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

(10) **DE 699 09 839 T3** 2009.10.08

(12) Übersetzung der geänderten europäischen Patentschrift

(97) **EP 1 143 337 B2**
(21) Deutsches Aktenzeichen: **699 09 839.4**
(96) Europäisches Aktenzeichen: **00 128 346.4**
(96) Europäischer Anmeldetag: **09.02.1999**
(97) Erstveröffentlichung durch das EPA: **10.10.2001**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **23.07.2003**
(97) Veröffentlichungstag des geänderten Patents beim EPA: **17.06.2009**
(47) Veröffentlichungstag im Patentblatt: **08.10.2009**

(51) Int Cl.[8]: ***G06F 9/50*** (2006.01)
G06F 17/30 (2006.01)

Patentschrift wurde im Einspruchsverfahren geändert

(30) Unionspriorität:
21506 **10.02.1998** **US**

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

(73) Patentinhaber:
Level 3 CDN International, Inc., Broomfield, Col., US

(72) Erfinder:
Farber, David A., Oak View, US; Greer, Richard E., Red Lodge, US; Swart, Andrew D, Westlake Village, US; Balter, James A., Santa Barbara, US

(74) Vertreter:
Patent- und Rechtsanwälte Bardehle, Pagenberg, Dost, Altenburg, Geissler, 81679 München

(54) Bezeichnung: **Optimierte Lokalisierung von Netzwerkbetriebsmittel**

Beschreibung

1. Gebiet der Erfindung

[0001] Diese Erfindung betrifft die Nachbildung von Ressourcen in Computernetzwerken.

2. Hintergrund der Erfindung

[0002] Das Aufkommen globaler Computernetzwerke wie des Internets hat vollkommen neue und unterschiedliche Wege der Informationsgewinnung mit sich gebracht. Ein Benutzer des Internets kann nun auf Informationen überall in der Welt zugreifen, unabhängig davon, wo sich der Benutzer oder die Informationen befinden. Der Benutzer erlangt Zugriff auf Informationen durch die bloße Kenntnis einer Netzwerkadresse für die Information und die Eingabe dieser Adresse in ein geeignetes Anwendungsprogramm, etwa einen Netzwerkbrowser.

[0003] Der rasche Popularitätsgewinn des Internets hat dem gesamten Netzwerk eine hohe Verkehrslast auferlegt. Lösungen für Nachfrageprobleme (z. B. bessere Erreichbarkeit. und schnellere Kommunikationsverbindungen) erhöhen die Auslastung der Versorgungssysteme nur noch. Internet-Websites (hier als "Publisher" bezeichnet) müssen permanent zunehmende Bandbreitenbedürfnisse bewältigen, dynamische Belastungswechsel unterbringen und die Leistung für entfernte Browsing-Clients verbessern, insbesondere ausländische. Die Einführung Content-reicher Anwendungen, wie live Audio und Video, hat das Problem noch zusätzlich verschärft.

[0004] Zur Erfüllung grundlegender Bandbreitenwachstumsbedürfnisse abonnniert ein Web-Publisher in der Regel zusätzliche Bandbreite bei einem Internet Service Provider (ISP), sei es in Form größerer oder zusätzlicher ”Pipes” (Datenaustauschkanäle) oder Kanäle vom ISP zum Publisher-Gebäude oder in Form großer Bandbreitenzuweisungen in der entfernten Hosting Server Collection eines ISPs. Diese inkrementellen Steigerungen sind nicht immer so feinabgestimmt, wie es den Bedürfnissen des Publishers entspricht, und nicht selten hinkt die Webseitenkapazität des Publishers aufgrund von Vorlaufzeiten hinter der Nachfrage her.

[0005] Zur Bewältigung ernsthafterer Bandbreitenwachstumsprobleme können die Publisher komplexere und kostspieligere Individuallösungen entwickeln. Die Lösung für das allgemeinste Bedürfnis, d. h. höhere Kapazität, basiert im allgemeinen auf der Nachbildung von Hardwareressourcen und Site-Content (Stichwort ”Mirroring” – Spiegelung) und der Duplikation von Bandbreitenressourcen. Diese Lösungen sind jedoch schwierig und teuer in der Installation und im Betrieb. Folglich können sich diese nur die größten Publisher leisten, weil nur diese Publisher die Kosten über die hohe Kundenzahl (und Websitezugriffe) amortisieren können.

[0006] Mehrere Lösungen wurden entwickelt, um Nachbildungen und Mirroring weiter zu entwickeln. Im allgemeinen sind diese Technologien zur Verwendung durch eine einzelne Website vorgesehen und besitzen keine Funktionen, anhand deren ihre Komponenten von vielen Websites gleichzeitig verwendet werden können.

[0007] Einige Lösungsmechanismen bieten Nachbildungssoftware, mit deren Hilfe gespiegelte Server auf dem neuesten Stand gehalten werden können. Diese Mechanismen erstellen im allgemeinen eine vollständige Kopie eines Dateisystems. Eines dieser Systeme funktioniert so, dass mehrere Kopien eines Dateisystems auf transparente Weise synchronisiert gehalten werden. Ein anderes System bietet Mechanismen zum expliziten und regelmäßigen Kopieren von Dateien, die sich geändert haben. Datenbanksysteme sind besonders schwer nachzubilden, zumal sie sich kontinuierlich ändern. Mehrere Mechanismen ermöglichen die Nachbildung von Datenbanken, obwohl keine Standardmethoden dafür vorliegen. Einige Unternehmen, die Proxy-Caches anbieten, beschreiben diese als Nachbildungs-Tools. Jedoch unterscheiden sich Proxy-Caches insofern als sie eher im Namen von Clients als von Publishers betrieben werden.

[0008] Wenn eine Website einmal von mehreren Servern bereitgestellt wird, liegt eine Herausforderung darin sicherzustellen, dass die Last unter den Servern angemessen verteilt oder ausgeglichen wird. Mit der Domainnamenserver-basierten Round-Robin-Adressauflösung werden unterschiedliche Clients zu unterschiedlichen Spiegeln geleitet.

[0009] Eine weitere Lösung, der Auslastungsausgleich, berücksichtigt die Auslastung der einzelnen Server (auf unterschiedliche Arten gemessen) bei der Auswahl, welcher Server eine bestimmte Anforderung abhandeln soll.

[0010] Auslastungsausgleicher setzen eine Reihe unterschiedlicher Techniken ein, um die Anforderung auf den entsprechenden Server zu lenken. Die meisten dieser Auslastungsausgleichstechniken verlangen, dass jeder Server eine exakte Abbildung der primären Website ist. Auslastungsausgleicher berücksichtigen die "Netzwerkdistanz" zwischen dem Client und den Spiegelserverkandidaten nicht.

[0011] Unter der Voraussetzung, dass Client-Protokolle nicht leicht veränderbar sind, ergeben sich in der Einführung nachgebildeter Ressourcen zwei Hauptprobleme. Das erste besteht in der Frage, wie welche Kopie der zu verwendenden Ressource ausgewählt werden soll. Das heißt, wenn eine Anforde-

rung nach einer Ressource an einen einzelnen Server ergeht, wie sollte die Wahl einer Nachbildung des Servers (oder dieser Daten) erfolgen? Wir bezeichnen dieses Problem als das "Rendezvous-Problem". Es gibt unterschiedliche Möglichkeiten, Clients zu Rendezvous mit entfernten Spiegelservern zu bringen. Diese Technologien müssen, wie Auslastungsausgleicher, eine Anforderung zu einem geeigneten Server leiten, doch zum Unterschied von Auslastungsausgleichern berücksichtigen sie die Netzwerkleistung und -topologie bei ihrer Entscheidung.

[0012] Mehrere Unternehmen bieten Produkte zur Verbesserung der Netzwerkleistung durch Priorisierung und Filterung von Netzwerkverkehr.

[0013] Proxy-Caches bieten eine Möglichkeit für Client-Aggregatoren, ihren Netzwerkressourcenverbrauch durch Speichern von Kopien beliebter Ressourcen nahe bei den Endbenutzern zu reduzieren. Ein Client-Aggregator ist ein Internet Service Provider oder eine andere Organisation, die eine große Anzahl von Clients, die Browser betreiben, ins Internet bringt. Client-Aggregatoren können mit Hilfe von Proxy-Caches die Bandbreiten reduzieren, die nötig sind, um diese Browser mit Web-Content zu versorgen. Traditionelle Proxy-Caches werden jedoch eher im Namen von Web-Clients als im Namen von Web-Publishers betrieben.

[0014] In Proxy-Caches werden die beliebtesten Ressourcen von allen Publishers gespeichert, was bedeutet, dass sie sehr groß sein müssen, um eine angemessene Cache-Effizienz zu erzielen. (Die Effizienz eines Cache ist definiert als die Zahl der Anforderungen an Ressourcen, die bereits im Cache abgelegt sind, dividiert durch die Gesamtanzahl an Anforderungen).

[0015] Proxy-Caches sind von Cache-Steuerungshinweisen abhängig, die mit den Ressourcen bereitgestellt werden, um zu entscheiden, wann die Ressourcen ersetzt werden sollten. Diese Hinweise sind prädiktiv und notwendigerweise oft inkorrekt, weshalb Proxy-Caches häufig überholte Daten bereitstellen. In vielen Fällen instruieren Proxy-Cache-Betreiber ihren Proxy, Hinweise zu ignorieren, um den Cache effizienter zu machen, auch wenn dies zur häufigeren Bereitstellung überholter Daten führt.

[0016] Proxy-Caches verstecken die Aktivitäten von Cleints vor den Publishers. Nachdem eine Ressource im Cache abgelegt ist, kann der Publisher nicht wissen, wie oft darauf aus dem Cache zugegriffen wurde.

[0017] Im WO 97/29423A wird ein Auslastungsausgleich über ein Netzwerk von Servern nach der Serverauslastung offenbart.

Zusammenfassung der Erfindung

[0018] Gemäß der vorliegenden Erfindung wird ein System geschaffen wie in Anspruch 1 beansprucht.

[0019] Nach einem zweiten Aspekt der vorliegenden Erfindung wird ein Verfahren geschaffen wie in Anspruch 6 beansprucht.

[0020] Diese Erfindung bietet Servern in einem Computernetzwerk die Möglichkeit, ihre Verarbeitung von Anforderungen für ausgewählte Ressourcen durch Bestimmung eines anderen Servers (eines "Repeaters") für die Verarbeitung dieser Anforderungen abzuladen. Die Auswahl der Repeater kann dynamisch auf der Grundlage von Informationen über mögliche Repeater vorgenommen werden. Wenn eine angeforderte Ressource Verweise auf andere Ressourcen enthält, können einige oder alle dieser Verweise durch Verweise auf Repeater ersetzt werden.

[0021] Zuerst stellt ein Client eine Anforderung nach einer bestimmten Ressource von einem Ausgangsserver, wobei die Anforderung einen Ressourcen-Identifikator für die bestimmte Ressource enthält, wobei der Ressourcen-Identifikator manchmal einen Hinweis auf den Ausgangsserver einschließt. Anforderungen, die beim Ausgangsserver ankommen, enthalten nicht immer einen Hinweis auf den Ausgangsserver; da sie zum Ausgangsserver gesendet werden, müssen sie ihn nicht benennen. Ein als Reflektor bezeichneter Mechanismus, der am selben Standort wie der Ausgangsserver lokalisiert ist, fängt die Anforderung vom Client zum Ausgangsserver ab und entscheidet, ob die Anforderung reflektiert oder lokal behandelt werden soll. Wenn der Reflektor entscheidet, die Anforderung lokal zu behandeln, leitet er sie an den Ausgangsserver weiter, ansonsten wählt er einen "besten" Repeater zur Verarbeitung der Anforderung. Wenn die Anforderung reflektiert wird, wird dem Client ein modifizierter Ressourcen-Identifikator zur Bestimmung des Repeaters bereitgestellt.

[0022] Der Client bekommt den modifizierten Ressourcen-Identifikator vom Reflektor und stellt eine Anforderung nach der bestimmten Ressource vom Repeater, der im modifizierten Ressourcen-Identifikator bestimmt ist.

[0023] Wenn der Repeater die Anforderung des Clients bekommt, reagiert er durch Übermittlung der angeforderten Ressource an den Client. Wenn der Repeater eine lokale Kopie der Ressource besitzt, übermittelt er diese Kopie, ansonsten leitet er die Anforderung an den Ausgangsserver weiter, um die Ressource zu erhalten, und speichert eine lokale Kopie der Ressource, um spätere Anforderungen bereitstellen zu können.

[0024] Die Auswahl eines geeigneten Repeaters zur Behandlung der Anforderung durch den Reflektor kann auf unterschiedliche Arten erfolgen. Im bevorzugten Ausführungsbeispiel erfolgt sie, indem das Netzwerk zuerst in "Kostengruppen" vorpartitioniert wird und dann bestimmt wird, in welche Kostengruppe der Client fällt. Als nächstes wird aus einer Mehrzahl von Repeater im Netzwerk eine Gruppe von Repeater ausgewählt, wobei die Mitglieder der Gruppe im Vergleich zu der Kostengruppe, zu der der Client gehört, niedrige Kosten aufweisen. Zur Feststellung der niedrigsten Kosten wird eine Tabelle eingerichtet und regelmäßig aktualisiert, um die Kosten zwischen jeder Gruppe und jedem Repeater zu definieren. Dann wird – vorzugsweise per Zufall – ein Mitglied der Gruppe als bester Repeater ausgewählt.

[0025] Wenn die bestimmte angeforderte Ressource selbst Identifikatoren anderer Ressourcen enthalten kann, kann die Ressource überschrieben werden (bevor sie dem Client bereitgestellt wird). Insbesondere wird die Ressource überschrieben, um mindestens einige der darin enthaltenen Ressourcen-Identifikatoren durch modifizierte Ressourcen-Identifikatoren zu ersetzen, die einen Repeater anstatt des Ausgangsservers bezeichnen. Wenn der Client andere Ressourcen gemäß Identifikatoren in der spezifisch angeforderten Ressource anfordert, so wird der Client diese Anforderungen als Konsequenz dieses Überschreibvorgangs direkt an den ausgewählten Repeater richten und den Reflektor und Ausgangsserver völlig umgehen.

[0026] Das Überschreiben von Ressourcen muss von Reflektoren ausgeführt werden. Es kann auch von Repeater ausgeführt werden, wenn sich die Repeater "zusammentun" ("peer") und Kopien von Ressourcen machen, die überschriebene Ressourcen-Identifikatoren enthalten, die einen Repeater bezeichnen.

[0027] In einem bevorzugten Ausführungsbeispiel ist das Netzwerk das Internet, und der Ressourcen-Identifikator ist ein Uniform Resource Locator (URL) zur Bezeichnung von Ressourcen im Internet, und der modifizierte Ressourcen-Identifikator ist eine URL, die den Repeater bezeichnet und auf den Ausgangsserver hinweist (wie in Schritt B3 unten beschrieben), und der modifizierte Ressourcen-Identifikator wird dem Client mit Hilfe einer REDIRECT-Nachricht (Umleitung) bereitgestellt. Es ist zu beachten, dass nur wenn der Reflektor eine Anforderung "reflektiert" der modifizierte Ressourcen-Identifikator unter Verwendung einer REDIRECT-Nachricht bereitgestellt wird.

KURZE BESCHREIBUNG DER ZEICHNUNGEN

[0028] Die oben aufgeführten und andere Ziele und Vorteile der Erfindung werden offensichtlich aus der folgenden detaillierten Beschreibung im Zusammenhang mit den begleitenden Zeichnungen, in denen die Bezugszeichen sich durchgehend auf gleiche Teile beziehen:

[0029] Fig. 1 stellt einen Abschnitt einer Netzwerkumgebung gemäß der vorliegenden Erfindung dar; und

[0030] Fig. 2–Fig. 6 sind Fließdiagramme der Ausführung der vorliegenden Erfindung.

