



(19)
Bundesrepublik Deutschland
Deutsches Patent- und Markenamt

(10) **DE 603 16 213 T2** 2008.05.29

(12) **Übersetzung der europäischen Patentschrift**

(97) **EP 1 347 619 B1**

(21) Deutsches Aktenzeichen: **603 16 213.4**

(96) Europäisches Aktenzeichen: **03 251 798.9**

(96) Europäischer Anmeldetag: **21.03.2003**

(97) Erstveröffentlichung durch das EPA: **24.09.2003**

(97) Veröffentlichungstag

der Patenterteilung beim EPA: **12.09.2007**

(47) Veröffentlichungstag im Patentblatt: **29.05.2008**

(51) Int Cl.⁸: **B60R 25/00** (2006.01)

H04L 29/08 (2006.01)

H04L 12/24 (2006.01)

(30) Unionspriorität:

104246 22.03.2002 US

(73) Patentinhaber:

Sun Microsystems, Inc., Santa Clara, Calif., US

(74) Vertreter:

derzeit kein Vertreter bestellt

(84) Benannte Vertragsstaaten:

AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LI, LU, MC, NL, PT, RO, SE, SI, SK, TR

(72) Erfinder:

Mocek, Darryl J., San Jose, 95136-2011 California, US; Grigoriev, Nikolay Georgievich, St Petersburg 192212, RU; Petrovsky, Sergey, Moscow 111141, RU

(54) Bezeichnung: **System und Verfahren zum Speichern von Benutzerpräferenzen**

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

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

Beschreibung**HINTERGRUND DER ERFINDUNG****1. GEBIET DER ERFINDUNG**

[0001] Diese Erfindung betrifft allgemein netzwerkzentrierte Telematik-Dienste und Anwendungen, die für Fahrzeuge bestimmt sind, und insbesondere ein Telematiksystem, das konfiguriert ist, um die Speicherung von Benutzerpräferenzen zu gewährleisten.

2. BESCHREIBUNG DES STANDS DER TECHNIK

[0002] Der elektronische Inhalt und die Ausgereiftheit des Kraftfahrzeugdesigns haben deutlich zugenommen. In immer zahlreicher werdenden Entertainment-, Sicherheits- und Steuerfunktionen sind Mikroprozessoren vorherrschend geworden. Demnach spielt dieser elektronische Inhalt in den Absätzen und Einnahmen der Autohersteller eine zunehmende Rolle. Zu den Funktionen, die vom elektronischen Inhalt gewährleistet werden, gehören Audiosysteme, die Fahrzeugstabilitätskontrolle, fahreraktivierte Getriebebesteuern, adaptive Geschwindigkeitsregelung, Routenplanung, Kollisionswarnsysteme usw. Die signifikante Vermehrung des elektronischen Inhalts von Landfahrzeugen ist gleichzeitig mit dem explosiven Wachstum des Internets und der zugehörigen datengesteuerten Anwendungen aufgetreten, die über mobile Anwendungen bereitgestellt werden.

[0003] Die Telematik, ein Allgemeinbegriff, der sich auf fahrzeugsbasierte drahtlose Kommunikationssysteme und Informationsdienste bezieht, stellt die Kombination von Fahrzeugsicherheits-, Entertainment- und Komfortfunktionen durch drahtlosen Zugriff auf verteilte Netzwerke wie z. B. das Internet in Aussicht. Die Telematik erlaubt die Umstellung vom hardwarezentrierten Modell der Audio- und Fahrzeugsteuersysteme, die in Geräte eingebaut sind, die für jedes Fahrzeug spezifisch sind, auf Infotainment durch Plug-and-play-Hardware, deren Funktionalität durch Laden von Software oder einfachen Modulaustausch aktualisiert werden kann. Durch die Produkte und Dienste, die durch die Telematik verfügbar gemacht werden, werden zudem neue Einnahmequellen für die Autohersteller und Dienstleister erschlossen.

[0004] Da diese Infotainment-Systeme Entertainment und Information in einem gemeinsamen Gerät integrieren, müssen die Systeme hochintegriert, offen und konfigurierbar sein. Doch die elektronischen Systeme, die aktuell auf dem Markt sind, sind spezifisch für die Marke, das Modell, das Baujahr und die Weltregion, in der das Fahrzeug angeboten wird. Zusätzlich werden die elektronischen Systeme, die heute benutzt werden, durch proprietäre Busse mit streng begrenzten Bandbreiten verbunden, die für datenintensive Dienste, die Information, Entertainment

und Sicherheit kombinieren, ungeeignet sind. Die proprietären und spezifisch ausgeführten Systeme erfordern, dass einem Entwickler die zugrunde liegenden Software- und Hardware-Anwendungsprogrammchnittstellen (APIs) bekannt sind, damit er Anwendungen für künftige Infotainment-Systeme entwickeln kann. Doch zahlreiche proprietäre und spezifische Systeme sind in den verschiedenen Fahrzeugmarken und -modellen auf dem Markt verbreitet, und selbst im selben Modell von einem Baujahr zum anderen. Daher beseitigt die heterogene Beschaffenheit der verschiedenen Systeme im Wesentlichen alle Massenproduktionsvorteile, weil die Ausrüstung und Software an jede Modelländerung angepasst werden muss.

[0005] Ferner müssen die gewünschten Einstellungen oder Präferenzen eines Benutzers für die Konfiguration eines Telematiksystems gespeichert werden, damit der Benutzer die Präferenzen nur einmal einstellen muss. Zudem benötigt der Benutzer einen einfachen Zugriff, um Präferenzen hinzuzufügen, zu löschen oder zu ändern. Zusätzlich müssen Änderungen an den Präferenzen, während die Telematik-Steereinheit im Fahrzeug in einem Ruhezustand ist, derart gehandhabt werden, dass die Änderungen auf der Serverseite gespeichert werden, bis die Telematik-Steereinheit „geweckt“ wird. Deshalb ist das Telematiksystem im Fahrzeug nicht in der Lage, die Präferenzen zu aktualisieren, bevor das Fahrzeug eingeschaltet wird.

[0006] WO 01/31515 beschreibt ein System zur Fernsammlung von Daten von einer Lokomotive. Ein Bordmonitor ist an Bord der Lokomotive und unter der Steuerung eines Fernüberwachungs- und -diagnosezentrums angeordnet, der Bordmonitor sammelt periodisch Information von der Lokomotive und überträgt diese zum Fernüberwachungs- und -diagnosezentrum. Eine zum Fernüberwachungs- und -diagnosezentrum gehörige Konfigurationsdatei kann geändert werden, und diese Änderung kann die Steuerung des Bordmonitors beeinflussen. Die geänderte Konfigurationsdatei wird zum Bordmonitor übertragen.

[0007] US 6.092.020 beschreibt ein Verfahren zum Erhalt von Daten über eine Verkehrslage in einem Straßennetz, wobei Fahrzeuge mit einer sensorischen Analysevorrichtung ausgerüstet sind, die geeignete Daten sammelt. Die Daten, die gesammelt werden sollen, werden durch Konfigurationseinstellungen bestimmt.

[0008] US 6.028.537 offenbart ein Fahrzeugkommunikations- und Fernsteuerungssystem. Das beschriebene System weist verschiedene Aspekte auf, einschließlich eines Aspekts, der die Personalisierung des Fahrzeugs über ein Funkübertragungssystem erlaubt. Mit diesem System kann ein Benutzer, der ein Fahrzeug umzuprogrammieren wünscht, ei-

nen Dienstanbieter anrufen, der die Umprogrammierung durchführt, indem er Funksignale an das Fahrzeug sendet.

[0009] In Anbetracht des Vorstehenden besteht ein Bedarf nach einem System und einem Verfahren zum Speichern von Benutzer- und Systempräferenzen für eine Telematik-Steuereinheit, damit ein Benutzer und das System beliebig Änderungen an den Präferenzen vornehmen können.

KURZDARSTELLUNG DER ERFINDUNG

[0010] Die Erfindung wird durch ein System nach Anspruch 1, eine Telematik-Steuereinheit nach Anspruch 8, Verfahren nach den Ansprüchen 12 und 21 und Datenträger nach Anspruch 20 und 23 definiert.

[0011] Allgemein erfüllt die vorliegende Erfindung diesen Bedarf indem sie eine Speicherung für neue Präferenzen und Präferenzänderungen vorsieht, wobei die Präferenzen und die Änderungen in einem Client auf einem Server gespeichert werden und über synchronisierte Kanäle übertragen werden. Es ist anzumerken, dass die vorliegende Erfindung auf zahlreiche Weisen implementiert werden kann, einschließlich als eine Einrichtung, ein System, ein Gerät oder ein Verfahren. Mehrere erfindungsgemäße Ausführungsformen der vorliegenden Erfindung werden im Folgenden beschrieben.

[0012] Der vorliegenden Erfindung gemäß wird ein System zum Speichern von Präferenzen auf einem Telematik-Client bereitgestellt, umfassend: einen Telematik-Server, konfiguriert, um eine Anforderung zu empfangen, die Änderungsdaten für eine Präferenz enthält, und um die Änderungsdaten für die Präferenz in einem Speicher eines Präferenzservers des Telematik-Servers zu speichern, wobei der Telematik-Server ein serverseitiges Communications-Framework aufweist, das mit dem Präferenzserver in Verbindung steht, eine Telematik-Steuereinheit, die einen Präferenz-Manager zum Speichern der Präferenzen aufweist, wobei die Telematik-Steuereinheit ein clientseitiges Communications-Framework einschließt, das mit dem Präferenz-Manager in Verbindung steht, wobei der Präferenz-Manager und der Präferenzserver konfiguriert sind, um sich über eine Netzwerkverbindung zu synchronisieren, um die Übertragung der Änderungsdaten zwischen dem Präferenzserver und dem Präferenz-Manager zu erlauben, wobei die Telematik-Steuereinheit einen Softwarestapel umfasst, der eine Hardware-Layer, eine Operating System-Layer, eine Java Virtual Machine-Layer, eine Open Services Gateway Initiative-Layer und eine Java Telematics Client-Layer einschließt, und wobei die Telematik-Steuereinheit einen Channel-Monitor umfasst, der konfiguriert ist, um verfügbare Kanäle zu überwachen, und einen Datenmultiplexer und einen Flow-Controller mit einem Con-

nection-Objekt, konfiguriert, um auf der Basis der Verfügbarkeit auf einen Kanal im Netzwerk umzuschalten, der eine höhere Bandbreite aufweist. Der Telematik-Server kann ein Java Provisioning-Server sein.

[0013] Die Erfindung stellt ferner eine Telematik-Steuereinheit bereit, umfassend: einen Präferenz-Manager zum Speichern von Präferenzen, ein clientseitiges Communications-Framework, das mit dem Präferenz-Manager in Verbindung steht; einen Softwarestapel, der eine Hardware-Layer, eine Operating System-Layer, eine Java Virtual Machine-Layer, eine Open Services Gateway Initiative-Layer und eine Java Telematics Client-Layer einschließt, und einen Channel-Monitor, konfiguriert, um verfügbare Bandbreiten zu überwachen, wobei der Channel-Monitor ferner konfiguriert ist, um auf der Basis der Verfügbarkeit auf einen Kanal im Netzwerk umzuschalten, der eine höhere Bandbreite aufweist, wobei der Präferenz-Manager konfiguriert ist, um sich mit einem Präferenzserver eines Telematik-Servers zu synchronisieren, der Telematik-Server konfiguriert ist, um eine Anforderung zu empfangen, die Änderungsdaten für eine Präferenz enthält, die Änderungsdaten auf diesem Präferenzserver gespeichert werden und die Synchronisation die Übertragung der Änderungsdaten zwischen dem Präferenzserver und dem Präferenz-Manager erlaubt.

