(19) Deutsches Patent- und Markenamt

(10) **DE 100 54 923 B4** 2013.11.14

(12) # Patentschrift

(21) Aktenzeichen: **100 54 923.3**
(22) Anmeldetag: **06.11.2000**
(43) Offenlegungstag: **31.05.2001**
(45) Veröffentlichungstag der Patenterteilung: **14.11.2013**

(51) Int Cl.: ***G06F 13/10*** (2006.01)

Innerhalb von drei Monaten nach Veröffentlichung der Patenterteilung kann nach § 59 Patentgesetz gegen das Patent Einspruch erhoben werden. Der Einspruch ist schriftlich zu erklären und zu begründen. Innerhalb der Einspruchsfrist ist eine Einspruchsgebühr in Höhe von 200 Euro zu entrichten (§ 6 Patentkostengesetz in Verbindung mit der Anlage zu § 2 Abs. 1 Patentkostengesetz).

(30) Unionspriorität:
439419 **13.11.1999** **US**

(73) Patentinhaber:
Inside Secure, Aix-en-Provence, FR

(74) Vertreter:
Uexküll & Stolberg, 22607, Hamburg, DE

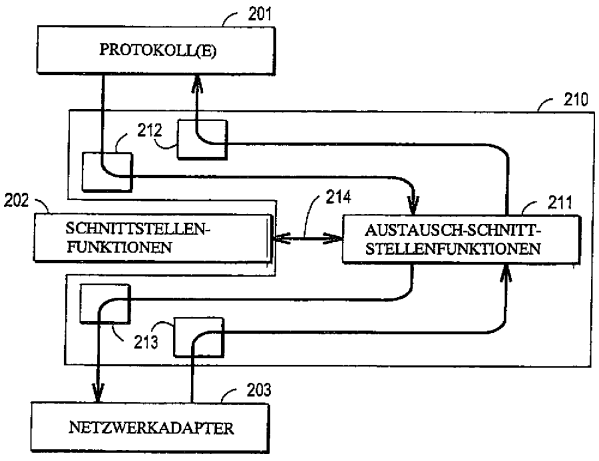
(72) Erfinder:
Haatainen, Niko, Kuopio, FI; Kivinen, Tero, Espoo, FI; Kukkonen, Jussi, Helsinki, FI; Ylönen, Tatu, Espoo, FI

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:

US	**5 781 550**	**A**
US	**5 786 770**	**A**
US	**5 862 362**	**A**
US	**5 983 274**	**A**

(54) Bezeichnung: **Verfahren zum Auffangen von Netzwerkpaketen in einem Computersystem, Computersystem zum Handhaben von Netzwerkpaketen sowie Paketauffängermodul zum Auffangen von Netzwerkpaketen in einem Computersystem**

(57) Hauptanspruch: Verfahren zum Auffangen von Netzwerkpaketen in einem Computersystem, wobei Netzwerkpakete zwischen einem ersten Netzwerkadapter und einer ersten Protokolleinrichtung kommuniziert werden, von denen der Netzwerkadapter eine bestimmte Netzwerk-Schnittstelle implementiert, wobei das Verfahren die Schritte aufweist
– Bereitstellen eines Satzes von Austauschfunktionen innerhalb eines Paketauffangmoduls;
– Einhängen wenigstens einer ersten Funktion, die zum Übertragen von Netzwerkpaketen von der ersten Protokolleinrichtung zu dem ersten Netzwerkadapter verwendet wird, in eine erste Austauschfunktion;
– Einhängen wenigstens einer zweiten Funktion, die zum Übertragen von Netzwerkpaketen von dem ersten Netzwerkadapter zu der ersten Protokolleinrichtung verwendet wird, in eine zweite Austauschfunktion; und
– Einhängen wenigstens einer dritten Funktion, die zum Empfangen von Information über den Zustand der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle verwendet wird, in eine dritte Austauschfunktion;
– wobei Einhängen definiert ist als Umleiten von Funktionsaufrufen an die erste, die zweite und die dritte Funktion auf entsprechende Austauschfunktionen, so dass die Austauschfunktionen durch die umgeleiteten Funktionsaufrufe aufgerufen werden.



Beschreibung

Technisches Fachgebiet

[0001] Die Erfindung betrifft Verfahren zum Auffangen von Netzwerkpaketen in einem Computersystem, ein Computersystem zum Handhaben von Netzwerkpaketen sowie ein Paketauffängermodul zum Auffangen von Netzwerkpaketen in einem Computersystem. Die Erfindung bezieht sich also auf das Handhaben von Datenpaketen, die über ein Netzwerk übertragen werden. Insbesondere betrifft die Erfindung das Thema des Auffangens von Datenpaketen, d. h. der Sicherung des Zugriffes zu im wesentlichen allen Datenpaketen, die von einem bestimmten System abgesendet und empfangen werden.

Zusammenfassung

[0002] Es wird ein Verfahren zum Auffangen von Netzwerkpaketen in einem Computersystem zur Verfügung gestellt, wobei eine Anzahl von Funktionen verwendet wird, um Netzwerkpakete zwischen einem Netzwerkadapter und einer Protokolleinrichtung zu übertragen. Ein erster Netzwerkadapter und eine erste Protokolleinrichtung, die in dem Computersystem installiert sind, werden identifiziert. Ein Satz von Austauschfunktionen wird in einem Paketauffangmodul zur Verfügung gestellt. Wenigstens eine Funktion, die für das Übertragen von Netzwerkpaketen von der ersten Protokolleinrichtung zu dem ersten Netzwerkadapter verwendet wird, ist in eine erste Austauschfunktion eingehängt. Wenigstens eine Funktion, die für das Übertragen von Netzwerkpaketen von dem ersten Netzwerkadapter zu der ersten Protokolleinrichtung verwendet wird, ist in eine zweite Austauschfunktion eingehängt. Wenigstens eine Funktion, die verwendet wird, um Informationen über den Zustand der Netzwerk-Schnittstelle, die von dem ersten Netzwerkadapter implementiert wird, zu empfangen, ist in eine dritte Austauschfunktion eingehängt.

Hintergrund der Erfindung

[0003] Das Anwachsen des Internet und seine Verwendung für neue Anwendungsgebiete haben die Einführung neuer Dienste vorteilhaft gemacht, welche die Art und Weise betreffen, in der Netzwerkpakete durch ein Netzwerk übertragen werden. Beispiele von Produkten und Diensten mit einem solchen Bedarf sind:

– Netzwerkebenen-Verschlüsselungsanwendungen, wie zum Beispiel VPNs (Virtual Private Networks – Virtuelle Privatnetzwerke), wie sie zum Beispiel in dem mit SWE98 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturnachweise beschrieben sind. Diese Anwendungen verschlüsseln und entschlüsseln Datenpakete, wenn sie in ein System und aus einem System heraus übertragen werden, um die Sicherheit der Daten bei der Übertragung zu gewährleisten. VPNs sind für das zuverlässige Durchführen von Geschäftstätigkeiten (Kaufen und Verkaufen) über ein öffentlich zugängliches paketvermitteltes Datenübertragungsnetzwerk, wie zum Beispiel über das Internet, und für das Verwenden des Internet für kritische Auftrags-Geschäftsanwendungen von Bedeutung.

– Firewalls (Feuerwände), wie sie zum Beispiel in dem mit CB94 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturnachweise beschrieben sind. Diese sind Sicherheitsvorrichtungen für das Netzwerk, die den Netzwerkverkehr gemäß spezifizierten Kriterien filtern und nur einigen Paketen den Durchlaß gestatten. Firewalls sind normalerweise als Erweiterungen von allgemeinen Zwecken dienenden Betriebssystemen implementiert, so daß sie überwachen und den Verkehr verändern können, der durch das System fließt. Sie können jedoch auch als zweckbestimmte Hardwarevorrichtungen implementiert sein.

– Erkennen von Eindringstörungen und Ausschnüffeln von Paketen. Viele Tools (Hilfsmittel) für das Erkennen von Eindringstörungen und für die Netzwerküberwachung benötigen Zutritt zu den Daten, die in Paketen durch ein Netzwerk übertragen werden. Ähnliche Tools werden auch verwendet, um statistische Daten über den Netzwerkverkehr zu sammeln (z. B. wie es in dem mit Waldbusser97 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist).

– Multimedia-Anwendungen. Es ist prognostiziert worden, daß 25% des Wertes des globalen Telekommunikationsmarktes in ein paar Jahren IP-basiert sein werden (wobei sich IP aus Internet Protocol-Internetprotokoll ableitet) und daß ein großer Teil des erforderlichen Datenverkehrs unter Verwendung von allgemeinen Zwecken dienenden Computern als Terminals über das öffentliche Internet übertragen wird. Eine garantierte QoS (Quality of Service – Qualität des Dienstes) ist für interaktive Video- und Audio-Anwendungen über solche Netzwerke erforderlich, wie es zum Beispiel in dem mit SCFJ96 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist.

– Mobilität der Terminals. Die Mobilität wird auch für paketvermittelte Datenübertragungsnetzwerke und für das Internet in wachsendem Maße wichtig, wie es zum Beispiel in dem mit Perkins96 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist. In vielen Fällen wird die Mobilitätsunterstützung als wertsteigernd für ein vorhandenes System vorgesehen und die Möglichkeit des Modifizierens und des Umleitens der ankommenden und der abgehenden Datenpakete ist erforderlich.

[0004] Alle vorher erwähnten Anwendungen verwenden spezifische Protokolle, die nicht in allen breit verwendeten Betriebssystemen vorhanden sind. In vielen Fällen wird der Verkäufer es wünschen, für diese Dienste Unterstützung auf breit installierten Plattformen zur Verfügung zu stellen, für die an sich keine Unterstützung für sie vorhanden ist. Das Implementieren solcher Unterstützungen erfordert oft, daß der Implementierende Zugriff zu allen Datenpaketen erhält, die von dem System abgesendet und empfangen werden. Der Modul, der einen solchen Zugriff zur Verfügung stellt, wird als Paketauffangmodul bezeichnet. Solche Module stellen der Anwendung normalerweise auch einige Informationen über die zur Verfügung stehenden Netzwerk-Schnittstellen und ihre Konfiguration (z. B. Netzwerkadressen) zur Verfügung. Die Anwendung besteht wiederum aus einer Komponente, die in Kernprogramm-Betriebsweise funktioniert und die Echtzeit-Paketverarbeitung handhabt und aus einer normalen Anwendung in Benutzer-Betriebsweise für die Verwaltung und für andere Funktionen, die nicht zeitkritisch sind und/oder einen Benutzerdialog erfordern.

[0005] Überall ist die Notwendigkeit des Auffangens von Paketen, die in ein System einfließen und daraus herausfließen äußerst wichtig. Das ist von den Programmierern und kommerziellen Betreibern in der Praxis sowie von den Verkäufern von Betriebssystemen, wie zum Beispiel von der Microsoft Corporation, erkannt worden. Eine wesentliche Arbeit ist aufgewendet worden, um Paketauffang-Funktionalitäten in Netzwerksystemen und verwandten Produkten zu implementieren.

[0006] Die vorhandenen Lösungen für das Paketauffangproblem fallen meistens unter die folgenden Kategorien A), B) und C):

A) Zwischentreiber. Ein TCP/IP-Protokollstapel (oder ein anderer Protokollstapel), wobei sich TCP/IP aus Transmission Control Protocol/Internet Protocol – Übertragungssteuerungs-Protokoll/Internet Protokoll ableitet, ist normalerweise geschichtet, so daß die Netzwerkgerätetreiber eines speziellen Hardwarebausteins, bekannt als Netzwerkadapter, eine Standardschnittstelle zur Verfügung stellen und die Protokollstapel verschiedene Netzwerkprotokolle implementieren. Die Protokollstapel sind durch die Standardschnittstelle, welche die Gerätetreiber implementieren müssen, hardware-unabhängig gestaltet. In Windows-Betriebssystemen, wobei Windows eine registrierte Handelsmarke der Microsoft Corporation ist, ist diese Schnittstelle, wie sie zum Beispiel in dem mit Win4DDK gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist, als NDIS bezeichnet (Network Driver Interface Specification – Netzwerktreiber-Schnittstellenspezifikation). In Sun Solaris, das auf die registrierten Handelsmarken Sun, Solaris und Sun Solaris von Sun Microsystems bezogen ist, ist eine ähnliche Funktionalität durch die STREAMS-Schnittstelle zur Verfügung gestellt, wie sie zum Beispiel in dem mit STREAMS93 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist. Zwischentreiber werden ohne weiteres zumindest von Microsoft Windows NT 4.0 (registrierte Handelsmarke von Microsoft) und von den Sun Solaris-Betriebssystemen unterstützt. Microsoft hat sogar einen Mustercode für das Entwickeln von Zwischentreibern für solche Anwendungen, wie sie vorher beschrieben sind, zur Verfügung gestellt. Mindestens zwei solcher Muster stehen zur Verfügung und viele Verkäufer haben implementierte Produkte, die auf dieser Technologie basieren.

B) Auffangen durch WINSOCK. In der Industrie ist gut bekannt, daß mehrere Produkte die WINSOCK. DLL ersetzen, wie sie zum Beispiel in den mit Bonner96 und QS96 gekennzeichneten Literaturhinweisen in der hierin enthaltenen Liste der Literaturanweisungen beschrieben sind. WINSOCK.DLL ist eine Datei in Window-Systemen. Einige Produkte verwenden Zwischentreiber auf der LSP-Ebene (Layered Service Provider- in Ebenen unterteilte Dienstanbieter), wie es zum Beispiel in dem mit Win4DDK gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise enthalten ist. Microsoft hat ebenfalls einen Mustercode für das Auffangen des Verkehrs auf dieser Ebene veröffentlicht.

C) Externe Bausteine außerhalb der Betriebssysteme. Dabei handelt es sich um Hardwareprodukte, die im wesentlichen kleine Kästen darstellen, die an der Rückseite des Computers befestigt sind oder die sogar in Netzwerkadapter eingebaut sind, und die den gesamten Netzwerkverkehr erkennen, der durch sie hindurchgeht. Solche Bausteine sind zumindest in Sicherheitsanwendungen verwendet worden, um Funktionalität zu implementieren, was aber anders auch durch das Auffangen des Verkehrs in dem Betriebssystem durchgeführt werden könnte.

[0007] Die bekannten Lösungen, die unter die vorher erwähnten Kategorien A bis C fallen, haben sich nicht als gute und robuste Hochleistungslösungen erwiesen, die in allen breit verwendeten Betriebssystemen funk-

tionieren würden. Insbesondere hat es sich für viele Verkäufer als äußerst schwierig herausgestellt, Paketauffangeinrichtungen für die Betriebssysteme Windows 95 und Windows 98 zu entwickeln, welche gegenwärtig sehr umfassend angewendet werden und für die das auch in den kommenden Jahren so bleiben wird.

[0008] Fast alle Softwareprodukte, die ein Auffangen von Paketen ausführen, verwenden Zwischentreiber, um das Auffangen auszuführen. Fig. 1 ist ein vereinfachtes Blockdiagramm, das die bekannte Verwendung eines Zwischentreibers, insbesondere in Verbindung mit einem Windows NT-Betriebssystem darstellt. Oben in Fig. 1 ist ein Anwendungsprogramm zu sehen, das einen Benutzer-Betriebsart-Clientbestandteil **101** hat. Zwischen ihm und einem Netzwerkprotokollblock **106** können sich Zwischeneinrichtungen befinden, die für die vorliegende Erfindung von geringer Bedeutung sind. Der Netzwerkprotokollblock **106** implementiert die Netzwerkprotokolle, zum Beispiel den TCP/IP-Protokollstapel. Der Zwischentreiber **107** befindet sich zwischen den Protokollstapeln und einem NIC-Treiber **108**. Er ist von ihnen durch die NDIS-Schnittstelle getrennt, von der in Fig. 1 Teile als **102** und **103** dargestellt sind. Der NIC-Treiberblock **108** ist dazu ausgestaltet, um eine NIC (Network Interface Card – Netzwerkschnittstellenkarte) **109** direkt zu verwalten. Bei der letzteren handelt es sich um eine Hardwarekomponente, normalerweise eine Erweiterungskarte, die mit dem internen Parallelbus eines Computers gekoppelt ist. Der NIC-Treiber **108** kann allgemeiner als Netzwerkadapter bezeichnet werden. Der NIC-Treiber **108** gestattet es den oberen Schichten durch das Netzwerk Pakete abzusenden und zu empfangen und Steuerungsoperationen auszuführen, wie zum Beispiel das Behandeln von Unterbrechungen, das Rückstellen oder das Anhalten der Schnittstellenkarte **109**. Er gestattet es den oberen Schichten auch die eigenen Betriebskennwerte abzufragen und einzustellen. Der Netzwerkadapter (d. h. der NIC-Treiber **108**) nimmt den Zwischentreiber **107** so wahr, als ob er ein Protokoll wäre und zu den Protokollblöcken **106** und zu dem Zwischentreiber **107** verhält er sich wie ein Netzwerkadapter. Dieselbe NDIS-Schnittstelle wird auf beiden Seiten des Zwischentreibers verwendet, d. h. zwischen den Blöcken **106** und **107** auf der einen Seite und zwischen den Blöcken **107** und **108** auf der anderen Seite. Die Netzwerkschnittstellenkarte **109** ist mit zumindest einem physikalischen Übertragungsmedium **110** gekoppelt, das zum Beispiel ein Glasfaserkabel, ein Koaxialkabel oder ein Paar von verdrillten Drähten sein kann.

[0009] Die Zwischentreibermethode funktioniert gut für Windows NT und Solaris, welche die Hauptplattformen für Firewall- und VPN-Anwendungen sind. In den letzten Jahren ist jedoch auch dem Implementieren von ähnlicher Funktionalität in andere Betriebssysteme, insbesondere in Windows 95 und Windows 98 erhöhte Aufmerksamkeit gewidmet worden. Das Entwickeln von Zwischentreibern für diese Systeme hat sich jedoch als extreme Wettbewerbsherausforderung erwiesen.

[0010] Ein Hauptproblem beim Programmieren solcher Treiber für diese Systeme ist die Unterstützung für Wählschnittstellen (dial-up interfaces), wie es zum Beispiel in dem mit Simpson94 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist. Eine Wählschnittstelle verwendet einen seriellen Port und ein Modem, um mit dem Internet oder mit einem anderen Netzwerk über das Telefonnetz in Verbindung zu treten. Solche Schnittstellen sind durch dynamische Netzwerkadressen gekennzeichnet und die Netzwerkverbindung steht nur für einen Teil der Zeit zur Verfügung. Die Verbindung wird aufgebaut (in einen aktiven, verbundenen Zustand) und abgebaut (in einen passiven, getrennten Zustand), wenn der Benutzer die Verbindung schaltet bzw. trennt.

[0011] Das größte Problem für das Schreiben von Auffangprogrammen für Wählschnittstellen ist gewesen, daß Windows 95, Windows 98 und Windows NT Erweiterungen für die NDIS-Schnittstelle verwenden, die von Microsoft urheberrechtlich geschützt sind, um die erweitere Funktionalität zur Verfügung zu stellen, die für die Wähl-Funktionalität und die dynamischen Adressen erforderlich ist. Angestellte von Microsoft haben empfohlen, daß das Auffangen von Paketen für diese Plattformen wegen der Art des Urheberschutzes der Schnittstelle nicht implementiert werden sollte. Von vielen Leuten ist bekannt, daß sie versucht haben, Zwischentreiber für diese Plattformen zu schreiben, ohne Erfolg zu haben oder mit begrenztem Erfolg erst nach Jahren mühsamer Ingenieurs- und Entwicklungsarbeit.

[0012] Einige wenige Firmen haben es zustande gebracht, Zwischentreiber selbst für diese Plattformen zu implementieren. Neuerdings hat es sich jedoch auch herausgestellt, daß Zwischentreiber Zuverlässigkeitsprobleme aufweisen. Die Treiber funktionieren nicht mit allen Netzwerkadaptern oder mit allen Versionen der Betriebssysteme. Je mehr Verkäufer Produkte auf den Markt bringen, die Zwischentreiber verwenden, desto mehr treten bei den Anwendern ernsthafte Kompatibilitätsprobleme zwischen den Zwischentreibern verschiedener Verkäufer auf, die in demselben Computer installiert sind. Die verschiedenen Zwischentreiber in einer korrekte Art und Weise zu verbinden, ist ein schwieriges, möglicherweise sogar unlösbares, Problem. Diese Sachverhalte können letztendlich bewirken, daß Zwischentreiber für die allgemeine Verwendung als zu unzuverlässig angesehen werden.

[0013] Ein weiteres Problem bei den Zwischentreibern ist ihre Leistungsfähigkeit. Die Zwischentreiber fügen dem Datenpfad eines Netzwerkpaketes einen wesentlichen Verarbeitungsumfang zu. Auch das Kopieren wesentlicher Datenmengen kann auftreten. Somit wird das Auffangen von Paketen durch die Verwendung von Zwischentreibern für großvolumige Anwendungen zu einem bedeutenden Engpaß für die Leistungsfähigkeit. Die NDIS-Bibliothek implementiert auch eine gewisse Sperrstrategie für die Zwischentreiber, die wahrscheinlich den Aufwand erhöht und bewirkt, daß die Protokoll-Zwischentreiber-NIC-Codepfade Halbduplexpfade sind. Das ist insbesondere für Leitweg-Anwendungen (Routing-Anwendungen) schlecht.