DETAILLIERTE BESCHREIBUNG DER DERZEIT BEVORZUGTEN EXEMPLARISCHEN AUSFÜHRUNGSBEISPIELE

Überblick

[0031] In Fig. 1 ist ein Abschnitt einer Netzwerkumgebung **100** gemäß der vorliegenden Erfindung dargestellt, wobei ein Mechanismus (Reflektor **108**, weiter unten detailliert beschrieben) an einem Server (hier der Ausgangsserver **102**) eine Anzahl teilweise nachgebildeter Server oder Repeater **104a**, **104b** und **104c** verwaltet und nachverfolgt. Jeder Repeater **104a**, **104b** und **104c** bildet einen Teil oder alle Informationen nach, die auf dem Ausgangsserver **102** bereitgestellt sind, sowie Informationen, die auf anderen Ausgangsservern im Netzwerk **100** zur Verfügung stehen. Der Reflektor **108** ist mit einem bestimmten Repeater verbunden, der als sein "Kontakt"-Repeater bezeichnet wird ("Repeater B" **104b** in dem in Fig. 1 dargestellten System). Vorzugsweise hält jeder Reflektor eine Verbindung mit einem einzelnen, als sein Kontakt bekannten Repeater, und jeder Repeater hält Verbindung mit einem speziellen Repeater, der als sein Master-Repeater bekannt ist (z. B. Repeater **104m** für die Repeater **104a**, **104b** und **104c** in Fig. 1).

[0032] Folglich kann ein Repeater als dedizierter Proxy-Server betrachtet werden, der einen partiellen oder spärlichen Spiegel des Ausgangsservers **102** verwaltet, indem er einen verteilten, kohärenten Cache des Ausgangsservers implementiert. Ein Repeater kann einen (partiellen) Spiegel von mehr als einem Ausgangsserver verwalten. In einigen Ausführungsbeispielen ist das Netzwerk **100** das Internet, und die Repeater spiegeln ausgewählte Ressourcen, die von Ausgangsservern als Reaktion auf die HTTP (Hypertext Transfer Protocol) und FTP (File Transfer Protocol) Anforderungen der Clients bereitgestellt werden.

[0033] Ein Client **106** schließt über das Netzwerk **100** an den Ausgangsserver **102** und möglicherweise an einen oder mehrere Repeater **104a** usw. an.

[0034] Der Ausgangsserver **102** ist ein Server, von dem Ressourcen ausgehen. Allgemeiner gesagt, ist

der Ausgangsserver **102** ein Prozess oder eine Sammlung von Prozessen, die Ressourcen als Reaktion auf Anforderungen von einem Client **106** bereitstellen. Der Ausgangsserver **102** kann jeder handelsübliche Webserver sein. In einem bevorzugten Ausführungsbeispiel ist der Ausgangsserver **102** typischerweise ein Webserver wie der Apache-Server oder der Enterprise™ Server der Netscape Communications Corporation.

[0035] Der Client **106** ist ein Prozessor, der im Namen eines Endbenutzers Ressourcen vom Ausgangsserver **102** anfordert. Der Client **106** ist typischerweise ein Benutzer-Vermittlungsprogramm (z. B. ein Webbrowser wie der Navigator™ der Netscape Communications Corporation) oder ein Proxy für ein Benutzer-Vermittlungsprogramm. Andere Komponenten als der Reflektor **108** und die Repeater **104a**, **104b** usw. können unter Verwendung allgemein verfügbarer Softwareprogramme implementiert werden. Insbesondere funktioniert diese Erfindung mit jedem HTTP-Client (z. B. einem Webbrowser), Proxy-Cache und Webserver. Zudem könnte der Reflektor **108** voll in den Datenserver **112** (z. B. in einen Webserver) integriert sein. Diese Komponenten könnten auf Basis der Benutzung von Erweiterungsmechanismen (wie etwa sogenannte Add-in-Module) lose integriert werden oder durch Modifikation der Servicekomponente zur spezifischen Unterstützung der Repeater fest integriert werden.

[0036] Ressourcen, die vom Ausgangsserver **102** ausgehen, können statisch oder dynamisch sein. Das heißt, die Ressourcen können fixiert sein, oder sie können vom Ausgangsserver **102** spezifisch als Reaktion auf eine Anforderung erstellt werden. Es ist zu beachten, dass die Bezeichnungen ”statisch” und ”dynamisch” relativ sind, da eine statische Ressource sich in regelmäßigen, wenn auch langen Intervallen ändern kann.

[0037] Ressourcenanforderungen vom Client **106** an den Ausgangsserver **102** werden vom Reflektor **108** abgefangen, der sie für eine gegebene Anforderung entweder an den Ausgangsserver **102** weiterleitet oder bedingt auf einen Repeater **104a**, **104b** usw. im Netzwerk **100** reflektiert. Das heißt, je nach der Art der Anforderung durch den Client **106** an den Ausgangsserver **102** stellt der Reflektor **108** die Anforderung lokal (am Ausgangsserver **102**) bereit oder wählt einen der Repeater (vorzugsweise den für die Aufgabe besten Repeater) und reflektiert die Anforderung auf den ausgewählten Repeater. Mit anderen Worten, der Reflektor **108** bewirkt, dass vom Client **106** gestellte Anforderungen nach Ressourcen vom Ausgangsserver **102** entweder lokal vom Ausgangsserver **102** bereitgestellt oder transparent auf den besten Repeater **104a**, **104b** usw. reflektiert werden. Das Konzept eines besten Repeaters und die Art und Weise, auf die der beste Repeater ausgewählt wird, werden unten detailliert beschrieben.

[0038] Die Repeater **104a**, **104b** usw. sind Zwischenprozessoren zur Bedienung von Clientanforderungen, wobei die Leistung verbessert und die Kosten auf die hier beschriebene Weise gesenkt werden. In den Repeater **104a**, **104b** usw. finden Prozesse oder Prozesssammlungen statt, die im Namen des Ausgangsservers **102** Ressourcen an den Client **106** liefern. Ein Repeater kann einen Repeater-Cache **110** enthalten, der zur Vermeidung unnötiger Transaktionen mit dem Ausgangsserver **102** benützt wird.

[0039] Der Reflektor **108** ist ein Mechanismus, vorzugsweise ein Softwareprogramm, der Anforderungen abfängt, die normalerweise direkt zum Ausgangsserver **102** gesendet würden. In der Zeichnung als getrennte Komponenten dargestellt, sind der Reflektor **108** und der Ausgangsserver **102** typischerweise zusammen auf einem bestimmten System angeordnet, etwa dem Datenserver **112**. (Wie unten näher erörtert, kann der Reflektor **108** sogar ein ”Plug-in”-Modul sein, das zu einem Teil des Ausgangsservers **102** wird).

[0040] In Fig. 1 ist nur ein Teil eines Netzwerks **100** gemäß dieser Erfindung dargestellt. Ein komplettes Betriebsnetzwerk besteht aus einer beliebigen Zahl von Clients, Repeatern, Reflektoren und Ausgangsservern. Die Reflektoren kommunizieren mit dem Repeater-Netzwerk, und die Repeater im Netzwerk kommunizieren untereinander.

Uniform Resource Locators (URL)

[0041] Jeder Ort in einem Computernetzwerk besitzt eine Adresse, die allgemein als Serie von Namen oder Zahlen spezifiziert werden kann. Um auf Informationen zuzugreifen, muss eine Adresse für diese Informationen bekannt sein. Beispielsweise ist im World Wide Web (das ”Web”), einem Teilnetz des Internets, das Verfahren zum Bereitstellen von Informationsadressorten in Form von Uniform Resource Locators (URLs) standardisiert. URLs spezifizieren den Ort von Ressourcen (Informationen, Dateien usw.) im Netzwerk.

[0042] Das URL-Konzept wird um so nützlicher, wenn Hypertextdokumente verwendet werden. Ein Hypertextdokument ist eines, das im Dokument selbst Verknüpfungen (Zeiger oder Verweise) auf das Dokument selbst oder auf andere Dokumente enthält. Beispielsweise kann in einem Onlinesystem für juristische Recherchen jeder Fall als Hypertext-Dokument präsentiert werden. Wenn andere Fälle zitiert werden, können Verknüpfungen zu diesen Fällen bereitgestellt werden. Wenn also eine Person einen Fall studiert, kann sie den zitierten Verknüpfungen folgen, um die entsprechenden Teile zitierter Fälle zu lesen.

[0043] Im Fall des Internets im allgemeinen und des World Wide Web im speziellen können Dokumente in einer standardisierten Form erzeugt werden, die als Hypertext Markup Language (HTML) bekannt ist. In HTML besteht ein Dokument aus Daten (Text, Bilder, Ton und ähnliches) einschließlich Verknüpfungen zu anderen Abschnitten desselben Dokuments oder anderer Dokumente.

[0044] Die Verknüpfungen werden allgemein als URLs bereitgestellt und können in relativer oder absoluter Form vorliegen. Relative URLs lassen einfach die Teile des URLs weg, die für das die Verknüpfung enthaltende Dokument identisch sind, wie etwa die Adresse des Dokuments (bei einer Verknüpfung zum selben Dokument) usw. Im allgemeinen füllt ein Browser-Programm fehlende Teile einer URL unter Verwendung der entsprechenden Teile vom aktuellen Dokument ein und erzeugt damit eine voll ausgebildete URL einschließlich eines voll qualifizierten Domainnamens usw.

[0045] Ein Hypertext-Dokument kann eine beliebige Anzahl von Verknüpfungen zu anderen Dokumenten enthalten, und jedes dieser anderen Dokumente kann sich auf einem anderen Server in einem anderen Teil der Welt befinden. Beispielsweise kann ein Dokument Verknüpfungen zu Dokumenten in Russland, Afrika, China und Australien besitzen. Ein Benutzer, der dieses Dokument an einem bestimmten Client betrachtet, kann jeder der Verknüpfungen auf transparente Weise folgen (d. h. ohne zu wissen, wo das Dokument, zu dem die Verknüpfung hergestellt wird, sich eigentlich befindet). Demgemäß können die Kosten (in Form von Zeit oder Geld oder Ressourcenzuteilung) des Verfolgen einer Verknüpfung im Vergleich zu einer anderen erheblich sein.

[0046] URLs haben im allgemeinen folgende Form (vgl. detaillierte Definition in T. Berners-Lee et al., Uniform Resource Locators (URL), Network Working Group, Request for Comments: 1738, Category: Standards Track, Dezember 1994, lokalisiert unter http://ds.internic.net/rfc/rfc1738.txt):
schema://host[port]/url-pfad
wobei ”schema” ein Symbol sein kann, wie ”datei” (für eine Datei im lokalen System), ”ftp” (für eine Datei auf einem anonymen FTP-Dateiserver), ”http” (für eine Datei in einer Datei in einem Webserver), und ”telnet” (für einen Anschluss an einen Telnet-basierten Dienst). Andere Schemata können ebenfalls benützt werden, und von Zeit zu Zeit werden neue Schemata hinzugefügt. Die Anzahl der Ports ist optional, wobei das System eine Standard-Portzahl (anhängig vom Schema) einsetzt, wenn keine vorgegeben wird. Das ”host”-Feld wird auf eine bestimmte Netzwerkadresse für einen bestimmten Computer abgebildet. Der ”url-pfad” ist abhängig von dem im ”host”-Feld angegebenen Computer. Ein url-pfad ist typischerweise, jedoch nicht notwendigerweise, der Pfadname eines Datei in einem Webserver-Verzeichnis.

[0047] Folgendes ist beispielsweise eine URL, welche eine Datei ”F” im Pfad ”A/B/C” auf einem Computer unter ”www.uspto.gov” identifiziert: http.//www.uspto.gov/A/B/C/F

[0048] Um auf die von obigem URL spezifizierte Datei ”F” (die Ressource) zuzugreifen, müsste ein auf einem Benutzercomputer (d. h. einem Client-Computer) laufendes Programm (z. B. ein Browser) zuerst den vom Hostnamen spezifizierten Computer (d. h. einen Server-Computer) lokalisieren. D. h. das Programm müsste den Server ”www.uspto.gov” lokalisieren. Dazu würde es auf einen Domainnamenserver (DNS) zugreifen und dem DNS den Hostnamen (”www.uspto.gov”) bereitstellen. Der DNS funktioniert als eine Art zentralisiertes Verzeichnis zur Auflösung von Adressen von Namen. Wenn der DNS feststellt, dass es einen (entfernten Server-)Computer in Entsprechung zum Namen www.uspto.gov gibt, stellt er dem Programm eine aktuelle Computernetzwerkadresse für diesen Servercomputer bereit. Im Internet wird diese als Internet Protocol (IP) Adresse bezeichnet, und sie hat die Form ”123.345.456.678”. Das Programm auf dem Computer des Benutzers (Client) würde dann die aktuelle Adresse für den Zugriff auf den entfernten (Server-)Computer nützen.

[0049] Das Programm öffnet eine Verbindung zum HTTP-Server (Webserver) auf dem entfernten Computer www.uspto.gov und benützt die Verbindung dazu, eine Anforderungsnachricht an den entfernten Computer zu senden (unter Verwendung des HTTP-Schemas). Die Nachricht ist typischerweise eine HTTP GET Anforderung, die den URL-Pfad der angeforderten Ressource ”A/B/C/F” enthält. Der HTTP-Server empfangt die Anforderung und benützt sie, um auf die vom url-Pfad ”A/B/C/F” spezifizierte Ressource zuzugreifen. Der Server sendet die Ressource über dieselbe Verbindung zurück.

[0050] So werden HTTP-Clientanforderungen nach Webressourcen auf einem Ausgangsserver **102** konventionell wie folgt verarbeitet (vgl. **Fig. 2**) (Dies ist eine Beschreibung des Verfahrens, wenn kein Reflektor **108** installiert ist):

A1. Ein Browser (z. B. Netscape Navigator) beim Client empfangt einen Ressourcen-Identifikator (eine URL) von einem Benutzer.

A2. Der Browser extrahiert den Host-(Ausgangsserver)-Namen vom Ressourcen-Identifikator und benützt einen Domainnamenserver (DNS) zum Nachschlagen der Netzwerk-(IP)-Adresse des entsprechenden Servers. Der Browser extrahiert auch eine Portnummer, wenn eine solche vorhanden ist, oder verwendet eine vorgegebene Portnummer (die Standardportnummer für http-Anforderungen ist 80).

A3. Der Browser bedient sich der Netzwerkadres-

se und Portnummer des Servers, um eine Verbindung zwischen dem Client **106** und dem Host- oder Ausgangsserver **102** herzustellen.
A4. Der Client **106** sendet dann eine (GET) Anforderung über die Verbindung, in der die angeforderte Ressource identifiziert wird.
A5. Der Ausgangsserver **102** empfängt die Anforderung und
A6. lokalisiert die entsprechende Ressource oder stellt sie zusammen.
A7. Der Ausgangsserver **102** sendet dann an den Client **106** eine Antwort zurück, in der die angeforderte Ressource (oder eine Form von Fehleranzeige, wenn die Ressource nicht verfügbar ist) enthalten ist. Die Antwort wird an den Client über die selbe Verbindung gesendet wie die, auf dem die Anforderung vom Client empfangen wurde.
A8. Der Client **106** empfängt die Antwort vom Ausgangsserver **102**.

[0051] Zu diesem Basismodell gibt es zahlreiche Variationen. Z. B. kann der Ausgangsserver in einer Variante, anstatt dem Client die Ressource bereitzustellen, den Client informieren, die Ressource unter einem anderen Namen erneut anzufordern. Dazu sendet der Server **102** in A7 an den Client **106** eine Antwort mit der Bezeichnung ”REDIRECT” zurück, die eine neue URL enthält, welche den anderen Namen angibt. Der Client **106** wiederholt dann die gesamte Sequenz, normalerweise ohne irgendeine Benutzerintervention, diesmal als Anforderung der mit der neuen URL identifizierten Ressource.

Systembetrieb

[0052] In dieser Erfindung nimmt der Reflektor **108** effektiv den Platz eines gewöhnlichen Webservers oder Ausgangsservers **102** ein. Der Reflektor **108** unternimmt dies, indem er die IP-Adresse und Portnummer des Ausgangsservers übernimmt. Wenn ein Client versucht, eine Verbindung mit dem Ausgangsserver **102** herzustellen, schließt er automatisch an den Reflektor **108** an. Der Ausgangs-Webserver (oder Ausgangsserver **102**) muss dann Anforderungen an einer unterschiedlichen Netzwerk-(IP)-Adresse annehmen, oder an der selben IP-Adresse, aber auf einer unterschiedlichen Portnummer. Bei Anwendung dieser Erfindung ist folglich der in A3–A7 genannte Server in Wahrheit ein Reflektor **108**.

