens (**1**), wobei
beim Empfangen der von dem ersten Knoten (**2**) übertragenen ersten Daten die ersten Daten über das erste Socket (**200**) empfangen werden, indem eine Empfangsfunktion in einer Anwendungsschicht (**100**) aufgerufen wird, und die Empfangsfunktion Informationen über das erste Socket (**200**) und die ersten Daten enthält, und
beim Senden der empfangenen ersten Daten zu dem zweiten Knoten (**3**) die empfangenen ersten Daten über das zweite Socket (**300**) gesendet werden, indem eine Sendefunktion in der Anwendungsschicht (**100**) aufgerufen wird, und die Sendefunktion Informationen über das zweite Socket (**300**) und die Informationen über die ersten Daten enthält.

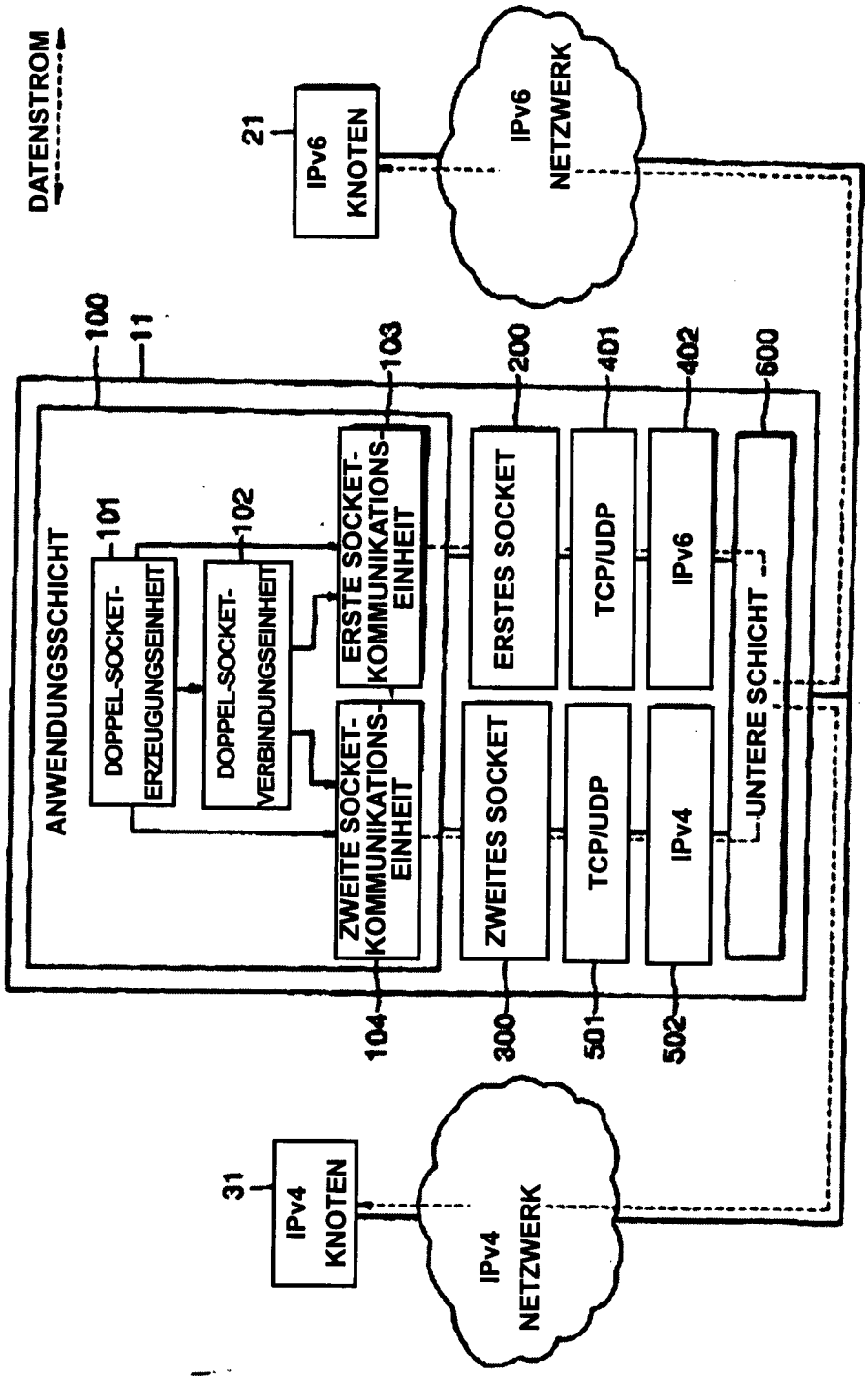
Es folgen 9 Blatt Zeichnungen

Anhängende Zeichnungen

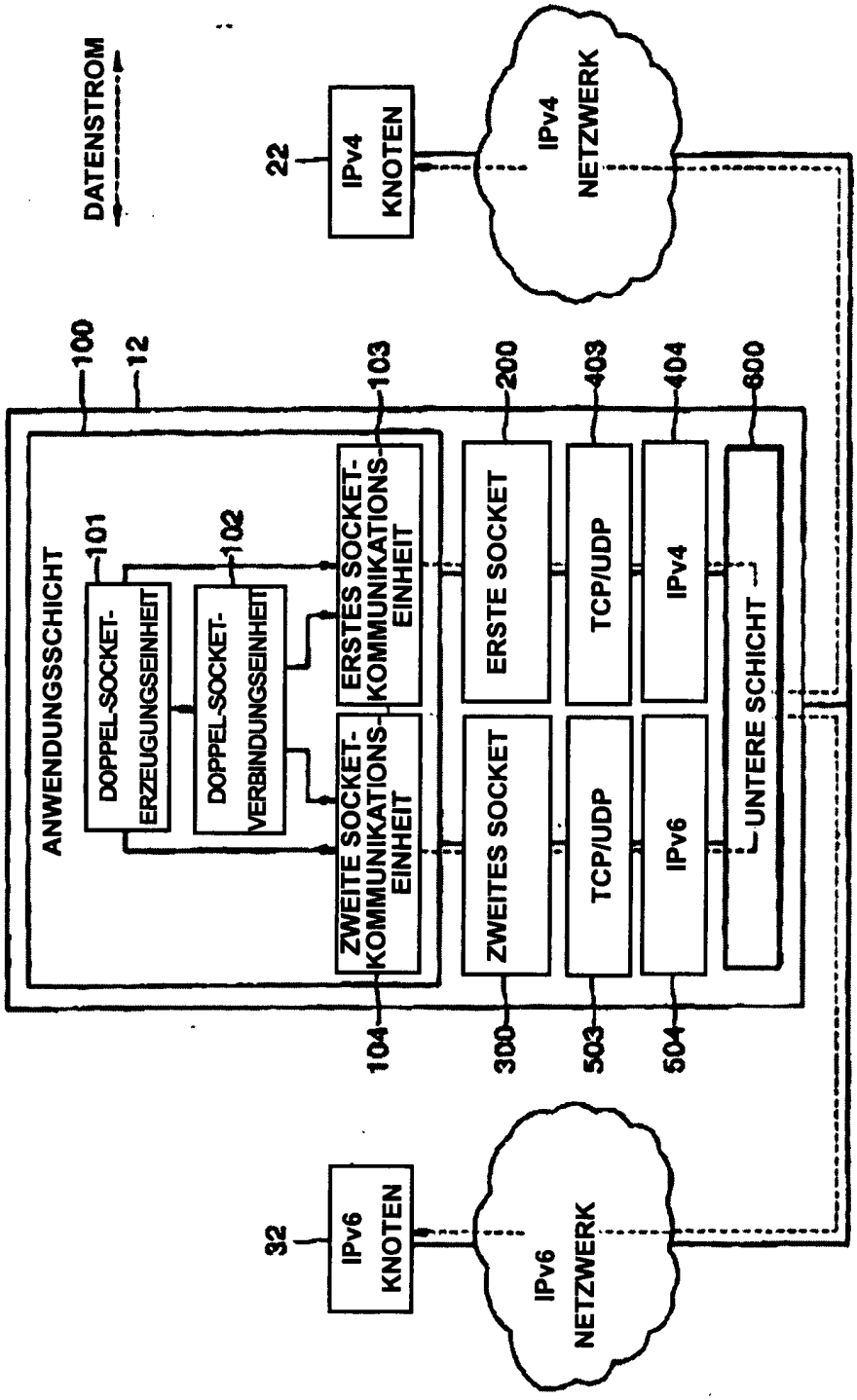
FIGUR 1



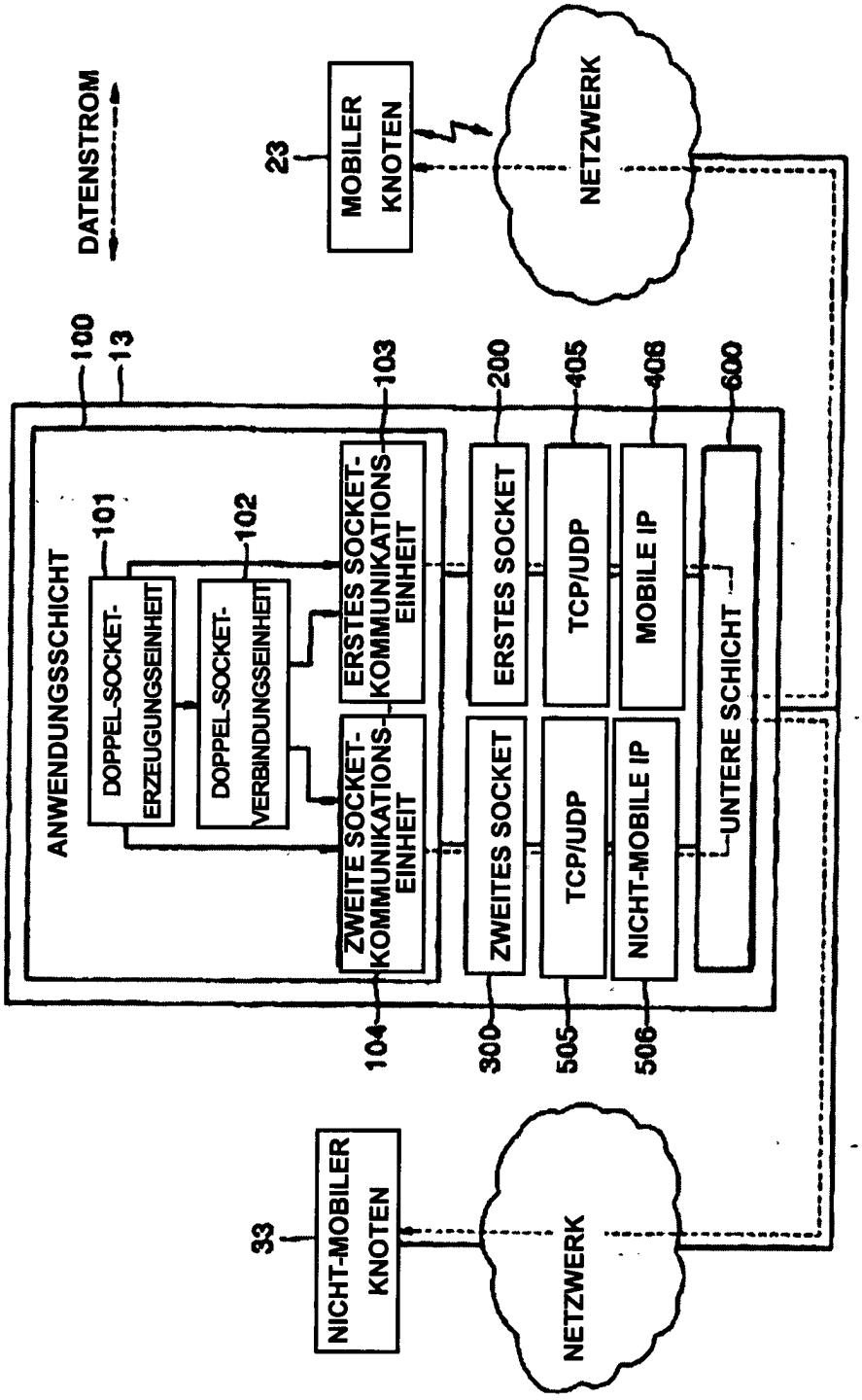
FIGUR 2


DATENSTROM
21
IPv6 KNOTEN
IPv6 NETZWERK
100
11
103
200
401
402
600
ANWENDUNGSSCHICHT
101
DOPPEL-SOCKET-ERZEUGUNGSEINHEIT
102
DOPPEL-SOCKET-VERBINDUNGSEINHEIT
ERSTE SOCKET-KOMMUNIKATIONS-EINHEIT
ZWEITE SOCKET-KOMMUNIKATIONS-EINHEIT
ERSTES SOCKET
ZWEITES SOCKET
TCP/UDP
TCP/UDP
IPv6
IPv4
UNTERE SCHICHT
104
300
501
502
31
IPv4 KNOTEN
IPv4 NETZWERK