[0014] Der vorliegenden Erfindung gemäß wird auch ein Verfahren zum Speichern von Präferenzen in Verbindung mit einem Telematik-System bereitgestellt, umfassend: das Empfangen, in einem Telematik-Server, einer Anforderung, die Änderungsdaten für eine Präferenz enthält, das Speichern der Änderungsdaten im Speicher eines Präferenzservers dieses Telematik-Servers, das Synchronisieren des Präferenzservers mit einem Präferenz-Manager einer Telematik-Steuereinheit, wobei die Telematik-Steuereinheit einen Softwarestapel umfasst, der eine Hardware-Layer, eine Operating System-Layer, eine Java Virtual Machine-Layer, eine Open Services Gateway Initiative-Layer und eine Java Telematics Client-Layer umfasst, wobei die Telematik-Steuereinheit außerdem einen Channel-Monitor umfasst, konfiguriert, um die verfügbare Bandbreite zu überwachen, der Channel-Monitor außerdem konfiguriert ist, um auf der Basis der Verfügbarkeit auf einen anderen Kanal im Netzwerk umzuschalten, der eine höhere Bandbreite aufweist, das Übertragen der Änderungsdaten zum Präferenz-Manager und das Speichern der Änderungsdaten im Speicher der Telematik-Steuereinheit.

[0015] Der vorliegenden Erfindung gemäß wird ein Verfahren zum Speichern von Präferenzen in einer Telematik-Steuereinheit bereitgestellt, wobei die Telematik-Steuereinheit einen Präferenz-Manager zum Speichern von Präferenzen aufweist, ein clientseitiges Communications-Framework, das mit dem Prä-

ferenz-Manager in Verbindung steht, und einen Softwarestapel, der eine Hardware-Layer, eine Operating System-Layer, eine Java Virtual Machine-Layer, eine Open Services Gateway Initiative-Layer und eine Java Telematics Client-Layer einschließt, wobei die Telematik-Steuereinheit außerdem einen Channel-Monitor aufweist, der konfiguriert ist, um verfügbare Bandbreiten zu überwachen, wobei der Channel-Monitor ferner konfiguriert ist, um auf Basis der Verfügbarkeit auf einen Kanal im Netzwerk umzuschalten, wobei das Verfahren umfasst: das Synchronisieren des Präferenz-Managers mit einem Präferenzserver eines Telematik-Servers, wobei diese Synchronisation bewirkt, dass die Änderungsdaten zwischen dem Präferenzserver und dem Präferenz-Manager übertragen werden, wobei der Telematik-Server konfiguriert ist, um eine Anforderung zu empfangen, die Änderungsdaten für eine Präferenz enthält, wobei die Änderungsdaten auf dem Präferenzserver gespeichert werden.

KURZBESCHREIBUNG DER ZEICHNUNGEN

[0016] Die Erfindung und ihre anderen Vorteile gehen aus der folgenden Beschreibung in Verbindung mit den beiliegenden Zeichnungen hervor.

[0017] [Fig. 1](#) ist eine codeunabhängige schematische Übersicht eines Kraftfahrzeug-Telematik-Systems nach einer Ausführungsform der Erfindung.

[0018] [Fig. 2](#) ist ein schematisches Diagramm eines Telematik-Clients, der über ein drahtloses Netzwerk mit einem Telematik-Server nach einer Ausführungsform der Erfindung kommuniziert.

[0019] [Fig. 3](#) ist eine dreidimensionale bildliche Darstellung einer Telematik-Client-Implementierung des clientseitigen Stapels von [Fig. 2](#) nach einer Ausführungsform der vorliegenden Erfindung.

[0020] [Fig. 4](#) ist ein codeunabhängiges schematisches Diagramm der Interaktion zwischen einem Carlet und einem Communications-Framework auf der Clientenseite eines Telematic-Systems nach einer Ausführungsform der Erfindung.

[0021] [Fig. 5](#) ist ein codeunabhängiges schematisches Diagramm eines Telematik-Systems, das einen Präferenzdienst zur Speicherung von Präferenzen nach einer Ausführungsform der Erfindung bereitstellt.

[0022] [Fig. 6](#) ist ein schematisches Diagramm, das eine zweidimensionale Ansicht von Lagers des Softwarestapels der Telematik-Steuereinheit (TCU) nach einer Ausführungsform der Erfindung zeigt.

[0023] [Fig. 7](#) veranschaulicht die Kommunikation zwischen einer Carlet-Anwendung und dem Commu-

nications-Framework auf der Clientseite nach einer Ausführungsform der Erfindung.

[0024] [Fig. 8](#) ist ein Flussdiagramm der Verfahrensschritte, die zum Einstellen einer Präferenz in einer Telematik-Steuereinheit nach einer Ausführungsform der Erfindung durchgeführt werden.

AUSFÜHRLICHE BESCHREIBUNG BEVORZUGTER AUSFÜHRUNGSFORMEN

[0025] Es wird eine Erfindung für ein Telematik-System offenbart, das die Fähigkeit hat, Präferenzen und Änderungsdaten für die Präferenzen zu speichern. In der folgenden Beschreibung werden zahlreiche spezifische Details dargelegt, um ein gründliches Verständnis der vorliegenden Erfindung zu ermöglichen. Doch für den Fachmann versteht es sich, dass die vorliegende Erfindung auch ohne all diese spezifischen Details in die Praxis umgesetzt werden kann. In anderen Fällen wurden wohlbekannte Verfahrensschritte nicht im Detail beschrieben, um das Verständnis der vorliegenden Erfindung nicht unnötig zu erschweren.

[0026] Die hierin beschriebenen Ausführungsformen der vorliegenden Erfindung stellen ein System und ein Verfahren zum Speichern von Präferenzen bereit, die von einem Benutzer in Bezug auf ein Telematik-System für ein Fahrzeug definiert wurden. Wie weiter unten ausführlicher erläutert, erlauben die Präferenzen für die Konfiguration einer Telematik-Steuereinheit die gewünschte Funktionalität, die vom Benutzer vorgegeben werden, bereitzustellen. Das heißt, wenn der Benutzer die Voreinstellung bestimmter Radiosender, Klimabedingungen im Fahrgeastraum, Infotainmenteeinstellungen, Komforteinstellungen usw. wünscht, kann der Benutzer diese gewünschten Einstellungen als Präferenzen spezifizieren. Diese Präferenzen müssen so gespeichert werden, dass die Präferenzen jedes Mal, wenn der Benutzer das Fahrzeug startet, eingestellt werden. In einer Ausführungsform werden die Präferenzen in einen Server eingegeben, hochgeladen oder dort geändert, wenn die Telematik-Steuereinheit des Fahrzeugs sich in einem Ruhemodus befindet, weshalb die Präferenzen auf dem Server gespeichert sind und nach dem Start und der Herstellung einer Kommunikationsverbindung mit dem Client zum Client übertragen werden. Es ist anzumerken, dass die Präferenzen auch vom Client gewählt und zum Server hochgeladen werden können.

[0027] Eine Telematik-Steuereinheit (TCU) im Fahrzeug kann mit jedem der Steuersysteme, Sicherheitssysteme, Entertainmentssysteme, Informationssysteme usw. des Fahrzeugs verbunden sein. Für den Fachmann versteht es sich, dass der clientseitige Stapel der TCU verwendet wird, um auf eine Fahrzeugschnittstellenkomponente für den Zugriff auf Ge-

rate im Fahrzeug wie z. B. Geschwindigkeits-, Drehzahl-, Öldruck-, Reifendruckanzeiger usw. zuzugreifen. Dadurch sorgen clientseitige Anwendungen, die in der TCU sitzen, für die Funktionalität in Bezug auf die Fahrzeugsysteme sowie auf Infotainmentanwendungen.

[0028] In einer Ausführungsform wendet das Telematik-System die Java-Technologie an. Es ist anzumerken, dass die Plattformunabhängigkeit und das überlegene Sicherheitsmodell der Java-Technologie eine plattformübergreifende Lösung für die heterogenen Systeme eines Fahrzeugs bereitstellen, während sie eine Sicherheitsarchitektur bietet, die vor Viren und unberechtigtem Zugriff schützt. Dadurch ist der Inhalt- oder Dienstanbieter von den unzähligen Autoplattformen isoliert, während Fahrzeughersteller vor Hackerangriffen geschützt sind. Zudem sind Java-Anwendungsprogrammierschnittstellen (APIs) verfügbar, um Telematik-Medien zu unterstützen, wie z. B. Spracherkennung durch die Java Speech API (JSAPI), Medienbereitstellung durch das Java Media Framework (JMF) und drahtlose Telefonie durch Wireless Telephony Communications APIs (WTCA), usw.

[0029] [Fig. 1](#) ist eine codeunabhängige schematische Übersicht eines Kraftfahrzeug-Telematik-Systems nach einer Ausführungsform der Erfindung. Eine Client/Server-Architektur, die sich auf die Standards und Prinzipien des modularen Designs stützt, erlaubt die Bereitstellung der Funktionalität des Telematik-Systems an den Kunden durch drahtlosen Zugriff. Die Serverseite umfasst einen Java Provisioning Server (JPS) **106**, der mit dem Netzwerk **104** in Verbindung ist.

[0030] Die Clientseite schließt die Telematik-Steuerereinheit (TCU) **102** ein, die in der Karosserie des Landfahrzeugs **100** enthalten ist. Die TCU **102** ist in der Lage, durch drahtlosen Zugriff mit dem Netzwerk **104** zu kommunizieren. Selbstverständlich kann das Netzwerk **104** jedes verteilte Netzwerk sein, wie z. B. das Internet, und das Wireless Access Protocol (WAP) kann ein Protokoll sein, das geeignet ist, um der TCU **102** genügend Bandbreite zur Kommunikation mit dem Netzwerk bereitzustellen. Es ist anzumerken, dass die Client/Server-Architektur von [Fig. 1](#) die Weiterentwicklung von festverdrahteten, abgeschlossenen Komponenten zu plattformbasierten Angeboten erlaubt, die sich auf Software und Updates stützen. Dadurch kann ein Dienstanbieter, der den JPS **106** steuert, eine urgebündelte, offene End-to-End-Lösung anbieten, die Plug-and-play-Anwendungen ermöglicht. Zum Beispiel kann der Dienst ein Dienst auf Rang-Basis sein, ähnlich der Satelliten oder Kabel Dienste. Für den Fachmann versteht es sich, dass eine offene Plattform wie z. B. Frameworks auf der Basis der Java-Technologie es einem Entwickler erlauben, ausführbare Programme

ohne Rücksicht auf das zugrunde liegende Hardware- oder Betriebssystem zu erstellen. Auch wenn [Fig. 1](#) ein Auto darstellt, ist anzumerken, dass die TCU **102** in jedes Fahrzeug oder in jeden Transportmodus integriert werden kann, ob an Land oder nicht. Zum Beispiel kann die TCU **102** in ein Boot, ein Flugzeug oder einen Hubschrauber usw. integriert werden.