[0053] Es ist zu beachten, dass es auch möglich ist, die Netzwerkadresse des Ausgangsservers so zu belassen, wie sie ist, und den Reflektor an einer unterschiedlichen Adresse oder auf einem anderen Port zu betreiben. Auf diese Weise fängt der Reflektor keine an den Ausgangsserver gesendete Anforderungen ab, sondern es können vielmehr nach wie vor direkt an den Reflektor adressierte Anforderungen versendet werden. So kann das System getestet und konfiguriert werden, ohne dass sein normaler Betrieb unterbrochen wird.

[0054] Der Reflektor **108** unterstützt die Verarbeitung wie folgt (vgl. Fig. 3):
Nach Empfang einer Anforderung
B1. analysiert der Reflektor **108** die Anforderung, um zu bestimmen, ob die Anforderung reflektiert werden soll oder nicht. Dazu legt der Reflektor zuerst fest, ob der Absender (Client **106**) ein Browser oder ein Repeater ist. Anforderungen, die von Repeatern ausgegeben werden, müssen lokal vom Ausgangsserver **102** bereitgestellt werden. Diese Bestimmung kann vorgenommen werden, indem die Netzwerk-(IP)-Adresse des Absenders in einer Liste bekannter Repeater-Netzwerk-(IP)-Adressen nachgeschlagen wird. Als Alternative könnte diese Bestimmung auch vorgenommen werden, indem Informationen an eine Anforderung geheftet werden, um anzuzeigen, dass die Anforderung von einem spezifischen Repeater stammt, oder Repeater können Ressourcen von einem speziellen Port anfordern, der sich von dem für gewöhnliche Clients verwendeten Port unterscheidet.
B2. Stammt die Anforderung nicht von einem Repeater, schlägt der Reflektor die angeforderte Ressource in einer Tabelle (genannt die ”Regelbasis”) nach, um festzustellen, ob die angeforderte Ressource ”repetierbar” ist. Auf der Grundlage dieser Feststellung reflektiert der Reflektor entweder die Anforderung (B3, Beschreibung unten) oder stellt die Anforderung lokal bereit (B4, Beschreibung unten).
Die Regelbasis ist eine Liste von Regelausdrücken und der zugehörigen Attribute. (Regelausdrücke sind in der Computerwissenschaft gut bekannt. Eine kleine Bibliografie ihrer Verwendung findet sich in Abo, et al., ”Compilers, Principles, techniques and tools”, Addison-Wesley, 1986, pp. 157–158). Der Ressourcen-Identifikator (URL) für eine gegebene Anforderung wird in der Regelbasis nachgeschlagen, indem sie sequenziell mit jedem Regelausdruck verglichen wird. Die erste Übereinstimmung identifiziert die Attribute für die Ressource, also repetierbar oder lokal. Findet sich in der Regelbasis keine Übereinstimmung, wird ein Standardattribut verwendet. Jeder Reflektor besitzt seine eigene Regelbasis, die vom Reflektor-Operatir manuell konfiguriert wird.
B3. Um eine Anforderung zu reflektieren (für die lokale Bereitstellung einer Anforderung vgl. B4), wie in Fig. 4 dargestellt, bestimmt (B3-1) der Reflektor den besten Repeater, zu dem die Anforderung reflektiert werden kann, wie weiter unten im Detail beschrieben. Der Reflektor erzeugt (B3-2) dann einen neuen Ressourcen-Identifikator (URL) (unter Verwendung der angeforderten URL und des besten Repeaters), der die selbe Ressource am ausgewählten Repeater identifiziert.
Es ist erforderlich, dass der Reflektierungsschritt

eine einzelne URL erzeugt, welche die URL der ursprünglichen Ressource sowie die Identität des ausgewählten Repeaters enthält. Zur Bereitstellung dieser Informationen wird eine Sonderform einer URL erzeugt. Dies erfolgt durch nachstehend beschriebene Erzeugung einer neuen URL:
D1. Unter Angabe eines Repeaternamens, Schemas, Ausgangsservernamens und Pfads wird eine neue URL erzeugt. Wenn das Schema "http" ist, benützt das bevorzugte Ausführungsbeispiel folgendes Format:

http://<repeater>/<server>/<pfad>
Wenn die benützte Form eine andere als "http" ist, benützt das bevorzugte Ausführungsbeispiel folgendes Format:

http://<repeater>/<server>@proxy=<schema>@/<pfad>
Der Reflektor kann an die Anforderung auch einen MIME-Typ anhängen, um den Repeater zu veranlassen, diesen MIME-Typ mit dem Ergebnis bereitzustellen. Dies ist nützlich, weil viele Protokolle (wie FTP) keine Möglichkeit bieten, an eine Ressource einen MIME-Typ anzuhängen. Das Format ist

http://<repeater>/<server>@proxy=<schema>:<typ>@/<pfad>
Diese URL wird nach Empfang durch den Repeater interpretiert.
Der Reflektor sendet dann (B3-3) eine REDIRECT-Antwort, welche diese neue URL enthält, zu dem anfordernden Client. Der HTTP REDIRECT-Befehl ermöglicht dem Reflektor, dem Browser eine einzelne URL zu senden, um die Anforderung erneut zu versuchen.
B4. Um eine Anforderung lokal bereitzustellen, wird die Anforderung vom Reflektor an den Ausgangsserver **102** gesendet ("weitergeleitet"). In diesem Modus fungiert der Reflektor als umgekehrter Proxy-Server. Der Ausgangsserver **102** verarbeitet die Anforderung auf die normale Weise (A5–A7). Der Reflektor empfängt dann die Antwort des Ausgangsservers auf die Anforderung, die er daraufhin untersucht, ob die angeforderte Ressource ein HTML-Dokument ist, d. h. ob die angeforderte Ressource eine ist, die ihrerseits Ressourcen-Identifikatoren enthält.
B5. Wenn die Ressource ein HTML-Dokument ist, überschreibt der Reflektor das HTML-Dokument durch Modifizierung der darin enthaltenen Ressourcen-Identifikatoren (URLs), wie unten beschrieben. Die Ressource, möglicherweise nach Modifikation durch Überschreiben, wird dann in einer Antwort an den anfordernden Client **106** zurückgesendet.
Wenn der anfordernde Client ein Repeater ist, kann der Reflektor vorübergehend allfällige Cache-Kontrollmodifikatoren, die der Ausgangsserver an die Antwort geheftet hat, deaktivieren. Diese deaktivierten Cache-Kontrollmodifikatoren werden später reaktiviert, wenn der Content vom Repeater bereitgestellt wird. Dieser Mechanismus ermöglicht es dem Ausgangsserver, das Ablegen von Ressourcen in normalen Proxy-Caches zu verhindern, ohne das Verhalten des Cache am Repeater zu beeinträchtigen.
B6. Unabhängig davon, ob die Anforderung reflektiert oder lokal behandelt wird, werden Details über die Transaktion, wie aktuelle Zeit, Adresse des Anforderungsprogramms, angeforderte URL und der Typ der generierten Antwort, vom Reflektor in eine lokale Protokolldatei geschrieben.

[0055] Durch Verwendung einer Regelbasis (B2) ist es möglich, Ressourcen selektiv zu reflektieren. Es gibt eine Reihe von Gründen dafür, dass bestimmte Ressourcen nicht effektiv repetiert werden können (und deshalb nicht reflektiert werden sollten), zum Beispiel:

– die Ressource wird für jede Anforderung eigens zusammengestellt;
– die Ressource beruht auf einem sogenannten Cookie (Browser senden keine Cookies zu Repeatern mit anderen Domainnamen);
– die Ressource ist eigentlich ein Programm (wie etwa ein Java Applet), das auf dem Client läuft und an einen Dienst angeschlossen werden möchte (Java verlangt, dass der Dienst auf der selben Maschine läuft, die auch das Applet bereitstellt).

[0056] Außerdem kann der Reflektor **108** so konfiguriert werden, dass Anforderungen von bestimmten Netzwerkadressen (z. B. Anforderungen von Clients auf dem selben Local Area Network wie der Reflektor selbst) nie reflektiert werden. Auch kann der Reflektor sich entscheiden, Anforderungen nicht zu reflektieren, weil der Reflektor seine gebundene aggregierte Informationsrate überschreitet, wie unten beschrieben.

[0057] Eine Anforderung, die reflektiert wird, wird automatisch am Repeater gespiegelt, wenn der Repeater die Anforderung empfängt und verarbeitet.

[0058] Die Kombination des hier beschriebenen Reflexionsverfahrens und des unten beschriebenen Caching-Verfahrens schafft auf wirksame Weise ein System, in dem repetierbare Ressourcen zum ausgewählten Reflektor migriert und an diesem gespiegelt werden, während nicht-repetierbare Ressourcen nicht gespiegelt werden.

Alternativmethode

[0059] Die Platzierung des Ausgangsservernamens in der reflektierten URL ist im allgemeinen eine gute Strategie, sie kann allerdings aus ästhetischen oder

(z. B. im Fall von Cookies) bestimmten technischen Gründen als unerwünscht betrachtet werden.

[0060] Es ist möglich, die Notwendigkeit zu vermeiden, sowohl den Repeaternamen wie auch den Servernamen in der URL zu platzieren. Stattdessen kann eine Namens-"Familie" für einen gegebenen Ausgangsserver geschaffen werden, wobei jeder Name einen von diesem Server benützten Repeater identifiziert.

[0061] Wenn beispielsweise www.beispiel.com der Ausgangsserver ist, könnten Namen für drei Repeater erzeugt werden:
wr1.beispiel.com
wr2.beispiel.com
wr3.beispiel.com

[0062] Der Name "wr1.beispiel.com" wäre ein Aliasname für Repeater 1, der auch unter anderen Namen bekannt sein könnte, wie "wr1.anderesBeispiel.com" und "wr1.beispiel.edu".

[0063] Wenn der Repeater bestimmen kann, unter welchem Namen er adressiert wurde, kann er diese Informationen (zusammen mit einer Tabelle, in der Repeater-Aliasnamen mit Ausgangsservernamen assoziiert werden) dazu verwenden, zu bestimmen, welcher Ausgangsserver adressiert wird. Zum Beispiel, wenn Repeater 1 als wr1.beispiel.com adressiert wird, ist der Ausgangsserver "www.beispiel.com"; wird er als "wr1.anderesBeispiel.com" adressiert, ist der Ausgangsserver "www.anderesBeispiel.com."

[0064] Der Repeater kann zwei Mechanismen benützen, um festzustellen, von welchem Aliasnamen er adressiert wird:

1. Jeder Aliasname kann mit einer anderen IP-Adresse assoziiert werden.
   Leider lässt sich diese Lösung nicht gut ausbauen, da IP-Adressen zur Zeit spärlich sind und die Anzahl der benötigten IP-Adressen mit dem Produkt von Ausgangsservern und Repeater wächst.
2. Der Repeater kann versuchen, den verwendeten Aliasnamen zu bestimmen, indem er den "Host"-Tag im HTTP-Header der Anforderung untersucht. Leider hängen einige noch in Gebrauch befindliche alte Browser den "Host"-Tag nicht an eine Anforderung an. Reflektoren müssten solche Browser identifizieren (die Identität des Browsers ist ein Teil jeder Anforderung) und diese Form der Reflexion vermeiden.

Wie ein Repeater eine Anforderung behandelt

[0065] Wenn ein Browser eine REDIRECT-Antwort erhält (wie in B3 produziert), gibt er unter Verwendung des neuen Ressourcen-Identifikators (URL) (A1–A5) eine erneute Anforderung nach der Ressource aus. Da sich der neue Identifikator auf einen Repeater anstatt auf den Ausgangsserver bezieht, sendet der Browser jetzt eine Anforderung nach der Ressource an den Repeater der eine Anforderung wie folgt bearbeitet (vgl. **Fig. 5**):

C1. Zuerst analysiert der Repeater die Anforderung, um die Netzwerkadresse des anfordernden Client und den Pfad der angeforderten Ressource festzustellen. Im Pfad enthalten ist ein Ausgangsservername (wie oben mit Bezug auf B3 beschrieben).

C2. Der Repeater benützt eine interne Tabelle, um zu prüfen, ob der Ausgangsserver einem bekannten "Subscriber" gehört. Ein Subscriber ist ein Rechtssubjekt (z. B. ein Unternehmen), das Ressourcen (z. B. Dateien) über einen oder mehrere Ausgangsserver veröffentlicht. Sobald das Rechtssubjekt sich als Subscriber anmeldet, ist es zur Nutzung des Repeater-Netzwerks berechtigt. Die unten beschriebenen Subscriber-Tabellen enthalten die Informationen, die verwendet werden, Reflektoren mit Subscribern zu verknüpfen.
Wenn die Anforderung nicht nach einer Ressource von einem bekannten Subscriber ist, wird sie abgelehnt. Um eine Anforderung abzulehnen, sendet der Repeater eine Antwort des Inhalts zurück, die angeforderte Ressource existiere nicht.

C3. Der Repeater stellt dann fest, ob die angeforderte Ressource lokal im Cache abgelegt ist. Wenn die angeforderte Ressource im Cache des Repeaters ist, wird sie abgerufen. Wenn sich im Cache des Repeaters allerdings keine gültige Kopie der angeforderten Ressource befindet, modifiziert der Repeater die ankommende URL und erzeugt eine Anforderung, die er direkt an den Ausgangsreflektor ausgibt, der sie verarbeitet (wie in B1–B6). Da diese Anforderung an den Ausgangsreflektor von einem Repeater kommt, gibt der Reflektor immer die angeforderte Ressource aus anstatt die Anforderung zu reflektieren. (Man vergesse nicht, dass Reflektoren Anforderungen von Repeatern immer lokal behandeln). Wenn der Repeater die Ressource vom Ausgangsserver erhalten hat, legt der Repeater die Ressource lokal in den Cache.
Wenn eine Ressource nicht lokal im Cache ist, kann der Cache seine "Peer-Caches" abfragen, um festzustellen, ob einer von diesen die Ressource enthält, bevor oder während er die Ressource vom Reflektor/Ausgangsserver abfragt. Wenn ein Peer-Cache innerhalb einer begrenzten Zeitspanne (vorzugsweise in einem kleinen Sekundenbruchteil) positiv antwortet, wird die Ressource aus dem Peer-Cache abgerufen.

C4. Der Repeater konstruiert dann eine Antwort einschließlich der angeforderten Ressource (die aus dem Cache oder dem Ausgangsserver abgerufen wurde) und sendet diese Antwort an den anfordernden Client.

C5. Details über die Transaktion, wie der assozi-

ierte Reflektor, die aktuelle Zeit, die Adresse des Anforderers, die angeforderte URL und der Typ der generierten Antwort werden in eine lokale Protokolldatei beim Repeater geschrieben.

[0066] Es ist zu beachten, dass sich die untere Zeile der Fig. 2 auf einen Ausgangsserver oder Reflektor oder Repeater bezieht, je nach dem, was die URL in Schritt A1 identifiziert.

Auswahl des besten Repeaters

[0067] Wenn der Reflektor **108** festlegt, dass er die Anforderung reflektieren wird, muss er den besten Repeater zur Behandlung dieser Anforderung auswählen (vgl. Schritt B3-1). Diese Auswahl wird durch den hier beschriebenen Best Repeater Selector (BRS) Mechanismus getroffen.

[0068] Das Ziel des BRS ist die rasche und heuristische Auswahl eines geeigneten Repeaters für einen gegebenen Client unter Vorgabe nur der Netzwerkadresse des Clients. Ein geeigneter Repeater ist einer, der nicht zu überlastet ist und gerechnet in einem bestimmten Maßstab für die Netzwerkdistanz nicht zu weit von Client entfernt ist. Der hier benützte Mechanismus beruht auf spezifischen, kompakten, vorberechneten Daten für eine schnelle Entscheidung. Andere, dynamische Lösungen können ebenfalls verwendet werden, um einen geeigneten Repeater auszuwählen.

[0069] Der BRS beruht auf drei vorberechneten Tabellen, namentlich der Gruppenreduzierungstabelle, der Verknüpfungskostentabelle und der Auslastungstabelle. Diese drei (unten beschriebenen) Tabellen werden offline berechnet und auf jeden Reflektor mittels dessen Kontakts im Repeaternetzwerk heruntergeladen. Die Gruppenreduzierungstabelle platziert jede Netzwerkadresse in einer Gruppe, mit dem Ziel, dass die Adressen in einer Gruppe die relativen Kosten teilen, so dass sie unter variierenden Umständen den selben besten Repeater haben würden (d. h. der BRS ist über die Mitglieder der Gruppe betrachtet invariabel).