[0014] Die US 5 862 362 A beschreibt einen Netzwerkfehlersimulator. Insbesondere offenbart diese Schrift ein Netzwerk-Simulationswerkzeug, das einen ”Ersatz”-Empfangspaket-bearbeiter und einen ”Ersatz”-Sendepaketbearbeiter aufweist. Aufrufe von den Sende- und Empfangspaketbearbeitern werden zu den ”Ersatz”-Sende- und Empfangsbearbeitern umgeleitet. Die ”Ersatz”-Sende- und Empfangsbearbeiter geben einen Statuswert an ihre jeweiligen Sende- und Empfangspaketbearbeiter zurück. In einigen Ausführungsbeispielen geben die ”Ersatz”-Sende- und Empfangsbearbeiter einen Statuswert zurück, der einen Netzwerkfehler anzeigt.

[0015] Die US 5 983 274 A betrifft einen Ansatz, der es einem Netzwerk zum Bearbeiten von Softwarekomponenten ermöglicht, mit den Daten Steuerinformation an eine andere Softwarekomponente zu kommunizieren und mit der anderen Softwarekomponente zusammenzuarbeiten, indem Steuerinformationen mit einem Paket von Netzwerkdaten assoziiert werden. Diese Schrift schlägt eine Steuerdatenstruktur vor, die getrennt von den aktuell über ein Netzwerk zu übertragenden Netzwerkdaten ist und Steuerinformationen enthält, die durch eine Softwarekomponente gesetzt und von anderen Softwarekomponenten benutzt wird. Einer der beanspruchten Vorteile dieses Ansatzes ist die Möglichkeit, Information zwischen einem Transportprotokolltreiber und einem Netzwerkkarten-Devicetreiber zu übermitteln, der über eine integrierende Komponente verbunden ist. Somit schlägt diese Schrift einen Mechanismus zum Leiten von Informationen durch eine zwischenliegende Komponente vor, wie die NDIS-Technologie, ohne dass die zwischenliegende Komponente dies weiß oder interveniert.

[0016] Die US 5 781 550 A beschreibt ein computerimplementiertes Ver- fahren, wobei Pakete transparent und sicher zwischen einem als vertrauenswert angesehenen Computer und einem nicht als vertrauenswert angesehenen Computer, die über ein Gateway verbunden sind, kommuniziert werden. Das vorgeschlagene Verfahren erfordert Modifikationen beim Bearbeiten eines IP-Layers und beim Bearbeiten eines Transport-Layers, zusammen mit einer unterstützenden Funktionalität an einem Anwendungs-Layer. Folglich führt das vorgeschlagene Verfahren zu einem speziellen Bearbeiten eintreffender Pakete und zu einem Umleiten von proxybestimmten Paketen an dem IP-Layer und an dem Transport-Layer, wobei das Umleiten von der aus den Kopfbereichen der empfangenen Pakete extrahierten Internetlayer-Adresse gesteuert wird. Somit erfordert dieses Verfahren als Konsequenz eine spezielle Bearbeitung in allen Layern oberhalb des IP-Layers, was ferner durch von einem Applikationslayer-Prozess gesteuerte Crosslayer-Operationen unterstützt wird.

[0017] Die US 5 786 770 A betrifft einen Message-Bearbeiter für die Übertragung und/oder den Empfang von Message-Paketen in einer Richtung oder in beiden Richtungen zwischen einer Telekommunikationsstation und einem Controller. Dises Schrift schlägt einen Simulator vor, der in einer Telekommunikationsleitung zwischen einer Telekommunikationsstation und einem Controller für eine oder mehrere Telekommunikationsstationen verbinden kann, um den Betrieb einer Telekommunikationsstation oder eines Controllers zu testen. Dies kann einschließen, die von der Telekommunikationsstation oder von dem Controller geschickten Messages zu überwachen, ebenso wie das Erzeugen/Routen von Controller/Telekommunikationsstations-Messages an die/den Telekommunikationsstation/Controller und das Überwachen von Messages, die in Antwort auf die von dem Simulator erzeugten bzw. gerouteten Messages gesendet werden.

Zusammenfassung der Erfindung

[0018] Es ist eine Aufgabe der vorliegenden Erfindung, ein Verfahren und eine Anordnung zum Auffangen von Paketen zur Verfügung zu stellen, das mit verschiedenen Netzwerkadaptern, verschiedenen Betriebssystemen, verschiedenen Versionen von Betriebssystemen und verschiedener Software Dritter kompatibel ist. Es ist eine weitere Aufgabe der Erfindung, ein solches Verfahren und eine solche Anordnung mit dem zusätzlichen Vorteil zur Verfügung zu stellen, daß selbst relativ unerfahrene Anwender sie sich leicht zu eigen machen können.

[0019] Die Aufgaben der Erfindung werden durch Verwendung eines Verfahrens erreicht, daß wir in übertragenem Sinne als Einhängen (hooking) bezeichnen, um Zugriff zu allen Paketen und den zu ihnen gehörenden

Informationen zu erhalten und indem eine Anordnung zur Verfügung gestellt wird, die programmiert ist, um ein solches Verfahren zu implementieren.

[0020] In seiner ersten Ausführung ist das erfindungsgemäße Verfahren ein Verfahren zum Auffangen von Netzwerkpaketen in einem Computersystem, wobei Netzwerkpakete zwischen einem ersten Netzwerkadapter und einer ersten Protokolleinrichtung kommuniziert werden, von denen der Netzwerkadapter eine bestimmte Netzwerk-Schnittstelle implementiert, und weist die Schritte auf:

– Bereitstellen eines Satzes von Austauschfunktionen innerhalb eines Paketauffangmoduls;

– Einhängen wenigstens einer ersten Funktion, die zum Übertragen von Netzwerkpaketen von der ersten Protokolleinrichtung zu dem ersten Netzwerkadapter verwendet wird, in eine erste Austauschfunktion;

– Einhängen wenigstens einer zweiten Funktion, die zum Übertragen von Netzwerkpaketen von dem ersten Netzwerkadapter zu der ersten Protokolleinrichtung verwendet wird, in eine zweite Austauschfunktion; und

– Einhängen wenigstens einer dritten Funktion, die zum Empfangen von Information über den Zustand der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle verwendet wird, in eine dritte Austauschfunktion;

– wobei Einhängen definiert ist als Umleiten von Funktionsaufrufen an die erste, die zweite und die dritte Funktion auf entsprechende Austauschfunktionen, so dass die Austauschfunktionen durch die umgeleiteten Funktionsaufrufe aufgerufen werden.

[0021] In einer zweiten Ausführung ist das erfindungsgemäße Verfahren ein Verfahren zum Auffangen von Netzwerkpaketen in einem Computersystem, wo eine Anzahl von Funktionen verwendet werden, um Netzwerkpakete zwischen einer Anzahl von Netzwerkadaptern und einer Anzahl von Protokolleinrichtungen zu kommunizieren, von denen die Netzwerkadapter bestimmte Netzwerk-Schnittstellen implementieren, und weist die Schritte auf:

– Bereitstellen eines Satzes von Austauschfunktionen innerhalb eines Paketauffängermoduls;

– Einhängen einer ersten Anzahl von Funktionen, die zum Übertragen von Netzwerkpaketen von Protokolleinrichtungen zu Netzwerkadaptern verwendet werden, in einen ersten Satz von Austauschfunktionen;

– Einhängen einer zweiten Anzahl von Funktionen, die zum Übertragen von Netzwerkpaketen von Netzwerkadaptern zu Protokolleinrichtungen verwendet werden, in einen zweiten Satz von Austauschfunktionen; und

– Einhängen einer dritte Anzahl von Funktionen, die zum Empfangen von Information über den Zustand der von Netzwerkadaptern implementierten Netzwerk-Schnittstellen verwendet werden, in einen dritten Satz Austauschfunktionen;

– wobei Einhängen definiert ist als Umleiten von Funktionsaufrufen an die erste, die zweite und die dritte Anzahl von Funktionen auf entsprechende Austauschfunktionen, so dass die Austauschfunktionen durch die umgeleiteten Funktionsaufrufe aufgerufen werden.

[0022] In einer dritten Ausführung ist das erfindungsgemäße Verfahren ein Verfahren zum Auffangen von Netzwerkpaketen in einem Computersystem, wo ein bestimmtes erstes Betriebssystemmodul verwendet wird, um Netzwerkfunktionalität zu implementieren, und das erste Betriebssystemmodul eine bestimmte Programmierschnittstelle mit einer Anzahl von Eintrittspunkten implementiert, und weist die Schritte auf:

– Austauschen des ersten Betriebssystemmoduls mit einem bestimmten ersten Austauschmodul, das eine Programmierschnittstelle gleich der Programmierschnittstelle des ersten Betriebssystemmoduls implementiert und das erste Betriebssystemmodul von einer Anzahl der Eintrittspunkte der Programmierschnittstelle ruft;

– Verwenden des Austauschmoduls, um wenigstens einen Netzwerkadapter und wenigstens eine Protokolleinrichtung zu identifizieren, die in dem Computersystem installiert sind;

– Verwenden des Austauschmoduls, um wenigstens eine erste Funktion zu ersetzen, die zum Übertragen von Netzwerkpaketen von der ersten Protokolleinrichtung zu dem Netzwerkadapter verwendet wird;

– Verwenden des Austauschmoduls, um wenigstens eine zweite Funktion zu ersetzen, die zum Übertragen von Netzwerkpaketen von dem ersten Netzwerkadapter zu der ersten Protokolleinrichtung verwendet wird;

– Verwenden des Austauschmoduls, um wenigstens eine dritte Funtion zu ersetzen, die zum Empfangen von Information über den Zustand der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle verwendet wird;

– Verwenden des Austauschmoduls, um zu bestimmen, ob der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle eine dynamische IP-Adresse zugeordnet worden ist oder nicht; und

– in einem Fall, wo der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle eine dynamische IP-Adresse zugeordnet worden ist, Verwenden des Austauschmoduls, um zu bestimmen, was diese dynamische IP-Adresse ist.

[0023] Weiterhin betrifft die Erfindung ein Computersystem zum Handhaben von Netzwerkpaketen, mit:
– einem ersten Netzwerkadapter, der dazu eingerichtet ist, eine Netzwerk-Schnittstelle zu implementieren;
– einer ersten Protokolleinrichtung;
– einer Anzahl vorbestimmter Funktionen zum Kommunizieren von Netzwerkpaketen zwischen dem Netzwerkadapter und der Protokolleinrichtung;
– einem Paketauffängermodul zum Bestimmen eines Satzes von Austauschfunktionen;
– innerhalb des Paketauffängermoduls, einer Einrichtung zum Einhängen wenigstens einer ersten Funktion, die zum Übertragen von Netzwerkpaketen von der ersten Protokolleinrichtung zu dem ersten Netzwerkadapter verwendet wird, in eine erste Austauschfunktion;
– innerhalb des Paketauffängermoduls, einer Einrichtung zum Einhängen wenigstens einer zweiten Funktion, die zum Übertragen von Netzwerkpaketen von dem ersten Netzwerkadapter zu der ersten Protokolleinrichtung verwendet wird, in eine zweite Austauschfunktion; und
– innerhalb des Paketauffängermoduls, einer Einrichtung zum Einhängen wenigstens einer dritten Funktion, die zum Empfangen von Information über den Zustand der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle verwendet wird, in eine dritte Austauschfunktion;
– wobei Einhängen definiert ist als Umleiten von Funktionsaufrufen an die erste, die zweite und die dritte Funktion auf entsprechende Austauschfunktionen, so dass die Austauschfunktionen durch die umgeleiteten Funktionsaufrufe aufgerufen werden.

[0024] In einer noch weiteren Ausführung betrifft die Erfindung ein Paketauffangmodul zum Auffangen von Netzwerkpaketen in einem Computersystem, das einen ersten Netzwerkadapter, eine erste Protokolleinrichtung und eine Anzahl von vorbestimmten Funktionen zum Kommunizieren von Netzwerkpaketen zwischen dem Netzwerkadapter und der Protokolleinrichtung aufweist; wobei das Paketauffängermodul aufweist:
– die Definition eines Satzes von Austauschfunktionen;
– eine Einrichtung zum Einhängen wenigstens einer ersten Funktion, die zum Übertragen von Netzwerkpaketen von der ersten Protokolleinrichtung zu dem ersten Netzwerkadapter verwendet wird, in eine erste Austauschfunktion;
– eine Einrichtung zum Einhängen wenigstens einer zweiten Funktion, die zum Übertragen von Netzwerkpaketen von dem ersten Netzwerkadapter zu der ersten Protokolleinrichtung verwendet wird, in eine zweite Austauschfunktion; und
– eine Einrichtung zum Einhängen wenigstens einer dritten Funktion, die zum Empfangen von Information über den Zustand der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle verwendet wird, in eine dritte Austauschfunktion;
– wobei Einhängen definiert ist als Umleiten von Funktionsaufrufen an die erste, die zweite und die dritte Funktion auf entsprechende Austauschfunktionen, so dass die Austauschfunktionen durch die umgeleiteten Funktionsaufrufe aufgerufen werden.

Kurzbeschreibung der Zeichnungen

[0025] Die neuen Merkmale, die als charakteristisch für die Erfindung angesehen werden, sind im einzelnen in den beigefügten Ansprüchen enthalten. Die Erfindung selbst jedoch, ihre Konstruktion und das Verfahren sowie die Ziele und Vorteile der Erfindung werden am besten aus der folgenden Beschreibung der speziellen Ausführungen in Verbindung mit den begleitenden Zeichnungen hervorgehen.

[0026] Fig. 1 zeigt das bekannte Konzept von Zwischentreibern,

[0027] Fig. 2a, Fig. 2b und Fig. 2c vergleichen schematisch eine erste Ausführung der Erfindung mit dem Stand der Technik,

[0028] Fig. 3 zeigt in allgemeiner Form die erforderlichen Schritte für das Anwenden der ersten Ausführung der Erfindung,

[0029] Fig. 4 zeigt das Registrieren eines Protokolls,

[0030] Fig. 5 zeigt einige Einzelheiten eines Funktionsaufrufes,

[0031] Fig. 6 zeigt die Handhabung der Einzelheiten von Fig. 5,

[0032] Fig. 7 zeigt die Verbindung eines Netzwerkadapters,

[0033] Fig. 8 zeigt das Handhaben eines abgehenden Paketes,

[0034] Fig. 9a zeigt einen bekannten Weg der Handhabung eines ankommenden Paketes,

[0035] Fig. 9b zeigt das Handhabungsprinzip für ein ankommendes Paket gemäß einer Ausführung der Erfindung,

[0036] Fig. 9c zeigt eine detaillierte Handhabungsweise eines ankommenden Paketes gemäß einer Ausführung der Erfindung,

[0037] Fig. 10 zeigt das Prinzip einer zweiten Ausführung der Erfindung,

[0038] Fig. 11 zeigt allgemein die erforderlichen Schritte für das Anwenden der zweiten Ausführung der Erfindung,

[0039] Fig. 12a und Fig. 12b zeigen die Verwendung der zweiten Ausführung der Erfindung,

[0040] Fig. 13 zeigt das Entladen eines Moduls gemäß der zweiten Ausführung der Erfindung und

[0041] Fig. 14 zeigt eine Hardware-Anordnung gemäß der Erfindung.

Ausführliche Beschreibung der Erfindung

[0042] Das Konzept des Einhängens ist in dem Fachgebiet der Computerprogrammierung allgemein bekannt. Es beinhaltet, daß ein Aufruf eines Standardsystemservice (wie zum Beispiel eine Funktion, eine Unterbrechung oder ein Speicherplatz) umgeleitet wird oder statt dessen in einen Austauschservice ”eingehängt wird. So stellt zum Beispiel das MSDOS(MicroSoft Disk Operation System)-Betriebssystem einen Systemaufruf zur Verfügung, um Unterbrechungen umzuleiten, wie es zum Beispiel in dem mit MSDOS5 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist. Viele objektorientierte Programmiersprachen stellen einen Weg zum Neudefinieren von Funktionen in abgeleiteten Klassen zur Verfügung (zum Beispiel virtuelle Funktionen von C++, siehe den mit ES90 gekennzeichneten Literaturhinweis). Lisp-ähnliche Programmiersprachen haben Einhängungen für bestimmte Operationen lange Zeit unterstützt, wie es zum Beispiel in dem mit Steele90 gekennzeichneten Literaturhinweis beschrieben ist.

[0043] Viele Betriebssysteme stellen Einhängungen oder bestimmte andere Typen des Umleitens mit einer begrenzten Funktionalität zur Verfügung. So werden zum Beispiel die bekannten Firewall-Einhängungen in Linux und FreeBSD normalerweise so genutzt, daß ein Codemodul in das Kernprogramm geladen wird. Der Modul registriert sich selbst in bezug auf den Netzwerkcode und der Netzwerkcode ruft den Modul immer dann auf, wenn ein Paket abgesendet oder empfangen wird. Der Code in dem Modul zeigt dann an, ob das Paket durchgelassen werden soll oder nicht. Normalerweise gestatten jedoch solche Einhängungen nicht das Modifizieren, Einfügen und Verzögern der Pakete.

[0044] Die Firewall-Einhängungsmethode steht nur für einen sehr eingeschränkten Bereich von Betriebssystemen, wie zum Beispiel Linux und FreeBSD, zur Verfügung. Weiterhin kann bei vielen Betriebssystemen die Funktionalität nicht durch Standardvorgaben aktiviert werden. Das Verwenden der Firewall-Einhängungs-Merkmale kann es daher für den Anwender erforderlich machen, neu zu kompilieren und ein neues Betriebssystem-Kernprogramm zu installieren. Das übersteigt die Fähigkeiten der meisten Systemadministratoren. Daher sind die bekannten Firewall-Einhängungs-Lösungen nicht allgemein nutzbar und lösen nicht das Problem, ein Paketauffangen für die bekanntesten Betriebssysteme oder für die große Anzahl der unerfahrenen Anwender bereitzustellen.

[0045] Spezialisierte Anwendungen, wie zum Beispiel Anti-Virus-Software, haben lokalisierte Systemdienste, um zum Beispiel die Dateisystemaktivität zu überwachen. Einige Betriebssysteme bieten eine spezielle Unterstützung für solche Anwendungen. So bietet zum Beispiel das Betriebssystem Windows 95 eine VMM.VxD-Hook_Device_Service-Funktion (Einhängbaustein-Servicefunktion), um Gerätedienste für solche Anwendungen einzuhängen, wie es zum Beispiel in dem mit Win95DDK gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist.

[0046] Ein anderes bekanntes Verfahren für die Überwachung des Netzwerkverkehrs ist, die bekannte WINSOCK.DLL-Datei in Windows zu ersetzen. Diese Datei erkennt keine einzelnen Datenpakete, sondern

erhält statt dessen Zugriff zu jedem Windows-SOCKET in dem System. Es ist möglich, die Originaldatei beiseite zu legen und sie durch eine neue DLL (Dynamic Loadable Library – dynamisch ladbare Bibliothek) zu ersetzen, so daß die neue DLL die ursprüngliche aufruft. In bekannten Systemen ist das Ersetzen während der Zeit des Installierens durchgeführt worden und der Austauschmodul wurde geladen, wenn eine Anwendung, die sie nutzt, startet (jedoch nicht, zum Beispiel wenn das System startet). Obwohl konzeptionell etwas mit dem Einhängen verwandt, ist dieses Verfahren nicht für das Auffangen von Paketen anwendbar, weil kein Zugriff auf die Pakete erhalten wird.

[0047] Die reine Basisidee, das Einhängen zu verwenden, um Zugriff zu den Netzwerkpaketen zu erhalten, ist bekannt, wie es zum Beispiel in den mit Lanciani98, Lanciani98PPPMAC und Lanciani98Reply gekennzeichneten Literaturhinweisen in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist. Sie ist in bestimmten Protokollkonvertierungsanwendungen verwendet worden. Der Stand der Technik auf diesem Gebiet besteht aus einem Programm, welches die Pakettreiberprogrammierungs-Schnittstelle an erster Stelle der NDIS-Programmierungs-Schnittstelle implementiert, wie es zum Beispiel in dem mit Lanciani97NDIS3PKT gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist. Andere bekannte Alternativen sind die Implementierung eines ODI-Pakettreibers über NDIS (Literaturhinweis Lanciani96ODIPKT3.1), einer älteren Pakettreiber-Schnittstelle (Literaturhinweis Lanciani92DIS_PKT), eines Banyan Vines Pakettreibers (Literaturhinweis Lanciani94BANPKT) und eines NFS-Clienten für Windows (Literaturhinweis Lanciani97NFSTDI). Ein Protokollkonvertierungs-Pakettreiber oder ein alternativer Pakettreiber kann nicht die komplizierten Netzwerksicherheits-Anwendungen unterstützen, da sie unerwartete und unzulässige Änderungen in der Form der Pakete verursachen können.

[0048] Die vorliegende Erfindung wendet im allgemeinen die Idee des Einhängens für das Auffangen sowohl ankommender als auch ausgehender Pakete und für das Erhalten und die Modifizierung der erforderlichen Information über das System an. Die Hauptanwendung der Erfindung ist ein vollständiges Paketauffangverfahren, welches das Einhängen in einer hochentwickelten Form verwendet, um Zugriff zu allen notwendigen Paketen und Informationen zu erhalten.