[0031] [Fig. 2](#) ist ein schematisches Diagramm eines Telematik-Clients, der über ein drahtloses Netzwerk mit einem Telematik-Server nach einer Ausführungsform der Erfindung kommuniziert. Der clientseitige Stapel **110** umfasst die benötigten Lagers für eine Client-Anwendung, auch Manager oder Carlet genannt, die ausgeführt wird, um Funktionalität zu gewährleisten. Wie weiter unten erläutert, hat das Carlet Zugriff auf jede Lager des clientseitigen Stapels **110**. Im clientseitigen Stapel **110** ist das Client-Communications-Framework **112** enthalten. Das Client-Communications-Framework **112** erlaubt die Kommunikation zwischen dem clientseitigen Stapel **110** und einer Anwendung auf dem Server **116** über das Netzwerk **114**. Es ist anzumerken, dass der Server **116** nicht auf eine drahtlose Kommunikation eingeschränkt ist. Zum Beispiel kann der Server **116** mit dem Netzwerk **114** festverdrahtet sein. Für den Fachmann versteht es sich, dass die Kommunikation über das Server-Communications-Framework **118** erfolgt, wenn der Server **116** über eine drahtlose Verbindung mit dem Netzwerk **114** kommuniziert. In Bezug auf eine Ausführungsform, wo der Server **116** mit einem Netzwerk **114** festverdrahtet ist, kann der Server über ein Netzwerkportal (z. B. das Internet) mit dem Netzwerk **114** kommunizieren, statt über das Server-Communications-Framework **118**. Zudem kann das Netzwerk **114** jedes geeignete verteilte Netzwerk sein, wie z. B. das Internet, ein lokales Netz (LAN), ein Stadtnetz (MAN), ein Weitverkehrsnetz (WAN) usw.

[0032] [Fig. 3](#) ist eine dreidimensionale bildliche Darstellung einer Telematik-Client-Implementierung des clientseitigen Stapels von [Fig. 2](#) nach einer Ausführungsform der vorliegenden Erfindung. Die clientseitige Implementierung **121** schließt die Hardware-Layer **120** des Clients mit einem Embedded Board ein, das eine Telematik-Steuerereinheit (TCU) enthält. Wie in Bezug auf [Fig. 1](#) erwähnt, ist die TCU in ein Landfahrzeug eingebaut. In einer Ausführungsform ist die TCU durch einen Fahrzeugbus oder andere Kommunikationsmittel mit den elektronischen Komponenten des Fahrzeugs verbunden. Diese Komponenten schließen die Messung der Betriebs- und Sicherheitsparameter des Fahrzeugs, wie z. B. Reifendruck, Geschwindigkeit, Öldruck, Motortemperatur usw., sowie Information- und Entertainment-Komponenten wie z. B. Audiosystemeinstellungen, Internetzugang, Klimatisierung im Fahrgastraum, Sitzstellungen usw. ein. Für den Fachmann versteht es sich, dass die Telematik-Steuerereinheit in

der Lage ist, die Funktionalität verschiedener tragbarer Information- und Entertainment (Infotainment)-Geräte wie z. B. Mobiltelefone, Personal Digital Assistants (PDA), MP3-Player usw. zu integrieren.

[0033] Weiter Bezug nehmend auf [Fig. 3](#), liegt die Operating System-Layer **122** auf der obigen Hardware-Layer **120**. Die Java Virtual Maschine (JVM)-Layer **124** liegt auf der Operating System (OS)-Layer **122**, und die Open Services Gateway Initiative-Layer **126** liegt auf der JVM-Layer. Es ist anzumerken, dass der Standard für die JVM-Layer **124** die Java 2 Platform Micro Edition (JEM2), Connected Device Configuration (CDC), Connected Limited Device Configuration (CLDC), Foundation Profile, Personal Profile oder Personal Basis Profile einschließt. Für den Fachmann versteht es sich, dass das J2ME Foundation Profile ein Satz von APIs ist, die für Anwendungen bestimmt sind, die auf kleinen Geräten ausgeführt werden, die eine Art von Netzwerkverbindung aufweisen, während das J2ME CDC-Personal Profile oder Personal Base Profile die J2ME CDC-Umgebung solcher Geräte mit einer Anforderung nach hoher Internet-Konnektivität und Web-Fidelity versieht. Die Standards für jede dieser Lagers des Stapels sind auf der rechten Seite der clientseitigen Implementierung **121** angegeben. Das heißt, OSGI **126a**, J2ME CDC **124a**, OS **122a** und Embedded Board **120a** sind Standards und links von den Standards sind Beispiele für tatsächliche Produkte angeführt, die diese Standards implementieren. Zum Beispiel wird der OSGI **126a**-Standard durch Java Embedded Server (JES) 2.1 **126b** von Sun implementiert, J2ME **124a** wird durch Virtual Machine **124b** von Insignia implementiert, OS **122a** wird durch das Echtzeitbetriebssystem VxWorks **122b** von Wind Rivers implementiert, und Embedded Board **120a** wird durch ein Embedded Board auf Personal Computer-Basis wie z.B. SH4 von Hitachi implementiert. Es ist anzumerken, dass die tatsächlichen Produkte nur beispielhaft und nicht einschränkend sind, da jedes geeignete Produkt verwendet werden kann, das die Standards implementiert.

[0034] Carlets **132** von [Fig. 3](#) haben Zugriff auf jede Lager über und einschließlich der OS-Layer **122**. Die Anwendungsprogrammierschnittstelle (API) **130** ist die Lager, die von Carlets benutzt wird, um mit dem JTC zu kommunizieren. Die Service Provider Interface (SPI)-Layer **128** ist eine private Schnittstelle, welche Manager untereinander aufweisen. Für den Fachmann versteht es sich, dass die OSGI-Layer **126** eine Grundstruktur bereitstellt, auf der Anwendungen laufen können. Zusätzliche Funktionalität wie z. B. Lifecycle-Management wird über der JVM von der OSGI-Layer **126** bereitgestellt. Es ist anzumerken, dass die Open Services Gateway Initiative eine branchenübergreifende Arbeitsgruppe ist, die einen Satz offener APIs für ein Service Gateway für ein Telematik-System definiert. Diese APIs bestehen aus

einem Satz Kern Framework-APIs. Um Services und ihre Implementierungen anzuwenden, definiert OSGI eine Paketeinheit, Service Bundle genannt. Ein Service Bundle ist eine Java-Archiv (JAR)-Datei, die einen Satz Service-Definitionen zusammen mit ihrer entsprechenden Implementierung enthält. Sowohl Infrastructure Services als auch Carlets werden als Service Bundles angewandt. Ein Teil der Funktionalität für die Arbitrierung, Steuerung und Verwaltung von Geräten und Ressourcen, z. B. Lautsprechern, Mobiltelefonen usw., erfolgt durch die OSGI-Schicht **126**. Für den Fachmann versteht es sich, dass auch ein separater Arbitration Service erforderlich sein kann. Hierin ist ein Carlet eine Java-Anwendung. Für jede Funktion oder Task, die auf der Clientseite oder zwischen der Client- und Serverseite zu verarbeiten ist, wird ein Carlet aufgerufen, um den Vorgang zu verwalten. Auf diese Weise können Carlets auf unabhängige Weise geschrieben, getestet und zum Gebrauch auf einem Telematik-System eingeführt werden. Zum Beispiel kann ein Carlet geschrieben werden, um die Aktivität von Fahrzeugkomponenten (z. B. Reifen, Motoröl, Scheibenwischeraktivität, Lenkfestigkeit, Wartungsempfehlungen, Airbagsteuerung, Getriebesteuerung usw.) zu steuern oder zu überwachen, und um Anwendungen zu steuern und zu überwachen, die von der Telematik-Steuereinheit (TCU) verarbeitet werden und über den Bordmonitor des Kraftfahrzeugs beeinflusst werden. Als solches können spezialisierte Carlets geschrieben werden, um das Audiosystem, Entertainment-Module (wie z. B. Online-Spiele oder Filme), Spracherkennung, Telekommunikation, Email-Nachrichten (text- und sprachgesteuert) usw. zu steuern. Demnach ist die Art der Carlets, die geschrieben werden können, nicht begrenzt. Carlets können vorinstalliert oder von einem Server heruntergeladen werden. Ein Carlet kann eine API, die von anderen Carlets aufgerufen werden kann, aufweisen oder nicht, und es kann eigene Threads aufweisen, die ausgeführt werden, oder nicht.

[0035] [Fig. 4](#) ist ein codeunabhängiges schematisches Diagramm der Interaktion zwischen einem Carlet und einem Communications-Framework nach einer Ausführungsform der Erfindung. Es ist anzumerken, dass die Serverseite ein entsprechendes Communications-Framework aufweist, um die synchronisierte Kommunikation zwischen der Clientseite (z. B. eine Telematik-Steuereinheit in einem Fahrzeug) und der Serverseite (z. B. einem Java Telematics Server) herzustellen und zu ermöglichen. Das Communications-Framework **416** umfasst einen Message Manager **417**, einen Stream-Manager **419**, einen Datenmultiplexer und Flow-Controller **415**, einen Policy-Manager **420**, einen Channel-Monitor **422** und eine Schnittstelle zu den verschiedenen physikalischen Kanälen, die dem Communications-Framework auf der Clientseite zur Verfügung stehen.

[0036] Weiter auf [Fig. 4](#) Bezug nehmend, wenn eine bestimmte Carlet-Anwendung **402** angefordert wird, kommuniziert das Carlet mit dem Stream-Manager **419** und fordert die Herstellung einer Verbindung an. In Antwort darauf fordert der Stream-Manager **419** ein Connection-Object (Conn. OBJ) **418a** vom Datenmultiplexer und Flow-Controller **415** an. Sobald ein Kanal, der die Anforderung erfüllt, verfügbar ist, gibt der Datenmultiplexer und Flow-Controller **415** ein Connection-Object (Conn. OBJ) **418b** an das Carlet zurück. Dadurch wird über die Connection-Objects **418a** und **418b** des Datenmultiplexers und Flow-Controllers **415** eine Kommunikationsverbindung mit der Carlet-Anwendung **402** hergestellt. In einer Ausführungsform weist das Connection-Object **418a** des Datenmultiplexers und Flow-Controllers **415** die Fähigkeit auf, zwischen Kanälen **425**, die dem Communications-Framework **416** der Clientseite verfügbar sind, umzuschalten. Hier erlaubt der im Policy-Manager enthaltene Code die Wahl verschiedener Kanäle auf der Basis der Verfügbarkeit, der gewünschten Kommunikationsart, des Bandbreitenbedarfs für eine oder mehrere Datenübertragungen, die Zahlung einer Bandbreitengebühr, der Art des Abonnements, usw.