[0070] Die Verknüpfungskostentabelle ist eine zweidimensionale Matrix, in der die aktuellen Kosten zwischen jedem Repeater und jeder Gruppe aufgeführt sind. Anfänglich werden die Verknüpfungskosten zwischen einem Repeater und einer Gruppe definiert als ”normalisierte Verknüpfungskosten” zwischen dem Repeater und der Gruppe definiert, wie weiter unten definiert. Im Laufe der Zeit wird die Tabelle mit Messungen aktualisiert, welche die relativen Kosten der Übertragung einer Datei zwischen dem Repeater und einem Mitglied der Gruppe präziser wiedergeben. Das Format der Verknüpfungskostentabelle ist <GruppenID><GruppenID><Verknüpfungskosten>, wobei die Gruppen-IDs als AS-Nummern angegeben werden.

[0071] Die Auslastungstabelle ist eine eindimensionale Tabelle, welche die aktuelle Auslastung an jedem Repeater identifiziert. Da Repeater unterschiedliche Kapazitäten haben können, ist die Auslastung ein Wert, der die Fähigkeit eines gegebenen Repeaters darstellt, zusätzliche Arbeit anzunehmen. Jeder Repeater sendet seine aktuelle Auslastung in regelmäßigen Intervallen zu einem zentralen Master-Repeater, vorzugsweise mindestens einmal pro Minute. Der Master-Repeater sendet die Auslastungstabelle über den Kontakt-Repeater an jeden Reflektor im Netzwerk.

[0072] Ein Reflektor enthält Einträge in der Auslastungstabelle nur für solche Repeater, deren Nutzung ihm zugeteilt ist. Die Zuteilung von Repeatern zu Reflektoren erfolgt zentral durch einen Repeater-Netzwerk-Operator am Master-Repeater. Diese Zuteilung ermöglicht die Modifizierung des Serviceniveaus eines bestimmten Reflektors. Beispielsweise kann ein sehr aktiver Reflektor viele Repeater verwenden, wohingegen ein relativ inaktiver Reflektor nur wenige Repeater benützen kann.

[0073] Die Tabellen können auch so konfiguriert werden, dass den Subscribern auf andere Weise selektive Repeater-Dienste bereitgestellt werden, z. B. für ihre Clients in spezifischen geografischen Regionen, wie etwa in Europa oder Asien.

Messen der Auslastung

[0074] In den gegenwärtig bevorzugten Ausführungsbeispielen wird die Repeaterauslastung in zwei Dimensionen gemessen, nämlich:

1. Vom Repeater empfangene Anforderungen pro Zeitintervall (RRPT), und
2. Vom Repeater gesendete Bytes pro Zeitintervall (BSPT).

[0075] Für jede dieser Dimensionen wird eine Maximalkapazität festgelegt. Die Maximalkapazität zeigt den Punkt an, an dem der Repeater als voll ausgelastet betrachtet wird. Eine höhere RRPT-Kapazität verweist allgemein auf einen schnelleren Prozessor, wohingegen eine höhere BSPT-Kapazität allgemein auf eine breitere Netzwerkleitung verweist. Diese Art der Auslastungsmessung geht davon aus, dass ein bestimmter Server der Aufgabe des Repetierens zugewiesen ist.

[0076] Jeder Repeater berechnet regelmäßig seinen aktuellen RRPT und BSPT durch Akkumulieren der Anzahl der über ein kurzes Zeitintervall empfangenen Anforderungen und gesendeten Bytes. Diese Messungen werden dazu verwendet, die Auslastung des Repeaters in jeder dieser Dimensionen zu bestimmen. Wenn die Auslastung eines Repeaters sei-

ne konfigurierte Kapazität überschreitet, wird an den Netzwerkadministrator des Repeaters eine Warnmeldung gesendet. Die beiden aktuellen Auslastungskomponenten werden zu einem einzigen Wert zusammengefasst, der für die Gesamtauslastung steht. Gleichermaßen werden die beiden Maximalkapazitätskomponenten zu einem einzigen Wert zusammengefasst, der für die maximale Gesamtkapazität steht. Die Komponenten werden wie folgt zusammengefasst:

Aktuelle Auslastung = B × aktueller RRPT + (1 – B) × aktueller BSPT

Maximalauslastung = B × max RRPT + (1 – B) × max BSPT

[0077] Der Faktor B, ein Wert zwischen 0 und 1, ermöglicht die Anpassung der relativen Gewichte von RRPT und BSPT, wodurch die Berücksichtigung von Verarbeitungsleistung oder Bandbreite begünstigt wird.

[0078] Die Werte der aktuellen Gesamtauslastung und der maximalen Gesamtkapazität werden periodisch von jedem Repeater zum Master-Repeater gesendet, wo sie in der Auslastungstabelle addiert werden, einer Tabelle, welche die Gesamtauslastung aller Repeater zusammenfasst. Änderungen in der Auslastungstabelle werden automatisch an jeden Reflektor verteilt.

[0079] Im bevorzugten Ausführungsbeispiel wird zwar ein zweidimensionales Maß für die Repeater-Auslastung verwendet, es kann aber auch jedes andere Auslastungsmaß verwendet werden.

Kombination von Verknüpfungskosten und Auslastung

[0080] Der BRS berechnet die Kosten der Bedienung eines bestimmten Client von jedem in Frage kommenden Repeater aus. Die Berechnung der Kosten erfolgt durch Kombination der verfügbaren Kapazität des Repeaterkandidaten mit den Kosten der Verknüpfung zwischen diesem Repeater und dem Client. Die Verknüpfungskosten werden durch einfaches Nachschlagen in der Verknüpfungskostentabelle errechnet.

[0081] Die Kosten werden anhand folgender Formel bestimmt:

Schwelle = K·max-Auslastung

Kapazität = max(max Auslastung – aktuelle-Auslastung, e)

Kapazität = min(Kapazität, Schwelle)

Kosten = Verknüpfungskosten·Schwelle/Kapazität

[0082] In dieser Formel ist e eine sehr kleine Zahl (Epsilon) und K ein Abstimmungsfaktor, der anfänglich auf 0,5 gestellt ist. Diese Formal verursacht den Anstieg der Kosten für einen bestimmten Repeater mit einer durch K definierten Rate, wenn seine Kapazität unter eine konfigurierbare Schwelle abfällt.

[0083] Anhand der Kosten jedes Repeaterkandidaten wählt der BRS alle Repeater in einem Deltafaktor des besten Wertes. Aus dieser Gruppe wird das Ergebnis per Zufall ausgewählt.

[0084] Der Deltafaktor hindert den BRS daran, bei ähnlichen Punktwerten wiederholt einen einzelnen Repeater auszuwählen. Er ist allgemein erforderlich, da die verfügbaren Informationen über Auslastungen und Verknüpfungskosten im Laufe der Zeit an Genauigkeit einbüßen. Dieser Faktor ist abstimmbar.

Best Repeater Selector (BRS)

[0085] Der BRS funktioniert wie folgt (vgl. **Fig. 6**): Gegeben sind eine Clientnetzwerkadresse und die drei oben beschriebenen Tabellen:

E1. Anhand der Gruppenreduzierungstabelle feststellen, in welcher Gruppe sich der Client befindet.
E2. Für jeden Repeater in der Verknüpfungskostentabelle und Auslastungstabelle die kombinierten Kosten dieses Repeaters wie folgt feststellen:
E2a. Maximal- und aktuelle Auslastung auf dem Repeater bestimmen (Verwendung der Auslastungstabelle);
E2b. Die Verknüpfungskosten zwischen dem Repeater und der Gruppe des Client bestimmen (Verwendung der Verknüpfungskostentabelle);
E2c. Kombinierte Kosten wie oben beschrieben bestimmen.
E3. Eine kleine Gruppe von Repeater mit den niedrigsten Kosten auswählen.
E4. Ein zufälliges Mitglied der Gruppe auswählen.

[0086] Vorzugsweise werden die Ergebnisse des BRS-Verfahrens in einem lokalen Cache am Reflektor **108** verwaltet. Wenn deshalb der beste Repeater kürzlich für einen bestimmten Client (d. h. für eine bestimmte Netzwerkadresse) bestimmt worden ist, kann der beste Repeater rasch wiederverwendet werden, ohne erneut bestimmt werden zu müssen. Da die oben beschriebene Berechnung auf statischen, vorberechneten Tabellen beruht, besteht kein Bedürfnis, den besten Repeater erneut zu bestimmen, sofern sich die Tabellen nicht geändert haben.

Bestimmung der Gruppenreduzierungs- und Verknüpfungskostentabellen.

[0087] Die im BRS-Verfahren benützte Gruppenreduzierungstabelle und Verknüpfungskostentabelle

werden in einem unabhängigen Verfahren erzeugt und regelmäßig aktualisiert, das hier als NetMap bezeichnet wird. Das NetMap-Verfahren wird je nach Bedarf mittels Absolvierung unterschiedlicher (weiter unten beschriebener) Phasen ausgeführt.

[0088] Die hier verwendete Bezeichnung Gruppe bezieht sich auf eine IP-"Adressengruppe".

[0089] Der Ausdruck Repeater-Gruppe bezieht sich auf eine Gruppe, welche die IP-Adresse eines Repeaters enthält.

[0090] Der Ausdruck Verknüpfungskosten bezieht sich auf statisch bestimmte Kosten zur Übertragung von Daten zwischen zwei Gruppen. In einer gegenwärtig bevorzugten Implementierung ist dies das Minimum der Summen der Kosten der Verknüpfungen entlang jedem Pfad zwischen den beiden. Die hier hauptsächlich interessierenden Verknüpfungskosten sind Verknüpfungskosten zwischen einer Gruppe und einer Repeatergruppe.

[0091] Der Ausdruck relative Verknüpfungskosten bezieht sich auf die Verknüpfungskosten in Relation zu anderen Verknüpfungskosten für die selbe Gruppe, die berechnet werden durch Subtrahieren der minimalen Verknüpfungskosten von einer Gruppe zu irgendeiner Repeatergruppe von jeder ihrer Verknüpfungskosten zu einer Repeatergruppe.

[0092] Der Ausdruck Kostenset bezieht sich auf ein Set von Gruppen, die bezüglich der Best Repeater Selection äquivalent sind. Das heißt, dass angesichts der verfügbaren Informationen für jede der selbe Repeater gewählt würde.

[0093] Das NetMap-Verfahren verarbeitet zuerst Eingabedateien, um eine interne Datenbank mit der Bezeichnung "Gruppenregister" zu erstellen. Diese Eingabedateien beschreiben Gruppen, die IP-Adressen innerhalb von Gruppen und Verknüpfungen zwischen den Gruppen; sie kommen aus unterschiedlichen Quellen, darunter öffentlich verfügbaren Internet Routing Registry (IRR) Datenbanken, BGP Routertabellen und Testservices, die an unterschiedlichen Punkten im Internet lokalisiert sind und öffentlich verfügbare Tools benützen (wie etwa "Traceroute"), um Datenpfade zu sampeln. Nach Abschluss dieser Verarbeitung enthält das Gruppenregister wesentliche Informationen für die weitere Verarbeitung, namentlich (1) die Identität jeder Gruppe, (2) die Serie von IP-Adressen in einer bestimmten Gruppe, (3) die Anwesenheit von Verknüpfungen zwischen Gruppen mit Hinweis auf Pfade, über die Informationen gehen können, und (4) die Kosten der Versendung von Daten über eine bestimmte Verknüpfung.

[0094] Die folgenden Prozesse werden dann mit der Gruppenregisterdatei durchgeführt.

Berechnung der Repeatergruppenverknüpfungskosten

[0095] Das NetMap-Verfahren berechnet "Verknüpfungskosten" zur Übertragung von Daten zwischen jeder Repeatergruppe und jeder Gruppe im Gruppenregister. Diese gesamten Verknüpfungskosten werden definiert als die Minimalkosten eines Pfades zwischen den beiden Gruppen, wobei die Kosten eines Pfads gleich der Summe der Kosten der einzelnen Verknüpfungen im Pfad sind. Der weiter unten präsentierte Verknüpfungskostenalgorithmus ist im wesentlichen der selbe wie der Algorithmus #562 aus dem ACM-Journal Transactions an Mathematical Software: "Shortest Path From a Specific Node to All Other Nodes in a Network" by U. Pape, ACM TOMS 6 (1980), pp. 450–455, http://www.netlib.org/toms/562.

[0096] In dieser Verarbeitung bezieht sich der Ausdruck "Repeatergruppe" auf eine Gruppe, die die IP-Adresse eines Repeaters enthält. Eine Gruppe ist eine Nachbarin einer anderen Gruppe, wenn das Gruppenregister anzeigt, dass eine Verknüpfung zwischen den beiden Gruppen besteht.

Für jede Ziel-Repeatergruppe T:

- Die Verknüpfungskosten zwischen T und ihr selbst auf Null initialisieren.
- Die Verknüpfungskosten zwischen T und jeder anderen Gruppe auf Unendlich initialisieren.
- Eine Liste L erstellen, die Gruppen enthält, welche sich in Äquidistanz von der Ziel-Repeatergruppe T befinden.
- Die Liste L initialisieren, dass sie nur die Ziel-Repeatergruppe T selbst enthält.
- Bei nicht leerer Liste L:

– Eine leere Liste L' von Nachbarn von Mitgliedern der Liste L erstellen.
– Für jede Gruppe G in der Liste L:
– Für jede Gruppe N, die eine Nachbarin von G ist:

- Sollen sich die Kosten auf die Summe der Verknüpfungskosten zwischen T und G und der Verknüpfungskosten zwischen G und N beziehen. Die Kosten zwischen T und G wurden im vorhergehenden Algorithmusdurchgang bestimmt, die Verknüpfungskosten zwischen G und N kommen aus dem Gruppenregister.
- Wenn die Kosten weniger sind als die Verknüpfungskosten zwischen T und N:

– Die Verknüpfungskosten zwischen T und N auf Kosten einstellen.
– N zu L' hinzufügen, wenn es nicht bereits dort ist.

- L auf L' setzen.

Kostensets berechnen.

[0097] Ein Kostenset ist eine Serie von Gruppen,

die bezüglich Best Repeater Selection äquivalent sind. Das heißt, angesichts der verfügbaren Informationen würde der selbe Repeater für jede davon gewählt.

[0098] Das ”Kostenprofil” einer Gruppe G ist hier definiert als das Set von Kosten zwischen G und jedem Repeater. Von zwei Kostenprofilen wird behauptet, sie seien äquivalent, wenn die Werte in einem Profil sich von den entsprechenden Werten im anderen Profil um einen konstanten Betrag unterscheiden.

[0099] Nachdem eine Client-Gruppe bekannt ist, greift der Algorithmus der Best Repeater Selection für Informationen über die Gruppe auf das Kostenprofil zurück. Wenn zwei Kostenprofile äquivalent sind, würde der BRS-Algorithmus bei beiden Profilen den selben Repeater wählen.

[0100] Ein Kostenset ist dann ein Set von Gruppen, die äquivalente Kostenprofile besitzen.

[0101] Die Effektivität dieser Methode lässt sich beispielsweise in dem Fall ersehen, bei dem alle Pfade zu einem Repeater von einer Gruppe A durch eine andere Gruppe B führen. Die beiden Gruppen haben äquivalente Kostenprofile (und sind deshalb im selben Kostenset), da welcher Repeater auch immer am besten für die Gruppe A ist, auch am besten für die Gruppe B sein wird, unabhängig davon, welcher Pfad zwischen den zwei Gruppen eingeschlagen wird.

[0102] Durch die Normalisierung von Kostenprofilen können äquivalente Kostenprofile identisch gemacht werden. Ein normalisiertes Kostenprofil ist ein Kostenprofil, bei dem die Minimalkosten den Wert Null haben. Ein normalisiertes Kostenprofil wird berechnet durch das Auffinden der Minimalkosten im Profil und Subtraktion dieses Werts von jedem Kostenpunkt im Profil.

[0103] Die Kostensets werden dann unter Anwendung des folgenden Algorithmus berechnet:

- Für jede Gruppe G:
  – das normalisierte Kostenprofil für G berechnen;
  – ein Kostenset mit dem selben normalisierten Kostenprofil suchen;
  – Wenn ein solches Set gefunden ist, G zum bestehenden Kostenset addieren;
  – ansonsten ein neues Kostenset mit dem berechneten normalisierten Kostenprofil erzeugen, das nur G enthält.