[0049] Die Verwendung des Einhängens gemäß der Erfindung ist eine neue Methode des Herangehens an das Problem, das andere Leute zur Zeit durch das Schreiben von Zwischentreibern zu lösen versuchen. Das neue Verfahren vermeidet vollständig die Notwendigkeit, einen Zwischentreiber zu schreiben oder Protokolle zu verwenden, die den vorhandenen technischen Urheberschutz in dem Betriebssystem verletzen. Der Umfang des Codes, der erforderlich ist, um das neue Verfahren zu implementieren, beträgt nur etwa die Hälfte von dem Umfang, der für einen Zwischentreiber erforderlich ist. Weiterhin kann die Herangehensweise gemäß der Erfindung oft verwendet werden, wenn überhaupt keine Unterstützung für das Auffangen in dem Hauptcomputersystem vorhanden ist. Das neue Verfahren ist einfacher, robuster und weist eine größere Leistungsfähigkeit als Zwischentreiber auf. Es bietet auch ein zuverlässigeres Zusammenarbeiten und Kommunizieren mit verschiedenen Softwarepaketen und Treibern Dritter.

[0050] Fig. 2a zeigt schematisch eine Anordnung einer dem Stand der Technik entsprechenden Netzwerk-Schnittstelle ohne eine Paketauffangeinrichtung. Ein Protokollblock **201** ist durch einen Schnittstellen-Funktionsblock **202** komplementiert, der wiederum eine Standardschnittstelle für einen oder mehrere Netzwerkadapter **203** darstellt. In Fig. 2b ist eine dem Stand der Technik entsprechende Paketauffangeinrichtung als ein Zwischentreiber **204** implementiert, so daß sich ein erster Schnittstellenfunktionsblock **202a** zwischen ihm und dem Protokollblock **201** und ein zweiter Schnittstellenfunktionsblock **202b** zwischen ihm und dem Netzwerkadapter oder den Netzwerkadaptern **203** befindet. Die Schnittstellenfunktionen beziehen sich auf die NDIS-Schnittstelle in einer Microsoft Windows-Umgebung und auf die STREAMS-Schnittstelle in einer Sun Solaris-Umgebung.

[0051] In Fig. 2c ist eine vereinfachte Anordnung gemäß der vorliegenden Erfindung dargestellt. Der Paketauffangmodul **210** weist einen Satz von Austauschfunktionen **211** auf, welcher dazu ausgestaltet ist, um zumindest teilweise die ursprünglichen Schnittstellenfunktionen **202** zu ersetzen. Zusätzlich weist der Paketauffangmodul **210** obere Einhängeinrichtungen **212** und untere Einhängeinrichtungen **213** auf, um die entsprechende Kommunikation, die auf die ursprünglichen Schnittstellenfunktionen **202** ausgerichtet war, zu den Austauschfunktionen in dem Block **211** umzuleiten und um die Pakete, Aufrufe und anderen Formen von Kommunikation in den ursprünglichen Signalweg nach der Verarbeitung zurückzuführen. Noch vorteilhafter ist es, wenn der Paketauffangmodul auch Einrichtungen für das Austauschen von Information zwischen den ursprünglichen Schnittstellenfunktionen und ihren Austauschfunktionen aufweist, wie es durch den Pfeil **214** dargestellt ist.

[0052] Die nachfolgende Beschreibung beginnt mit der Erläuterung einer ersten bevorzugten Ausführung der Erfindung in den Windows **95** und Windows 98-Umgebungen unter Bezugnahme auf Fig. 3 bis **Fig.** 9. In dieser Ausführung kann die Auffangeinrichtung verwendet werden, sowohl die unter diesen Betriebssystemen empfangenen als auch die übertragenen Pakete zu überprüfen und zu modifizieren. Der Paketauffangmodul (der auch den Code enthält, um die Pakete tatsächlich zu überprüfen und zu modifizieren) ist in einen VxD-Geräte-Treibermodul eingebaut, dessen formale Anforderungen allgemein bekannt sind.

[0053] Der "Treiber", der in Wirklichkeit ein Auffangmodul ist, ist in dem System vorzugsweise als ein statischer Gerätetreiber installiert, der sich hinter dem NDIS-Modul, jedoch in der Ladefolge vor den Netzwerkadaptern befindet. Die folgenden Schritte erläutern, wie der Modul das Verfahren in der vorliegenden Erfindung implementiert. Es ist zu bemerken, daß die hierin beschriebenen Schritte nicht in dem Sinne einschränkend sind, daß sie nicht umgeordnet oder in vielfältiger Art und Weise miteinander gemischt sein können (wenn es nicht anders festgelegt ist), wobei sie auch dann innerhalb des Schutzumfanges der Erfindung verbleiben.

[0054] Wir nehmen an, daß ein Computersystem mit einem Windows 95- oder Windows 98-Betriebssystem oder mit einem anderen Betriebssystem arbeitet, das auf eine Standardschnittstelle und eine jeweilige Sammlung von Funktionen und Handlern (Hilfsprogrammen) zwischen Protokollstapeln und Netzwerkadaptern bezogen ist. Wir nehmen weiterhin an, daß ein NDIS-Modul oder eine entsprechende Einrichtung, die eine Standardschnittstelle zwischen den Protokollstapeln und den Netzwerkadaptern definiert, in einigen vorhergehenden Schritten des Initialisierens oder des Betreibens des Computersystem geladen wurde. Das Laden des NDIS-Moduls ist in Fig. 3 als Schritt **301** dargestellt.

[0055] In Schritt **302** wird zuerst ein Paketauffangmodul, der die Form eines Gerätetreibers aufweist, initialisiert, vorzugsweise unter Verwendung der sogenannten SYS_Critical_Init-Funktion des Moduls. Der Modul leitet die Aufrufe bestimmter Standardfunktionen an der vorher angeführten Schnittstelle zu seinen eigenen Funktionen um (hängt sie ein). In Fig. 4 sind die einzuhängenden Funktionen als eine Absendefunktion **403**, ein Protokollregistrierfunktion **404**, eine Adapteröffnungsfunktion **405**, eine Anforderungsfunktion **406**, eine Zustandsanzeigefunktion **407** und eine Datenübertragungsfunktion **408** dargestellt. Angenommen, daß die Schnittstelle NDIS ist, sind diese Funktionen die bekannten NdisSend-, NdisRegisterProtocol-, NdisOpenAdapter-, NdisRequest-NdisIndicateStatus bzw. NdisTransferData-Funktionen. Weiterhin sichert in Schritt **302** der Paketauffangmodul die ursprünglichen Adressen der Funktionen **403** bis **408**. Das Windows 95-Betriebssystem stellt eine generische Hook_Device_Service(Einhängbaustein-Service)-Operation in Assemblersprache zur Verfügung, die verwendet werden kann, um das Einhängen bequem auszuführen. Der folgende Mustercode erläutert ihre Anwendung.

```
; Hook NDISRegisterProtocol
GetVxDServiceORDINAL eax, NDISRegisterProtocol
Mov esi, OFFSET32_ssh_ndisregisterprotocol_handler@16
VMMCallHook_Device_Sercice
jc not_hooked

mov [_ssh_orig_ndisregisterprotocol_fn], esi
```

[0056] Die Funktion des Codes sollte für jeden verständlich sein, der im Schreiben von Windows **95** VxD-Treibern in Assemblersprache erfahren ist. Der mit WIN95DDK gekennzeichnete Literaturhinweis in der beigefügten Liste der Literaturhinweise erläutert die aufgerufenen Systemdienste.

[0057] Für das korrekte Funktionieren beim Einhängen sollte der Paketauffangmodul **412** nach dem NDIS-Modul (Schritt **301**, jedoch vor den Netzwerkadaptern und Protokollen (Schritt **303** und **304**), geladen werden. Der Paketauffangmodul **412** kann zum Beispiel in Form eines statischen VxD-Treibers mit geeigneter Ladefolge, konfiguriert in einer INI-Datei oder Registratur (registry), vorliegen.

[0058] Nach dem Schritt **302** startet das System das Laden der Netzwerkprotokolle und der Netzwerkadapter in den Schritten **303** und **304**. Jedes in Schritt **303** zu ladende Protokoll sollte sich durch Aufrufen der entsprechenden Protokollregistrierfunktion, die im NDIS-Rahmenwerk die NdisRegisterProtocol-Funktion ist, selbst registrieren. In Fig. 4 ist das Protokoll zum Ausführen der Registrierung als **401** dargestellt. Weil jedoch das Einhängen dieser und anderer relevanter Funktionen in Schritt **302** ausgeführt wurde, wird der Aufruf **420**, der

ursprünglich auf die Funktion **404** zielte, zu der entsprechenden Austauschfunktion **414** umgeleitet, was das Ziel des Einhängens war.

[0059] Fig. 5 zeigt schematisch, wie die Austausch-Implementierung **414** der ursprünglichen Protokollregistrierfunktion, als Bestandteil des Aufrufes **420**, eine bestimmte Datenstruktur **501** empfängt, welche die relevanten Protokollmerkmale des Aufrufprotokolls beschreibt. In dem NDIS-Rahmenwerk ist das die NDIS_PROTOCOL_CHARACTERISTICS-Struktur. Die Funktion **414** findet aus dieser Struktur vorzugsweise eine Anzahl von Verweisen auf einige weitere Bezugsfunktionen **502** auf. Wiederum angenommen, daß es sich um NDIS handelt, sind das die ReceiveHandler-, ReceiveCompleteHandler-, RequestCompleteHandler-, TransferDataCompleteHandler- und die SendCompleteHandler-Funktionen. Fig. 6 zeigt schematisch, wie die Austauschfunktion **414** die ursprünglichen Adressen der in der Struktur **502** angeführten Funktionen in ihren eigenen Datenstrukturen sichert und die Funktionsverweise in der Struktur **502** durch Verweise ersetzt, die auf die entsprechenden Austauschfunktionen hinweisen. Der exakte Satz der zu ersetzenden Funktionen hängt vom Betriebssystem ab und innerhalb des NDIS-Rahmenwerkes selbst von der NDIS-Version. Die beispielhaft in der Struktur **502** aufgezeigten Funktionen sind nur einige der von NDIS geforderten Funktionen.

[0060] Fig. 4 zeigt weiterhin, wie die ursprüngliche NdisRegisterProtocol-Funktion dann aufgerufen wird, um die Registrierung fortzusetzen. Der Aufruf **420'** unterscheidet sich von dem ursprünglichen Aufruf **420**, der durch das Registrierprotokoll abgesendet wird, dadurch, daß die Verweise auf die Bezugsfunktionen, die in der Struktur **502** in Fig. 5 dargestellt sind, durch Verweise auf die entsprechenden Bezugs-Austauschfunktionen ersetzt sind. Wenn die Registrierung erfolgreich ist, wird das zurückgesendete Protokoll-Handlerprogramm ebenfalls in den Datenstrukturen der Austauschfunktion **414** gesichert.

[0061] Im Ergebnis dieser Schritte ist das Protokoll **401** ansonsten normal im System registrieret. Die Funktionen, die aufgerufen werden, wenn zum Beispiel Pakete von Netzwerkadaptern empfangen werden, verweisen jedoch in Wirklichkeit auf die Austauschfunktionen.

[0062] Nachdem ein Protokoll registriert worden ist, können ein Netzwerkadapter oder mehrere Netzwerkadapter damit verbunden werden. Das wird durch das System gesteuert und ist schematisch in Fig. 7 dargestellt. Immer wenn eine solche Verbindung erfolgt, ruft das Protokoll **701**, mit dem der Netzwerkadapter **702** verbunden werden soll, die Adapteröffnungsfunktion **405** (In NDIS: die Ndis-OpenAdapter-Funktion) auf. Dieser Aufruf **720** wird eingehängt und eine Austauschfunktion **415** wird statt dessen aufgerufen. Die Austauschfunktion **415** ruft die ursprüngliche Funktion **405** auf und sichert die Information über das Verbinden mit ihren eigenen Datenstrukturen (in Fig. 7 nicht gesondert dargestellt). Die Austauschfunktion **415** kann von dem Adapter **702** Informationen anfordern, wie zum Beispiel seinen Typ oder die Verbindungsschichtadresse. Sie kann auch bestimmen, ob der Adapter ein Wähladapter ist. Ein Weg, um das zu bestimmen ist, die Ethernet-Adresse mit 44:45:53:54:00:00 zu vergleichen, die von Windows für Wähladapter verwendet wird. Windows-Wähladapter sehen ansonsten wie Ethernet-Adapater (IEEE 802.3) aus. Im allgemeinen sind die Adapteröffnungsfunktionen (in NDIS: NdisOpenAdapter) ein Weg, auf dem die Auffangeinrichtung **412** Informationen über die zur Verfügung stehenden Netzwerk-Schnittstellen erhält.

[0063] Die Austauschfunktion **416** kann dazu ausgestaltet sein, um Anforderungsaufrufe und Antworten aufzufangen (in NDIS: NdisRequest call and replies) und bestimmte Anforderungen zu erkennen. So kann sie zum Beispiel MTU-Werte (Maximum Transmission Unit – Maximale Übertragungs-Einheit; beschrieben zum Beispiel in dem mit Postel8l gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise), die von dem Adapter zurückgeleitet werden, bevor sie zu dem Protokoll durchgelaufen sind, einstellen, so daß AH-(Acceptor Handshake) und ESP-Header (Vorsignale), wie sie zum Beispiel in dem mit KA98 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben sind, dem Paket hinzugefügt werden können, ohne das Paket zu fragmentieren, bevor es zu der Verbindungsschicht durchgelaufen ist.

[0064] Die Datenübertragung über einen bestimmten Netzwerkadapter kann beginnen, nachdem die entsprechende Adapteröffnungsfunktion (in NDIS: NdisOpenAdapter) abgeschlossen ist.

[0065] Fig. 8 zeigt, wie ein abgehendes Paket von einem Protokoll **801** in die Austauschabsendefunktion **413** weitergeleitet wird, die in NDIS die NdisSend-Funktion der Paketauffangeinrichtung ist. Das Paket wird in Obhut der Auffangeinrichtung genommen und die Auffangeinrichtung ruft die ursprüngliche Absendeabschluß-Handlerfunktion **803** auf (in NDIS: SendCompleteHandler), um dem Protokoll **801** mitzuteilen, daß das Paket **802** verarbeitet worden ist. Alternativ kann die Auffangeinrichtung den Aufruf **821** zu der ursprünglichen Ab-

sendeabschluß-Handlerfunktion **803** zurückstellen, bis das Paket vollständig verarbeitet ist. Die tatsächliche Bearbeitung des Paketes kann zu dieser Zeit erfolgen.

[0066] Die Auffangeinrichtung sendet die ausgehenden Pakete durch Aufrufen der ursprünglichen Absendefunktion **403** (in NDIS: Ndis-Send) gemäß dem Pfeil **820'** zu den Netzwerkadaptern. Ein Stern bedeutet in Fig. 8, daß das Paket die Auffangeinrichtung durchlaufen hat. Wenn der Aufruf vollständig abgeschlossen ist, kann das Paket freigegeben werden. Ansonsten ruft der Adapter schließlich die Austausch-Absendeabschluss-Handlerfunktion **813** auf (in NDIS: SendCompleteHandler) (siehe Pfeil **822**), welche das Paket freigibt.

[0067] Ein Vergleich von Fig. 9a und Fig. 9b zeigt gewisse allgemeine Unterschiede zwischen dem Stand der Technik und der Erfindung insbesondere in der Windows NT-Umgebung. In der dem Stand der Technik entsprechenden Anordnung von Fig. 9a sendet der NIC-Treiber eine ProtocolReceive-Nachricht (Protokollempfangsnachricht) **951** zu der Protokolleinrichtung, welche mit einem MiniportTransferData-Befehl **952** (Miniport-Datenübertragungsbefehl) antwortet. Die Operation endet mit einer ProtocolTransferDataComplete-Nachricht **953** (Nachricht über den Abschluss des Übertragens der Protokolldaten) von dem NIC-Treiber. In Fig. 9b, die einen Aspekt der Erfindung erläutert, geht die anfängliche ProtocolReceive-Nachricht **951** von dem NIC-Treiber nicht zu der Protokolleinrichtung, sondern zu dem Paketauffangmodul, der nun mit dem MiniportTransferData-Befehl **962** antwortet. Die ProtocolTransferDataComplete-Nachricht **953'** von dem NIC-Treiber ist optional und das tatsächliche Verarbeiten des aufgefangenen Paketes erfolgt in dem Paketauffangmodul in Schritt **963**.

[0068] Um das verarbeitete Paket zu der Protokolleinrichtung weiterzuleiten, überträgt der Paketauffangmodul eine neue ProtocolReceive-Nachricht **951"** zu der Protokolleinrichtung, welche nun dem Paketauffangmodul (und nicht dem NIC-Treiber) mit seinem MiniportTransferData-Befehl **952'** antwortet. Eine ProtocolReceive-Complete-Nachricht **964**, die von dem NIC-Treiber kommt, wird im Schritt **965** von dem Paketauffangmodul zu der Protokolleinrichtung geleitet.

[0069] Eine etwas detaillierte Ansicht der Behandlung eines ankommenden Paketes ist in Fig. 9c dargestellt. Die Auffangeinrichtung empfängt die ankommenden Pakete von den Netzwerkadaptern **803** in einem Aufruf **901** an seine Austausch-Handlerempfangsfunktion (in NDIS: ReceiveHandler) **913**, wie es in Fig. 9c dargestellt ist. Die Auffangeinrichtung überträgt durch Aufrufen der ursprünglichen Datenübertragungsfunktion **408** (in NDIS: NdisTransferData) die Daten zu ihrem Speicher. Wenn das abgeschlossen ist, wird das Paket in dem Verarbeitungsspeicher **902** sofort den empfangenen Paketen zugefügt. Wenn das nicht sofort abgeschlossen wird, wird die Datenübertragungsabschluß-Handler-Funktion **903** (in NDIS: TransferDataComplete Handler) durch den Adapter **803** aufgerufen, wenn die Übertragung abgeschlossen ist, wie es durch den gestrichelten Pfeil **904** in Fig. 9c dargestellt ist und das Paket wird im Speicher **902** den zu diesem Zeitpunkt empfangenen Paketen zugefügt. Die empfangenen Pakete können entweder sofort verarbeitet werden, oder sie können in eine Warteschlange eingeordnet und asynchron verarbeitet werden. Die Austausch-Empfangsabschluß-Handlerfunktion **905** (in NDIS: ReceiveComplete-Handler) wird aufgerufen, nachdem einige Pakete empfangen worden sind und diese Funktion ist eine mögliche Stelle für die Echtverarbeitung der in die Warteschlange eingeordneten Pakete.

[0070] Die Auffangeinrichtung sendet die ankommenden Pakete zu den Protokollen, indem die ursprüngliche Empfangshandlerfunktion **906** (in NDIS: Receive Handler) aufgerufen wird. Das Protokoll kann die Austausch-Datenübertragungsfunktion (in NDIS: NdisTransferData-Funktion) während dieses Aufrufes aufrufen.

[0071] Der Austausch kopiert die Daten zu dem Protokoll. Wenn der Austausch immer vollständig abschließt, ist es möglich, das Paket freizugeben, nachdem die Empfangshandlerfunktion **906** zurückgeleitet wurde. Nachdem ein Paket oder mehrere Pakete zu der Empfangshandlerfunktion **906** gelangt sind, wird die ursprüngliche Empfangsabschluß-Handlerfunktion (in NDIS: ReceiveCompleteHandler) durch die Austausch-ReceiveCompleteHandler-Funktion aufgerufen.

[0072] In jedem Falle verwendet die Auffangeinrichtung Informationen, die über die Verknüpfungen in den Aufrufen der Adapteröffnungs-Austauschfunktion (in NDIS: NdisOpenAdapter), die als 415 in Fig. 4 enthalten ist, gesichert wurden, um die Pakete mit einem speziellen Netzwerkadapter und mit dem Protokoll zu verbinden. Das kann zum Beispiel durch Verwendung der Kontextargumente für die verschiedenen Funktionen als Ordnungsbegriffe (Schlüssel) zu den Prüftabellen (hash tables) erfolgen.

[0073] Für Ethernet-Adapter sind die ersten Pakete bereits normaler Datenverkehr. Für Wähladapter sind jedoch die ersten Pakete Verbindungssteuerungsverkehr, wie zum Beispiel die PPP LCP-Pakete (Point-to-Point Protocol Linc Control Protocol – Punkt-zu-Punkt-Protokoll-Verbindungssteuerungs-Protokoll), wie sie zum Bei-

spiel in dem mit Simpson94 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise enthalten sind, und IPCP-Pakete (Internet Protocol Control Protocol – Steuerungsprotokoll für Internetprotokoll), wie sie zum Beispiel in dem mit McGregor92 gekennzeichneten Literaturhinweis beschrieben sind. Die Auffangeinrichtung kann die Protokolle unter Verwendung des Ethertyp-Feldes, das in dem Ethernet-Header in den Paketen enthalten ist, differenzieren.