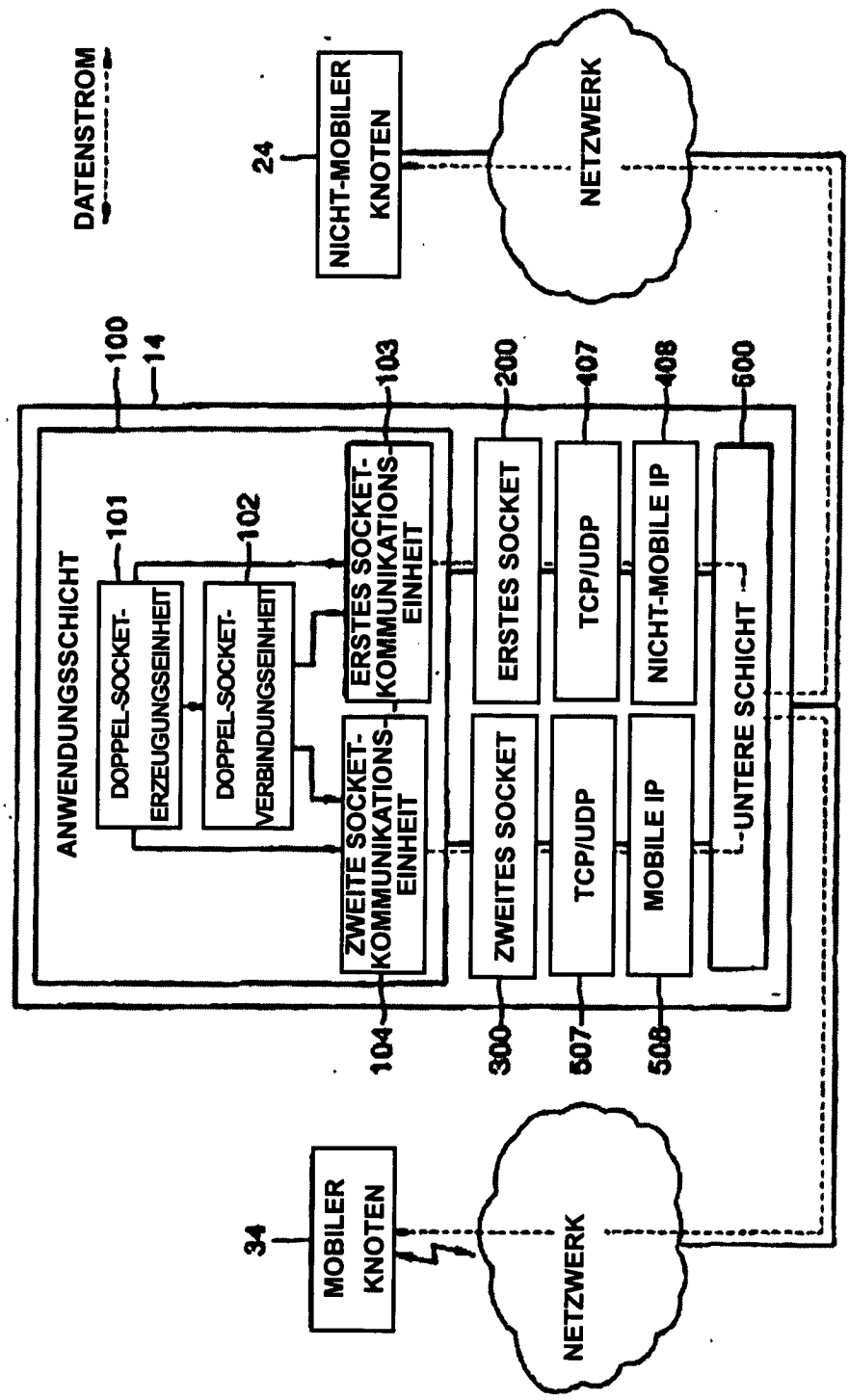
FIGUR 3


DATENSTROM
22
IPv4 KNOTEN
IPv4 NETZWERK
100
12
103
200
403
404
800
ANWENDUNGSSCHICHT
101
102
DOPPEL_SOCKET-ERZEUGUNGSEINHEIT
DOPPEL_SOCKET-VERBINDUNGSEINHEIT
ERSTES SOCKET-KOMMUNIKATIONS-EINHEIT
ZWEITE SOCKET-KOMMUNIKATIONS-EINHEIT
ERSTE SOCKET
ZWEITES SOCKET
TCP/UDP
TCP/UDP
IPv4
IPv6
UNTERE SCHICHT
104
300
503
504
32
IPv6 KNOTEN
IPv6 NETZWERK