[0037] [Fig. 5](#) ist ein codeunabhängiges schematisches Diagramm eines Telematik-Systems, das einen Präferenzdienst zur Speicherung von Präferenzen nach einer Ausführungsform der Erfindung bereitstellt. Hier kann ein Benutzer eine Präferenz von einer Arbeitsstation wie einem Personal Computer **440** aus einstellen. Der Personal Computer **440** ist mit dem Netzwerk **446** verbunden, um auf die Webseite **442** zuzugreifen, die einem Benutzer erlaubt, Präferenzen einzustellen oder zu ändern. Zum Beispiel kann ein Benutzer wünschen, von einem Heimcomputer aus einen Radiosender als Präferenz einzustellen, sodass der Radiosender als Präferenz verfügbar ist, wenn das Fahrzeug des Benutzers, das eine Telematik-Steuereinheit (TCU) enthält, das nächste Mal gestartet wird. In einer Ausführungsform enthält die Webseite **442** eine Liste von Präferenzen **444** des Benutzers. Natürlich kann der Benutzer auch eine Präferenz angeben, statt sie beim Anpassen der Präferenzen aus einer Liste auszuwählen. Der Benutzer hat die Möglichkeit, Präferenzen **444** zu ändern, hinzuzufügen oder aus der Liste zu löschen. Für den Fachmann versteht es sich, dass ein Benutzer auch über ein tragbares Gerät **441** wie z. B. ein PDA, ein Web-fähiges Mobiltelefon usw. auf die Webseite **442** zugreifen kann. Wenn der Benutzer im Beispiel wünscht, die Präferenz **444** zu Präferenzen hinzuzufügen, die in der TCU des Fahrzeugs gespeichert sind, wird eine Anforderung zum Speichern der Präferenz **444** über das Netzwerk **446** an den Java Provisioning Server (JPS) **448** gesendet. Der JPS **448** speichert die Daten für die Präferenz **444** im Speicher **450**. Es ist anzumerken, dass der JPS **448** eine Komponente des Backends (Serverseite) des

Telematik-Systems ist.

[0038] Weiter Bezug nehmend auf [Fig. 5](#), umfasst der JPS **448** ein Communications-Framework, konfiguriert, um einen Präferenzserver des JPS über ein Communications-Framework der TCU mit einem Präferenz-Manager der TCU **452** zu synchronisieren, wie Bezug nehmend auf [Fig. 7](#) ausführlicher erläutert. Bei der Synchronisation des Präferenzservers und des Präferenz-Managers werden die Daten für die Präferenz **444** über das Netzwerk **446** an den Speicher **454** der TCU **452** (Clientseite) übertragen. In einer Ausführungsform erfolgt die Synchronisation zwischen dem Präferenzserver und dem Präferenz-Manager, wenn das die TCU **452** enthaltende Fahrzeug gestartet wird. Sobald die Daten für die Präferenz **444** in der TCU **452** gespeichert worden sind, ist die Präferenz für den Benutzer verfügbar. Das heißt, wenn die Präferenz ein Radiosender ist, ist der Radiosender für den Benutzer verfügbar. Es ist anzumerken, dass Präferenzen entweder Systempräferenzen **456a** oder Benutzerpräferenzen **456b** sein können. Systempräferenzen schließen Daten ein, die vom Telematik-System gespeichert werden, wie z. B. eine Fahrzeug-Identifizierungsnummer (VIN), eine Zulassungsnummer des Fahrzeugs, eine Telefonnummer zur Verbindung mit dem JPS **448** usw. Benutzerpräferenzen schließen Infotainment- oder Anpassungspräferenzen ein, d. h. Präferenzen, die die Information und das Entertainment betreffen, wie z. B. Radiosender-Einstellungen, Klimatisierung, Email, Nachrichten usw. In einer Ausführungsform sind Anwendungspräferenzen eingeschlossen. Für den Fachmann versteht es sich, dass Anwendungspräferenzen Präferenzen sind, welche die Anwendung für sich selbst speichert.

[0039] [Fig. 6](#) ist ein schematisches Diagramm, das eine zweidimensionale Ansicht der Lagers des Softwarestapels der Telematik-Steuereinheit nach einer Ausführungsform der Erfindung zeigt. Der Softwarestapel der TCU **452** definiert Standards für jede Layer. Das heißt, die Operating System (OS)-Layer **122a** liegt auf der Hardware (HW)-Layer **120**. Die Java Virtual Maschine (JVM)-Layer **124a** liegt auf der OS-Layer **122a**, und die Open Services Gateway Initiative (OSGI)-Layer **126a** liegt auf der JVM-Layer **124a**. Die Java Telematics Client (JTC)-Layer **460** liegt auf der OSGI-Layer **126a**. In der JTC-Layer **460** ist ein User Interface (UI)-Manager **464**, ein Communications-Framework **517a** und ein Präferenz-Manager **466** eingeschlossen. Es ist anzumerken, dass der UI-Manager **464** eine Benutzerschnittstellenanzeige im Fahrzeug ermöglicht, damit ein Benutzer die TCU beeinflussen kann. Natürlich kann die Benutzerschnittstelle auch nur eine Sprachschnittstelle sein. Das Communications-Framework **516a** der TCU **452** ist konfiguriert, um über das Netzwerk **446** mit dem Communications-Framework **516b** des JPS **448** zu kommunizieren, und wird Bezug nehmend auf [Fig. 7](#)

ausführlicher erläutert. Carlets C_1 bis C_n **462** sind Client-Anwendungen, die die Funktionalitäten gewährleisten, die ein Benutzer in einer Ausführungsform der Erfindung durch die Präferenzen des Präferenz-Managers spezifiziert. Es ist anzumerken, dass die Synchronisation zwischen den Präferenzen auf der Clientseite und den Präferenzen auf der Serverseite durch die Kommunikation zwischen dem Präferenz-Manager der Clientseite und dem Präferenzserver der Serverseite erfolgt. Die Kommunikationsverbindung wird über ein Netzwerk durch ein Communications-Framework **516a** auf der Clientseite und **516b** auf der Serverseite hergestellt.

[0040] [Fig. 7](#) veranschaulicht die Kommunikation zwischen einer Carlet-Anwendung und dem Communications-Framework auf der Clientseite nach einer Ausführungsform der Erfindung. Der Einfachheit halber werden die detaillierten Komponenten des Communications-Frameworks **516a** nur von der Perspektive der Clientseite aus gezeigt, obwohl es sich versteht, dass die Serverseite ein ähnliches Communications-Framework **516b** aufweist, um die synchronisierte Kommunikation zwischen der Clientseite (z. B. einer Telematik-Steuereinheit in einem Fahrzeug) und der Serverseite (z. B. einem Java Telematics Provisioning Server) herzustellen und zuzulassen. Der Java Provisioning Server ist für die Kommunikation mit jeder Zahl von Clients verantwortlich. Solche Clients können für jedes Fahrzeug, jede Marke, jedes Modell usw. vorgesehen sein, während die Clientseite für ein bestimmtes Fahrzeug und eine bestimmte TCU-Implementierung spezifisch ist.

[0041] Das Communications-Framework **516a** umfasst einen Message-Manager **517**, einen Stream-Manager **519**, einen Datenmultiplexer und Flow-Controller **518c** (d. h., um als Datenpumpe zu dienen), einen Policy-Manager **520**, einen Channel-Monitor **522** und eine Schnittstelle zu den verschiedenen physikalischen Kanälen, die dem Communications-Framework auf der Clientseite zur Verfügung stehen. Eine Synchronisationssteuerung **527** ist vorgesehen, um als Schnittstelle zwischen der Clientseite und der Serverseite zu dienen. Das heißt, die Synchronisationssteuerung **527** erlaubt die Kommunikation zwischen dem Datenmultiplexer und Flow-Controller **518c** der Clientseite und dem Datenmultiplexer und Flow-Controller **518s** der Serverseite.

[0042] Wenn in Betrieb eine bestimmte Carlet-Anwendung **502** angefordert wird, kommuniziert **515** das Carlet mit dem Stream-Manager **519** und fordert die Herstellung einer Verbindung an. In der Anforderung schließt das Carlet in einer Ausführungsform Eigenschaften ein, die angeben, welche Art von Verbindung benötigt wird, um den Bandbreitenbedarf des Carlets zu decken. Wenn das Carlet ein MP3-Carlet ist, können die Eigenschaften, wie oben erwähnt,

eine bestimmte Mindestübertragungsrate bezeichnen. In Antwort darauf fordert der Stream-Manager **519** ein Connection-Object (Conn. OBJ) **518a** vom Datenmultiplexer und Flow-Controller **518c** an. Wenn ein Kanal, der die gewünschte Bandbreite erfüllt, verfügbar ist, gibt der Datenmultiplexer und Flow-Controller **518c** ein Connection-Object (Conn. OBJ) **514a** an das Carlet zurück. Der Message-Manager **517** wird allgemein benutzt, um Connection-Objects für die Einwegkommunikation zu erhalten, ähnlich wie bei einem Einweg-Email.

[0043] Dadurch wird über die Connection-Objects **514a** und **518b** des Datenmultiplexers und Flow-Controllers **518c** eine Kommunikationsverbindung mit der Carlet-Anwendung **502** hergestellt. In einer Ausführungsform weist das Connection-Object **518a** des Datenmultiplexers und Flow-Controllers **518c** die Fähigkeit auf, zwischen Kanälen **525**, die dem Communications-Framework **516a** der Clientseite verfügbar sind, umzuschalten. Zum Beispiel kann das Connection-Object **518a** des Datenmultiplexers und Flow-Controllers zuerst eine Verbindung **524** zu einem Kanal 1 (CH 1) herstellen. Die Verbindung **524** kommuniziert dadurch mit einem komplementären Kanal 1 (CH 1) auf der Serverseite. Der Policy-Manager **520** ist bevorzugt eine plug-fähige Richtlinie, die an die jeweilige Anwendung oder der Benutzerspezifikationen entsprechend angepasst ist. Zum Beispiel kann der Policy-Manager Code enthalten, der die Wahl verschiedener Kanäle abhängig von der Verfügbarkeit, der gewünschten Kommunikationsart, dem Bandbreitenbedarf für eine oder mehrere Datenübertragungen, der Zahlung einer Bandbreitengebühr, der Art des Abonnements usw. ermöglicht.

[0044] Angenommen in einem Beispiel, dass die Connection-Objects **514a** und **518a** hergestellt worden sind und den Datenstrom über die Verbindung **524** durch den Kanal 1 ermöglichen. An einem bestimmten Zeitpunkt, möglicherweise, wenn der Client (z. B. ein Fahrzeug mit einer Telematik-Steuereinheit) in eine Zone mit höherer Bandbreite eintritt (z. B. eine Tankstelle, die einen Dienst mit hoher drahtloser Bandbreite anbietet), wird der Kanal 2 (CH 2) verfügbar. Seine Verfügbarkeit wird vom Channel-Monitor **522** des Communications-Frameworks **516a** erkannt. Wenn der Kanal 2 wünschenswerter ist als Kanal 1, leitet das Connection-Objekt **518a** auf der Basis der Richtlinien, die im Policy-Manager **520** festgelegt sind, eine Umschaltung zum Kanal 2 ein.

[0045] Die Umschaltung zum Kanal 2 wird dann von der Synchronisationssteuerung **527** so synchronisiert, dass Daten, die zwischen der Clientseite und der Serverseite übertragen werden, die Synchronisation erreichen (wodurch Datenverluste während der Umschaltung verhindert werden). Zum Beispiel kann der Datenstrom über die Verbindung **524** gestoppt

werden, um eine Datensicherung auf der Carlet-Anwendungsseite zu veranlassen. Übertragungen von Daten über den Kanal 1, die noch im Gange sind, werden abgeschlossen, bevor die Umschaltung des Connection-Objects **518a** auf den Kanal 2 zugelassen wird. Die Synchronisation zwischen der Clientseite und der Serverseite erlaubt die Kanalschaltung, wodurch Datenverluste vermieden werden. Sobald das Connection-Object **518a** die Synchronisation zwischen der Clientseite und der Serverseite erreicht hat und das Connection-Object **518a** vom Kanal 1 auf Kanal 2 umgeschaltet hat, wird die Fortsetzung des Datentroms über die Connection-Objects **514a** und **518a** durch Kanal 2 zugelassen. Wenn Daten auf der Clientseite gesichert wurden, wird der Strom dieser Daten dann durch den Kanal 2 zugelassen.