[0074] Für Wählverbindungen ruft der Adapter, wenn die Verbindungsebenen-Vermittlung abgeschlossen ist, die Zustandsanzeige-Funktion auf (in NDIS: NdisIndicateStatus), um dem Protokoll mitzuteilen, daß die Verbindung nun zur Verfügung steht. Es wird die unter **417** in Fig. 4 angeführte Austauschversion verwendet.

[0075] Während der Verarbeitung (zum Beispiel in Austausch-NdisOpen-Adapter, in NdisIndicateStatus oder an einem anderen geeigneten Punkt während der Operation) sucht die Auffangeinrichtung die Protokolladresse (z. B. die IP-Adresse) für den Adapter aus dem Register (registry). Die Adresse steht nur zur Verfügung, wenn sie statisch konfiguriert ist. Wenn die Adresse nicht statisch konfiguriert ist, kann jedes aus einer Anzahl von Verfahren verwendet werden, um die dynamische Adresse zu bestimmen. Diese beinhalten zum Beispiel das Analysieren von Verbindungsschicht-Paketen, die zum Protokoll durchgelaufen sind und das Extrahieren der Adresse aus dem Verbindungsschicht-Paket (z. B. PPP ICCP-Paket, wie es zum Beispiel in dem mit McGregor 92 gekennzeichneten Literaturhinweis in der hierin enthaltenen Liste der Literaturhinweise beschrieben ist). Alternativ kann die Adresse aus dem ersten ARP-Paket, das von dem Protokoll abgesendet wird, extrahiert werden, wie es zum Beispiel in dem mit Plummer82 gekennzeichneten Literaturhinweis beschrieben ist. Eine weitere Alternative ist, die Adresse aus einem DHCP-Paket zu extrahieren, wie es zum Beispiel in dem mit Droms97 gekennzeichneten Literaturhinweis beschrieben ist. In jedem Falle enthält ein geeignetes Feld oder eine Nutzlast (payload), wie sie in den zutreffenden Standards dokumentiert ist, die Adresse. Externe DHCP-Clients (getrennt von dem IP-Stapel) können auch das Register dynamisch aktualisieren, indem sie die neu erhaltene IP-Adresse der TCP/IP-Konfiguration zuschreiben. Somit könnte die Auffangeinrichtung auch das Register auf Veränderungen in solcher Information abfragen.

[0076] Während der Verarbeitung erkennt die Auffangeinrichtung einen neuen Adapter (z. B. Wählverbindung), der verfügbar wird, wenn die Austausch-Zustandsanzeigefunktion (in NDIS: NdisIndicate-Status) mit einer entsprechenden Anzeige aufgerufen wird, die in NDIS die bekannte NDIS_STATUS_WAN_LINE-UP-Anzeige ist. Entsprechend wird durch dieselbe Funktion, die mit einer anderen Anzeige aufgerufen wird, angezeigt, wenn die Verbindung zu einer nichtverfügbaren Verbindung wird. Diese Anzeige ist in NDIS die bekannte NDIS_STATUS_WAN_LINE_DOWN-Anzeige.

[0077] Der Ausdruck Verbinden wird in der vorhergehenden Beschreibung in der Windows NDIS-Bedeutung verwendet. Eine Verbindung ist eine Verbindung zwischen einem Netzwerkadapter und einem Protokoll. Jeder Adapter kann mit null oder mehr Protokollen verbunden werden und null oder mehr Adapter können mit jedem Protoll verbunden werden. Es ist auch möglich, Module zu verwenden, die sowohl als ein Adapter als auch als ein Protokoll wirken (Zwischentreiber) und somit kann der Verbindungsgraph in Wirklichkeit sehr kompliziert sein. In Windows werden die Verbindungen normalerweise automatisch konfiguriert. Dem Anwender kann sie sich jedoch auch sichtbar machen und sie modifizieren. Die Verbindungen werden intern durch Windows gehandhabt und sie sind normalerweise in dem Register gespeichert. Das entsprechende Konzept ist auch in anderen Betriebssystemen vorhanden, kann jedoch implizit sein. So sind zum Beispiel in FreeBSD alle Adapter implizit mit allen Protokollen verbunden und das entsprechende Protokoll wird aus dem Pakettyp bestimmt. Das Paket wird nur an das Protokoll abgesendet, welches erkennt, wie es zu verarbeiten ist.

[0078] Register (registry) bedeutet in der vorhergehenden Beschreibung das Windows-Register oder seine ältere Alternative, die INI-Dateien. Im allgemeinen ist es eine reine Informationsspeicherung. Das Windows-Register gestattet zumindest eine eingeschränkte Baumstruktur und Namenswert-Zuweisungen. Windows-Konfigurations-Dateien (INI-Dateien) haben eine ähnliche, jedoch eingeschränktere Struktur. Anstelle eines solchen Registers können andere bekannte Formen von Informationsspeicherung verwendet werden, oder es sind offensichtlich sogar solche Formen, die einem Fachmann zur Zeit der Einreichung der vorliegenden Patentanmeldung nicht bekannt sind, jedoch verfügbar geworden sind, für eine ähnliche Art der Informationsspeicherung geeignet.

[0079] Die vorher beschriebene Ausführung der Erfindung war nur ein Beispiel. Eine andere mögliche Ausführung der Erfindung ist das Implementieren eines dynamisch ladbaren Auffangmoduls für IPSEC, Firewall- und andere Anwendungen im FreeBSD-Betriebssystem. Eine ähnliche Verfahrensweise funktioniert z. B. für Linux und NetBSD. Eine solche ladbare Kernprogrammmodul-Auffangeinrichtung folgt dem in Fig. 10 dargestellten Betriebsprinzip. Der Paketauffangmodul **1010** weist einen Satz von Austauschfunktionen **1011** sowie eine

obere Einhängeinrichtung **1012** und eine untere Einhängeinrichtung **1013** auf, wobei jedoch die Anordnung zwischen den Übertragungsrichtungen asymmetrisch ist. In Absenderichtung erfolgt das Einhängen durch die obere Einhängeinrichtung **1012** zwischen dem Protokollblock **201** und dem Block der ursprünglichen Schnittstellenfunktionen **202**. In Empfangsrichtung erfolgt das Einhängen durch die untere Einhängeinrichtung **1012** zwischen dem Netzwerkadapterblock **203** und dem Block der ursprünglichen Schnittstellenfunktionen **202**. In beiden Richtungen werden die Pakete, welche die Auffangeinrichtung durchlaufen haben, zu dem Einhängpunkt in dem Sinne ”zurückgeleitet”, daß die ursprüngliche(n) Schnittstellenfunktion(en) verwendet werden, um die Pakete weiter zu ihrem Zielort zu bringen.

[0080] Der Paketauffangmodul gemäß dieser zweiten Ausführung der Erfindung ist am vorteilhaftesten ein dynamisch ladbarer Kernprogrammmodul. Er wird in den Kernprogrammspeicher wie jeder andere Treiber in einem Schritt vor seiner Inbetriebnahme geladen. Der Ladeschritt ist als Schritt **1101** in Fig. 11 dargestellt.

[0081] Wenn er im Schritt **1102** gestartet ist, sperrt der Modul sofort die Unterbrechungen und modifiziert in Schritt **1103** den Binärcode des Kernprogramms des Betriebssystems, so daß bestimmte Funktionen zu Austauschfunktionen umgeleitet werden, die in dem Modul enthalten sind. So könnten zum Beispiel in FreeBSD die auszutauschenden Funktionen die ip_input-, ip_output- und ifioctl-Funktionen sein. Von diesen empfängt ip_input normalerweise Pakete von dem Netzwerk, ip_output wird aufgerufen, um sie zu dem Netzwerk zu senden und ifioctl wird zum Beispiel aufgerufen, wenn sich der Zustand der Netzwerk-Schnittstelle verändert. Die ursprünglichen Funktionen werden gesichert, so daß sie von den Austauschfunktionen aufgerufen werden können. Die Wege dafür werden nachfolgend beschrieben. Nachdem die Funktionen ausgetauscht wurden, können in Schritt **1104** die Unterbrechungen wieder wirksam gemacht werden. Es kann auch erforderlich sein, in Abhängigkeit von der CPU-Architektur, den Befehlspufferspeicher (Cache) anzupassen, wenn der Code modifiziert wurde.

[0082] Nach den Schritten **1101** bis **1104** von Fig. 11 erfolgt die Datenübertragung gemäß Schritt **1105** unter Verwendung der Austauschfunktionen, die in Schritt **1101** geladen und in Schritt **1103** mit den Einhängungen versehen wurden.

[0083] Fig. 12a und Fig. 12b zeigen, wie unter Verwendung der zweiten Ausführung der Erfindung die Pakete übertragen und empfangen werden. Um die von den Protokollen **1201** abgehenden Pakete zu empfangen, wird die Austauschfunktion **1202** (in FreeBSD: die ip_output-Funktion) in der Paketauffangeinrichtung aufgerufen. Um die zu den Adaptern **1204** abgehenden Pakete abzusenden, ruft die Auffangeinrichtung die ursprüngliche Ausgangsfunktion **1203** auf (ip_output).

[0084] Um die ankommenden Pakete von den Adaptern **1204** zu empfangen, wird die Austausch-Eingangsfunktion **1205** (ip_input) in der Paketauffangeinrichtung aufgerufen. Um die ankommenden Pakete zu den Protokollen **1201** abzusenden, ruft die Auffangeeinrichtung die ursprüngliche Eingangsfunktion **1206** (ip-input) auf.

[0085] In der zweiten Ausführung der Erfindung erhält die Auffangeinrichtung Informationen über aktive Netzwerk-Schnittstellen, indem sie die Gesamtvariable der Netzinformation anspricht oder, in einem anderen, ähnlich funktionierenden Betriebssystem, eine entsprechende andere Variable in dem Kernprogramm. Diese Variable enthält eine Liste der Netzwerk-Schnittstellen und jede Schnittstellen-Datenstruktur enthält die Protokolladressen der Schnittstelle.

[0086] Weiterhin erkennt die Auffangeinrichtung in der zweiten Ausführung der Erfindung neue Netzwerk-Schnittstellen (z. B. Wählleitungen, die aufgebaut oder abgebaut werden), wenn die ifioctl-Funktion aufgerufen wird. Nach dem Aufrufen der ursprünglichen ifioctl-Funktion kann die Auffangeinrichtung die Liste der Netzwerk-Schnittstellen erneut lesen und kann alle Änderungen, die erfolgten, erkennen.

[0087] Wenn das Entladen des Kernprogramm-Moduls gemäß der zweiten Ausführung der Erfindung gestartet ist, wie es in Schritt **1301** von Fig. 13 dargestellt ist, ist es am vorteilhaftesten, die Unterbrechungen unwirksam zu machen und die Funktionen in Schritt **1302** zu ihren Originalen zu speichern, in Schritt **1303** die Unterbrechungen wieder wirksam zu machen und möglicherweise den Befehls-Pufferspeicher anzupassen und im Schritt **1304** das Entladen fortzusetzen, indem die Speicherplätze, die vorher der Paketauffangeinrichtung zugewiesen wurden, erneut entleert werden.

[0088] In einigen Anwendungen kann die Auffangeinrichtung daran interessiert sein, nur spezifische Adapter oder Protokolle aus allen im System installierten Adaptern oder Protokollen herauszufinden. Somit möchte die Auffangeinrichtung nicht die Informationen über alle Adapter bestimmen. Statt dessen kann sie beim Suchen

nach solchen Informationen zum Beispiel stoppen, wenn sie alle Netzwerkadapter gefunden hat, nach denen sie gesucht hat.

[0089] An verschiedenen Stellen der vorhergehenden Beschreibung wurde Bezug auf das Einhängen oder Austauschen einer Funktion genommen. In den beigefügten Ansprüchen wird der Begriff Einhängen gemeinsam für solche Operationen verwendet. Es gibt viele Wege für das Implementieren der Einhängung. Viele von diesen Wegen sind relativ lange bekannt, sind jedoch nicht im Zusammenhang mit dem Auffangen von Paketen verwendet worden. Eine gebräuchliche Erklärung des Begriffes ”Einhängen” ist die, daß Einhängen das Austauschen einer bestimmten ursprünglichen Funktion in einem Programm (zum Beispiel in dem Kernprogramm des Betriebssystems) bedeutet, so das eine andere Funktion, oft als Austauschfunktion bekannt, anstelle der ursprünglichen Funktion aufgerufen wird. Einhängungen werden oft als ein Erweiterungsmechanismus zur Verfügung gestellt. Die Einhängungen können irgendeine von mehreren Bedeutungen haben: Die Austauschfunktion kann anstelle, vor oder nach der ursprünglichen Funktion aufgerufen werden.

[0090] Die Implementierung des tatsächlichen Einhängverfahrens variiert daher. In einigen Fällen ist normalerweise eine Liste von Funktionen vorhanden, die aufgerufen werden und es können der Liste neue Funktionen zugefügt werden (oft wird eine Liste von Einhängungen oder Einhängfunktionen aufgerufen). In anderen Fällen kann eine Funktion durch einen Funktionsverweis aufgerufen werden, selbst dann, wenn der Funktionsverweis nicht dazu ausgestaltet ist, eingehängt zu werden. In solchen Fällen kann das Einhängen durch dahingehende Veränderung des Funktionsverweises erfolgen, daß er auf die Austauschfunktion verweist. Die Austauschfunktion kann dann, als ihre Option, die ursprüngliche Funktion an jedem Punkt aufrufen. Eine ähnliche Situation kann implizit vorhanden sein. So haben zum Beispiel einige Implementierungen von dynamisch verbundenen Bibliotheken eine Tabelle oder eine Verbindungsliste von Eingangspunkten, die modifiziert werden können, um denselben Effekt zu erreichen (eine solche Tabelle wird auch als Abfertigungstabelle bezeichnet). Einige Systeme stellen auch einen Mechanismus für das Wiederauffinden und das Modifizieren von Hardware- oder Software-Unterbrechungsanzeigern zur Verfügung, was als ein möglicher Mechanismus für das Einhängen angesehen werden kann.

[0091] Manchmal steht kein Mechanismus für Umleitungsaufrufe für eine Funktion zur Verfügung. In diesen Fällen ist es möglich, das Einhängen durch momentanes Modifizieren des binären Formatcodes zu implementieren. Eine mögliche Implementierung eines solchen Einhängens ist, einige wenige Bytes vom Beginn der ursprünglichen Funktion zu sichern und sie dann durch einen Sprungbefehl an die Austauschfunktion zu ersetzen. Ein Weg zum Aufrufen der ursprünglichen Funktion ist in einem solchen Fall, die ersten gesicherten Befehle der ursprünglichen Funktion zu analysieren und nachdem der erste Befehl, der nicht durch den in die ursprüngliche Funktion geschriebenen Sprungbefehl überschrieben ist, einen Sprung an die entsprechende Stelle in der ursprünglichen Funktion zu setzen. Mit kleinen Veränderungen des Codes, der auf den Sprung in dem ursprünglichen Code hinweist, kann die Austauschfunktion anstelle, vor oder nach der ursprünglichen Funktion aufgerufen werden.

[0092] Die Austauschfunktion wird manchmal als die Einhängversion der Funktion, im Gegensatz zu der ursprünglichen Funktion, bezeichnet. Diese Funktion wird insbesondere dann verwendet, wenn die Austauschfunktion anstelle der ursprünglichen Funktion aufgerufen wird (was natürlich nicht die Möglichkeit ausschließt, daß ein expliziter Aufruf des ursprünglichen Codes von irgendeiner Stelle innerhalb ihrer Ausführung erfolgen kann).

[0093] Zur Klarstellung ist zu bemerken, daß die Verwendung normaler APIs (Application Programming Interfaces – Anwender-Programmierungsschnittstellen) und anderer Schnittstellen normalerweise nicht als Einhängen angesehen wird, selbst wenn Funktionsverweise oder Rückaufrufe durch die API geleitet werden. Ein normaler Zwischentreiber oder ein STREAMS-Modul führt zum Beispiel kein Einhängen aus, sondern verwendet statt dessen die dokumentierten (oder umgekehrt-gestalteten) Schnittstellen für Netzwerkadapter und Protokolle und wirkt als Protokoll in eine Richtung und als Adapter in die andere Richtung.

[0094] Ein Verfahren ähnlich dem der vorher beschrieben Erfindung kann anstelle des Einhängens in einigen Umgebungen zur Anwendung kommen. Das Kernprogramm vieler Betriebssysteme, wie zum Beispiel Windows NT, enthält mehrere dynamisch ladbare Bibliotheken (Dynamic Loadable Libraries – DLL) oder Module. Jeder Modul stellt eine gut definierte Programmierungs-Schnittstelle zu dem Rest des Kernprogramms dar und implementiert die Schnittstelle unter Verwendung eines internen Verfahrens. Das Netzwerkprotokollgerüst in Windows NT wird durch die NDIS.SYS-Datei implementiert. Diese Datei implementiert die Funktionsaufrufe, die von den Treibern der Netzwerkadapter und von den Netzwerkprotokollen verwendet werden.

[0095] Windows NT stellt keine geeignete Schnittstelle für das Einhängen der DLL-Eingabepunkte zur Verfügung, wie sie für Windows 95- und Windos 98-VxD-Treiber vorhanden ist. Es wäre möglich, die NDIS-Funktionen durch Austauschen des Codes ab dem Starten der ursprünglichen Funktionen einzuhängen. Das ist jedoch ziemlich kompliziert. Eine andere, jedoch konzeptionell ähnliche Verfahrensweise ist hierin beschrieben.

[0096] Grundlegend kann man eine Funktionalität ähnlich dem Einhängen durch Beiseite-Legen (Sichern) des ursprünglichen Betriebssystemmoduls (NDIS.SYS) und durch sein Ersetzen durch einen neuen Modul erreichen, der dieselbe Programmierungs-Schnittstelle implementiert, die Unterbrechungen ausführt und den ursprünglichen Betriebssystemmodul jedesmal dann aufruft, wenn die ursprüngliche Operation ausgeführt werden soll.

[0097] Bei dieser einfachen Verfahrensweise gibt es jedoch drei Probleme. Die Programmier-Schnittstelle (API), die durch den Kernprogramm-Modul (NDIS.SYS) zur Verfügung gestellt wird, kann sich zwischen jeder Version des Betriebssystems ändern, sogar zwischen verschiedenen OEM-Versionen (Original Equipment Manufacturer – Originalgerätehersteller-Versionen) oder Service-Paketen. Somit führt das einfache Austauschen des Moduls zu ernsthaften Kompatibilitätsproblemen (sehr wahrscheinlich würde der Computer nicht starten, wenn der Austauschmodul mit einem nicht kompatiblen Service-Paket verwendet wird). Das zweite Problem ist ähnlich. Wenn eine Aktualisierung oder ein Service-Paket installiert wird, kann eine Warnanzeige an den Benutzer über die veränderte Datei gegeben werden. Das verwirrt die Benutzer und sie brechen möglicherweise die Installation von Service-Paketen ab. Das dritte Problem ist ähnlich: Wenn ein neues Service-Paket installiert wurde und es den alten Betriebssystemmodul durch einen neuen ersetzt hat, kann die Auffangeinrichtung nicht mehr in den geladenen Zustand gelangen.

[0098] Eine mögliche Lösung für dieses Problem ist, nach jedem Neustart einen separaten Programmlauf durchzuführen, mit diesem Programm zu prüfen, ob der Modul verändert wurde und, wenn das der Fall ist, es durch den Auffangmodul zu ersetzen.

[0099] Eine zweite Lösung ist, eine sogenannte systemeigene Anwendung zu erzeugen, d. h. eine Anwendung, die läuft, bevor das Betriebssystem die Netzwerkmodule lädt. Eine systemeigene Anwendung läuft über das Kernprogramm von NT, jedoch ohne Win32 oder andere Untersysteme in Anwenderbetriebsweise. Eine systemeigene Anwendung ist somit eine Anwendung in Anwenderbetriebsweise, die über NT läuft, jedoch in Wirklichkeit keine ”Windows”-Anwendung ist. Solche Anwendungen werden traditionell zum Beispiel für Startprogramm-Bereinigungen (boot-time clean-ups) wie zum Beispiel Dateisystemkonsistenz und Integritätsprüfungen verwendet. Ein Verfahren, das alle drei Probleme löst, funktioniert folgendermaßen:

– Beim Starten läuft ein systemeigenes Programm. Dieses Programm überprüft zuerst, ob der Kernprogramm-Modul (z. B. NDIS.SYS) bereits ein durch sich selbst erzeugter Austausch ist. Ist das der Fall, entfernt es den Austausch und speichert den ursprünglichen Modul. Dann beginnt es wirklich zu arbeiten. Es legt das Original beiseite, liest seine Eingabepunkttabelle, liest die Eingabepunkttabelle des Auffangmoduls und erzeugt dynamisch einen neuen Modul, der dieselben Eingabepunkte wie der ursprüngliche Modul enthält und schickt jeden Eingabepunkt zu dem Auffangmodul, wenn dieser Eingabepunkt in dem Modul der Auffangeinrichtung vorhanden ist, sowie zu dem ursprünglichen Modul, wenn der Eingabepunkt nicht in dem Modul der Auffangeinrichtung vorhanden ist. Der erzeugte Modul wird an die Stelle des ursprünglichen Moduls geschrieben.