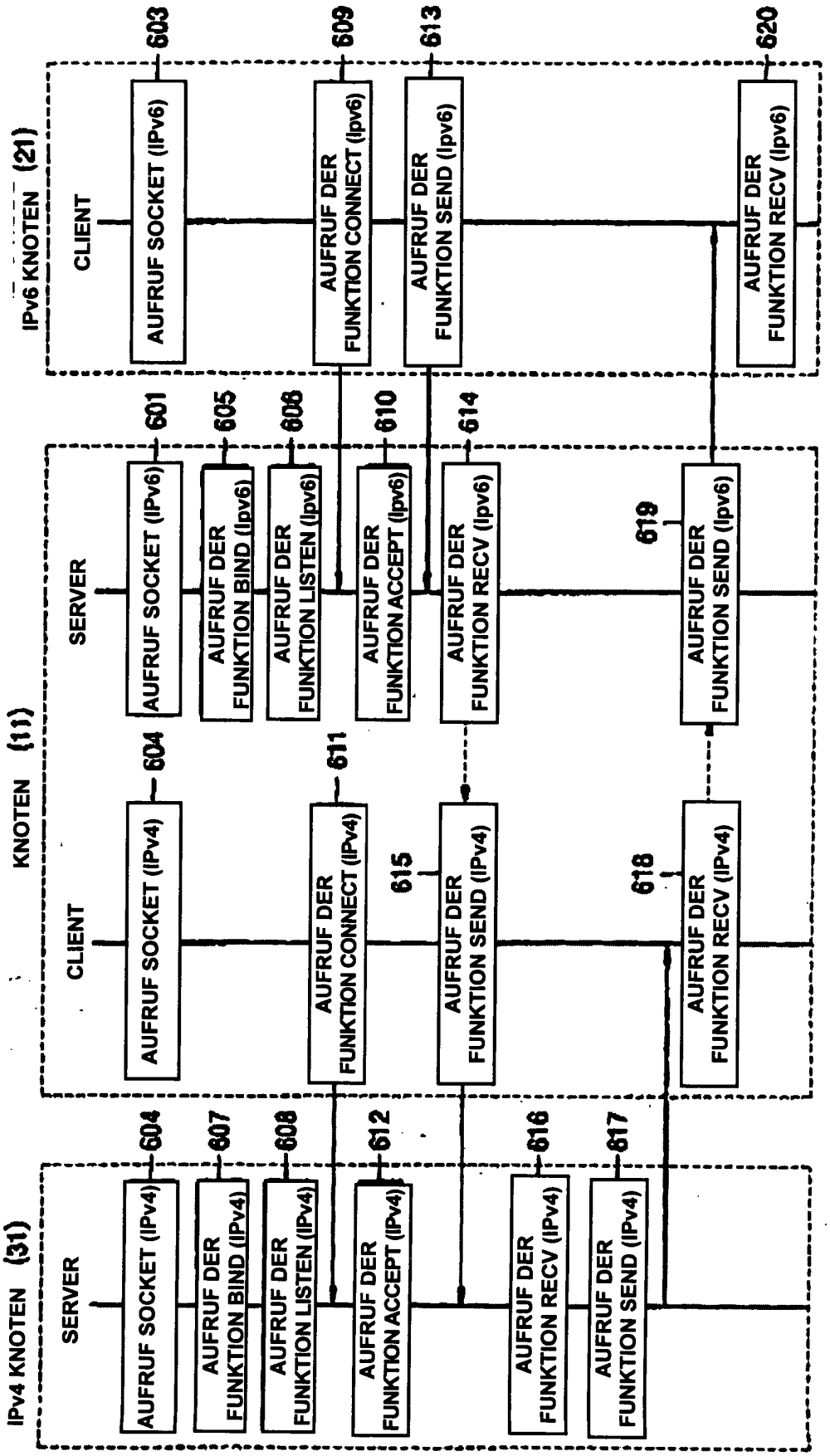
FIGUR 4


DATENSTROM
23
MOBILER KNOTEN
NETZWERK
100
13
ANWENDUNGSSCHICHT
101
DOPPEL-SOCKET-ERZEUGUNGSEINHEIT
102
DOPPEL-SOCKET-VERBINDUNGSEINHEIT
103
ERSTES SOCKET-KOMMUNIKATIONS-EINHEIT
ZWEITE SOCKET-KOMMUNIKATIONS-EINHEIT
200
ERSTES SOCKET
ZWEITES SOCKET
405
TCP/UDP
406
MOBILE IP
NICHT-MOBILE IP
600
UNTERE SCHICHT
104
300
505
506
33
NICHT-MOBILER KNOTEN
NETZWERK