[0104] Der Algorithmus zum Suchen von Kostensets verwendet eine Hash-Tabelle zur Reduzierung der Zeit, die notwendig ist, um zu bestimmen, ob das gewünschte Kostenset bereits existiert. Die Hash-Tabelle benützt einen aus dem Kostenprofil von G berechneten Hash-Wert.

[0105] Jedes Kostenset wird dann mit einer eindeutigen Kostenset-Indexnummer nummeriert. Die Kostensets werden dann auf geradlinige Weise dazu verwendet, die Verknüpfungskostentabelle zu generieren, welche die Kosten von jedem Kostenset zu jedem Repeater wiedergeben.

[0106] Wie unten beschrieben, bildet die Gruppenreduzierungstabelle jede IP-Adresse auf einem dieser Kostensets ab.

Aufbau der IP-Abbildung

[0107] Die IP-Abbildung (IP Map) ist eine sortierte Liste mit Einträgen, die IP-Adressenbereiche auf Verknüpfungskostentabellenschlüsseln abbilden. das Format der IP-Abbildung ist:

<Ausgangs-IP-Adresse><max IP-Adresse><Verknüpfungskostentabellenschlüssel>
wobei die IP-Adressen derzeit durch 32-bit-Ganzzahlen dargestellt werden. Die Einträge sind nach absteigender Ausgangsadresse und ansteigender Maximaladresse unter gleichen Ausgangsadressen und nach aufsteigendem Verknüpfungskostentabellenschlüssel unter gleichen Ausgangsadressen und Maximaladressen sortiert. Die Bereiche können auch überlappen.

[0108] Das NetMap-Verfahren erzeugt eine Zwischen-IP-Abbildung mit einer Abbildung zwischen IP-Adressenbereichen und Kostensetnummern wie folgt:

- Für jedes Kostenset S:
  – Für jede Gruppe G in S:
- Für jeden IP-Adressenbereich in G:
- Ein Dreifaches (niedrige Adresse; hohe Adresse; Kostensetnummer von S) zu der IP-Abbildung hinzufügen.

[0109] Die IP-Abbildungsdatei wird dann sortiert nach abnehmender Ausgangsadresse und nach ansteigender Maximaladresse unter gleichen Ausgangsadressen, und nach ansteigender Kostensetnummer unter gleichen Ausgangsadressen und Maximaladressen. Die Sortierreihenfolge für die Ausgangsadresse und die Maximaladresse minimiert die Zeit zum Aufbau der Gruppenreduzierungstabelle und produziert die richtigen Ergebnisse für überlappende Einträge.

[0110] Schließlich erzeugt das NetMap-Verfahren die Gruppenreduzierungstabelle durch Verarbeitung der sortierten IP-Abbildung. Die Gruppenreduzierungstabelle bildet (nach Bereichen spezifizierte) IP-Adressen in Kostensetnummern ab. Eine spezielle Verarbeitung der IP-Abbildungsdatei ist erforderlich, um überlappende Adressenbereiche zu entdecken und angrenzende Adressenbereiche zu verschmelzen, um die Größe der Gruppenreduzierung-

stabelle zu minimieren.

[0111] Eine geordnete Liste von Adressenbereichssegmenten wird verwaltet, wobei jedes Segment aus einer Ausgangsadresse B und einer Kostensetnummer N besteht, sortiert nach der Ausgangsadresse B. (Die Maximaladresse eines Segments ist die Ausgangsadresse des nächsten Segments minus eins). Folgender Algorithmus wird verwendet:

• Die Liste mit den Elementen [–unendlichkeit, NOGROUP], [+unendlichkeit, NOGROUP] initialisieren;
– Für jeden Eintrag in der IP-Abbildung in sortierter Reihenfolge, bestehend aus (b, m, s),
– (b, m, s) in die Liste einsetzen (Erinnerung: IP-Abbildungseinträge haben die Form (niedrige Adresse, hohe Adresse, Kostensetnummer von S)).
– Für jeden reservierten LAN-Adressenbereich (b, m): (b, m, LOCAL) in die Liste einsetzen.
– Für jeden Repeater an Adresse a: (a, a, REPEATER) in die Liste einsetzen.
– Für jedes Segment S in der geordneten Liste:
– S mit den folgenden Segmenten mit dem selben Kostenset verschmelzen;
– Einen Eintrag in die Gruppenreduzierungstabelle mit Ausgangsadresse von der Ausgangsadresse von S erstellen,
• Max Adresse = Ausgangsadresse d. nächsten Segments –1
• Gruppen-ID = Kostensetnummer von S.

[0112] Ein reservierter LAN-Adressenbereich ist ein Adressenbereich, der zur Nutzung durch LANs reserviert ist, die nicht als globale Internetadresse erscheinen sollten. LOCAL ist ein besonderer Kostensetindex, der sich von allen anderen unterscheidet und darauf verweist, dass der Bereich auf einen Client abbildet, der nie reflektiert werden sollte. REPEATER ist ein besonderer Kostensetindex, der sich von allen anderen unterscheidet und darauf verweist, dass der Adressenbereich auf einen Repeater abbildet. NOGROUP ist ein besonderer Kostensetindex, der sich von allen anderen unterscheidet und darauf verweist, dass dieser Adressenbereich keine bekannte Abbildung besitzt.

[0113] Bei gegebenen (B, M, N) ist ein Eintrag in die geordnete Adressenliste wie folgt zu erstellen:
Das letzte Segment (AB, AN) suchen, für das AB kleiner oder gleich B ist. Wenn AB kleiner als B ist, nach (AB, AN) ein neues Segment (B, N) einfügen. Das letzte Segment (YB, YN) suchen, für das YP kleiner oder gleich M ist. Jedes Segment (XB, NOGROUP), für das XB größer als B und kleiner als YB ist, durch (XB, N) ersetzen.
Wenn YN nicht N ist und entweder YN NOGROUP ist oder YB kleiner oder gleich B ist, Sei (ZB, ZN) das Segment, welches auf (YB, YN) folgt.
Wenn M + 1 kleiner als ZB ist, ein neues Segment (M + 1, YN) vor (ZB, ZN) einfügen.
(YB, YN) durch (YB, N) ersetzen.

Überschreiben von HTML-Ressourcen

[0114] Wie oben unter Bezugnahme auf Fig. 3 (B5) erklärt, wird, wenn ein Reflektor oder Repeater eine Ressource bereitgestellt, die ihrerseits Ressourcen-Identifikatoren (z. B. eine HTML-Ressource) enthält, diese Ressource modifiziert (überschrieben), um Ressourcen-Identifikatoren (URLs) repetierbarer Ressourcen, die in der Ressource erscheinen, vorzureflektieren. Das Überschreiben gewährleistet, dass wenn ein Browser repetierbare Ressourcen anfordert, die durch die angeforderte Ressource identifiziert werden, er diese von einem Repeater erhält, ohne zum Ausgangsserver zurück zu gehen, doch wenn er nicht-repetierbare Ressourcen anfordert, die durch die angeforderte Ressource identifiziert werden, er direkt zum Ausgangsserver zurückgeht. Ohne diese Optimierung würde der Browser entweder alle Anforderungen am Ausgangsserver stellen (wodurch das Verkehrsaufkommen am Ausgangsserver vergrößert würde und wesentlich mehr Umleitungen vom Ausgangsserver nötig wären) oder alle Anforderungen am Repeater stellen (wodurch der Repeater zu redundanten Anforderungen und zum Kopieren von Ressourcen veranlasst würde, die nicht im Cache abgelegt werden könnten, wodurch sich der Verwaltungsaufwand für die Bereitstellung solcher Ressourcen vergrößern würde).

[0115] Das Überschreiben setzt voraus, dass ein Repeater ausgewählt wurde (wie oben unter Bezugnahme auf den Best Repeater Selector beschrieben). Das Überschreiben erfolgt unter Verwendung einer sogenannten BASE-Anweisung. Die BASE-Anweisung lässt die HTML einen anderen Ausgangsserver identifizieren. (Die Ausgangsadresse ist normalerweise die Adresse der HTML-Ressource).

[0116] Das Überschreiben geht wie folgt vor sich:

F1. Eine BASE-Anweisung wird am Anfang der HTML-Ressource hinzugefügt oder wenn nötig modifiziert. Normalerweise interpretiert ein Browser relative URLs als relativ zu der voreingestellten Ausgangsadresse, namentlich der URL der HTML-Ressource (Seite), in der sie zusammentreffen. Die hinzugefügte BASE-Adresse spezifiziert die Ressource am Reflektor, der die Ressource ursprünglich bereitgestellt hat. Dies bedeutet, dass unbearbeitete relative URLs (wie die von Javascript™-Programmen generierten) als relativ zum Reflektor interpretiert werden. Ohne diese BASE-Adresse würden Browser relative Adressen mit Repeaternamen kombinieren, um URLs zu schaffen, die nicht in der von den Repeatern geforderten Form waren (gemäß Beschreibung oben in Schritt D1).

F2. Der Überschreiber identifiziert Anweisungen,

wie eingebettete Bilder und Verweisstichwörter, welche URLs enthalten. Wenn der Überschreiber in einem Reflektor läuft muss er die HTML-Datei parsen (analysieren), um diese Anweisungen zu identifizieren.
Wenn er in einem Repeater läuft, kann der Überschreiber Zugriff auf vorberechnete Informationen haben, welche den Standort jeder URL (die im Schritt F4 in der HTML-Datei platziert wurden) identifizieren.
F3. Für jede URL, die in der zu überschreibenden Ressource angetroffen wird, muss der Überschreiber bestimmen, ob die URL repetierbar (wie in den Schritten B1–B2) ist. Wenn die URL nicht repetierbar ist, wird sie nicht modifiziert. Ist die URL aber repetierbar, wird sie so modifiziert, dass sie sich auf den ausgewählten Repeater bezieht.
F4. Nachdem alle URLs identifiziert und modifiziert wurden, wird, wenn die Ressource einem Repeater bereitgestellt wird, am Anfang der Ressource eine Tabelle angehängt, in der der Ort und Inhalt jeder in der Ressource anzutreffenden URL identifiziert wird. (Dies ist ein Optimierungsschritt, der das Bedürfnis zum Parsen der HTML-Ressourcen am Repeater eliminiert).
F5. Nachdem alle Änderungen identifiziert wurden, wird eine neue Länge für die Ressource (Seite) berechnet. Die Länge wird in den HTTP-Header eingefügt, bevor die Ressource bereitgestellt wird.

[0117] Zur Zeit wird eine Erweiterung der HTML unter der Bezeichnung XML entwickelt. Das Verfahren des Überschreibens von URLs wird für XML ähnlich sein, mit einigen Unterschieden im Mechanismus, der die Ressource analysiert (Parsing) und eingebettete URLs identifiziert.

Bearbeitung von Nicht-HTTP-Protokollen

[0118] Diese Erfindung ermöglicht die Reflexion von Verweisen zu Ressourcen, die von anderen Protokollen als HTTP bereitgestellt werden, beispielsweise vom File Transfer Protocol (FTP) und Audio-/Videostromprotokollen. Allerdings bieten viele Protokolle nicht die Fähigkeit, Anforderungen umzuleiten. Es ist jedoch möglich, Verweise vor der Ausführung von Anforderungen durch Überschreiben von URLs, die in HTML-Seiten eingebettet sind, umzuleiten. Mit folgenden Modifizierungen an den obenstehenden Algorithmen wird diese Fähigkeit unterstützt.

[0119] In F4 überschreibt der Überschreiber URLs für Server, wenn diese Server in einer konfigurierbaren Tabelle kooperierender Ausgangsserver oder sogenannter Co-Server erscheinen. Der Operator des Reflektors kann diese Tabelle so definieren, dass sie FTP-Server und andere Server enthält. Ein überschriebenes URL, das auf eine Nicht-HTTP-Ressource verweist, hat folgende Form:

http://<repeater>/<ausgangsserver>@proxy=<schema>[:<typ>]@/ressource
wobei <schema> ein unterstützter Protokollname ist, wie ”ftp”. Dieses URL-Format ist eine Alternative zu der in B3 dargestellten Form.

[0120] In C3 sucht der Repeater ein Protokoll, das in die ankommende Anforderung eingebettet ist. Wenn ein Protokoll präsent ist und die angeforderte Ressource nicht bereits im Cache abgelegt ist, verwendet der Repeater das ausgewählte Protokoll statt dem Standard-HTTP-Protokoll zur Anforderung der Ressource, wenn er diese bereitstellt und im Cache speichert.

Systemkonfiguration und Verwaltung

[0121] Zusätzlich zu der oben beschriebenen Verarbeitung verlangt das Repeater-Netzwerk unterschiedliche Mechanismen für Systemkonfiguration und Netzwerkverwaltung. Einige dieser Mechanismen werden hier beschrieben.

[0122] Reflektoren ermöglichen ihren Operators, Repeater-Caches mittels der Durchführung von Publishing-Operationen zu synchronisieren. Wie Repeater-Caches synchronisiert gehalten werden, wird unten beschrieben. Publishing zeigt an, dass eine Ressource oder Sammlung von Ressourcen sich geändert hat.

[0123] Repeater und Reflektoren nehmen an unterschiedlichen Arten von Protokollverarbeitungen teil. Die Ergebnisse von Protokollen, die an Repeatern gesammelt wurden, werden gesammelt und mit Protokollen verschmolzen, die an Reflektoren gesammelt wurden, wie unten beschrieben.

Hinzufügen von Subscribern zum Repeater-Netzwerk

[0124] Wenn ein neuer Subscriber zum Netzwerk hinzugefügt wird, werden die Informationen über den Subscriber in eine Subscriber-Tabelle beim Master-Repeater eingegeben und an alle Repeater im Netzwerk verteilt. Zu diesen Informationen gehören die Committed Aggregate Information Rate (CAIR – Gebundene aggregierte Informationsrate) für Server, die dem Subscriber gehören, und eine Liste von Repeatern, die von Servern benützt werden können, die zum Subscriber gehören.

Hinzufügen von Reflektoren zum Repeater-Netzwerk

[0125] Wenn ein neuer Repeater zum Netzwerk hinzugefügt wird, verbindet er sich einfach mit einem Kontaktrepeater und meldet sich bei diesem an, wofür er vorzugsweise ein sicher verschlüsseltes Zertifikat einschließlich des Subscriber-Identifikators des

Repeaters verwendet.

[0126] Der Kontaktrepeater bestimmt, ob die Reflektornetzwerkadresse für diesen Subscriber zugelassen ist. Ist dies der Fall, akzeptiert der Kontaktrepeater die Verbindung und aktualisiert den Reflektor mit allen notwendigen Tabellen (unter Verwendung von Versionsnummern zur Feststellung, welche Tabellen veraltet sind).

[0127] Der Reflektor verarbeitet während dieser Zeit Anforderungen, ist aber nicht ”aktiviert” (zur Reflexion von Anforderungen befugt), bevor alle seine Tabellen aktuell sind.

Die Repeater-Caches synchronisiert halten

[0128] Repeater-Caches sind kohärent in dem Sinn, dass wenn ein Reflektor eine Änderung an einer Ressource identifiziert, alle Repeater-Caches verständigt werden und die Änderung in einer einzigen Transaktion akzeptieren.

[0129] Nur der Identifikator der veränderten Ressource (und nicht die gesamte Ressource) wird auf die Repeater übertragen; der Identifikator wird dazu verwendet, die entsprechende im Cache abgelegte Ressource am Repeater wirksam zu annullieren. Dieses Verfahren ist wesentlich effizienter als den Inhalt der veränderten Ressource an jeden Repeater zu senden.

[0130] Ein Repeater lädt die frisch modifizierte Ressource bei deren nächster Anforderung.

[0131] Eine Ressourcenänderung wird am Reflektor entweder manuell durch den Operator oder über ein Skript, wenn Dateien auf dem Server installiert werden, oder automatisch durch einen Änderungsdetektionsmechanismus (z. B. ein separates Verfahren, das regelmäßig auf Änderungen prüft) identifiziert.

[0132] Eine Ressourcenänderung veranlasst den Reflektor, eine ”Annuliert”-Nachricht an seinen Kontaktrepeater zu senden, der die Nachricht an den Master-Repeater weiterleitet. Die Annulliert-Nachricht enthält eine Liste von Ressourcen-Identifikatoren (oder regulären Ausdrücken, die Muster von Ressourcen-Identifikatoren identifizieren), die sich geändert haben. (Reguläre Ausdrücke werden verwendet, ein Verzeichnis oder einen gesamten Server zu annullieren). Das Repeaternetzwerk verwendet ein zweiphasiges Bindungsverfahren, um sicherzustellen, dass alle Repeater eine gegebene Ressource korrekt annullieren.