[0046] Dieser Prozess wird dann den vom Policy-Manager eingestellten Richtlinien entsprechend fortgesetzt, und auf der Basis der ständigen Überwachung jedes der verfügbaren Kanäle durch den Channel-Monitor **522**. Zum Beispiel kann ein Carlet mehr als eine offene Verbindung haben, wie durch das Connection-Object **514b** und das Connection-Object **518b** des Datenmultiplexers und Flow-Controllers **518c** veranschaulicht.

[0047] Unter bestimmten Umständen kann ein Connection-Object **518b** aufgrund einer Unterbrechung in der Übertragungsfähigkeit eines bestimmten Kanals (z. B. durch Verlassen des Bereichs eines aktuellen drahtlosen Netzwerks) eine Verbindung **526** verlieren. Wenn dies auftreten sollte, wird die Erkennung der Nichtverfügbarkeit des Kanals 4 (CH 4) vom Channel-Monitor **522** erkannt. Das Connection-Object **518b** wird dann bestimmen, ob der Kanal, der nicht verfügbar geworden ist, tatsächlich genutzt wurde. In einem Beispiel kann der Kanal ungenutzt sein, seine Nichtverfügbarkeit wird aber dennoch erkannt, um seine Wahl zu verhindern. In einem anderen Beispiel wird angenommen, dass der Kanal 4 genutzt wurde. In solch einem Fall kann es aufgrund des plötzlichen Ausfalls der Verbindung zu Datenverlusten gekommen sein. Wenn dies austritt, kommuniziert das Connection-Object **518b** mit einem Connection-Object **514b** des Carlets, um zu bestimmen, ob tatsächlich Daten verloren worden sind. Wenn Daten verloren wurden, werden die verlorenen Daten vom Carlet angefordert, falls Daten vom Carlet an den Server gesendet wurden, oder die verlorenen Daten würden vom Server angefordert, falls Daten vom Server an das Carlet gesendet wurden.

[0048] Der Policy-Manager wird dann abgefragt, um zu bestimmen, auf welchen der verbleibenden Kanäle, die vom Channel-Monitor **522** überwacht werden, am besten umgeschaltet werden sollte, um die zwischen den Connection-Objects **514b** und **518b** hergestellte Verbindung fortzusetzen. In diesem Beispiel

würde das Connection-Object **518b** zur Verbindung **526'** über den Kanal 5 (CH 5) umschalten, was eine langsamere Verbindung sein kann, und die Verbindung wird auf transparente Weise wiederhergestellt, um die kontinuierliche Datenübertragung zu gewährleisten. Um die Umschaltung abzuschließen, wirkt die Synchronisationssteuerung **527** mit der Clientseite und der Serverseite zusammen, um sicherzustellen, dass Daten, die zwischen jedem der Datenmultiplexer und Flow-Controller **518c** und **518s** übertragen werden, synchronisiert sind und alle verlorenen Daten neu übertragen werden. Da der Channel-Monitor **522** jeden der Kanäle weiter überwacht, wird erneut eine transparente Umschaltung auftreten, wenn der mehr bevorzugte Kanal wieder verfügbar wird, wie in Bezug auf die Kanalschaltung zwischen den Verbindungen **524** und **524'** erläutert.

[0049] [Fig. 8](#) ist ein Flussdiagramm der Verfahrensschritte, die zum Einstellen einer Präferenz in einer TCU nach einer Ausführungsform der Erfindung durchgeführt werden. Das Verfahren beginnt mit Schritt **802**, wo eine Präferenz gewählt wird. In einer Ausführungsform ist die Präferenz eines von einer Systempräferenz und einer Benutzerpräferenz. Das Verfahren geht dann zu Schritt **804** über, wo die gewählte Präferenz im zum Präferenzserver gehörigen Speicher gespeichert wird. Für den Fachmann versteht es sich, dass die gewählte Präferenz über eine Webseite benannt werden kann, auf die mit einem Web-fähigen Gerät wie einem Personal Computer, PDA, Mobiltelefon usw. zugegriffen werden kann. Das Verfahren geht dann zu Schritt **806** über, wo der Präferenzserver mit dem Präferenz-Manager synchronisiert wird. Ein geeignetes Verfahren, um den Präferenzserver mit dem Präferenzmanager zu synchronisieren, wird Bezug nehmend auf [Fig. 7](#) beschrieben. In einer Ausführungsform aktiviert eine Synchronisationssteuerung, die mit den Communications-Frameworks des Servers und des Clients in Verbindung ist, die Synchronisation. Dann geht das Verfahren zu Schritt **808** über, wo die gewählte Präferenz zum Präferenz-Manager übertragen wird. In einer Ausführungsform, in der die Präferenz hochgeladen wird, während die TCU in einem Ruhezustand ist, wird die Präferenz vom Server übertragen, wenn das die TCU enthaltende Fahrzeug gestartet wird. Das Verfahren geht dann zu Schritt **810** über, wo die gewählte Präferenz im zur TCU gehörigen Speicher gespeichert wird. Es ist anzumerken, dass der Benutzer nach der Übertragung der Präferenz an die TCU auf die gewählte Präferenz zugreifen kann.

[0050] Auch wenn [Fig. 8](#) ein Verfahren zum Speichern einer Präferenz vom Server zum Client erläutert, ist anzumerken, dass das Verfahren auch vom Client zum Server angewendet werden kann. Das heißt, ein Benutzer hat die Möglichkeit, Präferenzen vom Fahrzeug aus über eine Benutzerschnittstelle der TCU hinzuzufügen, zu ändern oder zu modifizie-

ren. Die gewählte Präferenz wird in der TCU gespeichert, und dann werden der Präferenz-Manager und der Präferenzserver synchronisiert. Die gewählte Präferenz wird dann vom Präferenz-Manager zum Präferenzserver übertragen, wo sie auf der Serverseite gespeichert wird.

[0051] Zusammenfassend wird die Java Virtual Machine (JVM) als ein Interpreter benutzt, um die Portabilität von Java-Anwendungen zu gewährleisten. Allgemein legen Entwickler Java-Anwendungen als hardwareunabhängige Softwaremodule aus, die von Java Virtual Machines ausgeführt werden. Die Java Virtual Machine-Layer wird entwickelt, um in Verbindung mit dem nativen Betriebssystem der speziellen Hardware zu funktionieren, auf welcher das Communications-Framework **516c** laufen soll. Auf diese Weise können die Java-Anwendungen (z. B. Carlets) von einem Hardwaregerät zum anderen portiert werden, ohne ein Update des Anwendungscodes zu erfordern.

[0052] Im Unterschied zu den meisten Programmiersprachen, in welchen ein Programm zu einem maschinenabhängigen, ausführbaren Programmcode kompiliert wird, werden Java-Klassen zu maschinenunabhängigen Bytecode-Klassendateien kompiliert, die von einer maschinenabhängigen virtuellen Maschine ausgeführt werden. Die virtuelle Maschine stellt eine Abstraktionsebene zwischen der Maschinenunabhängigkeit der Bytecode-Klassen und dem maschinenabhängigen Befehlssatz der zugrunde liegenden Computerhardware dar. Ein Klassenlader sorgt dafür, dass die Bytecode-Klassendateien nach Bedarf geladen werden, und ein Interpreter oder Just-in-Time-Compiler sorgt für die Umwandlung von Bytecodes in Maschinencode.

[0053] Das heißt, Java ist eine Programmiersprache, die ausgelegt ist, um Anwendungen zu erzeugen, die unverändert auf allen Hardwareplattformen laufen können, ob klein, mittel oder groß. Von Sun entwickelt, hat Java durch das Web Verbreitung gefunden und ist sowohl für öffentliche Web-Sites als auch für Intranets geeignet. Allgemein können Java-Programme von HTML-Dokumenten aus aufgerufen oder selbstständig gestartet werden. Wenn ein Java-Programm von einer Web-Seite aus ausgeführt wird, wird es ein „Java-Applet“ genannt, und wenn es auf einem Web-Server läuft, wird die Anwendung ein „Servlet“ genannt.

[0054] Java ist eine interpretierte Sprache. Der Quellcode eines Java-Programms wird zu einer Zwischensprache kompiliert, „Bytecode“ genannt. Der Bytecode wird dann bei der Ausführung in Maschinencode umgewandelt (interpretiert). Wenn der Web-Browser ein Java-Applet findet, ruft der Web-Browser einen Java-Interpreter (Java Virtual Machine) auf, der den Bytecode in Maschinencode

umwandelt, und führt diesen aus. Daher sind Java-Programme von keiner spezifischen Hardware abhängig und laufen auf jedem Computer mit der Java Virtual Machine-Software. Auf der Serverseite können Java-Programme zur schnelleren Ausführung auch zu Maschinensprache kompiliert werden. Ein kompiliertes Java-Programm verliert dadurch aber seine Hardwareunabhängigkeit.

[0055] Auch wenn die vorliegende Erfindung auf der Basis der Java-Programmiersprache beschrieben wurde, können andere Programmiersprachen verwendet werden, um die Ausführungsformen der vorliegenden Erfindung zu implementieren, wie z. B. andere objektorientierte Programmiersprachen. Das objektorientierte Programmieren ist ein Verfahren zum Erstellen von Computerprogrammen, bei dem bestimmte grundlegende Bausteine kombiniert werden und Beziehungen unter und zwischen den Bausteinen hergestellt werden. Die Bausteine in objektorientierten Programmiersystemen werden „Objekte“ genannt. Ein Objekt ist eine Programmereinheit, in welcher eine Datenstruktur (Instanzvariable) und die Operationen (Methoden), die diese Daten verwenden oder beeinflussen können, zusammengefasst sind. Demnach besteht ein Objekt aus Daten und einer oder mehreren Operationen oder Prozeduren, die an diesen Daten vorgenommen werden können. Die Verbindung von Daten und Operationen zu einem einheitlichen Baustein wird „Einkapselung“ genannt.

[0056] Ein Objekt kann angewiesen werden, eine seiner Methoden auszuführen, wenn es eine „Message“ empfängt. Eine Message ist ein Befehl oder eine Anweisung an ein Objekt, eine bestimmte Methode auszuführen. Es besteht aus einer Methodenwahl (Name) und mehreren Argumenten, die an ein Objekt gesendet werden. Eine Message teilt dem empfangenden Objekt mit, welche Operationen durchzuführen sind.

[0057] Ein Vorteil der objektorientierten Programmierung ist die Art und Weise, wie Methoden aufgerufen werden. Wenn eine Message an ein Objekt gesendet wird, ist es für die Message nicht notwendig, das Objekt anzuweisen, wie eine bestimmte Methode auszuführen ist. Es ist nur notwendig, vom Objekt die Ausführung der Methode anzufordern. Dies vereinfacht die Programmentwicklung sehr.

[0058] Objektorientierte Programmiersprachen basieren vorwiegend auf einem System von „Klassen“. Eine Klasse definiert einen Objekttyp, der typischerweise sowohl Instanzvariablen als auch Methoden für die Klasse einschließt. Eine Objektklasse wird verwendet, um eine spezielle Instanz eines Objekts zu erzeugen. Eine Instanz einer Objektklasse schließt die Variablen und Methoden ein, die für die Klasse definiert wurden. Mehrfache Instanzen derselben Klasse können aus einer Objektklasse er-

zeugt werden. Jede Instanz, die aus der Objektklasse erzeugt wurde, gehört diesem Typ oder dieser Klasse an.