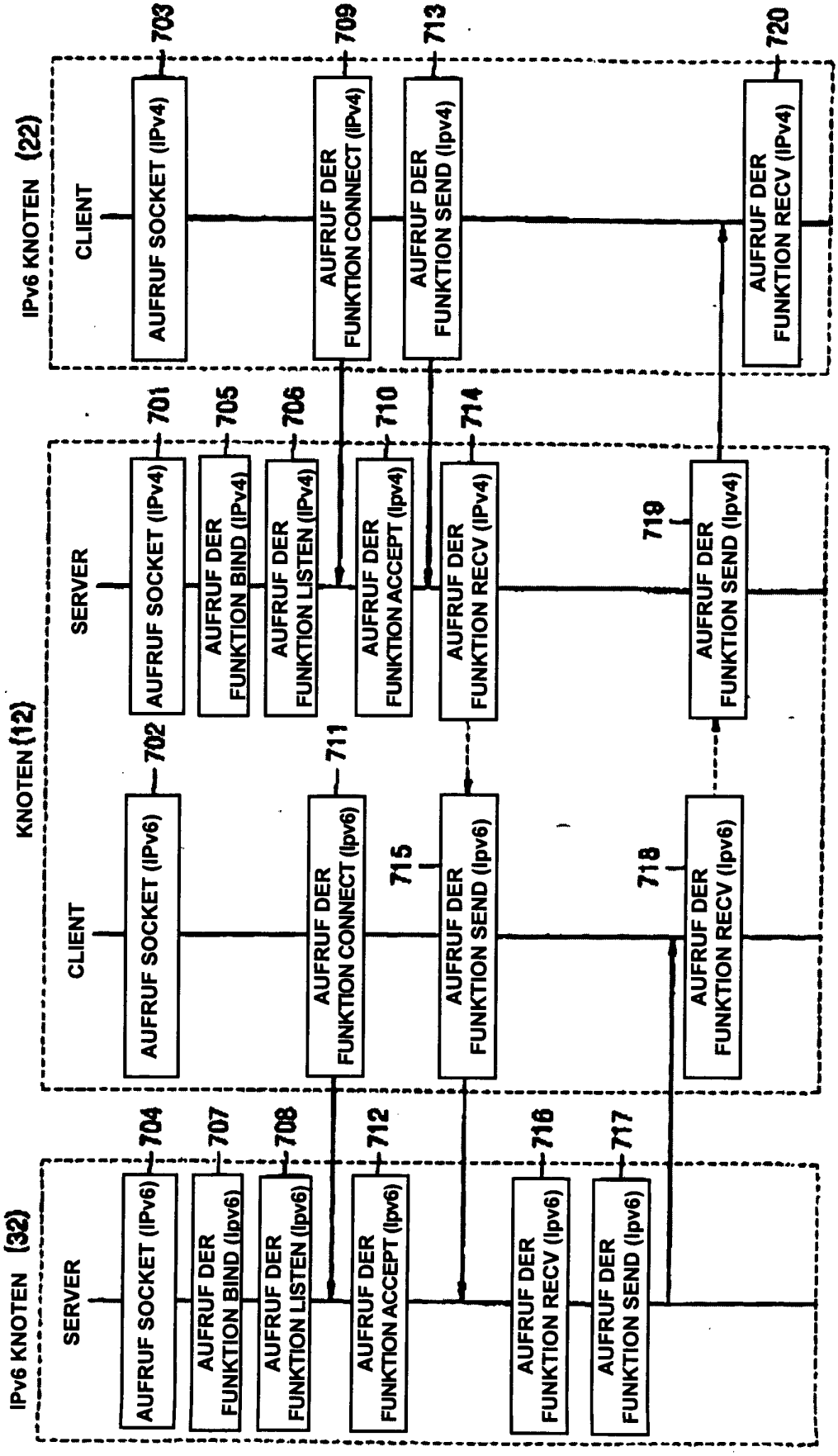
FIGUR 5


DATENSTROM
24
NICHT-MOBILER KNOTEN
NETZWERK
100
14
103
200
107
108
600
ANWENDUNGSSCHICHT
101
102
DOPPEL-SOCKET-ERZEUGUNGSEINHEIT
DOPPEL-SOCKET-VERBINDUNGSEINHEIT
ERSTES SOCKET-KOMMUNIKATIONS-EINHEIT
ZWEITE SOCKET-KOMMUNIKATIONS-EINHEIT
ERSTES SOCKET
ZWEITES SOCKET
TCP/UDP
TCP/UDP
NICHT-MOBILE IP
MOBILE IP
UNTERE SCHICHT
104
300
507
508
34
MOBILER KNOTEN
NETZWERK