[0133] Das Annullierungsverfahren funktioniert wie folgt:
Der Master sendet eine ”Phase 1”-Annullierungsanforderung an alle Repeater mit Angabe der Ressourcen und regulären Ausdrücke, in denen zu annullierende Ressourcen-Serien beschrieben werden.

[0134] Wenn ein Repeater die Phase 1 Nachricht erhält, platziert er zuerst die Ressourcen-Identifikatoren oder regulären Ausdrücke in eine Liste von Ressourcen-Identifikatoren, die auf eine Annullierung warten.

[0135] Eine (in C3) angeforderte Ressource, die in der Annullierungswarteliste ist, kann aus dem Cache nicht bereitgestellt werden. Dies hindert den Cache daran, die Ressource von einem Peer-Cache anzufordern, der möglicherweise keine Annullierungsnachricht erhalten hat. Würde er auf diese Weise eine Ressource anfordern, könnte diese die frisch annullierte Ressource durch dieselben, nun veralteten, Daten ersetzen.

[0136] Der Repeater vergleicht dann den Ressourcen-Identifikator jeder Ressource in seinem Cache mit den Ressourcen-Identifikatoren und regulären Ausdrücken in der Liste.

[0137] Jede Übereinstimmung wird annulliert, indem sie als veraltet markiert und optional aus dem Cache entfernt wird. Dies bedeutet, dass eine zukünftige Anforderung nach der Ressource sie dazu bringt, eine neue Kopie der Ressource vom Reflektor abzurufen.

[0138] Wenn der Repeater die Annullierung abgeschlossen hat, sendet er eine Bestätigung an den Master. Der Master wartet, bis alle Repeater die Annullierungsanforderung bestätigt haben.

[0139] Wenn ein Repeater nicht innerhalb eines bestimmten Zeitraums bestätigt, wird er vom Master-Repeater getrennt. Wenn er sich wieder anschließt, wird er aufgefordert, seinen gesamten Cache zu entleeren, wodurch jegliches Inkonsistenzproblem beseitigt wird. (Um das Entleeren des gesamten Cache zu vermeiden, könnte der Master ein nach Datum sortiertes Protokoll aller durchgeführten Annullierungen anlegen und nur solche Dateien entleeren, die annulliert wurden, seit der wiederanschließende Repeater zum letzten Mal erfolgreich eine Annullierung durchgeführt hat. In den gegenwärtig bevorzugten Ausführungsbeispielen wird dies nicht durchgeführt, weil angenommen wird, dass es selten zu Trennungen von Repeater kommt).

[0140] Wenn alle Repeater die Annullierung bestätigt haben (oder nach Zeitablauf), sendet der Repeater eine ”Phase 2”-Annullierungsanforderung an alle Repeater. Dies veranlasst die Repeater dazu, die entsprechenden Ressourcen-Identifikatoren und regulären Ausdrücke von der Liste der eine Annullierung erwartenden Ressourcen-Identifikatoren zu entfernen.

[0141] In einem anderen Ausführungsbeispiel wird die Annullierungsanforderung so erweitert, dass ein "Server-Schub" möglich wird. Wenn in solchen Anforderungen die Phase 2 des Annullierungsverfahrens abgeschlossen ist, fordert der die Annullierungsanforderung erhaltende Repeater sofort eine neue Kopie der annullierten Ressource zur Ablage in seinem Cache an.

Protokoll und Protokollverarbeitung

[0142] Webserver-Aktivitätsprotokolle sind fundamental für die Überwachung der Aktivitäten auf einer Website. Diese Erfindung schafft "verschmolzene Protokolle", die die Aktivität an Reflektoren mit der Aktivität an Repeatern kombinieren, so dass am Ausgangsserver ein einziges Aktivitätsprotokoll erscheint, in dem sämtliche Webressourcenanforderungen enthalten sind, die im Namen dieser Site an einem Repeater vorgenommen wurden.

[0143] Dieses verschmolzene Protokoll kann von Standardverarbeitungs-Tools verarbeitet werden, so als ob es lokal generiert worden wäre.

[0144] Der Master-Repeater (oder dessen Delegierter) sammelt in periodischen Abständen Protokolle von jedem Repeater. Die gesammelten Protokolle werden verschmolzen, nach Ressourcen-Identifikator und Zeitstempel sortiert und in einer datierten Datei auf Reflektorbasis gespeichert. Das verschmolzene Protokoll für einen bestimmten Reflektor repräsentiert die Aktivität aller Repeater im Namen dieses Reflektors. Ein Reflektor kontaktiert auf periodischer Grundlage, wie vom Reflektor-Operator vorgegeben, den Master-Repeater, um seine verschmolzenen Protokolle anzufordern. Er lädt diese herunter und verschmilzt sie mit seinen lokal verwalteten Protokollen, nach Zeitstempel sortierend. Das Ergebnis ist ein verschmolzenes Protokoll, das alle Aktivitäten im Namen von Repeater und dem gegebenen Reflektor darstellt.

[0145] Aktivitätsprotokolle werden optional erweitert mit Informationen, die für das Repeater-Netzwerk wichtig sind, wenn der Reflektor vom Operator des Reflektors dazu konfiguriert wird. Insbesondere zeigt ein "erweiterter Statuscode" Informationen über jede Anforderung an, wie etwa:

1. Anforderung wurde von einem Reflektor lokal bereitgestellt;
2. Anforderung wurde zu einem Repeater reflektiert;*
3. Anforderung wurde von einem Reflektor einem Repeater bereitgestellt;*
4. Anforderung nach nicht-repetierbarer Ressource wurde vom Repeater bereitgestellt;*
5. Anforderung wurde von einem Repeater aus dem Cache bereitgestellt;
6. Anforderung wurde von einem Repeater nach Auffüllen des Cache bereitgestellt;
7. Auf Annullierung wartende Anforderung wurde von einem Repeater bereitgestellt.

[0146] (Die mit * markierten Aktivitäten stellen Zwischenstadien einer Anforderung dar und erscheinen normalerweise nicht in einem fertigen Aktivitätsprotokoll).

[0147] Zusätzlich enthalten Aktivitätsprotokolle eine Dauer und erweiterte Präzisionszeitstempel. Die Dauer ermöglicht die Analyse der zum Bereitstellen einer Ressource benötigten Zeit, der benützten Bandbreite, der Anzahl der zu einem bestimmten Zeitpunkt parallel bearbeiteten Anforderungen und anderer nützlicher Informationen. Der erweiterte Präzisionszeitstempel ermöglicht die präzise Verschmelzung von Aktivitätsprotokollen.

[0148] Repeater benützen das Network Time Protocol (NTP), um synchronisierte Uhren zu verwalten. Reflektoren können entweder das NTP benützen oder eine Zeitasymmetrie berechnen, um einigermaßen präzise Zeitstempel relativ zu ihrem Kontaktrepeater bereitzustellen.

Durchsetzung der Gebundenen Aggregierten Informationsrate

[0149] Das Repeater-Netzwerk überwacht und limitiert die aggregierte Rate, mit der Daten im Namen eines gegebenen Subscribers von allen Repeater bereitgestellt werden. Dieser Mechanismus bietet folgende Vorteile:

1. stellt ein Mittel zur Preisgestaltung von Repeater-Dienstleistungen bereit;
2. stellt ein Mittel zur Einschätzung und Reservierung von Kapazitäten bei Repeatern bereit;
3. stellt ein Mittel bereit, um Clients einer belegten Site davon abzuhalten, den Zugriff auf andere Sites zu beschränken.

[0150] Für jeden Subscriber wird ein "Schwellwert für die aggregierte Informationsrate" (TAIR – Threshold Aggregate Information Rate) konfiguriert und am Master-Repeater verwaltet. Dieser Schwellwert ist nicht notwendigerweise die gebundene Rate, sie kann ein Vielfaches dieser Rate sein, basierend auf einer Preispolitik.

[0151] Jeder Repeater misst periodisch (typischerweise etwa einmal pro Minute) die Informationsratenkomponente jedes Reflektors, für den er Ressourcen bereitgestellt, indem er die Anzahl der im Namen dieses Reflektors übertragenen Bytes bei jeder Überbringung einer Anforderung aufzeichnet. Die so erzeugte Tabelle wird einmal pro Zeitintervall zum Master-Repeater gesendet. Der Master-Repeater kombiniert die Tabellen von jedem Repeater und summiert

die gemessenen Informationen jedes Reflektors über alle Repeater hinweg, welche Ressourcen für diesen Reflektor bereitstellen, um die ”gemessene aggregierte Informationsrate” (MAIR – Measured Aggregated Information Rate) jedes Reflektors zu ermitteln.

[0152] Wenn die MAIR eines gegebenen Reflektors größer ist als die TAIR für diesen Reflektor, wird die MAIR vom Master auf alle Repeater und zum entsprechenden Reflektor gesendet.

[0153] Wenn ein Reflektor eine Anforderung erhält, stellt er fest, ob seine zuletzt berechnete MAIR größer ist als seine TAIR. Ist dies der Fall, entscheidet der Reflektor nach dem Wahrscheinlichkeitsprinzip, ob er die Reflexion unterdrückt, indem er die Anforderung lokal bereitstellt (in B2). Die Wahrscheinlichkeit einer Unterdrückung der Reflexion steigt exponentiell mit der Differenz zwischen MAIR und CAIR.

[0154] Die lokale Abgabe einer Anforderung während einer Spitzenperiode kann den lokalen Ausgangsserver belasten, doch hindert sie diesen Subscriber daran, mehr als die zugeteilte Bandbreite aus dem gemeinsamen Repeater-Netzwerk zu beanspruchen.

[0155] Wenn ein Repeater eine Anforderung nach einem gegebenen Subscriber erhält (in C2), stellt er fest, ob der Subscriber nahe an seinem Schwellwert für die aggregierte Informationsrate läuft. Ist dies der Fall, entscheidet er nach dem Wahrscheinlichkeitsprinzip, ob er seine Auslastung durch Umleiten der Anforderung zum Reflektor zurück reduzieren soll. Die Wahrscheinlichkeit steigt exponentiell mit der Annäherung der aggregierten Informationsrate des Reflektors an ihr Limit.

[0156] Wenn eine Anforderung zurück zu einem Reflektor reflektiert wird, wird an den Ressourcen-Identifikator eine besondere Zeichenfolge angehängt, so dass der empfangende Reflektor nicht versucht, sie erneut zu reflektieren. Im aktuellen System hat die Zeichenfolge die Form:

”src=overload”.

[0157] Der Reflektor prüft in B2 auf diese Zeichenfolge.

[0158] Der oben beschriebene Mechanismus zur Limitierung der aggregierten Informationsrate ist ziemlich grob. Er limitiert auf der Ebene von Sessions mit Clients (zumal wenn ein Client zu einem bestimmten Repeater reflektiert worden ist, das Überschreibverfahren dazu neigt, den Client zur Rückkehr zu diesem Repeater zu bewegen) und bestenfalls einzelnen Anforderungen nach Ressourcen. Ein feinerer Mechanismus zur Durchsetzung von TAIR-Grenzwerten in Repeatern funktioniert mittels Senkung des Bandbreitenverbrauchs eines belegten Subscribers, wenn andere Subscriber um Bandbreite konkurrieren.

[0159] Der feine Mechanismus ist eine Form von Daten-”Ratenformung”. Er erweitert den Mechanismus, der Ressourcendaten in eine Verbindung kopiert, wenn eine Antwort an einen Client gesendet wird. Wenn zu dem Zeitpunkt, zu dem eine Anforderung empfangen wird, ein Ausgabekanal eingerichtet ist, identifiziert der Repeater in C2, für welchen Subscriber der Kanal tätig ist, und trägt den Subscriber in ein mit dem Kanal assoziiertes Datenfeld ein. Jedesmal wenn eine ”Schreib”-Operation zum Kanal bevorsteht, untersucht der Gemessene Ausgabestrom zuerst die oben berechneten aktuellen Werte der MAIR und der TAIR für den gegebenen Subscriber. Wenn die MAIR größer ist als die TAIR, unterbricht der Mechanismus kurz, bevor er die Schreiboperation durchführt. Die Länge der Pause ist proportional zu der Menge, um die die MAIR die TAIR übersteigt. Die Pause stellt sicher, dass Aufgaben, die andere Ressourcen an andere Clients senden, vielleicht im Namen anderer Subscriber, eine Gelegenheit erhalten, ihre Daten zu senden.

Elastizität des Repeater-Netzwerks

[0160] Das Repeater-Netzwerk ist in der Lage, sich zu erholen, wenn eine Repeater- oder Netzwerkverbindung ausfällt.

[0161] Ein Repeater kann nur funktionieren, wenn er an den Master-Repeater angeschlossen ist. Der Master-Repeater tauscht kritische Informationen mit anderen Repeater aus, darunter Informationen über Repeaterauslastung, aggregierte Informationsrate, Subscriber und Verbindungskosten.

[0162] Wenn ein Master ausfällt, stellt ein ”Nachfolge”-Prozess sicher, dass ein anderer Repeater die Rolle des Masters übernimmt und das Netzwerk als Ganzes funktionsfähig bleibt. Wenn ein Master ausfällt oder eine Verbindung zu einem Master infolge eines Netzwerkproblems ausfallt, entdeckt jeder Repeater, der versucht, mit dem Master zu kommunizieren, den Ausfall entweder durch einen Hinweis vom TCP/IP oder durch Zeitablauf von einer regulären ”Herzschlag”-Nachricht, die er zu dem Master sendet.

[0163] Wenn ein Repeater von seinem Master getrennt wird, versucht er auf Basis einer konfigurierbaren Datei, die als seine ”Nachfolgeliste” bezeichnet wird, sofort sich einer Serie potenzieller Master anzuschließen.

[0164] Der Repeater versucht hintereinander jedes System in der Liste, bis ein erfolgreicher Anschluss an einen Master hergestellt ist. Wenn er in diesem Verfahren auf seinen eigenen Namen trifft, über-

nimmt er die Rolle des Masters und akzeptiert Anschlüsse von anderen Repeater. Wenn ein Repeater, der nicht an der Spitze der Liste ist, zum Master wird, wird er als ”Provisorischer Master” bezeichnet.

[0165] Eine Netzwerkpartition kann zwei Gruppen von Repeater jeweils zur Wahl eines Masters veranlassen. Wenn die Partition korrigiert ist, ist es erforderlich, dass der übergeordnete Master das Netzwerk übernimmt. Wenn also ein Repeater als provisorischer Master fungiert, versucht er regelmäßig, sich an irgendeinen höhergestellten Master in der Nachfolgeliste anzuschließen. Ist er erfolgreich, trennt er sofort die Verbindungen zu allen an ihn angeschlossenen Repeatern. Wenn diese erneut ihre Nachfolgelisten versuchen, schließen sie sich dem übergeordneteren Master-Repeater an.

[0166] Um Datenverluste zu vermeiden, akzeptiert ein provisorischer Master keine Konfigurationsänderungen und verarbeitet keine Protokolldateien. Um diese Aufgaben übernehmen zu können, muss er mittels manueller Manipulation seiner Nachfolgeliste informiert werden, dass er ein primärer Master ist. Jeder Repeater lädt regelmäßig seine Nachfolgeliste neu, um festzustellen, ob er seine Vorstellung davon, wer der Master ist, ändern solle.

[0167] Wenn ein Repeater vom Master getrennt wird, muss er seinen Cache neu synchronisieren, wenn er sich wieder an den Master anschließt. Der Master kann eine Liste neuester Cache-Annullierungen verwalten und alle Annullierungen, die er während der Verbindungstrennung nicht verarbeiten konnte, an den Repeater senden. Wenn diese Liste aus irgendeinem Grund nicht verfügbar ist (beispielsweise weil der Reflektor zu lange getrennt war), muss der Reflektor seinen gesamten Cache annullieren.

[0168] Ein Reflektor darf Anforderungen nur dann reflektieren, wenn er mit einem Repeater verbunden ist. Der Reflektor ist für kritische Informationen von seinem Kontaktrepeater abhängig, etwa für Auslastungs- und Verknüpfungskostentabellen und die aktuelle aggregierte Informationsrate. Ein Reflektor, der nicht mit einem Repeater verbunden ist, kann weiterhin Anforderungen empfangen und lokal behandeln.