[0059] Es kann eine Klassenhierarchie definiert werden, sodass eine Objektklassendefinition eine oder mehrere Subklassen aufweist. Ein Subklasse erbt die Definition ihrer Eltern (und Großeltern usw.). Jede Subklasse in der Hierarchie kann zum Verhalten beitragen, das durch seine Elternklasse spezifiziert wird, oder dieses verändern.

[0060] Zur Veranschaulichung kann eine Angestelltenobjektklasse die Instanzvariablen „Name“ und „Lohn“ und eine Methode „Lohn festlegen“ umfassen. Instanzen der Angestelltenobjektklasse können für jeden Angestellten in einem Betrieb angelegt oder instanziiert werden. Jede Objektinstanz ist vom Typ „Angestellte“. Jede Angestelltenobjektinstanz umfasst die Instanzvariablen „Name“ und „Lohn“ und das Verfahren „Lohn festlegen“. Die Werte, die mit den Variablen „Name“ und „Lohn“ in jeder Angestelltenobjektinstanz verknüpft sind, enthalten den Namen und den Lohn jedes Angestellten im Betrieb. Eine Message kann an die Angestelltenobjektinstanz eines Angestellten gesendet werden, um die Methode „Lohn festlegen“ aufzurufen, damit der Lohn des Angestellten (d. h., der Wert, der mit der Variablen „Lohn“ im Angestelltenobjekt des Angestellten verknüpft ist) geändert wird.

[0061] Ein Objekt ist ein Allgemeinbegriff, der in der objektorientierten Programmierumgebung benutzt wird, um sich auf ein Modul zu beziehen, das diesbezüglichen Code und Variablen enthält. Eine Softwareanwendung kann mit einer objektorientierten Programmiersprache geschrieben werden, wodurch die Funktionalität des Programms mit Objekten implementiert wird. Beispiele für objektorientierte Programmiersprachen schließen C++ sowie Java ein.

[0062] Überdies kann die Erfindung mit anderen Computersystemkonfigurationen einschließlich tragbarer Geräte, Mikroprozessorsystemen, mikroprozessorbasierter oder programmierbarer Unterhaltungselektronik, Minirechnern, Großrechnern und dergleichen in die Praxis umgesetzt werden. Die Erfindung kann auch in verteilten Rechenumgebungen, in denen Tasks von Fernverarbeitungsvorrichtungen durchgeführt werden, die über ein Netzwerk verbunden sind, in die Praxis umgesetzt werden.

[0063] In Anbetracht der obigen Ausführungsformen versteht es sich, dass die Erfindung verschiedene computerimplementierte Vorgänge unter Verwendung von Daten anwenden kann, die in Computersystemen gespeichert sind. Diese Vorgänge sind solche, die die physikalische Manipulation von physikalischen Größen erfordern. Diese Größen nehmen gewöhnlich, aber nicht unbedingt die Form von elektri-

schen oder magnetischen Signalen an, die gespeichert, übertragen, kombiniert, verglichen und auf sonstige Weise manipuliert werden können. Ferner werden die Manipulationen, die durchgeführt werden, oft als Erzeugung, Erkennung, Bestimmung oder Vergleich bezeichnet.

[0064] Jede der hierin beschriebenen Operationen, die Bestandteil der Erfindung sind, sind nutzbare Maschinenoperationen. Die Erfindung betrifft auch eine Vorrichtung oder eine Einrichtung zur Durchführung dieser Operationen. Die Einrichtung kann speziell zu diesem Zweck konstruiert worden sein, wie die oben erläuterte TCU, oder sie kann ein Universalrechner sein, der durch ein im Computer gespeichertes Computerprogramm auf selektive Weise aktiviert oder konfiguriert wird. Das heißt, verschiedene Universalrechner können mit Computerprogrammen benutzt werden, die den hierin beschriebenen Lehren entsprechend geschrieben wurden, oder es kann praktischer sein, ein spezialisiertere Vorrichtung zur Durchführung der notwendigen Operationen zu konstruieren.

[0065] Die Erfindung kann auch als computerlesbarer Code auf einem computerlesbaren Datenträger implementiert sein. Der computerlesbare Datenträger ist ein Datenspeichergerät, das Daten speichern kann, die danach von einem Computersystem gelesen werden können. Beispiele für computerlesbare Datenträger schließen Festplatten, Network Attached Storage (NAS), Nur-Lesen-Speicher, Schreib-/Lesespeicher, CD-ROMs, CD-Rs, CD-RWs, Magnetbänder und sonstige optische und nichtoptische Datenspeichergeräte ein. Der computerlesbare Datenträger kann auch über ein netzwerkgekoppeltes Computersystem verteilt werden, sodass der computerlesbare Code auf eine verteilte Weise gespeichert und ausgeführt wird. Der computerlesbare Datenträger kann auch ein flüchtiges Medium umfassen, d. h., ein Signal wie z. B. ein optisches, magnetisches, elektrisches, akustisches, Mikrowellen, Funk- oder elektromagnetisches Signal.

[0066] Auch wenn die obige Erfindung der Klarheit halber auf detaillierte Weise beschrieben wurde, versteht es sich, dass bestimmte Änderungen und Modifikationen im Rahmen der beiliegenden Ansprüche vorgenommen werden können. Deshalb sind die vorliegenden Ausführungsformen als veranschaulichend und nicht einschränkend zu betrachten.

Patentansprüche

1. System zum Speichern von Präferenzen auf einem Telematik-Client, umfassend:
einen Telematik-Server (**106**; **116**; **448**), konfiguriert, um eine Anforderung zu empfangen, die Änderungsdaten für eine Präferenz enthält, und um die Änderungsdaten für die Präferenz in einem Speicher (**450**)

eines Präferenz-Servers des Telematik-Servers zu speichern, wobei der Telematik-Server ein serverseitiges Communications-Framework (**118**) aufweist, das mit dem Präferenz-Server in Verbindung steht; eine Telematik-Steuereinheit (**102**), die einen Präferenz-Manager (**466**) zum Speichern der Präferenzen aufweist, wobei die Telematik-Steuereinheit ein clientseitiges Communications-Framework (**112**; **416**; **5168**) einschließt, das mit dem Präferenz-Manager (**466**) in Verbindung steht, wobei der Präferenz-Manager und der Präferenz-Server (**458**) konfiguriert sind, um sich über eine Netzwerkverbindung (**104**; **114**; **446**) zu synchronisieren, um die Übertragung der Änderungsdaten zwischen dem Präferenz-Server (**458**) und dem Präferenz-Manager (**466**) zu erlauben, wobei die Telematik-Steuereinheit einen Softwarestapel (**110**; **121**; **452**) umfasst, der eine Hardware-Layer (**120**), eine Betriebssystem-Layer (**122**; **122a**), eine Java Virtual Machine-Layer (**124**, **124a**), eine Open Services Gateway Initiative-Layer (**126**; **126a**) und eine Java Telematics Client-Layer (**460**) einschließt, und wobei die Telematik-Steuereinheit einen Channel-Monitor umfasst, der konfiguriert ist, um verfügbare Kanäle zu überwachen, und einen Datenmultiplexer und einen Flow-Controller mit einem Connection-Objekt, konfiguriert, um auf der Basis der Verfügbarkeit auf einen Kanal im Netzwerk umzuschalten, der eine höhere Bandbreite aufweist.

2. System nach Anspruch 1, wobei die Präferenz eines von einer Systempräferenz, einer Anwendungspräferenz und einer Benutzerpräferenz ist.

3. System nach einem der obigen Ansprüche, wobei die Synchronisation zwischen dem Telematik-Server (**106**; **116**; **448**) und der Telematik-Steuereinheit (**102**) durch ein Connection-Objekt (**418a**) hergestellt wird.

4. System nach einem der obigen Ansprüche, wobei eine Synchronisationssteuerung die Kommunikation zwischen dem clientseitigen Communications-Framework (**112**; **416**; **5168**) und dem serverseitigen Communications-Framework (**118**) erlaubt.

5. System nach einem der obigen Ansprüche, wobei das clientseitige Communications-Framework (**112**; **416**; **5168**) den Datenmultiplexer und den Flow-Controller (**415**, **518c**) umfasst.

6. System nach einem der obigen Ansprüche, wobei das serverseitige Communications-Framework (**118**) einen Datenmultiplexer und einen Flow-Controller (**518s**) umfasst.

7. System nach Anspruch 3, wobei das Connection-Objekt (**418a**) der Clientseite die Fähigkeit aufweist, zwischen Kanälen (**425**) umzuschalten, die dem clientseitigen Communications-Framework (**112**; **416**; **516a**) verfügbar sind.

8. Telematik-Steuereinheit (**102**), umfassend: einen Präferenz-Manager (**466**) zum Speichern von Präferenzen; ein clientseitiges Communications-Framework (**112**; **416**; **516x**), das mit dem Präferenz-Manager in Verbindung steht; einen Softwarestapel (**110**; **452**), der eine Hardware-Layer (**120**), eine Betriebssystem-Layer (**122**), eine Java Virtual Machine-Layer (**124**, **124a**), eine Open Services Gateway Initiative-Layer (**126**; **126a**) und eine Java Telematics Client-Layer (**460**) einschließt; und einen Channel-Monitor, konfiguriert, um verfügbare Kanäle zu überwachen, und einen Datenmultiplexer und einen Flow-Controller mit einem Connection-Objekt, konfiguriert, um auf der Basis der Verfügbarkeit auf einen Kanal im Netzwerk umzuschalten, der eine höhere Bandbreite aufweist; wobei der Präferenz-Manager konfiguriert ist, um sich mit einem Präferenz-Server eines Telematik-Servers (**106**; **116**; **448**) zu synchronisieren, der Telematik-Server konfiguriert ist, um eine Anforderung zu empfangen, die Änderungsdaten für eine Präferenz enthält, die Änderungsdaten auf diesem Präferenz-Server gespeichert werden, und die Synchronisation die Übertragung der Änderungsdaten zwischen dem Präferenz-Server und dem Präferenz-Manager (**466**) erlaubt.

9. Telematik-Steuereinheit nach Anspruch 8, wobei die Präferenz eines von einer Systempräferenz, einer Anwendungspräferenz und einer Benutzerpräferenz ist.

10. Telematik-Steuereinheit nach Anspruch 8 oder 9, wobei die Java Telematics Client-Layer (**460**) außerdem umfasst:

Carlets (**402a**, **402b**, **402n**), die von der Java Telematics Client-Layer ausgeführt werden, wobei die Carlets Funktionen ausführen, die durch die Präferenz spezifiziert werden.

11. Telematik-Steuereinheit nach einem der Ansprüche 8 bis 10, wobei das clientseitige Communications-Framework (**112**; **416**; **516a**) außerdem umfasst:

einen Message-Manager (**417**; **517**), der eine ein-fachgerichtete Kommunikation gewährleistet; einen Stream-Manager (**419**; **519**), der eine zwei-fachgerichtete Kommunikation gewährleistet, wobei der Stream-Manager (**419**; **519**) konfiguriert ist, um eine Anforderung zu empfangen, dass eine Verbindung von einem Carlet (**402a**, **402b**, **402n**) herzustellen ist; und der Datenmultiplexer und der Flow-Controller (**415**; **518c**) konfiguriert sind, um vom Stream-Manager (**419**; **519**) eine Anforderung nach einem Connection-Objekt (**418a**) zu empfangen, wobei der Datenmultiplexer und der Flow-Controller (**415**; **518c**) geeignet sind, in Antwort auf die Anforderung ein Con-

nection-Objekt (**418a**) an das Carlet (**402a**, **402b**, **402n**) zurückzugeben, um eine Verbindung herzustellen.