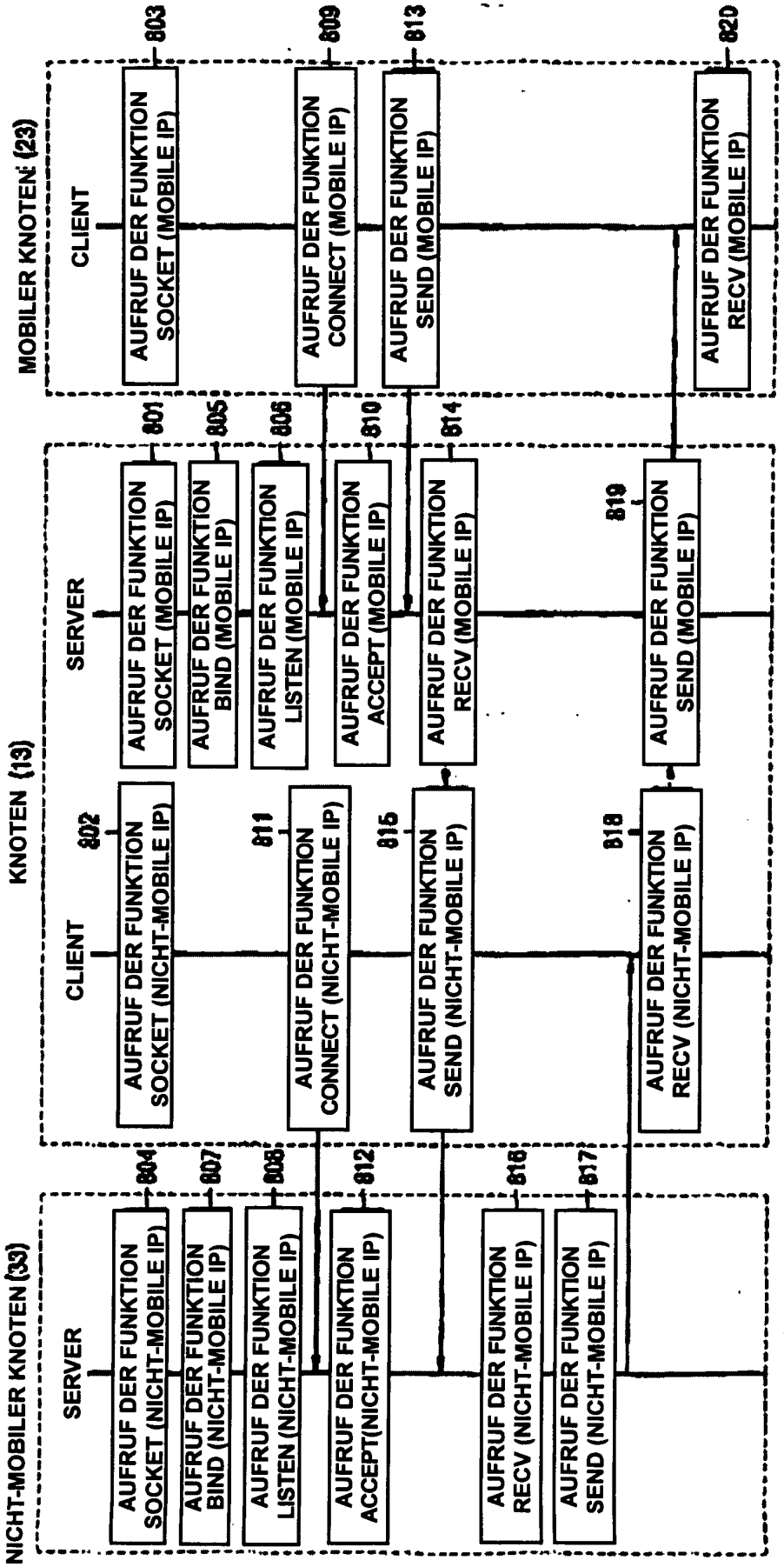
FIGUR 6


IPv6 KNOTEN (21)
CLIENT
AUFRUF SOCKET (IPv6)
603
AUFRUF DER FUNKTION CONNECT (Ipv6)
609
AUFRUF DER FUNKTION SEND (Ipv6)
613
AUFRUF DER FUNKTION RECV (Ipv6)
620
KNOTEN (11)
SERVER
AUFRUF SOCKET (IPv6)
601
AUFRUF DER FUNKTION BIND (Ipv6)
605
AUFRUF DER FUNKTION LISTEN (Ipv6)
606
AUFRUF DER FUNKTION ACCEPT (Ipv6)
610
AUFRUF DER FUNKTION RECV (Ipv6)
614
AUFRUF DER FUNKTION SEND (Ipv6)
619
CLIENT
AUFRUF SOCKET (IPv4)
604
AUFRUF DER FUNKTION CONNECT (IPv4)
611
AUFRUF DER FUNKTION SEND (IPv4)
615
AUFRUF DER FUNKTION RECV (IPv4)
618
IPv4 KNOTEN (31)
SERVER
AUFRUF SOCKET (IPv4)
604
AUFRUF DER FUNKTION BIND (IPv4)
607
AUFRUF DER FUNKTION LISTEN (IPv4)
608
AUFRUF DER FUNKTION ACCEPT (IPv4)
612
AUFRUF DER FUNKTION RECV (IPv4)
616
AUFRUF DER FUNKTION SEND (IPv4)
617