– Der Start wird dann fortgesetzt und der erzeugte Modul wird geladen. Der ursprüngliche Modul und der Auffangmodul werden ebenfalls geladen.

– Wenn das Betriebssystem geladen ist, läuft ein Programm (z. B. System-Service-Anwendungsprogramm), das die erzeugte DLL entfernt und statt dessen den ursprünglichen Betriebssystem-Modul (NDIS.SYS) zurückbewegt.

[0100] Dieses Verfahren bietet einen hohen Grad von Unempfindlichkeit gegenüber Unterschieden zwischen den Service-Paketen, da es sehr selten vorkommt, daß sich Schnittstellen in vollständig unkompatibler Weise verändern (im Gegensatz zum Hinzufügen von neuen Schnittstellen). Es vermeidet auch die Probleme beim Installieren der Service-Pakete, da der ursprüngliche Kernprogramm-Modul immer vorhanden ist, wenn ein Service-Paket installiert wird und der neue Modul verwendet wird, nachdem das Service-Paket installiert ist.

[0101] Es sollte bemerkt werden, daß das vorher angeführte Verfahren in vielfältiger Weise implementiert werden kann. Die Operationen können in unterschiedlicher Reihenfolge ausgeführt werden, andere Verschachtelungsoperationen können hinzugefügt werden und einige der Operationen schließen aus, daß die Gedankengänge hinter dem Verfahren wesentlich beeinträchtigt werden.

[0102] Im Gegensatz zu einigen vorhergehenden Verfahrensweisen, legt das Verfahren der vorliegenden Erfindung den Anwendungen, welche die Auffangeinrichtung verwenden, sehr wenige Einschränkungen auf. Die Anwendung kann alle Datenpakete durch die Auffangeinrichtung schicken, kann sie in eine Warteschlange einordnen, kann sie abnehmen, sie modifizieren und sie kann ganz neue Pakete in den Paketstrom einbringen, die von den Adaptern und Protokollen erkannt werden.

[0103] Wenn auch die Erfindung vorher im Zusammenhang mit Windows 95-, Windows 98-, Windows NT-, Solaris- und FreeBSD-Betriebssystemen beschrieben wurde, ist es auf eine Anzahl anderer Betriebssysteme, wie zum Beispiel Windows 2000, NetBSD und Linux direkt anwendbar. Es sollte auch so verstanden werden, daß, obwohl die Erfindung im Zusammenhang mit einem einzigen Kernprogrammtreiber beschrieben wurde, es möglich ist, einen Teil der Funktionalität auf separate Kernprogrammtreiber oder sogar auf eine Anwendung in Benutzer-Betriebsweise aufzuteilen. In einigen Betriebssystemen gibt es sogar keine Unterscheidung zwischen Kernprogramm- und Benutzer-Betriebsweise-Operationen.

[0104] Fig. 14 zeigt eine Hardware-Anordnung, die verwendet werden kann, um die vorliegende Erfindung zu implementieren. Die Computervorrichtung von Fig. 14 ist um einen Prozessor **1401** herum aufgebaut, der mit einer Benutzerschnittstelle **1402** (vorzugsweise eine Tastatur- und eine Bildschirm-Schnittstelle) zum Empfangen von Benutzerbefehlen und Anzeigeinformationen für den Benutzer, einem Programmspeicher **1403** zum Speichern des Betriebssystem-Kernprogramms und von anderen Programmen während der Operation und mit einem Verarbeitungsspeicher **1404** gekoppelt ist, der als zeitweiliger Arbeitsspeicherplatz für Informationen verwendet wird, zum Beispiel Pakete, die sich mitten in dem Prozeß des Absendens oder des Empfangens befinden. Der Prozessor ist ferner mit einem internen Bus **1405** zum Austauschen von Informationen mit anderen Teilen des Computers gekoppelt, wie zum Beispiel mit einer örtlichen Daten-Eingabe/Ausgabe-Vorrichtung (vorzugsweise eine CD-ROM-Station) **1406**, einem Massenspeicher (vorzugsweise eine Festplatte) **1407** und mit einer Netzwerk-Schnittstellenkarte **1408**. Die letztere ist mit dem physikalischen Übertragungsmedium **1409** eines Datenpakete-Übertragungsnetzwerkes über einen Netzwerkanschluß **1410** gekoppelt.

[0105] Für die Zwecke der Erfindung muß die Computeranordnung gemäß Fig. 14 erstens in der Lage sein, den Programmcode zu empfangen und zu laden, der aus durch den Prozessor ausführbaren Befehlen und Implementierungen der Einhäng- und Austauschfunktionen, die den Paketauffangmodul darstellen, besteht. Diese Forderung ist leicht zu erfüllen, weil der erfindungsgemäße Paketauffangmodul am bevorzugtesten in einer gespeicherten Form entweder auf einem mobilen Speichermedium, welches durch die lokale Daten-Eingabe/Ausgabe-Vorrichtung **1406** lesbar ist, oder als eine Datei, die über das Netzwerk zugänglich ist, mit dem die Computeranordnung über die Netzwerk-Schnittstellenkarte **1408** und den Netzwerkanschluß **1410** gekoppelt ist, vorliegt. Es sind nur allgemein übliche Ladebefehle einer bekannten Art, die durch den Benutzer über die Benutzer-Schnittstelle erteilt werden, erforderlich, um das Laden eines solchen Programmcodes von der lokalen Daten-Eingabe/Ausgabe-Vorrichtung **1406** oder von dem Netzwerkanschluß **1410** in den Programmspeicher **1403** zu initiieren. Es ist weiterhin empfehlenswert, daß der Programmcode in den Massenspeicher **1407** gespeichert wird, so daß er leicht beim Neustarten oder wenn es ansonsten erforderlich ist neu geladen werden kann.

[0106] Zweitens muß die Computeranordnung gemäß Fig. 14 in der Lage sein, den in den Programmspeicher **403** geladenen Programmcode, der die Einhängungen und die Austauschfunktionen, welche den Paketauffangeinrichtung-Modul darstellen, implementiert, auszuführen. Das ist ebenfalls leicht durchzuführen, solange der Programmcode mit dem Betriebssystem kompatibel ist, welches den Betrieb der Computeranordnung steuert. In den vorhergehenden Teilabschnitten der Patentanmeldung haben wir deutlich aufgezeigt, wie bestimmte Funktionen und Einhängprozeduren zusammen mit bestimmten, klar benannten und bekannten Betriebssystemen verwendet werden. Fachleuten ist jedoch nach dem Lesen und Verstehen der Prinzipien der Erfindung, die in dieser Patentanmeldung offenbart werden, klar, daß es möglich ist, die Erfindung so zu verallgemeinern, daß sie mit einem beliebigen Betriebssystem (das entweder am Einreichungsdatum der vorliegenden Patentanmeldung bekannt ist oder in der Zukunft noch zu entwickeln ist) verwendet werden kann, indem der Einhängmechanismus, der in diesem Betriebssystem verfügbar ist und die jeweiligen Funktionen ersetzt, welche sich auf die Handhabung der ankommenden und abgehenden Pakete beziehen, verwendet wird.

Liste der Literaturhinweise

AT93
Adams, P., Tondo, C.: Writing Unix Device Drivers in C (Schreiben von Unix-Gerätetreibern in der Programmiersprache C),
Prentice Hall, 1993.

Baker97
Baker, A.: The Windows NT Device Driver Book (Das Windows NT-Gerätetreiber-Buch),
Prentice Hall, 1997.
Bonner96
Bonner, P.: Programming with Windows Sockets (Programmieren mit Windows-Sockets),
Prentice Hall, 1996.
Brain96
Brain, M.: Win32 System Services (System-Serviceleistungen von Win32),
Prentice Hall 1996.
CB94
Cheswick, W., Bellovin, S.: Firewalls and Internet Security (Firewalls und Internet-Sicherheit),
Addison-Wesley, 1994.
CZ95
Chapman, B., Zwicky, E.: Building Internet Firewalls (Das Aufbauen von Internet-Firewalls),
O'Reilly, 1995.
DEJANEWS
Public discussions in the comp.os.ms-windows.programmer.nt.kernel-mode and comp.os.ms-windows.programmer.win95vxd newsgroups (Öffentliche Diskussionen in comp.os.ms-windows.programmer.nt.kernel-mode und comp.os.ms-windows.programmer.win95vxd-Newsgroups),
1997–1999, (archiviert z. B. unter www.dejanews.com).
Dhawan95
Dhawan, S.: Networking Device Drivers (Netzwerk-Gerätetreiber),
Van Nostrand Reinhold, 1995.
Droms97
Droms, R.: Dynamic Host Configuration Protocol RFC 2131 (Protokoll RFC 2131 zum Konfigurieren eines dynamischen Verarbeitungsrechners),
Internet Engineering Task Force, 1997.
ES90
Ellis, M., Stroustrup, B.: The Annotated C++ Reference Manual (Das annotierte C++-Anleitungshandbuch),
Addison-Wesley, 1990.
Ezzel97
Ezzell, B.: NT4/Windows 95 Developers Handbuch (NT4/Windows95-Entwickler-Handbuch),
SYBEX, 1997.
KA98
Kent, S., Atkinson, R.: Security Architecture for the Internet Protocol, RFC 2401 (Sicherheitsarchitektur für das Internet-Protokoll RFC 2401),
Internet Engineering Task Force, 1998.
Kauler97
Kauler, B.: Windows Assembly Language and System Programming (Windows-Assemblersprache und Systemprogrammierung),
R&D Books, 1997.
Lanciani92DIS_PKT
Lanciani, D.: DIS-PKT.ASM file, 1992 (DIS_PKT-Datei, 1992), Verfügbar unter www.danlan.com.
Lanciani94IBANPKT
Lanciani, D.: IBANPKT-program, 1994 (Das IBANPKT-Programm, 1994),
verfügbar unter www.danlan.com.
Lanciani96OdiPKT3.1
Lanciani, D.: ODIPKT 3.1 program, 1996 (Das ODIPKT 3.1-Programm, 1996),
verfügbar unter www.danlan.com.
Lanciani97NFSTDI
Lanciani, D.: NFDTDI-program, 1997 (Das NFSTDI-Programm, 1997), verfügbar unter www.danlan.com.
Lanciani97NDIS3PKT
Lanciani, D.: NDIS3PKT program, 1997 (Das NDIS3PKT-Programm, 1997),
verfügbar unter www.danlan.com.
Lanciani98
Lanciani, Dan: RE: Intercepting Packets at NDIS Level, article posted to comp.os.ms-windows.programmer.nt.kernel-mode, February 18, 1998 (Betr: Das Auffangen von Datenpaketen im NDIS-Level. Beitrag abgeschickt an comp.os.ms-windows.programmer.nt.kernel-mode am 18. Februar 1998),
(archiviert zum Beispiel unter www.dejanews.com.).
Lanciani98PPPMAC

Lanciani, Dan: RE: Help about PPPMAC, article posted to comp.os.ms-windows.programmer.nt.kernel-mode, December 17, 1998 (Betr: Hilfe zu PPPMAC. Beitrag abgeschickt an comp.os.ms-windows.programmer.nt.kernel-mode am 17. Dezember 1998),
(archiviert zum Beispiel unter www.dejanews.com.).
Lanciani98Reply
Lanciani, Dan: RE: Interceptor packets at NDIS level, article posted to corp. os.ms-windows.programmer.nt.kernel-mode, March 1, 1998 (Betr: Das Auffangen von Datenpaketen im NDIS-Level, Beitrag abgeschickt am 1. März 1998 an comp.os.ms-windows.programmer.nt.kernel-mode am 1. März 1998),
(archiviert zum Beispiel unter www.dejanews.com.).
McGregor92
McGregor, G.: The PPP Internet Protocol Control Protocol (IPCP), RFC 1332 (Das PPP-Internetprotokoll-Steuerungsprotokoll (IPCP) RFC 1332),
Internet Engineering Task Force 1992.
MSDOS5
MSDOS5.x programmer's manual (Das Handbuch des MSDOS5.x-Programmierers),
Microsoft, 1986 (?).
Nogar97
Nogar, R.: Windows NT File Systems Internals (Interne Systeminhalte der Windows NT-Datei)
O'Reilly, 1997.
PCAUSA99
Implementing a intermediate driver that works with RAS under NT (Das Implementieren eines Zwischentreibers, der mit RAS unter NT arbeitet)
Web-Seite http://www.pcausa.com/resources/im-ras.html, März 1999.
Pentium
Pentium Processor Family User's Manual (Anleitungshandbuch für die Pentium-Prozessor-Familie),
Intel, 1994, Bd. 3
Perkins96
Perkins, C.: IP Mobility Support, RFC 2290 (IP-Mobilitäts-Unterstützung RFC 2290),
Internet Engineering Task Force, 1996.
Petzold92
Petzold, C.: Programming Windows 3.1 (Programmieren von Windows 3.1),
Microsoft Press, 1992.
Plummer82
Plummer, D.: Ethernet Address Resolution Protocol RFC 826 (Ethernet-Adressenauflösungsprotokoll RFC 826),
Internet Engineering Task Force 1982.
Poste181
Postel, J.: Internet Protocol RFC 791 (Internet-Protokoll RFC 791),
Internet Engineering Task Force, 1981.
QS96
Quinn, B., Shute, D.: Windows Socket Programming (Windows-Soket-Programmierung),
Addison-Wesley, 1996.
SCFJ96
Schulzrinne, H., Casner, S., Frederick, R. u. Jacobson, V. RTP: A Transport Protocol for Real-Time Application, RFC 1889 (RTP: Ein Transportprotokoll für Echtzeit-Anwendungen RFC 1889),
Internet Engineering Task Force 1996.
Simpson94
Simpson, W.: The Point-to-Point Protokoll (PPP), RFC 1661 (Das Punkt-zu-Punkt-Protokoll RFC 1661),
Internet Engineering Task Force 1994.
Sinha96
Sinha, A.: Network Programming in Windows NT (Netzwerkprogrammierung in Windows NT),
Addison-Wesley, 1996.
Steele90
Steele, G.: Common Lisp- The Language (Lisp allgemein – die Programmiersprache),
1990, 2. Ausgabe.
STREAMS93
SunOS 5.2 STREAMS Programmer's Guide (SunOS 5.2-STREAMS-Progammierer-Handbuch),
Sun Microsystems, 1993.
SWE98
Scott, C., Wolfe, P. u. Erwin, M.: Virtual Private Networks (Virtuelle private Netzwerke)

O'Reilly, 1998.
VM99
Viscarola, P., Mason W.: Windows NT Device Driver Development (Windows NT-Gerätetreiberentwicklung), OSR Open System Resources Inc. 1999.
Waldbusser 97
Waldbusser, S.: Remote Network Monitoring Management Information Base, RFC 2021 (Netzwerk-Fernüberwachungs-Verwaltungsinformations-Basis RFC 2021),
Internet Engineering Task Force 1997.
Win4DDDK
Microsoft Windows NT 4.0 Device Driver Kit (Gerätetreibersatz für Microsoft Windows NT 4.0),
Microsoft 1996.
Win95DDK
Microsoft Windows NT 95 Device Driver Kit (Gerätetreibersatz für Microsoft Windows NT 95),
Microsoft 1996.
Win98DDK
Microsoft Windows NT 98 Device Driver Kit (Gerätetreibersatz für Microsoft Windows NT 98),
Microsoft, 1998.
WS95
Wright, G., Stevens, W.: TCP/IP Illustrated (TCP/IP-Erläuterung),
Addison-Wesley, 1995, Bd. 2.

Patentansprüche

1. Verfahren zum Auffangen von Netzwerkpaketen in einem Computersystem, wobei Netzwerkpakete zwischen einem ersten Netzwerkadapter und einer ersten Protokolleinrichtung kommuniziert werden, von denen der Netzwerkadapter eine bestimmte Netzwerk-Schnittstelle implementiert, wobei das Verfahren die Schritte aufweist
– Bereitstellen eines Satzes von Austauschfunktionen innerhalb eines Paketauffangmoduls;
– Einhängen wenigstens einer ersten Funktion, die zum Übertragen von Netzwerkpaketen von der ersten Protokolleinrichtung zu dem ersten Netzwerkadapter verwendet wird, in eine erste Austauschfunktion;
– Einhängen wenigstens einer zweiten Funktion, die zum Übertragen von Netzwerkpaketen von dem ersten Netzwerkadapter zu der ersten Protokolleinrichtung verwendet wird, in eine zweite Austauschfunktion; und
– Einhängen wenigstens einer dritten Funktion, die zum Empfangen von Information über den Zustand der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle verwendet wird, in eine dritte Austauschfunktion;
– wobei Einhängen definiert ist als Umleiten von Funktionsaufrufen an die erste, die zweite und die dritte Funktion auf entsprechende Austauschfunktionen, so dass die Austauschfunktionen durch die umgeleiteten Funktionsaufrufe aufgerufen werden.

2. Verfahren nach Anspruch 1, das zusätzlich die Schritte aufweist
– Bestimmen, ob eine dynamische IP-Adresse der Netzwerk-Schnittstelle, die von dem ersten Netzwerkadapter implementiert wird, zugeordnet worden ist; und
– in einem Fall, wo eine dynamische IP-Adresse der Netzwerk-Schnittstelle, die von dem ersten Netzwerkadapter implementiert wird, zugeordnet worden ist, Bestimmen, welche die dynamische IP-Adresse ist.

3. Verfahren nach Anspruch 1, das zusätzlich einen Schritt des Identifizierens eines ersten Netzwerkadapters und einer ersten Protokolleinrichtung, die in dem Computersystem installiert sind, aufweist, der so angeordnet ist, dass dieser Schritt zunächst den Unterschritt aufweist
– Einhängen eines bestimmten Mechanismus, der von Netzwerkadaptern und Protokolleinrichtungen benutzt werden soll, um sie selbst in dem Computersystem zu registrieren, in einen bestimmten Austauschmechanismus;
– und danach ohne irgendeinen spezifischen Befehl die Unterschritte
– Identifizieren des ersten Netzwerkadapters, wenn er den Austauschmechanismus verwendet, um sich selbst bei dem Computersystem zu registrieren und
– Identifizieren der ersten Protokolleinrichtung, wenn sie den Austauschmechanismus verwendet, um sich selbst bei dem Computersystem zu registrieren.

4. Verfahren nach Anspruch 3, wobei der Schritt des Einhängens eines bestimmten, von Netzwerkadaptern und Protokolleinrichtungen verwendeten Mechanismus zum Registrieren davon, ohne irgendeinen speziellen Befehl, die Unterschritte aufweist

– Laden eines Schnittstellenmoduls, der den bestimmten Mechanismus bestimmt;
– Laden eines Paketauffangmoduls, der den Austauschmechanismus bestimmt; und
– Einhängen vorbestimmter Teile des bestimmten Mechanismus in vorbestimmte Teile des Austauschmechanismus.

5. Verfahren nach Anspruch 4, wobei der Schritt des Ladens eines Schnittstellenmoduls, der den bestimmten Mechanismus bestimmt, den Schritt des Ladens eines NDIS(Network Driver Interface Specification)-Schnittstellenmoduls aufweist, und der Schritt des Einhängens vorbestimmter Teile des bestimmten Mechanismus in vorbestimmte Teile des Austauschmechanismus die Unterschritte aufweist
– Einhängen der von dem NDIS-Schnittstellenmodul bestimmten NDIS-Register-Protokoll-Funktion in eine von dem Paketauffangmodul bestimmte Austausch-Protokollregistrierfunktion; und
– Einhängen der von dem NDIS-Schnittstellenmodul bestimmten NDIS-Offen-Adapter-Funktion in eine von dem Paketauffangmodul bestimmte Austausch-Netzwerkadapter-Öffnungs-Funktion.

6. Verfahren nach Anspruch 5, wobei der Schritt des Einhängens der NDIS-Register-Protokoll-Funktion den Schritt des Austauschens einer Anzahl von Funktionen in der von dem NDIS-Schnittstellenmodul bestimmten NDIS-Protokoll-Eigenschaften-Struktur aufweist.

7. Verfahren nach Anspruch 6, wobei der Schritt des Austauschens einer Anzahl der Funktionen in der NDIS-Protokoll-Eigenschaften-Struktur den Schritt des Austauschens der Empfangs-Handhaber-, Empfang-Vollständig-Handhaber- und Datenübertragung-Vollständig-Handhaber-Funktionen aufweist, die von dem NDIS-Schnittstellenmodul bestimmt werden.

8. Verfahren nach Anspruch 7, wobei der Schritt des Austauschens einer Anzahl der Funktionen in der NDIS-Protokoll-Eigenschaften-Struktur zusätzlich den Schritt des Austauschens der in dem NDIS-Schnittstellen-Modul bestimmten Sendung-Vollständig-Handhaber- und Anfrage-Vollständig-Handhaber-Funktionen aufweist.

9. Verfahren nach Anspruch 5, das zusätzlich den Schritt des Bestimmens, welche Bindungen den ersten Netzwerkadapter und die erste Protokolleinrichtung verbinden, durch Rufen der Austausch-Adapter-Öffnungs-Funktion aufweist.

10. Verfahren nach Anspruch 1, das zusätzlich die Schritte aufweist
– Laden des ersten Netzwerkadapters und der ersten Protokolleinrichtung und
– Bestimmen, welche Bindungen den ersten Netzwerkadapter und die erste Protokolleinrichtung verbinden, durch Analysieren von Datenstrukturen, nachdem der erste Netzwerkadapter und die erste Protokolleinrichtung geladen worden sind.