12. Verfahren zum Speichern von Präferenzen in Verbindung mit einem Telematik-System, umfassend:

das Empfangen, in einem Telematik-Server (**106**; **116**; **448**), einer Anforderung, die Änderungsdaten für eine Präferenz enthält;

das Speichern der Änderungsdaten im Speicher eines Präferenz-Servers dieses Telematik-Servers (**106**; **116**; **448**);

das Synchronisieren des Präferenz-Servers mit einem Präferenz-Manager (**466**) einer Telematik-Steuereinheit (**102**), wobei die Telematik-Steuereinheit einen Softwarestapel (**110**; **121**; **452**) umfasst, der eine Hardware-Layer (**120**), eine Betriebssystem-Layer (**122**; **122a**), eine Java Virtual Machine-Layer (**124**, **124a**), eine Open Services Gateway Initiative-Layer (**126**; **126a**) und eine Java Telematics Client-Layer (**460**) einschließt, und wobei die Telematik-Steuereinheit außerdem einen Channel-Monitor und einen Datenmultiplexer und einen Flow-Controller mit einem Connection-Objekt umfasst, wobei das Verfahren umfasst:

Das Überwachen der verfügbaren Kanäle durch den Channel-Monitor,

Das Umschalten des Connection-Objekts des Datenmultiplexers und Flow Controllers auf einen Kanal im Netzwerk mit einer höheren Bandbreite auf der Basis der Verfügbarkeit, und

das Übertragen der Änderungsdaten zum Präferenz-Manager (**466**), wobei die Änderungsdaten im Speicher der Telematik-Steuereinheit (**102**) gespeichert werden.

13. Verfahren nach Anspruch 12, wobei der Verfahrensschritt des Synchronisieren des Präferenz-Servers mit einem Präferenz-Manager außerdem umfasst:

das Anfordern eines Connection-Objekts (**418a**) von einem Datenmultiplexer und Flow-Controller (**415**; **518c**); und das Bestimmen, ob ein gewünschter Kanal (**425**) verfügbar ist.

14. Verfahren nach Anspruch 12 oder 13, wobei die Präferenz eine von einer Systempräferenz, einer Anwendungspräferenz und einer Benutzerpräferenz ist.

15. Verfahren nach einem der Ansprüche 12 oder 13, wobei die Präferenz eine Systempräferenz ist, die eines von einer Fahrzeug-Identifizierungsnummer, einer Führerscheinnummer, einer Zulassungsnummer und einer Telefonnummer zur Verbindung mit einem Server einschließt.

16. Verfahren nach einem der Ansprüche 12 oder 13, wobei die Präferenz eine Benutzerpräferenz ist,

die eine Infotainment- oder Komfortpräferenz ist.

17. Verfahren nach einem der Ansprüche 12 oder 16, wobei das Empfangen der Änderungsdaten außerdem umfasst:

das Empfangen dieser Änderungsdaten über ein verteiltes Netzwerk (**446**), das mit dem Präferenz-Server (**466**) in Verbindung steht, wobei die Änderungsdaten eine Präferenzeingabe durch eine Benutzerschnittstelle umfassen.

18. Verfahren nach Anspruch 17, wobei die Benutzerschnittstelle eine Webseite ist.

19. Verfahren nach einem der Ansprüche 12 bis 18, wobei der Verfahrensschritt des Synchronisieren des Präferenz-Servers mit einem Präferenz-Manager einer Telematik-Steuereinheit (**102**) außerdem umfasst:

das Initiieren der Synchronisation beim Einschalten der Telematik-Steuereinheit (**102**).

20. Datenträger, der prozessorlesbaren Code zur Steuerung eines Prozessors trägt, um alle Verfahrensschritte nach jedem der Ansprüche 12 bis 19 auszuführen.

21. Verfahren zum Speichern von Präferenzen in einer Telematik-Steuereinheit (**102**), wobei die Telematik-Steuereinheit einen Präferenz-Manager (**466**) zum Speichern von Präferenzen aufweist, ein clientseitiges Communications-Framework (**112**; **416**; **516a**), das mit dem Präferenz-Manager in Verbindung steht, und einen Softwarestapel (**110**; **452**), der eine Hardware-Layer (**120**), eine Betriebssystem-Layer (**122**), eine Java Virtual Machine-Layer (**124**, **124a**), eine Open Services Gateway Initiative-Layer (**126**; **126a**) und eine Java Telematics Client-Layer (**460**) einschließt, wobei die Telematik-Steuereinheit außerdem einen Channel-Monitor und einen Datenmultiplexer und einen Flow-Controller mit einem Connection-Objekt aufweist, wobei das Verfahren umfasst:

das Überwachen verfügbarer Kanäle durch den Channel-Monitor;

das Umschalten des Connection-Objekts des Datenmultiplexers und Flow Controllers auf einen Kanal im Netzwerk mit einer höheren Bandbreite auf der Basis der Verfügbarkeit; und

das Synchronisieren des Präferenz-Managers mit einem Präferenz-Server eines Telematik-Servers, wobei diese Synchronisation bewirkt, dass die Änderungsdaten zwischen dem Präferenz-Server und dem Präferenz-Manager (**466**) übertragen werden; wobei der Telematik-Server (**106**; **116**; **448**) konfiguriert ist, um eine Anforderung zu empfangen, die Änderungsdaten für eine Präferenz enthält, wobei die Änderungsdaten auf dem Präferenzserver gespeichert werden.

22. Verfahren nach Anspruch 21, wobei die Präferenz eines von einer Systempräferenz, einer Anwendungspräferenz und einer Benutzerpräferenz ist.

23. Datenträger, der prozessorlesbaren Code zur Steuerung eines Prozessors trägt, um alle Verfahrensschritte von Anspruch 21 oder 22 auszuführen.