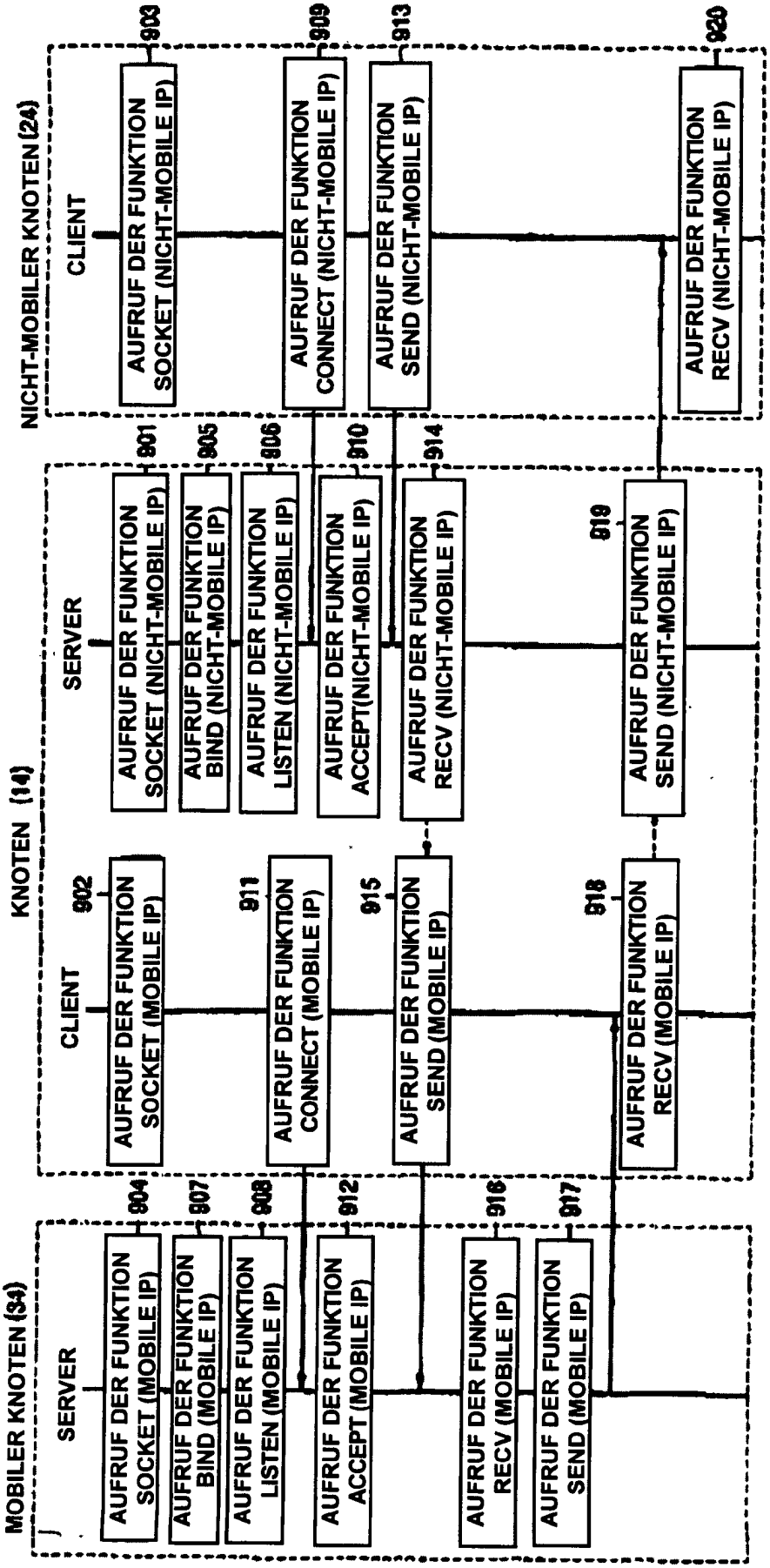
FIGUR 7


IPv6 KNOTEN (22)
CLIENT
AUFRUF SOCKET (IPv4)
703
AUFRUF DER FUNKTION CONNECT (IPv4)
709
AUFRUF DER FUNKTION SEND (Ipv4)
713
AUFRUF DER FUNKTION RECV (IPv4)
720
KNOTEN (12)
SERVER
AUFRUF SOCKET (IPv4)
701
AUFRUF DER FUNKTION BIND (IPv4)
705
AUFRUF DER FUNKTION LISTEN (IPv4)
706
AUFRUF DER FUNKTION ACCEPT (Ipv4)
710
AUFRUF DER FUNKTION RECV (IPv4)
714
AUFRUF DER FUNKTION SEND (Ipv4)
719
CLIENT
AUFRUF SOCKET (IPv6)
702
AUFRUF DER FUNKTION CONNECT (Ipv6)
711
AUFRUF DER FUNKTION SEND (Ipv6)
715
AUFRUF DER FUNKTION RECV (Ipv6)
718
IPv6 KNOTEN (32)
SERVER
AUFRUF SOCKET (IPv6)
704
AUFRUF DER FUNKTION BIND (Ipv6)
707
AUFRUF DER FUNKTION LISTEN (Ipv6)
708
AUFRUF DER FUNKTION ACCEPT (Ipv6)
712
AUFRUF DER FUNKTION RECV (Ipv6)
716
AUFRUF DER FUNKTION SEND (Ipv6)
717

FIGUR 8


MOBILER KNOTEN (23)
CLIENT
AUFRUF DER FUNKTION SOCKET (MOBILE IP)
AUFRUF DER FUNKTION CONNECT (MOBILE IP)
AUFRUF DER FUNKTION SEND (MOBILE IP)
AUFRUF DER FUNKTION RECV (MOBILE IP)
803
809
813
820
KNOTEN (13)
SERVER
AUFRUF DER FUNKTION SOCKET (MOBILE IP)
AUFRUF DER FUNKTION BIND (MOBILE IP)
AUFRUF DER FUNKTION LISTEN (MOBILE IP)
AUFRUF DER FUNKTION ACCEPT (MOBILE IP)
AUFRUF DER FUNKTION RECV (MOBILE IP)
AUFRUF DER FUNKTION SEND (MOBILE IP)
801
805
806
810
814
819
CLIENT
AUFRUF DER FUNKTION SOCKET (NICHT-MOBILE IP)
AUFRUF DER FUNKTION CONNECT (NICHT-MOBILE IP)