[0169] Wenn ein Reflektor seine Verbindung mit einem Repeater aufgrund eines Repeaterausfalls oder eines Netzwerkausfalls verliert, bleibt er weiterhin in Betrieb, während er versucht, die Verbindung zu einem Repeater herzustellen. Jedesmal wenn ein Reflektor versucht, die Verbindung zu einem Repeater herzustellen, verwendet er DNS zur Identifizierung einer Serie von Repeaterkandidaten mit einem Domainnamen, der das Repeater-Netzwerk repräsentiert. Der Reflektor probiert jeden Repeater in dieser Serie, bis er einen erfolgreichen Kontakt herstellt. Bis ein erfolgreicher Kontakt hergestellt ist, stellt der Reflektor sämtliche Anforderungen lokal bereit. Wenn sich ein Reflektor mit einem Repeater verbindet, kann ihn der Repeater beauftragen, die Kontaktierung eines anderen Repeaters zu versuchen; so kann das Repeater-Netzwerk sicherstellen, dass kein einzelner Repeater zu viele Kontakte hat.

[0170] Wenn der Kontakt hergestellt ist, stellt der Reflektor seinem Kontaktrepeater die Versionsnummern jeder seiner Tabellen bereit. Der Repeater entscheidet dann, welche Tabellen aktualisiert werden sollten, und sendet entsprechende Aktualisierungen an den Reflektor. Nachdem alle Tabellen aktualisiert wurden, verständigt der Repeater den Reflektor, dass dieser nunmehr mit dem Reflektieren von Anforderungen beginnen kann.

Verwendung eines Proxy-Caches in einem Repeater

[0171] Repeater sind absichtlich so angelegt, dass jeder Proxy-Cache als darin enthaltene Komponente verwendet werden kann. Dies ist deshalb möglich, weil der Repeater HTTP-Anforderungen erhält und diese in eine Form konvertiert, die von dem Proxy-Cache erkannt wird.

[0172] Anderseits sind mehrere Modifikationen an einem Standard-Proxy-Cache als Optimierungen vorgenommen worden bzw. können vorgenommen werden. Dies umfasst insbesondere die Fähigkeit, eine Ressource passend zu annullieren, die Fähigkeit, Cache-Quoten zu unterstützen, und die Fähigkeit, das Herstellen einer Zusatzkopie jeder Ressource zu vermeiden, wenn diese vom Proxy-Cache durch den Repeater zum Anforderungsprogramm geht.

[0173] In einem bevorzugten Ausführungsbeispiel wird ein Proxy-Cache dazu verwendet, C3 zu implementieren. Der Proxy-Cache ist für die ausschließliche Benützung durch einen oder mehrere Repeater dediziert. Jeder Repeater, der eine Ressource vom Proxy-Cache anfordert, konstruiert eine Proxy-Anforderung von der ankommenden Ressourcenanforderung. Eine normale HTTP GET Anforderung an einen Server enthält nur den Pfadnamensteil der URL – Schema und Servername sind impliziert. (In einer HTTP GET Anforderung an einen Repeater enthält der Pfadnamensteil der URL den Namen des Ausgangsservers, in dessen Auftrag die Anforderung erfolgt, wie oben beschrieben). Allerdings benötigt eine Proxy Agent GET Anforderung eine vollständige URL. Deshalb muss der Repeater eine Proxy-Anforderung konstruieren, welche die gesamte URL vom Pfadabschnitt der URL, die er empfängt, enthält. Insbesondere wenn die ankommende Anforderung die Form

GET/<Ausgangsserver>/<pfad>
annimmt, konstruiert der Repeater eine Proxy-Anfor-

derung der Form:

GET http://<Ausgangsserver>/<pfad>
und wenn die ankommende Anforderung die Form:

GET <Ausgangsserver>@proxy=<schema>:<typ>@/<pfad>
annimmt, konstruiert der Repeater eine Proxy-Anforderung der Form:

GET <schema>://<Ausgangsserver>/<pfad>

Cache-Kontrolle

[0174] HTTP-Antworten enthalten Anweisungen, die als Cache-Kontrollanweisungen bezeichnet werden. Sie zeigen einem Cache an, ob die angehängte Ressource in den Cache gelegt werden kann, und wenn dies der Fall ist, wann sie ablaufen sollte. Ein Website-Administrator konfiguriert die Website, so dass entsprechende Anweisungen angehängt werden. Vielfach weiß ein Administrator nicht, wie lange eine Seite frisch ist, und er muss eine kurze Ablaufzeit definieren, um zu verhindern, dass Caches veraltete Daten bereitstellen. In vielen Fällen gibt ein Website-Operator nur deshalb eine kurze Ablaufzeit an, um die Anforderungen (oder Zugriffe) zu erhalten, die ansonsten durch die Präsenz eines Cache maskiert würden. Dies ist in der Branche als ”Cache-Busting” bekannt. Zwar halten einige Cache-Betreiber das Cache-Busting für unhöflich, doch Werbetreibende, die von solchen Informationen abhängig sind, betrachten es möglicherweise als unverzichtbar.

[0175] Wenn eine Ressource in einem Repeater gespeichert wird, können ihre Cache-Anweisungen vom Repeater ignoriert werden, weil der Repeater explizite Annullierungsereignisse erhält, um festzustellen, wann eine Ressource veraltet ist. Wenn als Cache am Repeater ein Proxy-Cache verwendet wird, können die zugehörigen Cache-Anweisungen vorübergehend deaktiviert werden. Sie müssen allerdings reaktiviert werden, wenn die Ressource vorn Cache einem Client bereitgestellt wird, um die Cache-Kontrollpolitik (einschließlich allenfalls Cache-Busting) wirksam werden zu lassen.

[0176] Die vorliegende Erfindung enthält Mechanismen, mit denen der Proxy-Cache in einem Repeater daran gehindert wird, Cache-Kontrollanweisungen zu befolgen; es ist aber zulässig, dass die Anweisungen vom Repeater bereitgestellt werden.

[0177] Wenn ein Reflektor eine Ressource einem Repeater in B4 bereitstellt, ersetzt er alle Cache-Anweisungen durch modifizierte Anweisungen, die vom Repeater-Proxy-Cache ignoriert werden. Er stellt dazu eine bestimmte Zeichenfolge, wie etwa ”wr-”, an den Anfang des HTTP-Tags. So wird ”läuft ab” zu ”wr-läuft ab”, und aus ”Cache-Kontrolle” wird ”wr-Cache-Kontrolle”. Auf diese Weise wird der Proxy-Cache selbst an der Befolgung der Anweisungen gehindert. Wenn ein Repeater eine Ressource in C4 bereitstellt und der anfordernde Client kein anderer Repeater ist, sucht er die HTTP-Tags, die mit ”wr-” beginnen und entfernt das ”wr-”. So kann der Client die Ressource abrufen, um die Anweisungen zu befolgen.

Revalidierung der Ressource

[0178] Es gibt mehrere Fälle, in denen eine Ressource so lange im Cache sein kann, wie der Ausgangsserver jedes Mal bei ihrer Bereitstellung konsultiert wird. In einem Fall wird die Anforderung nach der Ressource an ein sogenanntes ”Cookie” angehängt. Dem Ausgangsserver muss das Cookie übermittelt werden, um die Anforderung aufzuzeichnen und zu bestimmen, ob die im Cache befindliche Ressource bereitgestellt werden kann oder nicht. In einem anderen Fall ist die Anforderung nach der Ressource an einen Authentizierungs-Header (der das Anforderungsprogramm mit Benutzer-ID und Passwort identifiziert) angehängt. Jede neue Anforderung nach der Ressource muss am Ausgangsserver geprüft werden, um sicherzustellen, dass das Anforderungsprogramm für den Zugriff auf die Ressource autorisiert ist.

[0179] Die HTTP 1.1 Spezifikation definiert einen Antwort-Header mit dem Titel ”Must-Revalidate”, der einem Ausgangsserver ermöglicht, einen Proxy-Cache anzuweisen, eine Ressource jedes Mal, wenn eine Anforderung empfangen wird, zu ”revalidieren”. Normalerweise wird dieser Mechanismus verwendet, um festzustellen, ob eine Ressource noch frisch ist. In der vorliegenden Erfindung ermöglicht es Must-Revalidate, einen Ausgangsserver zu ersuchen, eine Anforderung zu validieren, die ansonsten von einem Repeater bereitgestellt wird.

[0180] Die Reflektor-Regelbasis enthält Informationen, die festlegen, welche Ressourcen repetiert werden können, aber bei jeder Bereitstellung revalidiert werden müssen. Für jede solche Ressource hängt der Reflektor in B4 einen Must-Revalidate-Header an. Jedes Mal wenn an einen Repeater eine Anforderung nach einer Ressource im Cache kommt, die mit einem Must-Revalidate-Header markiert ist, wird die Anforderung zum Reflektor zur Validierung weitergeleitet, bevor die angeforderte Ressource bereitgestellt wird.

Cache-Quoten

[0181] Die Cache-Komponente eines Repeaters ist jenen Subscribern gemeinsam, die Clients zu diesem Repeater reflektieren. Um den Subscribern einen fairen Zugriff auf die Speichereinrichtungen zu erlauben, kann der Cache zur Unterstützung von Quoten

erweitert werden.

[0182] Normalerweise kann ein Proxy-Cache mit einem Plattenplatz-Schwellwert T konfiguriert werden. Immer wenn mehr als T Bytes im Cache gespeichert sind, versucht der Cache, zu eliminierende Ressourcen zu finden.

[0183] Typischerweise verwendet ein Cache den Least-recently-used-Algorithmus (LRU-Algorithmus; der Algorithmus nach dem Prinzip der am weitesten zurückliegenden Verwendung), um festzustellen, welche Ressource eliminiert werden soll; anspruchsvollere Caches benützen andere Algorithmen. Ein Cache kann auch mehrere Schwellwerte unterstützen, beispielsweise einen tieferen Schwellwert, der, sobald er erreicht ist, ein Low-Priority-Hintergrundverfahren zum Entfernen von Einträgen aus dem Cache auslöst, und einen höheren Schwellwert, der, sobald er erreicht ist, Ressourcen daran hindert, im Cache abgelegt zu werden, bis ausreichend freier Plattenplatz gewonnen wurde.

[0184] Wenn zwei Subscriber A und B einen Cache teilen und während einer Zeitperiode auf mehr Ressourcen des Subscribers A zugegriffen wird als auf Ressourcen des Subscribers B, sind weniger der Ressourcen Bs im Cache, wenn neue Anforderungen eintreffen. Es ist möglich, dass infolge des Verhaltens von A die Ressourcen von B nie in den Cache kommen, wenn sie angefordert werden. In der vorliegenden Erfindung ist dieses Verhalten nicht wünschenswert. Um dieses Problem zu bekämpfen, erweitert die Erfindung den Cache an einem Repeater zur Unterstützung von Cache-Quoten.

[0185] Der Cache zeichnet die von jedem Subscriber in D_S benützte Platzmenge auf und unterstützt eine konfigurierbare Schwelle T_S für jeden Subscriber.

[0186] Immer wenn eine Ressource zum Cache hinzugefügt wird (in C3), wird der Wert D_S für den die Ressource bereitstellenden Subscriber aktualisiert. Wenn D_S größer ist als T_S, versucht der Cache, aus den mit dem Subscriber S assoziierten Ressourcen zu eliminierende Ressourcen zu finden. Der Cache wird wirksam in getrennte Bereiche für jeden Subscriber partitioniert.

[0187] Der ursprüngliche Schwellwert T wird noch immer unterstützt. Wenn die Summe reservierter Segmente für jeden Subscriber kleiner ist als der gesamte im Cache reservierte Platz, so ist der restliche Bereich ”allgemein” und unterliegt dem Wettbewerb zwischen den Subscribern.

[0188] Es ist zu beachten, dass dieser Mechanismus implementiert werden könnte, indem der oben erörterte, bestehende Proxy-Cache modifiziert wird, oder er könnte auch implementiert werden, ohne dass der Proxy-Cache modifiziert wird – wenn der Proxy-Cache zumindest einem externen Programm ermöglicht, eine Liste von Ressourcen im Cache zu erhalten und eine gegebene Ressource aus dem Cache zu entfernen.

Überschreiben von Repeatern

[0189] Wenn ein Repeater eine Anforderung nach einer Ressource abruft, kann sein Proxy-Cache so konfiguriert sein, dass er feststellt, ob ein Peer-Cache die angeforderte Ressource enthält. Ist dies der Fall, erhält der Proxy-Cache die Ressource vom Peer-Cache, was schneller sein kann als sie vom Ausgangsserver (dem Reflektor) zu erhalten. Eine Folge daraus ist jedoch, dass überschriebene, vom Peer-Cache abgerufene HTML-Ressourcen, den falschen Repeater identifizieren würden. Um kooperierende Proxy-Caches zu erlauben, werden deshalb Ressourcen vorzugsweise am Repeater überschrieben.

[0190] Wenn eine Ressource für einen Repeater überschrieben wird, wird ein spezielles Tag an den Anfang der Ressource gesetzt. Bei der Konstruktion einer Antwort untersucht der Repeater das Tag, um festzustellen, ob die Ressource anzeigt, dass zusätzliches Überschreiben erforderlich ist. Ist dies der Fall, modifiziert der Repeater die Ressource, indem er Verweise auf den alten Repeater durch Verweise auf den neuen Repeater ersetzt.

[0191] Diese Überschreibung ist nur dann erforderlich, wenn eine Ressource dem Proxy-Cache an einem anderen Repeater bereitgestellt wird.

Repeater-Seiten-Einbezug (Repeater-Side Include)

[0192] Manchmal konstruiert ein Ausgangsserver eine individualisierte Ressource für jede Anforderung (beispielsweise wenn eine Werbung auf der Grundlage der Biografie des anfordernden Clients eingeschoben wird). In so einem Fall muss diese Ressource lokal bereitgestellt anstatt repetiert werden. Im allgemeinen enthält eine individualisierte Ressource zusammen mit den individuellen Informationen Text und Verweise auf andere, repetierbare Ressourcen.

[0193] Das Verfahren, mit dem eine ”Seite” von einer Textressource und möglicherweise einer oder mehreren Bildressourcen zusammengestellt wird, wird vom Webbrowser unter Lenkung von HTML ausgeführt. Es ist jedoch nicht möglich, HTML dazu zu verwenden, einen Browser zur Zusammenstellung einer Seite unter Verwendung von Text oder Anweisungen von einer getrennten Ressource zu veranlassen. Deshalb enthalten individualisierte Ressourcen oftmals notwendigerweise große Mengen statischen Texts, der ansonsten repetierbar wäre.

[0194] Um diese potenzielle Ineffizienz aufzulösen, erkennen Repeater eine spezielle Anweisung mit der Bezeichnung "Repeater-Seiten-Einbezug". Diese Anweisung ermöglicht es dem Repeater, unter Verwendung einer Kombination repetierbarer und lokaler Ressourcen eine individualisierte Ressource zusammenzustellen. Auf diese Weise kann der statische Text repetierbar gemacht werden, und nur die spezielle Anweisung muss lokal vom Reflektor bereitgestellt werden.

[0195] Beispielsweise könnte eine Ressource X aus individualisierten Anweisungen bestehen, die eine Werbeleiste auswählen, gefolgt von einem großen Textartikel. Um diese Ressource repetierbar zu machen, muss der Website-Administrator eine zweite Ressource Y nutzbar machen, um die Leiste zu wählen. Die Ressource X ist so modifiziert, dass sie eine Repeater-Seiten-Einbezug-Anweisung enthält, in der die Ressource Y zusammen mit dem Artikel identifiziert wird. Die Ressource Y wird geschaffen und enthält nur die individualisierten Anweisungen, welche eine Werbeleiste auswählen. Jetzt ist die Ressource X repetierbar, und nur die Ressource Y, die relativ klein ist, ist nicht repetierbar.

[0196] Wenn ein Repeater eine Antwort konstruiert, legt er fest, ob die bereitgestellte Ressource eine HTML-Ressource ist, und ist dies der Fall, sucht er sie nach Repeater-Seiten-Einbezug-Anweisungen ab. Jede solche Anweisung enthält eine URL, die der Repeater auflöst und anstelle der Anweisung einsetzt. Die gesamte Ressource muss zusammengestellt werden, bevor sie bereitgestellt wird, um ihre endgültige Größe zu bestimmen, da die Größe in einen Antwort-Header vor der Ressource aufgenommen ist.