11. Verfahren nach Anspruch 10, wobei der Schritt des Analysierens von Datenstrukturen, nachdem der erste Netzwerkadapter und die erste Protokolleinrichtung geladen worden sind, den Schritt des Lesens eines Stücks Systemkonfigurationsinformation aus einem Speicher aufweist.

12. Verfahren nach Anspruch 11, wobei der Schritt des Lesens eines Stücks Systemkonfigurationsinformation den Schritt des Lesens einer Registry aufweist.

13. Verfahren nach Anspruch 1, das zusätzlich den Schritt des Identifizierens des ersten Netzwerkadapters und der ersten Protokolleinrichtung aufweist, so dass dieser Schritt den Unterschritt des Lesens eines Stücks Systemkonfigurationsinformation aus einem Speicher aufweist.

14. Verfahren nach Anspruch 13, wobei der Unterschritt des Lesens eines Stücks Systemkonfigurationsinformation aus einem Speicher das Lesen einer Registry aufweist.

15. Verfahren nach Anspruch 1, das zusätzlich die Schritte aufweist
– Laden des ersten Netzwerkadapters und der ersten Protokolleinrichtung und
– Identifizieren des ersten Netzwerkadapters und der ersten Protokolleinrichtung durch Durchfahren von Datenstrukturen, nachdem Adapter und Protokolle in das Computersystem geladen worden sind.

16. Verfahren nach Anspruch 15, das zunächst ohne irgendeinen spezifischen Befehl die Schritte aufweist
– Laden der ersten Protokolleinrichtung in das Computersystem; und
– Laden des ersten Netzwerkadapters in das Computersystem;
und danach, in der folgenden Reihenfolge, die Schritte

– Laden eines dynamisch ladbaren Paketauffängermoduls in das Computersystem; und
– Durchfahren von Datenstrukturen, um den ersten Netzwerkadapter und die erste Protokolleinrichtung zu identifizieren.

17. Verfahren nach Anspruch 1, wobei wenigstens einer der Einhängschritte die Unterschritte aufweist
– Lokalisieren des Anfangs des ausführbaren Programmcodes einer bestimmten ersten Funktion, die in eine bestimmte erste Austauschfunktion eingehängt werden soll;
– Speichern einer Kopie einer bestimmten Passage ausführbaren Programmcodes, die an diesem Anfang anfängt; und
– Austauschen der bestimmten Passage ausführbaren Programmcodes, die an diesem Anfang anfängt, mit einer anderen Passage ausführbaren Programmcodes, die die Ausführung auf die erste Austauschfunktion überträgt.

18. Verfahren nach Anspruch 1, wobei wenigstens einer der Einhängschritte die Unterschritte aufweist
– Lokalisieren, in einer Datenstruktur, eines Funktionszeigers, der auf eine bestimmte erste Funktion zeigt, die in eine bestimmte erste Austauschfunktion eingehängt werden soll;
– Speichern einer Kopie dieses Funktionszeigers; und
– Austauschen dieses Funktionszeigers mit einem anderen Funktionszeiger, der auf die erste Austauschfunktion zeigt.

19. Verfahren nach Anspruch 1, wobei wenigstens einer der Einhängschritte die Unterschritte aufweist
– Lokalisieren einer Abfertigungstabelle in einem dynamisch geladenen Modul; und
– Modifizieren dieser Abfertigungstabelle.

20. Verfahren nach Anspruch 1, wobei wenigstens einer der Einhängschritte den Unterschritt des Rufens einer Systemfunktion aufweist, die einen Haken (hook) für eine Systemdienstleistung installiert.

21. Verfahren nach Anspruch 1, wobei wenigstens einer der Einhängschritte den Unterschritt des Hinzufügens einer ersten Austauschfunktion, in die eine bestimmte erste Funktion eingehängt wird, zu einer vom System bereitgestellten Einhängliste aufweist.

22. Verfahren nach Anspruch 1, wobei wenigstens einer der Einhängschritte den Unterschritt des Umleitens eines Interrupt-Vektors aufweist.

23. Verfahren nach Anspruch 1, das zusätzlich den Schritt des Handhabens eines Netzwerkpakets mit einer bestimmten ersten Austauschfunktion aufweist, ohne dieses Netzwerkpaket zu der Funktion, die in die erste Austauschfunktion eingehängt ist, zu leiten.

24. Verfahren nach Anspruch 1, das zusätzlich den Schritt des Rufens einer bestimmten ersten Funktion von einer bestimmten ersten Austauschfunktion, in die die erste Funktion eingehängt ist, aufweist.

25. Verfahren nach Anspruch 1, das zusätzlich, in der folgenden Reihenfolge, die Schritte aufweist
– Modifizieren eines Netzwerkpakets mit einer bestimmten ersten Austauschfunktion und
– Leiten des modifizierten Netzwerkpakets zu der Funktion, die in die erste Austauschfunktion eingehängt ist.

26. Verfahren nach Anspruch 1, das zusätzlich den Schritt des Kopierens eines Netzwerkpakets durch Anwenden einer bestimmten ersten Austauschfunktion aufweist.

27. Verfahren nach Anspruch 1, das zusätzlich den Schritt des Rufens einer bestimmten ersten Funktion aufweist, die in eine bestimmte erste Austauschfunktion eingehängt ist, ohne zunächst die erste Austauschfunktion zu rufen.

28. Verfahren nach Anspruch 1, das zusätzlich die Schritte aufweist
– Bestimmen, ob eine Anwahlverbindung an oder aus ist; und
– Bereitstellen von Information darüber, ob diese Anwahlverbindung an oder aus ist, an das Paketauffängermodul.

29. Verfahren nach Anspruch 1, das zusätzlich die Schritte aufweist
– Bestimmen wenigstens einer Netzwerkadresse, die für die erste Netzwerkschnittstelle verwendet wird; und
– Bereitstellen von Information über bestimmte Netzwerkadressen an das Paketauffängermodul.

30. Verfahren nach Anspruch 29, wobei der Schritt des Bestimmens wenigstens einer Netzwerkadresse den Unterschritt des Prüfens von Verbindungslagen(link-layer)-Protokollpaketen aufweist.

31. Verfahren nach Anspruch 30, wobei der Schritt des Prüfens von Verbindungslagen-Protokollpaketen den Unterschritt des Prüfens von IPCP(IP Control Protocol)-Paketen aufweist, wobei IPCP ein Unterprotokoll von PPP (Point-to-Point Protocol) ist.

32. Verfahren nach Anspruch 30, wobei der Schritt des Prüfens von Verbindungslagen-Protokollpaketen den Unterschritt des Prüfens von ARP(Address Resolution Protocol)-Protokollpaketen aufweist.

33. Verfahren nach Anspruch 29, wobei der Schritt des Bestimmens wenigstens einer Netzwerkadresse den Unterschritt des Prüfens des DHCP(Dynamic Host Configuration Protocol)-Protokolls aufweist.

34. Verfahren nach Anspruch 29, wobei der Schritt des Bestimmens wenigstens einer Netzwerkadresse die Unterschritte aufweist
– Einhängen einer bestimmten ersten Funktion, die gerufen werden soll, wenn eine Änderung in der Adressinformation vorliegt, in eine bestimmte Austauschfunktion;
– Durchfahren einer Anzahl vorbestimmter Datenstrukturen zu der Zeit des Rufens der Austauschfunktion; und
– Vergleichen von aus den Datenstrukturen gelesener Information mit einem vorbestimmten Stück früher gespeicherter entsprechender Information.

35. Verfahren nach Anspruch 1, das ferner den Schritt aufweist
– Modifizieren von zwischen dem ersten Netzwerkadapter und der ersten Protokolleinrichtung geleiteter Information in Bezug auf Verbindungslageneigenschaften; und
– als Ergebnis dieser Modifikation der Information, Reduzieren der maximalen übertragenen Paketgröße, die der ersten Protokolleinrichtung an einer Verbindung bekannt ist.

36. Verfahren zum Auffangen von Netzwerkpaketen in einem Computersystem, wo eine Anzahl von Funktionen verwendet werden, um Netzwerkpakete zwischen einer Anzahl von Netzwerkadaptern und einer Anzahl von Protokolleinrichtungen zu kommunizieren, von denen die Netzwerkadapter bestimmte Netzwerk-Schnittstellen implementieren, wobei das Verfahren die Schritte aufweist
– Bereitstellen eines Satzes von Austauschfunktionen innerhalb eines Paketauffängermoduls;
– Einhängen einer ersten Anzahl von Funktionen, die zum Übertragen von Netzwerkpaketen von Protokolleinrichtungen zu Netzwerkadaptern verwendet werden, in einen ersten Satz von Austauschfunktionen;
– Einhängen einer zweiten Anzahl von Funktionen, die zum Übertragen von Netzwerkpaketen von Netzwerkadaptern zu Protokolleinrichtungen verwendet werden, in einen zweiten Satz von Austauschfunktionen; und
– Einhängen einer dritte Anzahl von Funktionen, die zum Empfangen von Information über den Zustand der von Netzwerkadaptern implementierten Netzwerk-Schnittstellen verwendet werden, in einen dritten Satz Austauschfunktionen;
– wobei Einhängen definiert ist als Umleiten von Funktionsaufrufen an die erste, die zweite und die dritte Anzahl von Funktionen auf entsprechende Austauschfunktionen, so dass die Austauschfunktionen durch die umgeleiteten Funktionsaufrufe aufgerufen werden.

37. Verfahren nach Anspruch 36, das zusätzlich den Schritt des Identifizierens einer Anzahl von Netzwerkadaptern und Protokolleinrichtungen aufweist, die in dem Computersystem installiert sind.

38. Verfahren nach Anspruch 37, wobei der Schritt des Identifizierens einer Anzahl von Netzwerkadaptern und Protokolleinrichtungen den Unterschritt des Ignorierens einer oder mehrerer Mitglieder der Gruppe der Netzwerkadapter und Protokolle aufweist.

39. Verfahren zum Auffangen von Netzwerkpaketen in einem Computersystem, wo ein bestimmtes erstes Betriebssystemmodul verwendet wird, um Netzwerkfunktionalität zu implementieren, und das erste Betriebssystemmodul eine bestimmte Programmierschnittstelle mit einer Anzahl von Eintrittspunkten implementiert, wobei das Verfahren die Schritte aufweist
– Austauschen des ersten Betriebssystemmoduls mit einem bestimmten ersten Austauschmodul, das eine Programmierschnittstelle gleich der Programmierschnittstelle des ersten Betriebssystemmoduls implementiert und das erste Betriebssystemmodul von einer Anzahl der Eintrittspunkte der Programmierschnittstelle ruft;
– Verwenden des Austauschmoduls, um wenigstens einen Netzwerkadapter und wenigstens eine Protokolleinrichtung zu identifizieren, die in dem Computersystem installiert sind;

– Verwenden des Austauschmoduls, um wenigstens eine erste Funktion zu ersetzen, die zum Übertragen von Netzwerkpaketen von der ersten Protokolleinrichtung zu dem Netzwerkadapter verwendet wird;
– Verwenden des Austauschmoduls, um wenigstens eine zweite Funktion zu ersetzen, die zum Übertragen von Netzwerkpaketen von dem ersten Netzwerkadapter zu der ersten Protokolleinrichtung verwendet wird;
– Verwenden des Austauschmoduls, um wenigstens eine dritte Funtion zu ersetzen, die zum Empfangen von Information über den Zustand der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle verwendet wird;
– Verwenden des Austauschmoduls, um zu bestimmen, ob der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle eine dynamische IP-Adresse zugeordnet worden ist oder nicht; und
– in einem Fall, wo der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle eine dynamische IP-Adresse zugeordnet worden ist, Verwenden des Austauschmoduls, um zu bestimmen, was diese dynamische IP-Adresse ist.

40. Verfahren nach Anspruch 39, das zusätzlich den Schritt des Rufens des ersten Betriebssystemmoduls von dem Austauschmodul aufweist.

41. Verfahren nach Anspruch 39, wobei der Schritt des Ersetzens des ersten Betriebssystemmoduls mit dem Austauschmodul die Unterschritte aufweist
– Beiseitestellen des ersten Betriebssystemmoduls zur Installationszeit und
– Ersetzen des ersten Betriebssystemmoduls durch das erste Austauschmodul.

42. Verfahren nach Anspruch 39, wobei der Schritt des Ersetzens des ersten Betriebssystemmoduls mit dem Austauschmodul durchgeführt wird, wenn das Computersystem booted, aber bevor das erste Betriebssystemmodul geladen wird.

43. Verfahren nach Anspruch 42, das zusätzlich den Schritt des Rückgängigmachens des Ersetzens durch das Austauschmodul aufweist, nachdem das erste Betriebssystemmodul geladen worden ist.

44. Verfahren nach Anspruch 39, das zusätzlich den Schritt des automatischen Erzeugens des auf dem ersten Betriebssystemmodul basierenden Austauschmoduls aufweist.

45. Computersystem zum Handhaben von Netzwerkpaketen, mit
– einem ersten Netzwerkadapter, der dazu eingerichtet ist, eine Netzwerk-Schnittstelle zu implementieren;
– einer ersten Protokolleinrichtung;
– einer Anzahl vorbestimmter Funktionen zum Kommunizieren von Netzwerkpaketen zwischen dem Netzwerkadapter und der Protokolleinrichtung;
– einem Paketauffängermodul zum Bestimmen eines Satzes von Austauschfunktionen;
– innerhalb des Paketauffängermoduls, einer Einrichtung zum Einhängen wenigstens einer ersten Funktion, die zum Übertragen von Netzwerkpaketen von der ersten Protokolleinrichtung zu dem ersten Netzwerkadapter verwendet wird, in eine erste Austauschfunktion;
– innerhalb des Paketauffängermoduls, einer Einrichtung zum Einhängen wenigstens einer zweiten Funktion, die zum Übertragen von Netzwerkpaketen von dem ersten Netzwerkadapter zu der ersten Protokolleinrichtung verwendet wird, in eine zweite Austauschfunktion; und
– innerhalb des Paketauffängermoduls, einer Einrichtung zum Einhängen wenigstens einer dritten Funktion, die zum Empfangen von Information über den Zustand der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle verwendet wird, in eine dritte Austauschfunktion;
– wobei Einhängen definiert ist als Umleiten von Funktionsaufrufen an die erste, die zweite und die dritte Funktion auf entsprechende Austauschfunktionen, so dass die Austauschfunktionen durch die umgeleiteten Funktionsaufrufe aufgerufen werden.

46. Computersystem nach Anspruch 45, das zusätzlich eine Einrichtung zum Identifizieren des ersten Netzwerkadapters und der ersten Protokolleinrichtung aufweist.

47. Paketauffängermodul zum Auffangen von Netzwerkpaketen in einem Computersystem, das einen ersten Netzwerkadapter, eine erste Protokolleinrichtung und eine Anzahl von vorbestimmten Funktionen zum Kommunizieren von Netzwerkpaketen zwischen dem Netzwerkadapter und der Protokolleinrichtung aufweist; wobei das Paketauffängermodul aufweist
– die Definition eines Satzes von Austauschfunktionen;

– eine Einrichtung zum Einhängen wenigstens einer ersten Funktion, die zum Übertragen von Netzwerkpaketen von der ersten Protokolleinrichtung zu dem ersten Netzwerkadapter verwendet wird, in eine erste Austauschfunktion;
– eine Einrichtung zum Einhängen wenigstens einer zweiten Funktion, die zum Übertragen von Netzwerkpaketen von dem ersten Netzwerkadapter zu der ersten Protokolleinrichtung verwendet wird, in eine zweite Austauschfunktion; und
– eine Einrichtung zum Einhängen wenigstens einer dritten Funktion, die zum Empfangen von Information über den Zustand der von dem ersten Netzwerkadapter implementierten Netzwerk-Schnittstelle verwendet wird, in eine dritte Austauschfunktion;
– wobei Einhängen definiert ist als Umleiten von Funktionsaufrufen an die erste, die zweite und die dritte Funktion auf entsprechende Austauschfunktionen, so dass die Austauschfunktionen durch die umgeleiteten Funktionsaufrufe aufgerufen werden.