AUFRUF DER FUNKTION SEND (NICHT-MOBILE IP)
AUFRUF DER FUNKTION RECV (NICHT-MOBILE IP)
802
811
815
818
NICHT-MOBILER KNOTEN (33)
SERVER
AUFRUF DER FUNKTION SOCKET (NICHT-MOBILE IP)
AUFRUF DER FUNKTION BIND (NICHT-MOBILE IP)
AUFRUF DER FUNKTION LISTEN (NICHT-MOBILE IP)
AUFRUF DER FUNKTION ACCEPT(NICHT-MOBILE IP)
AUFRUF DER FUNKTION RECV (NICHT-MOBILE IP)
AUFRUF DER FUNKTION SEND (NICHT-MOBILE IP)
804
807
808
812
816
817

FIGUR 9


MOBILER KNOTEN (34)
SERVER
AUFRUF DER FUNKTION SOCKET (MOBILE IP)
AUFRUF DER FUNKTION BIND (MOBILE IP)
AUFRUF DER FUNKTION LISTEN (MOBILE IP)
AUFRUF DER FUNKTION ACCEPT (MOBILE IP)
AUFRUF DER FUNKTION RECV (MOBILE IP)
AUFRUF DER FUNKTION SEND (MOBILE IP)
904
907
908
912
916
917
KNOTEN (14)
CLIENT
AUFRUF DER FUNKTION SOCKET (MOBILE IP)
AUFRUF DER FUNKTION CONNECT (MOBILE IP)
AUFRUF DER FUNKTION SEND (MOBILE IP)
AUFRUF DER FUNKTION RECV (MOBILE IP)
902
911
915
918
SERVER
AUFRUF DER FUNKTION SOCKET (NICHT-MOBILE IP)
AUFRUF DER FUNKTION BIND (NICHT-MOBILE IP)
AUFRUF DER FUNKTION LISTEN (NICHT-MOBILE IP)
AUFRUF DER FUNKTION ACCEPT(NICHT-MOBILE IP)
AUFRUF DER FUNKTION RECV (NICHT-MOBILE IP)
AUFRUF DER FUNKTION SEND (NICHT-MOBILE IP)
901
905
906
910
914
919
NICHT-MOBILER KNOTEN (24)
CLIENT
AUFRUF DER FUNKTION SOCKET (NICHT-MOBILE IP)
AUFRUF DER FUNKTION CONNECT (NICHT-MOBILE IP)
AUFRUF DER FUNKTION SEND (NICHT-MOBILE IP)
AUFRUF DER FUNKTION RECV (NICHT-MOBILE IP)
903
909
913
920