Es folgen 8 Blatt Zeichnungen

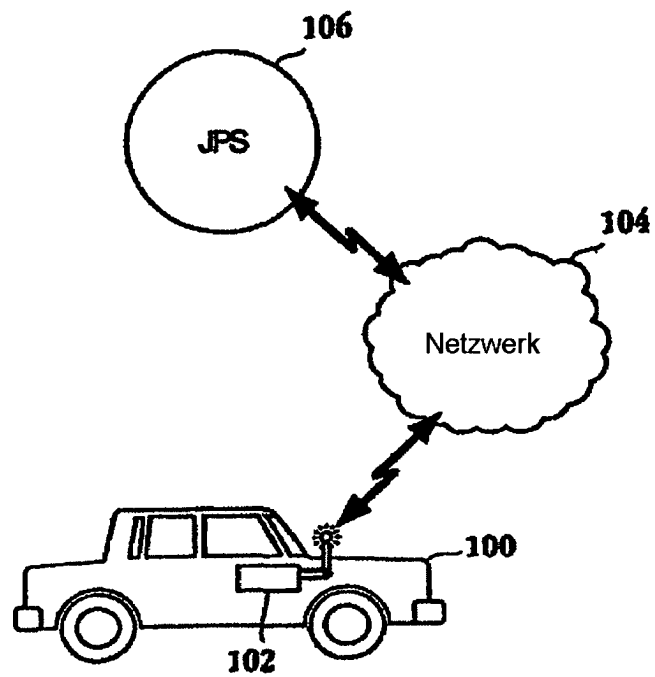


Fig. 1

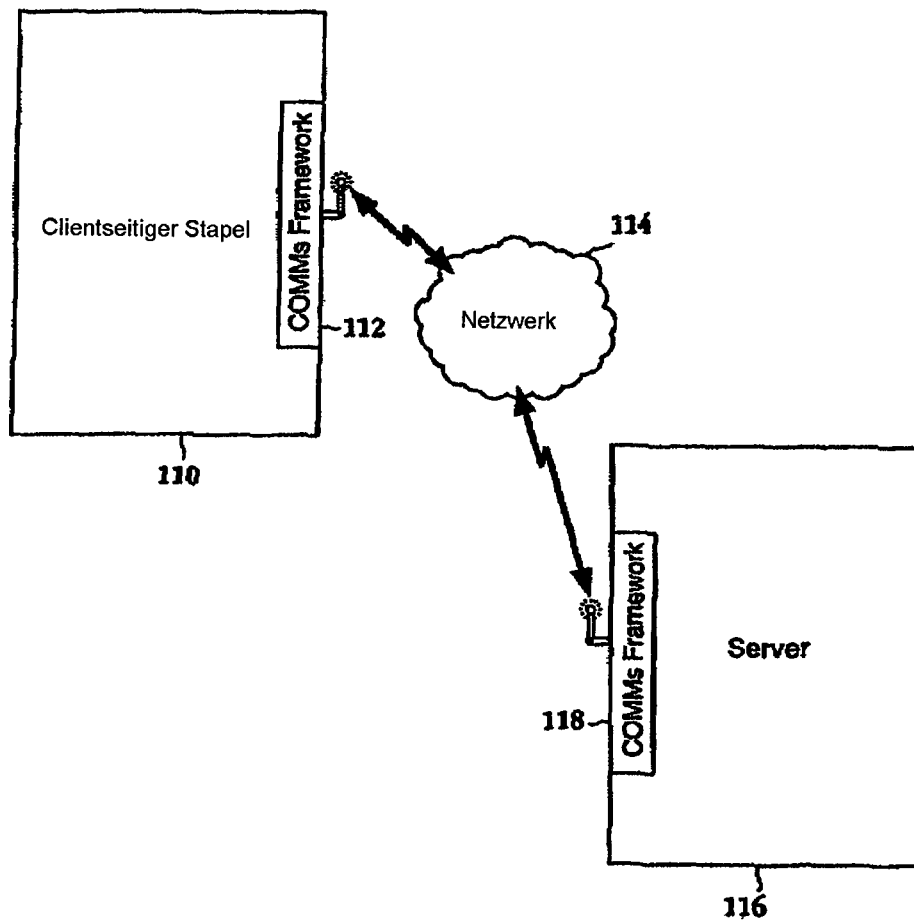


Fig. 2

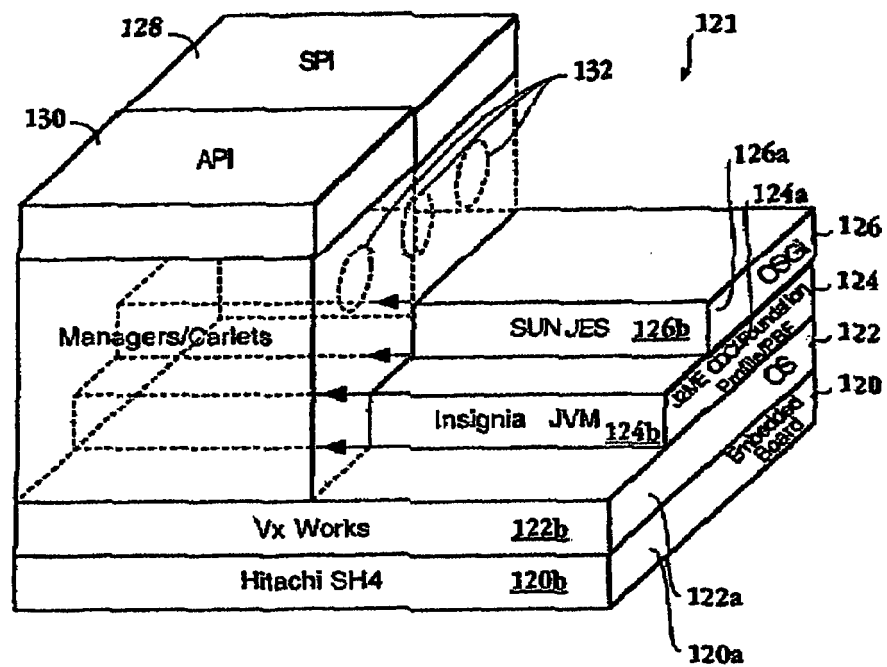


Fig. 3

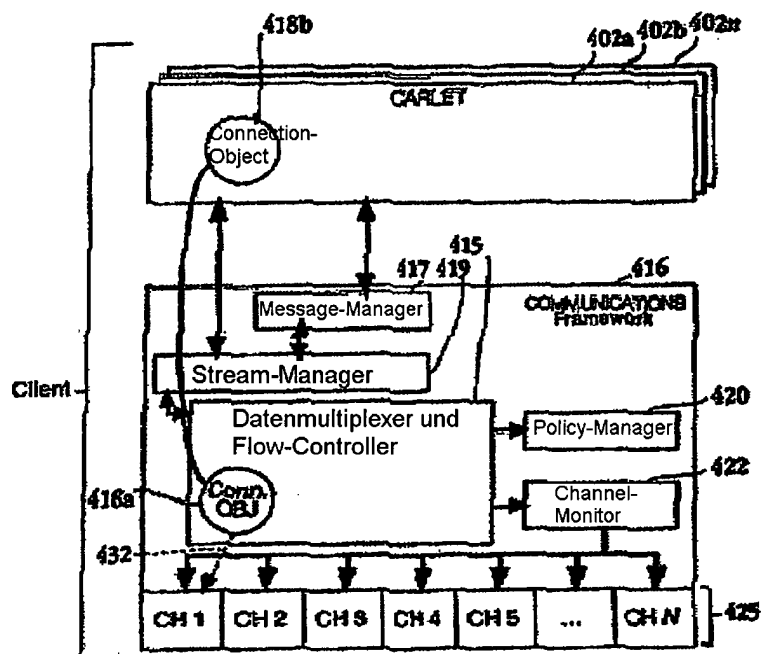


Fig. 4

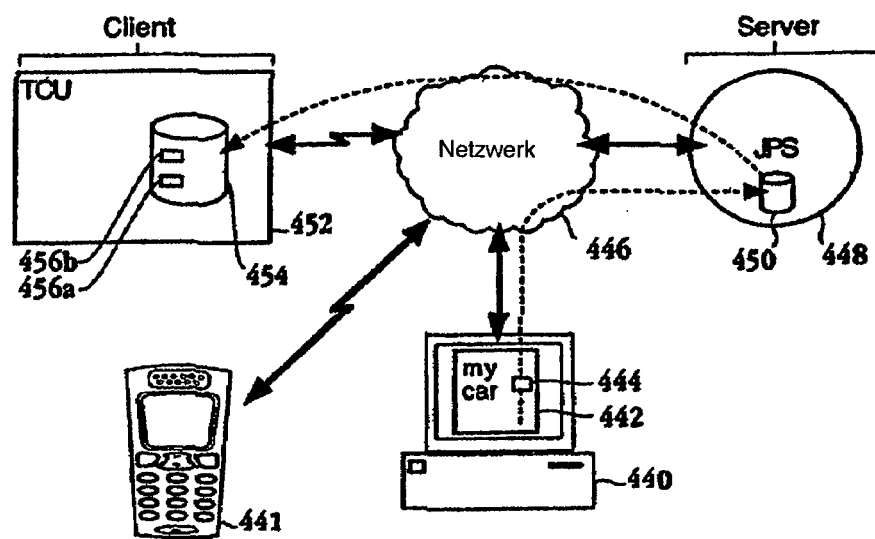
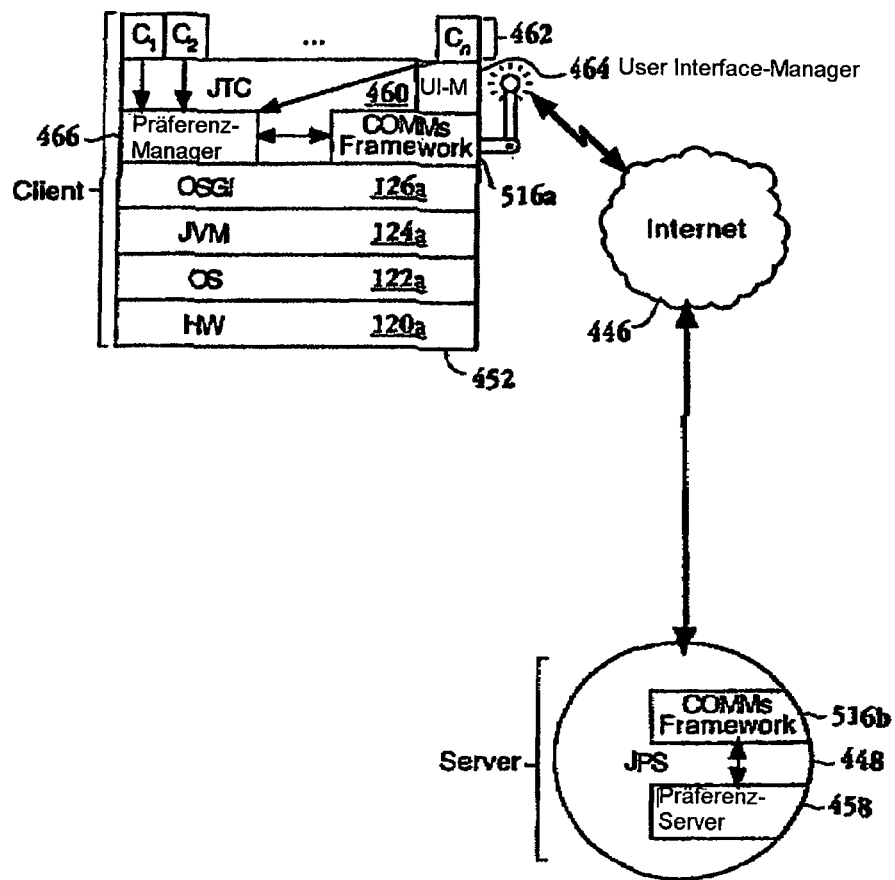


Fig. 5

**Fig. 6**

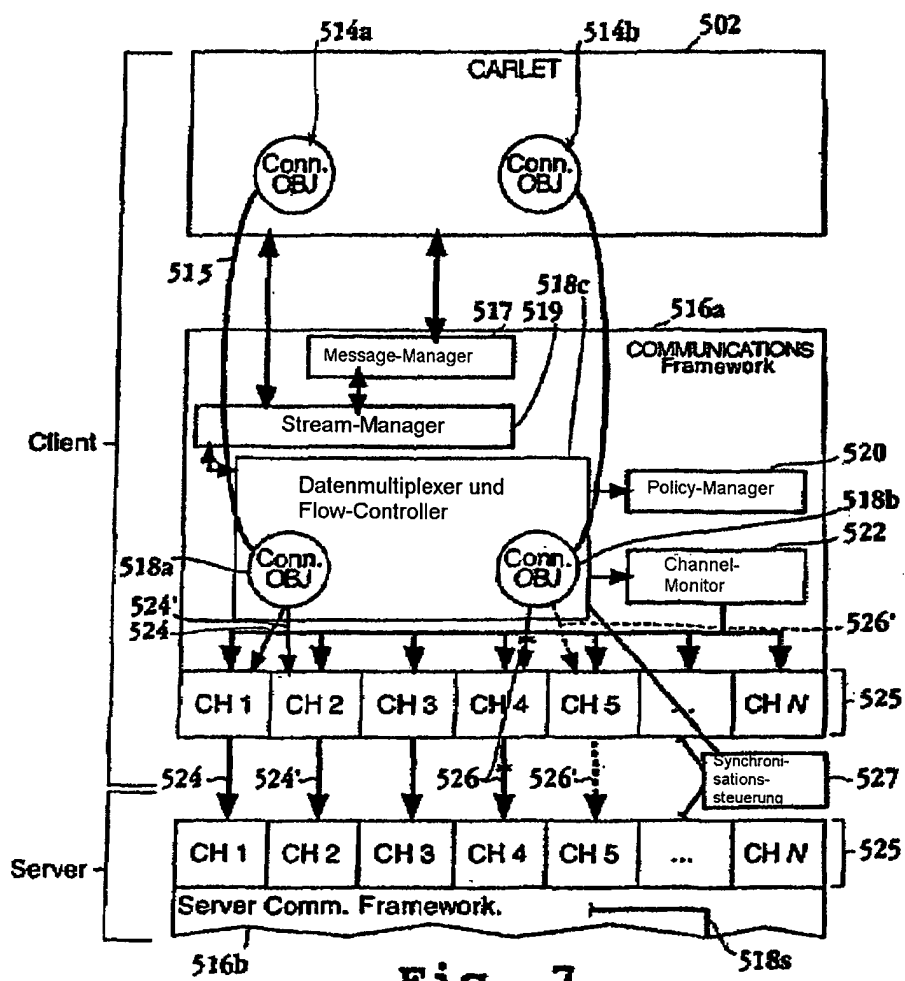


Fig. 7

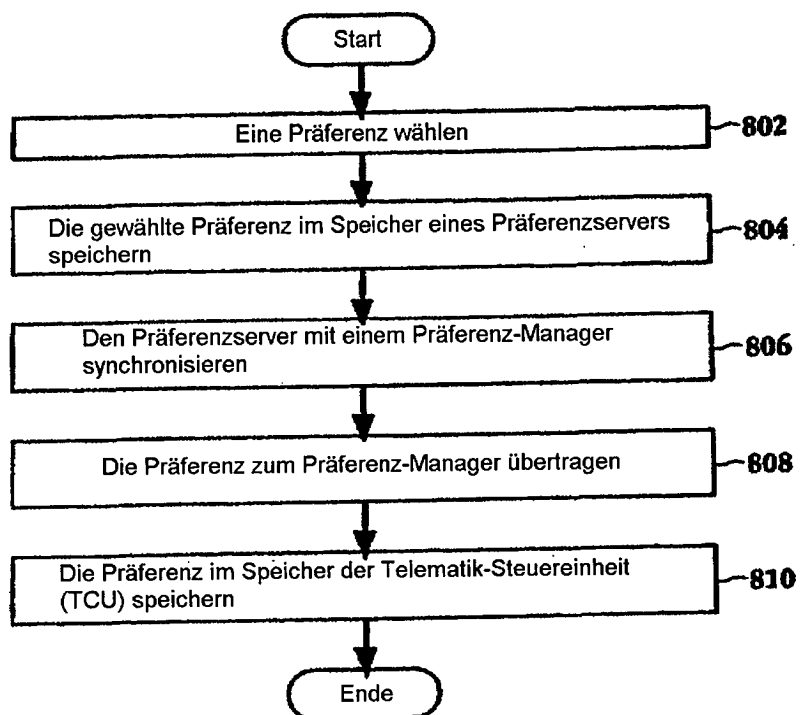


Fig. 8