Es folgen 11 Blatt Zeichnungen

Anhängende Zeichnungen

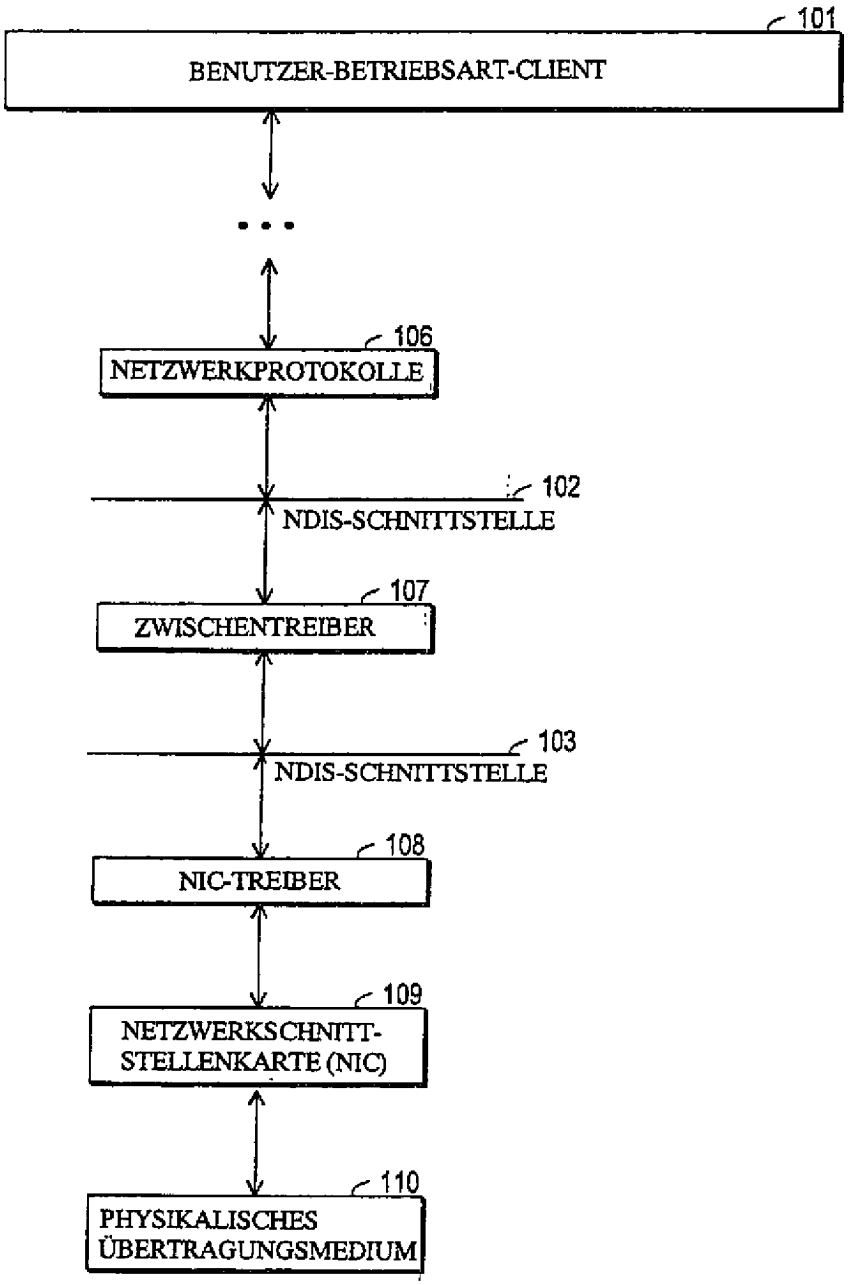


Fig. 1
STAND DER TECHNIK

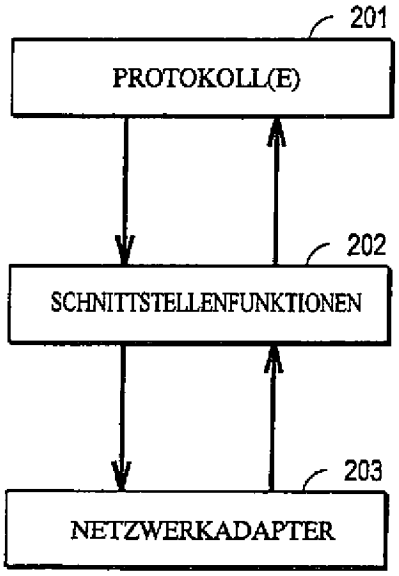


Fig. 2a
STAND DER TECHNIK

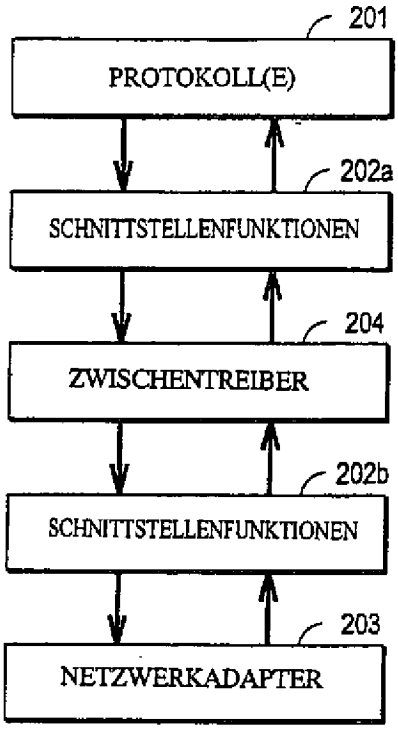


Fig. 2b
PRIOR ART

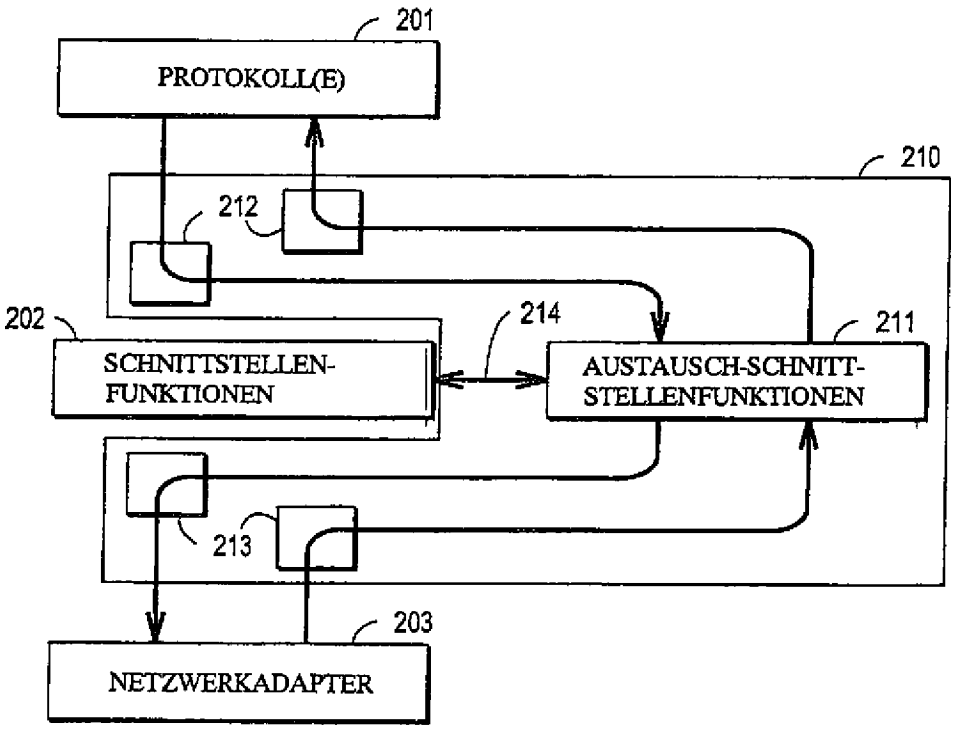


Fig. 2c

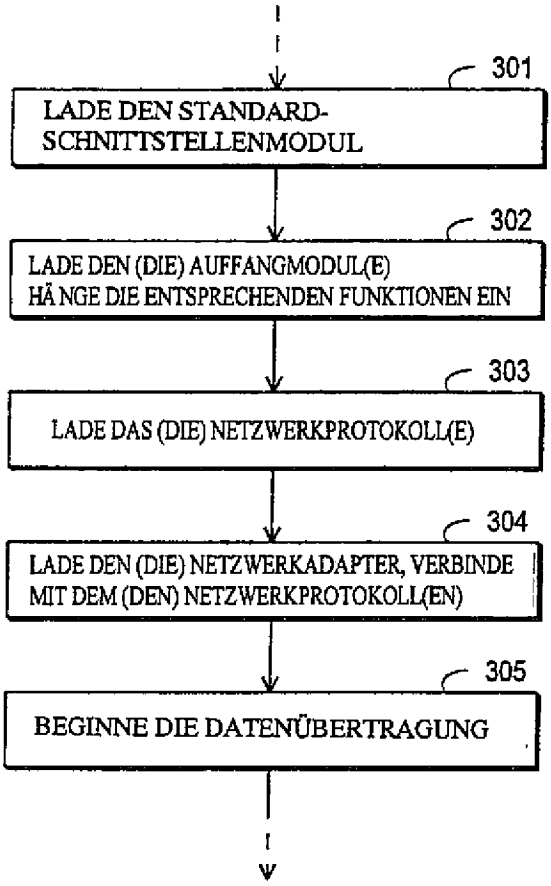

301
LADE DEN STANDARD-SCHNITTSTELLENMODUL
302
LADE DEN (DIE) AUFFANGMODUL(E) HÄNGE DIE ENTSPRECHENDEN FUNKTIONEN EIN
303
LADE DAS (DIE) NETZWERKPROTOKOLL(E)
304
LADE DEN (DIE) NETZWERKADAPTER, VERBINDE MIT DEM (DEN) NETZWERKPROTOKOLL(EN)
305
BEGINNE DIE DATENÜBERTRAGUNG

Fig. 3

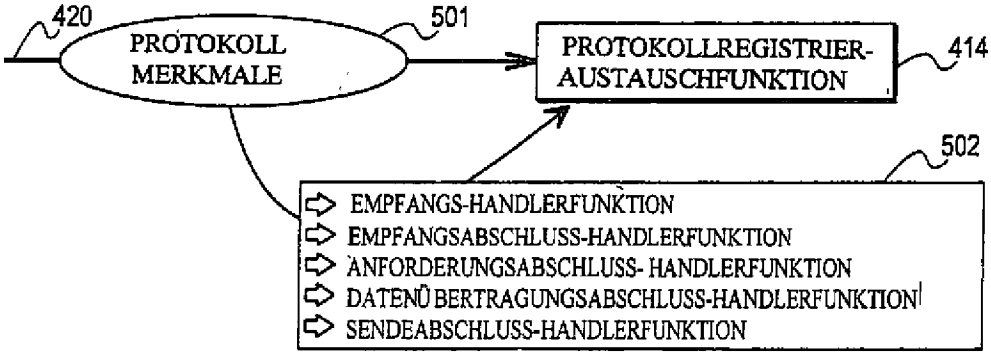

420
PROTOKOLL MERKMALE
501
PROTOKOLLREGISTRIER-AUSTAUSCHFUNKTION
414
502
EMPFANGS-HANDLERFUNKTION
EMPFANGSABSCHLUSS-HANDLERFUNKTION
ANFORDERUNGSABSCHLUSS- HANDLERFUNKTION
DATENÜBERTRAGUNGSABSCHLUSS-HANDLERFUNKTION
SENDEABSCHLUSS-HANDLERFUNKTION

Fig. 5

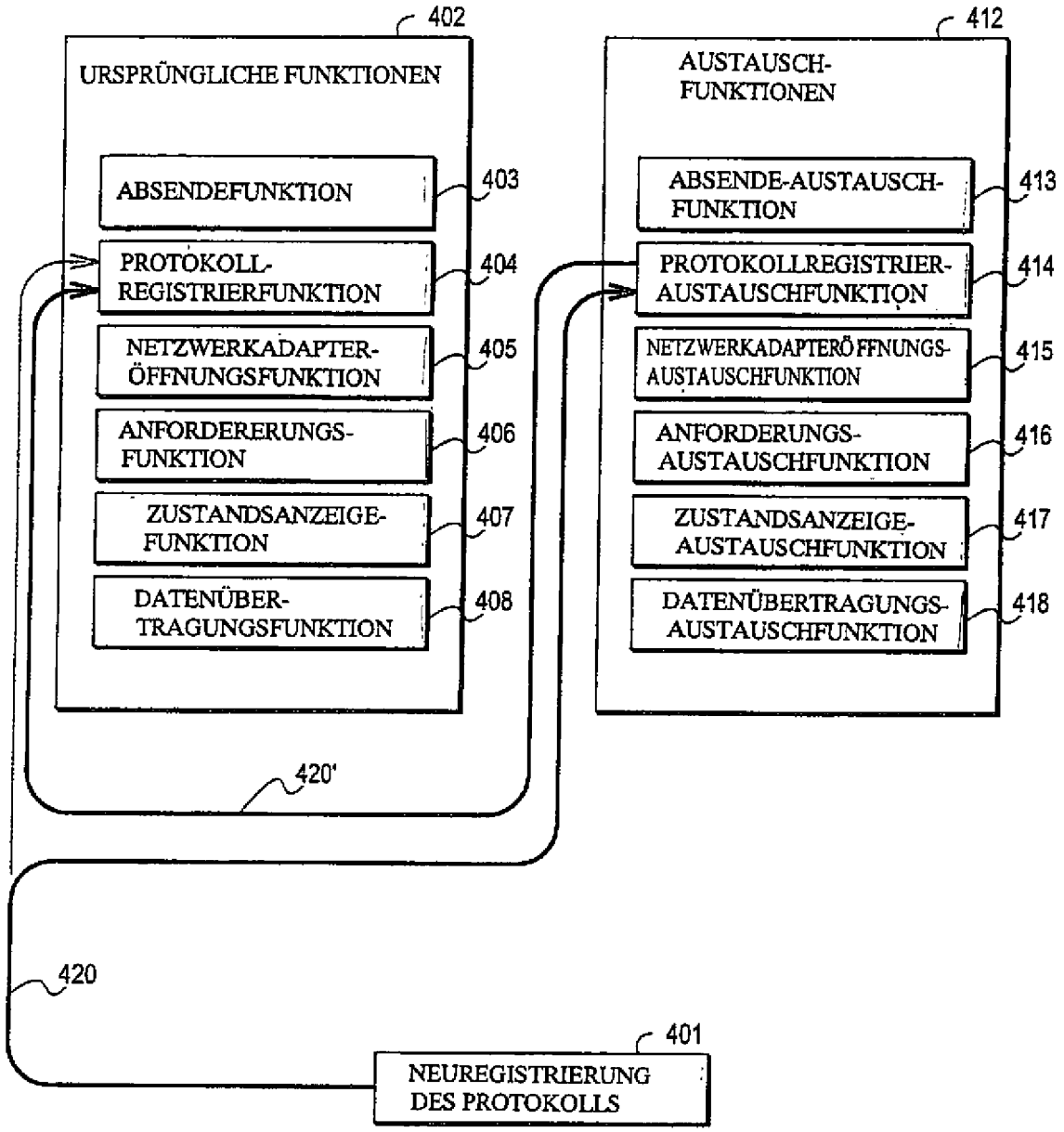

402
URSPRÜNGLICHE FUNKTIONEN
ABSENDEFUNKTION
403
PROTOKOLL-REGISTRIERFUNKTION
404
NETZWERKADAPTER-ÖFFNUNGSFUNKTION
405
ANFORDERERUNGS-FUNKTION
406
ZUSTANDSANZEIGE-FUNKTION
407
DATENÜBER-TRAGUNGSFUNKTION
408
412
AUSTAUSCH-FUNKTIONEN
ABSENDE-AUSTAUSCH-FUNKTION
413
PROTOKOLLREGISTRIER-AUSTAUSCHFUNKTION
414
NETZWERKADAPTERÖFFNUNGS-AUSTAUSCHFUNKTION
415
ANFORDERUNGS-AUSTAUSCHFUNKTION
416
ZUSTANDSANZEIGE-AUSTAUSCHFUNKTION
417
DATENÜBERTRAGUNGS-AUSTAUSCHFUNKTION
418
420'
420
401
NEUREGISTRIERUNG DES PROTOKOLLS

Fig. 4

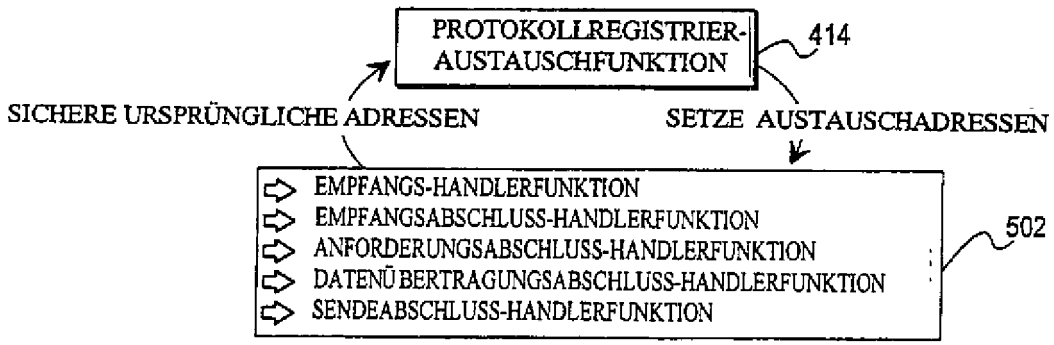

PROTOKOLLREGISTRIER-AUSTAUSCHFUNKTION
414
SICHERE URSPRÜNGLICHE ADRESSEN
SETZE AUSTAUSCHADRESSEN
EMPFANGS-HANDLERFUNKTION
EMPFANGSABSCHLUSS-HANDLERFUNKTION
ANFORDERUNGSABSCHLUSS-HANDLERFUNKTION
DATENÜBERTRAGUNGSABSCHLUSS-HANDLERFUNKTION
SENDEABSCHLUSS-HANDLERFUNKTION
502

Fig. 6

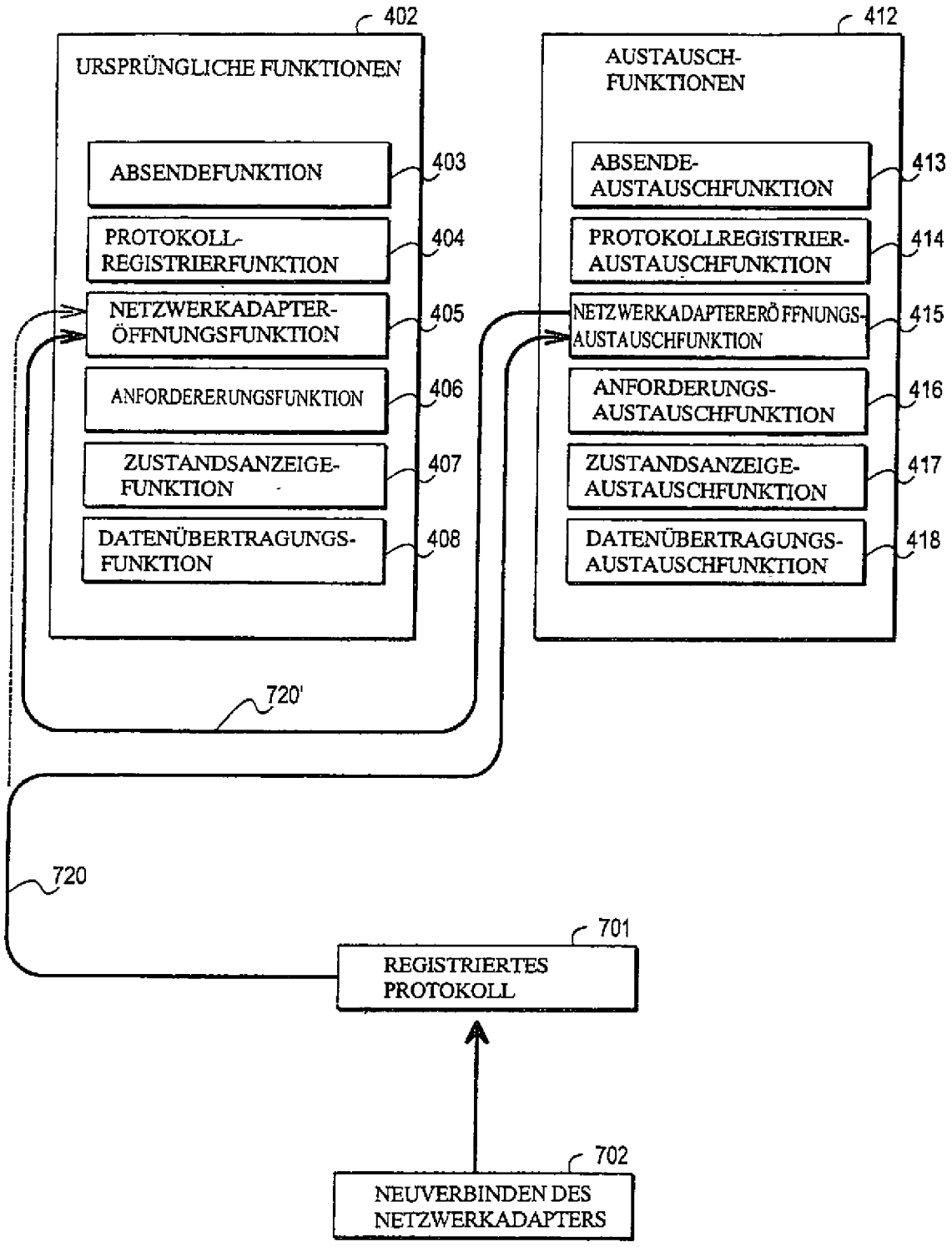

402
URSPRÜNGLICHE FUNKTIONEN
ABSENDEFUNKTION
403
PROTOKOLL-REGISTRIERFUNKTION
404
NETZWERKADAPTER-ÖFFNUNGSFUNKTION
405
ANFORDERERUNGSFUNKTION
406
ZUSTANDSANZEIGE-FUNKTION
407
DATENÜBERTRAGUNGS-FUNKTION
408
412
AUSTAUSCH-FUNKTIONEN
ABSENDE-AUSTAUSCHFUNKTION
413
PROTOKOLLREGISTRIER-AUSTAUSCHFUNKTION
414
NETZWERKADAPTERERÖFFNUNGS-AUSTAUSCHFUNKTION
415
ANFORDERUNGS-AUSTAUSCHFUNKTION
416
ZUSTANDSANZEIGE-AUSTAUSCHFUNKTION
417
DATENÜBERTRAGUNGS-AUSTAUSCHFUNKTION
418
720'
720
701
REGISTRIERTES PROTOKOLL
702
NEUVERBINDEN DES NETZWERKADAPTERS

Fig. 7

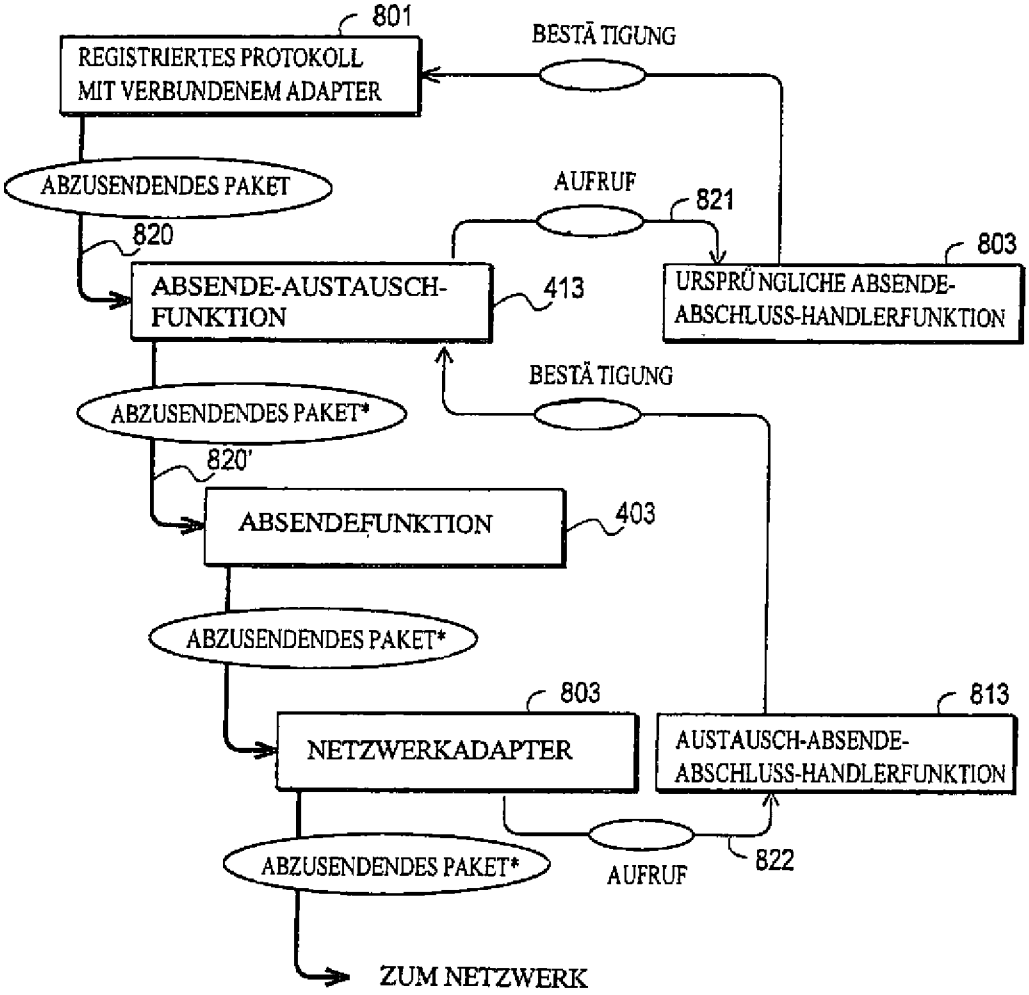

801
REGISTRIERTES PROTOKOLL MIT VERBUNDENEM ADAPTER
BESTÄTIGUNG
ABZUSENDENDES PAKET
AUFRUF
821
820
ABSENDE-AUSTAUSCH-FUNKTION
413
803
URSPRÜNGLICHE ABSENDE-ABSCHLUSS-HANDLERFUNKTION
BESTÄTIGUNG
ABZUSENDENDES PAKET*
820'
ABSENDEFUNKTION
403
ABZUSENDENDES PAKET*
803
NETZWERKADAPTER
813
AUSTAUSCH-ABSENDE-ABSCHLUSS-HANDLERFUNKTION
ABZUSENDENDES PAKET*
AUFRUF
822
ZUM NETZWERK

Fig. 8

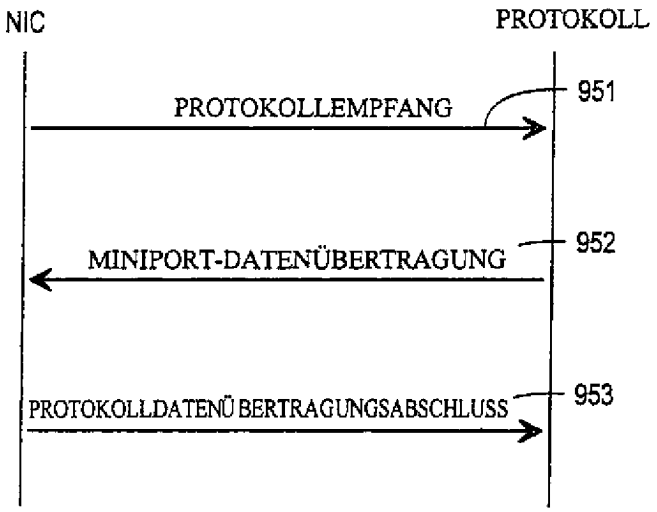


Fig. 9a
STAND DER TECHNIK

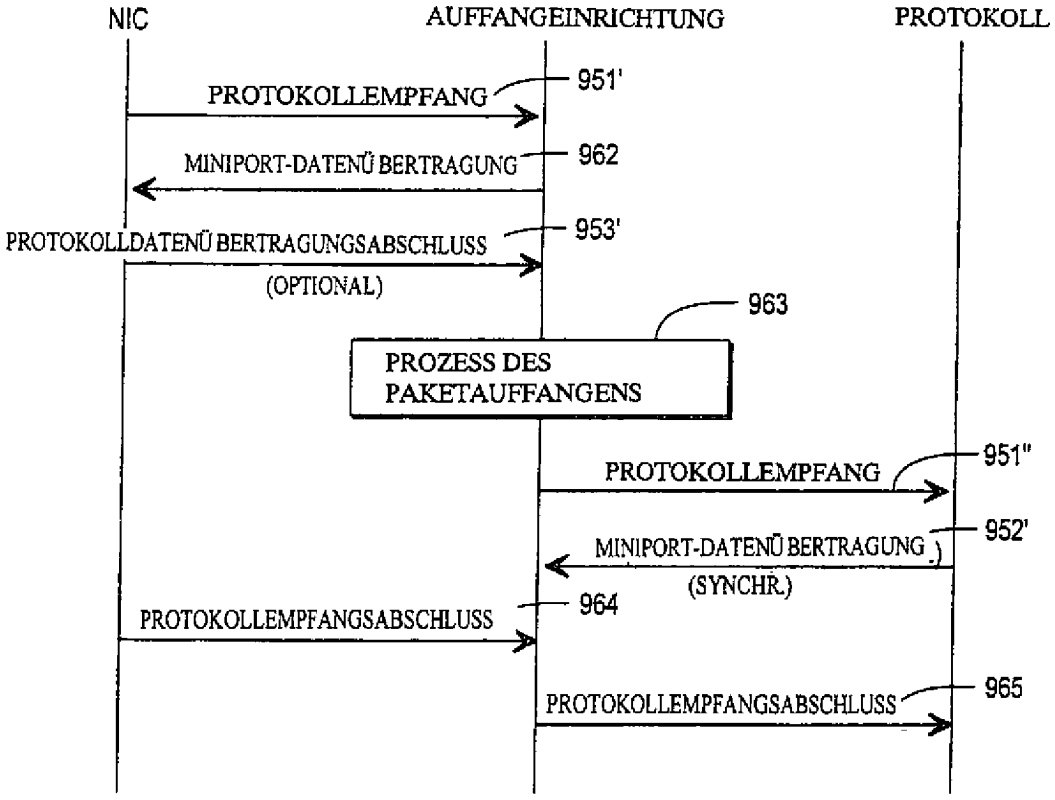


Fig. 9b

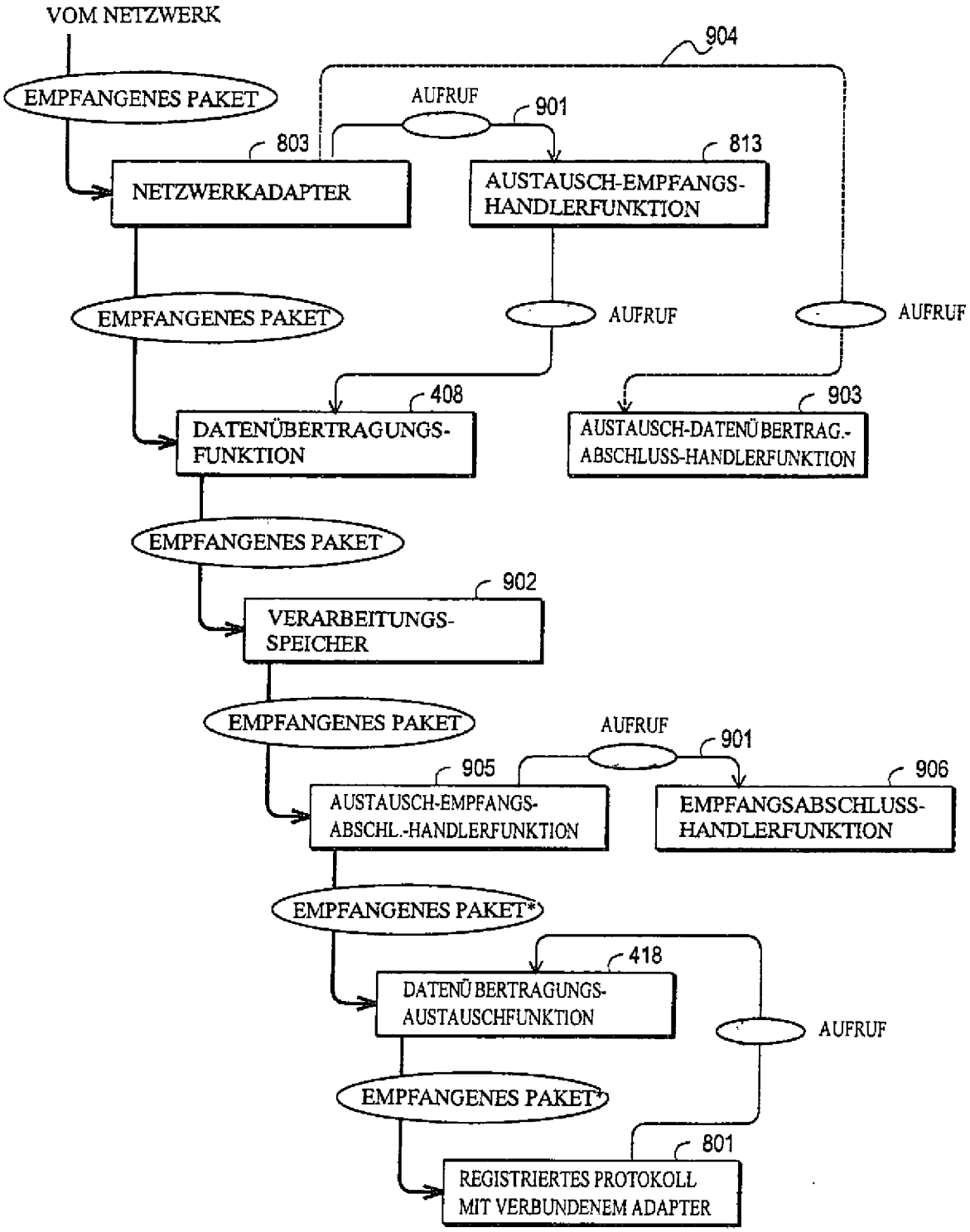

VOM NETZWERK
EMPFANGENES PAKET
904
AUFRUF
901
803
NETZWERKADAPTER
813
AUSTAUSCH-EMPFANGS-HANDLERFUNKTION
EMPFANGENES PAKET
AUFRUF
AUFRUF
408
DATENÜBERTRAGUNGS-FUNKTION
903
AUSTAUSCH-DATENÜBERTRAG.-ABSCHLUSS-HANDLERFUNKTION
EMPFANGENES PAKET
902
VERARBEITUNGS-SPEICHER
EMPFANGENES PAKET
AUFRUF
901
905
AUSTAUSCH-EMPFANGS-ABSCHL.-HANDLERFUNKTION
906
EMPFANGSABSCHLUSS-HANDLERFUNKTION
EMPFANGENES PAKET*
418
DATENÜBERTRAGUNGS-AUSTAUSCHFUNKTION
AUFRUF
EMPFANGENES PAKET*
801
REGISTRIERTES PROTOKOLL MIT VERBUNDENEM ADAPTER

Fig. 9c

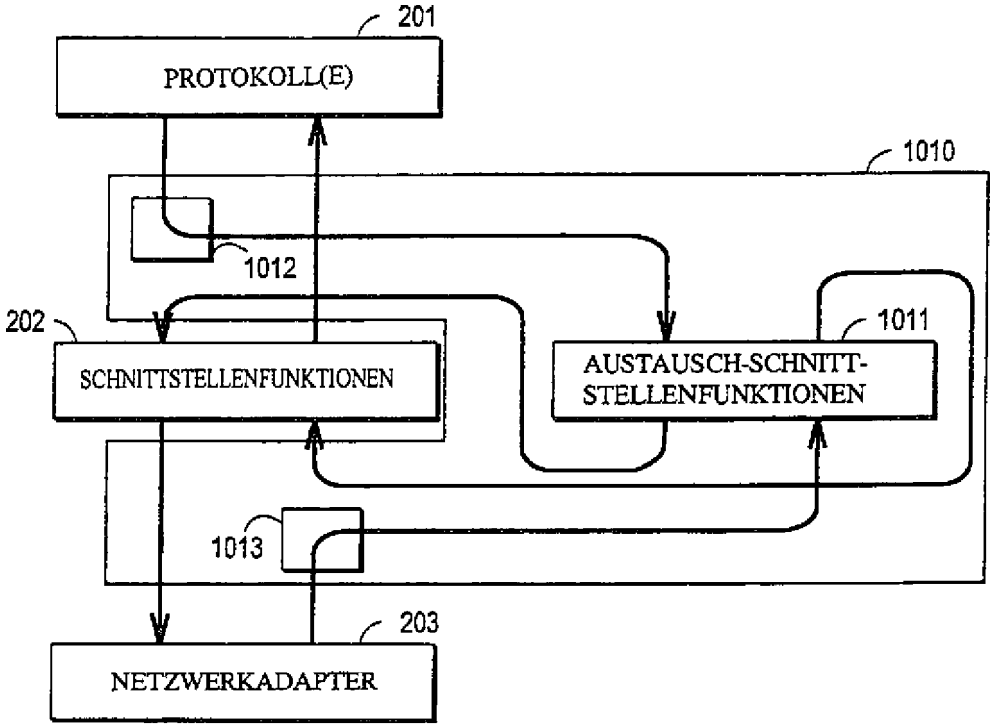

201
PROTOKOLL(E)
1010
1012
202
SCHNITTSTELLENFUNKTIONEN
1011
AUSTAUSCH-SCHNITT-
STELLENFUNKTIONEN
1013
203
NETZWERKADAPTER

Fig. 10

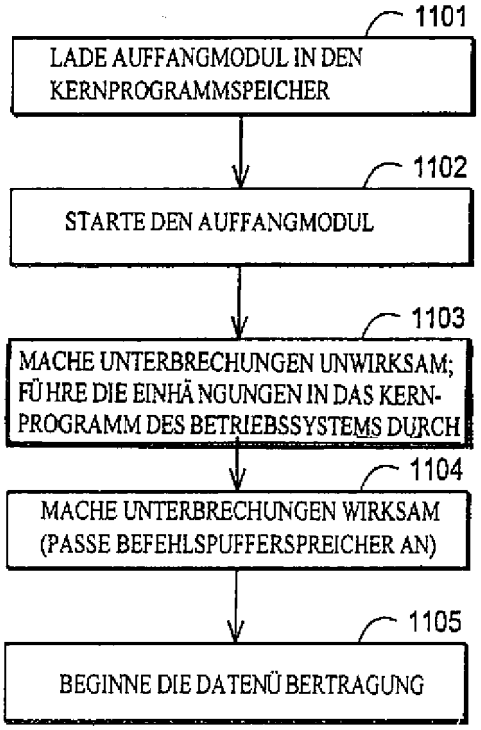

1101
LADE AUFFANGMODUL IN DEN
KERNPROGRAMMSPEICHER
1102
STARTE DEN AUFFANGMODUL
1103
MACHE UNTERBRECHUNGEN UNWIRKSAM;
FÜHRE DIE EINHÄNGUNGEN IN DAS KERN-
PROGRAMM DES BETRIEBSSYSTEMS DURCH
1104
MACHE UNTERBRECHUNGEN WIRKSAM
(PASSE BEFEHLSPUFFERSPREICHER AN)
1105
BEGINNE DIE DATENÜBERTRAGUNG

Fig. 11

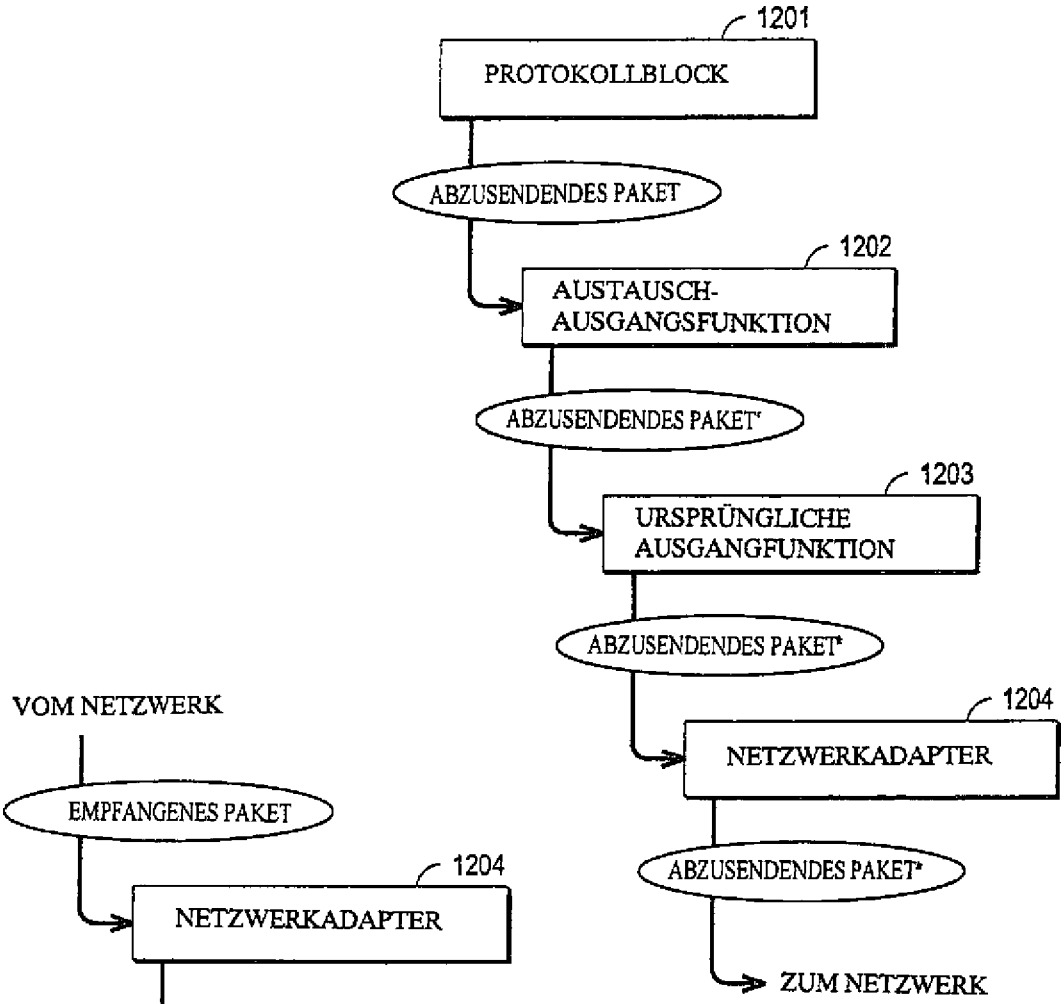


Fig. 12a

Fig. 12b

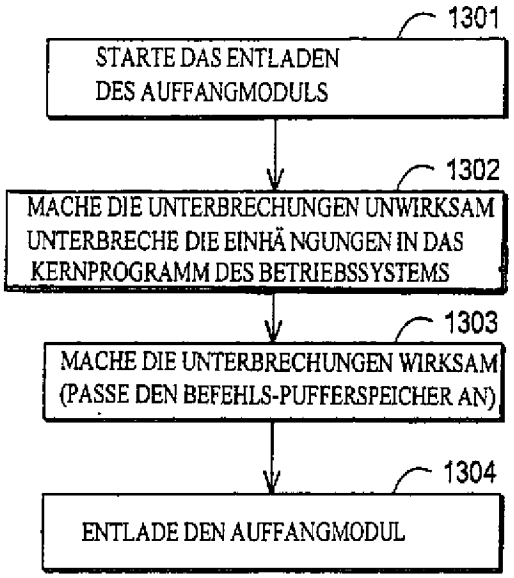

1301
STARTE DAS ENTLADEN DES AUFFANGMODULS
1302
MACHE DIE UNTERBRECHUNGEN UNWIRKSAM UNTERBRECHE DIE EINHÄNGUNGEN IN DAS KERNPROGRAMM DES BETRIEBSSYSTEMS
1303
MACHE DIE UNTERBRECHUNGEN WIRKSAM (PASSE DEN BEFEHLS-PUFFERSPEICHER AN)
1304
ENTLADE DEN AUFFANGMODUL

Fig. 13

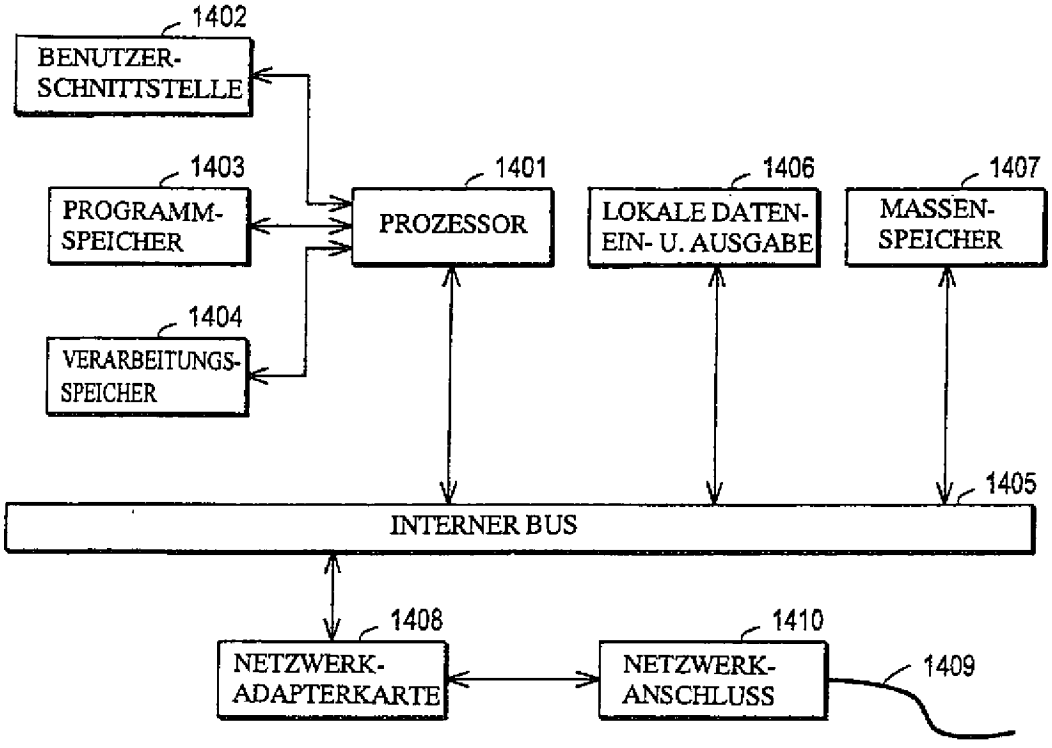

1402
BENUTZER-SCHNITTSTELLE
1403
PROGRAMM-SPEICHER
1401
PROZESSOR
1406
LOKALE DATEN-EIN- U. AUSGABE
1407
MASSEN-SPEICHER
1404
VERARBEITUNGS-SPEICHER
1405
INTERNER BUS
1408
NETZWERK-ADAPTERKARTE
1410
NETZWERK-ANSCHLUSS
1409

Fig. 14