[0197] So wird eine Methode und eine Vorrichtung zur dynamischen Nachbildung ausgewählter Ressourcen in Computernetzwerken geschaffen. Fachpersonen erkennen, dass die vorliegende Erfindung auch in anderen als den beschriebenen Ausführungsbeispielen ausgeführt werden kann, die hier nur aus illustrativen Gründen dargestellt werden und keinen einschränkenden Charakter haben. Die vorliegende Erfindung wird nur durch die angehängten Patentansprüche eingeschränkt.

Patentansprüche

1. Ein in einem Computernetzwerk (**100**) betriebenes System, in welchem Clients (**106**) mit einem Ausgangsserver (**102**) Verbindung aufnehmen, wobei das System beinhaltet:
eine Serie von Repeaterservern (**104a** ...), die vom Ausgangsserver (**102**) verschieden sind und als Host für mindestens einige der eingebetteten Objekte von Webseiten dienen, die normalerweise vom Ausgangsserver (**102**) als Host verwaltet werden;
einen Reflektor (**108**), in dem enthalten ist:
eine Routine, die mindestens eine URL eines eingebetteten Objekts einer Webseite so modifiziert, dass sie zu einem Server in der Serie von Repeaterservern anstatt zum Ausgangsserver aufgelöst wird; und
der Reflektor weiter enthält:
einen Server-Auswahlmechanismus, der so aufgebaut und angepasst ist, dass er auf eine bestimmte Client-Anforderung hin einen geeigneten Repeaterserver aus der Serie von Repeaterservern identifiziert,
wobei als Reaktion auf Anforderungen nach der Webseite, die von den Clients erzeugt werden, die Webseite einschließlich der modifizierten URL des eingebetteten Objekts vom Reflektor bereitgestellt wird, wobei der Reflektor Teil des Ausgangsservers ist und das eingebettete Objekt, das durch die modifizierte URL des eingebetteten Objekts identifiziert wird, von einem gegebenen der vom Server-Auswahlmechanismus identifizierten Repeaterserver bereitgestellt wird,
dadurch gekennzeichnet,
dass der Server-Auswahlmechanismus einen Repeaterserver anhand der Netzwerkadresse des anfordernden Clients und anhand von Daten identifiziert, die sich auf Netzwerkverbindungs-Kosten beziehen, wobei die genannten Verbindungskosten Daten enthalten, die bestimmt wurden unter Verwendung von Sondendiensten, die Datenpfade im Netzwerk abfragen, um Messungen zu erhalten, die die relativen Kosten der Übertragung von Daten von Repeaterservern zu anderen Lokationen im Netzwerk reflektieren.

2. System nach Anspruch 1, in welchem bei einem Ausfall eines der Repeaterserver (**104a** ...) ein anderer Repeaterserver die Rolle eines ausgefallenen Servers übernimmt.

3. System nach Anspruch 1, in welchem der Server-Auswahlmechanismus außerdem einen Repeaterserver in Abhängigkeit von der Auslastung der Repeaterserver identifiziert.

4. System nach Anspruch 1, in welchem der Server-Auswahlmechanismus eine Netzwerkübersicht beinhaltet, anhand derer eine von einem Client erzeugte Anforderung nach dem eingebetteten Objekt zielgerichtet versandt werden kann.

5. System nach Anspruch 1, in dem weiter enthalten ist:
mindestens ein DNS-Server, um eine Namen-zu-Adresse Auflösung von Hostnamen in URLs von eingebetteten Objekten zur Verfügung zu stellen.

6. Verfahren zum Bereitstellen einer auf einem Ausgangsserver unterstützten Seite, wobei das Verfahren zur Nutzung eines in einem Computernetzwerk betriebenen Systems dient und das System fol-

gendes beinhaltet:
eine Serie von Repeaterservern (**104a** ...), die vom Ausgangsserver (**102**) verschieden sind und als Hostsystem für mindestens einige der eingebetteten Objekte von Webseiten dienen, die normalerweise vom Ausgangsserver als Host verwaltet werden; und
einen Reflektor (**108**), in dem enthalten ist:
eine Routine, die bewirkt, dass mindestens eine URL eines eingebetteten Objekts einer Webseite zu einem Server in der Serie von Repeaterservern anstatt zum Ausgangsserver aufgelöst wird; und der Reflektor weiter enthält:
einen Server-Auswahlmechanismus, der so aufgebaut und angepasst ist, dass er auf eine bestimmte Client-Anforderung hin einen geeigneten Repeaterserver aus der Serie von Repeaterservern identifiziert;
wobei der Server-Auswahlmechanismus einen Repeaterserver anhand der Netzwerkadresse des anfordernden Clients und anhand von Daten identifiziert, die sich auf Netzwerk-Verbindungskosten beziehen, wobei die genannten Verbindungskosten Daten enthalten, die unter Verwendung von Sondendiensten bestimmt wurden, die Datenpfade im Netzwerk abtasten, um Messungen zu erhalten, die relative Kosten der Übertragung von Daten von Repeaterservern zu anderen Lokationen im Netzwerk reflektieren,
wobei die Seite ein Basisdokument beinhaltet, welchem eingebettete Objekte zugeordnet sind, von denen jedes durch eine URL (Uniform Resource Locator) identifiziert wird, wobei das Verfahren Folgendes beinhaltet:
Modifizieren der URL eines eingebetteten Objekts zur Erzeugung einer modifizierten URL mit einem zusätzlichen neuen Hostnamen, der auf einen Repeaterserver in der Serie der Repeaterserver verweist, entsprechend der Identifikation durch den Serverauswahl-Mechanismus;
Bereitstellen der Seite mit der modifizierten URL vom Reflektor als Reaktion auf eine Anforderung zum Bereitstellen der Seite;
Versuch zum Bereitstellen des eingebetteten Objekts von einem durch den neuen Host-Namen identifizierten Repeaterserver aus;
Abrufen des eingebetteten Objekts durch den Repeaterserver vom Ausgangsserver, wenn eine Kopie des eingebetteten Objekts auf dem Repeaterserver nicht verfügbar ist.

7. Verfahren nach Anspruch 6, wobei das Modifizieren der URL das Hinzufügen eines neuen Hostnamens zur ursprünglichen URL beinhaltet.

8. Verfahren nach Anspruch 7, wobei der ursprüngliche Hostname als Bestandteil der modifizierten URL beibehalten wird.

9. Verfahren nach Anspruch 6, wobei die ursprüngliche URL einen Ausgangsserver-Namen und einen Pfad beinhaltet, und wobei die modifizierte URL einen Repeaterserver-Namen beinhaltet.

10. Verfahren nach Anspruch 9, wobei die ursprüngliche URL die Form hat
„http://<server>/<pfad>“,
wobei „<server>“ der Ausgangsservername ist, wobei <pfad> ein Pfad ist, der zu der durch die URL angegebenen Ressource führt, und wobei die modifizierte URL die Form hat
“http://<repeater>/<server>/<pfad>“,
wobei ‘<repeater>“ der Hostname des Repeaterservers ist.

11. Verfahren nach Anspruch 9, wobei die ursprüngliche URL die Form hat
‘http://<server>/<pfad>‘,
wobei ‘<server>“ der Ausgangsserver-Name ist, wobei “<pfad>“ ein Pfad ist, der zu der durch die URL angegebenen Ressource führt, und wobei die modifizierte URL die Form hat
”http://<repeater>/<server>/<pfad>”,
wobei ‘<repeater>“ ein Hostname einer Serie von Repeaterservern ist.

12. Verfahren nach Anspruch 7, wobei
das Modifizieren der URL die Schritte beinhaltet, die URL für das eingebettete Seitenobjekt so zu modifizieren, dass sie einen Hostnamen enthält, der einem vom Ausgangsserver bereitgestellten Domainnamen und Pfad vorangestellt ist;
das Bereitstellen der Seite das Bereitstellen der Seite mit der modifizierten URL vom Ausgangsserver gegenüber dem Browser eines Clients beinhaltet,
das Verfahren weiterhin Folgendes beinhaltet:
Zurücksenden einer IP-Adresse des identifizierten Repeaterservers an den Browser des Clients, so dass der Browser in die Lage versetzt wird, einen Versuch zum Abrufen des Objekts von diesem Repeaterserver zu unternehmen.

13. Verfahren nach Anspruch 12, wobei Kopien des eingebetteten Seitenobjekts auf einer Untermenge der Serie von Repeaterservern gespeichert sind.

14. Verfahren nach Anspruch 6, wobei das Modifizieren der URL des eingebetteten Objekts bewirkt, dass die Adresse des eingebetteten Objekts auf der Seite zu einer anderen Adresse als einer Ausgangsserver-Adresse aufgelöst wird, indem einer vom Ausgangsserver bereitgestellten Adresse vorgegebene Daten hinzugefügt werden, um so eine Alternativadresse zu erzeugen.

15. Verfahren nach Anspruch 14, wobei das Computernetzwerk das Internet ist und die Adresse des Objekts eine URL (Uniform Resource Locator) ist.

16. Verfahren nach Anspruch 15, wobei die Alter-

nativadresse gebildet wird, indem die gegebenen Daten einem Teil der vom Ausgangsserver bereitgestellten Adresse vorangestellt werden.

17. Verfahren nach einem oder mehreren der Ansprüche 14 bis 16, wobei das Auflösen der Alternativadresse den Schritt beinhaltet:
unter Verwendung der Netzwerkadresse eines anfordernden Clients sowie von Daten im Zusammenhang Internet-Verbindungskosten den Repeaterserver zu identifizieren.

18. Verfahren nach Anspruch 6, wobei das Verfahren weiterhin das Nachbilden einer Serie von Seitenobjekten über ein Weitverkehrsnetz von Repeaterservern beinhaltet.

19. Verfahren nach Anspruch 6, wobei das Verfahren weiterhin für jede modifizierte URL eines eingebetteten Objekts das Identifizieren mindestens eines Repeaterservers beinhaltet, von welchem aus das eingebettete Objekt abgerufen werden kann.

20. Verfahren nach Anspruch 19, wobei das Identifizieren das Auflösen einer Anforderung nach einem eingebetteten Objekt als Funktion der Netzwerkadresse eines anfordernden Clients beinhaltet.

21. Verfahren nach Anspruch 19, wobei das Identifizieren das Auflösen einer Anforderung nach einem eingebetteten Objekt als Funktion der Netzwerkadresse eines anfordernden Clients sowie der dann aktuellen Internet-Verbindungskosten beinhaltet.

22. Verfahren nach Anspruch 6, wobei das Verfahren weiterhin als Reaktion auf eine Anforderung nach dem eingebetteten Objekt das Auflösen des Hostnamens zu einer IP-Adresse eines bestimmten Repeaterservers beinhaltet.

23. Verfahren nach Anspruch 22, wobei das Auflösen des Hostnamens das Identifizieren einer Untermenge von Repeaterservern (**104a**, ...) beinhaltet, die in der Lage sein können, das eingebettete Objekt bereitzustellen, sowie das Identifizieren des bestimmten Repeaterservers aus der identifizierten Untermenge von Repeaterservern.

24. Verfahren nach Anspruch 23, wobei das Identifizieren des bestimmten Repeaterservers anhand der Netzwerkadresse der Client-Maschine und der Internet-Verbindungskosten erfolgt.

25. Verfahren nach Anspruch 6, wobei das Verfahren weiterhin beinhaltet:
Verteilen einer Serie von Seitenobjekten über ein Netzwerk von Repeaterservern (**104a**, ...), wobei das Netzwerk von Repeaterservern als Serie von Repeatergruppen organisiert ist,
als Reaktion auf eine Client-Anforderung nach einem eingebetteten Objekt einer Seite
Auflösen der Client-Anforderung als Funktion einer Netzwerkadresse des Clients sowie der Internet-Verbindungskosten zum Identifizieren einer gegebenen Repeatergruppe;
Zurücksenden einer IP-Adresse eines bestimmten Repeaterservers innerhalb der gegebenen Repeatergruppe an den Client.

26. System nach Anspruch 1, in welchem die Routine zum Modifizieren eine URL so modifiziert, dass sie einen Hostnamen enthält, der einem Domainnamen und Pfad vorangestellt ist.

27. System nach Anspruch 1, in welchem die Routine zum Modifizieren der ursprünglichen URL einen neuen Hostnamen hinzufügt.

Es folgen 6 Blatt Zeichnungen

Anhängende Zeichnungen

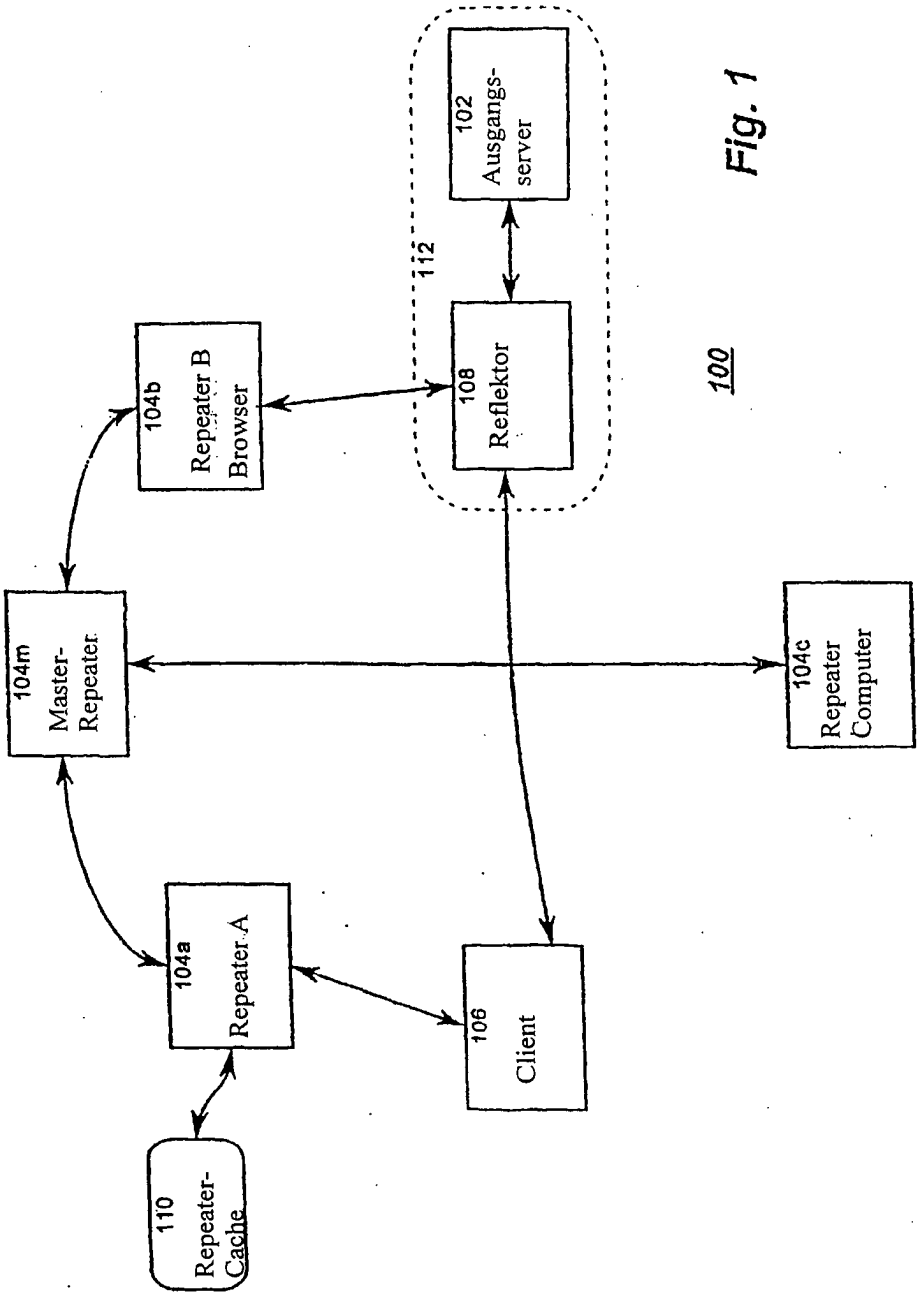

Fig. 1

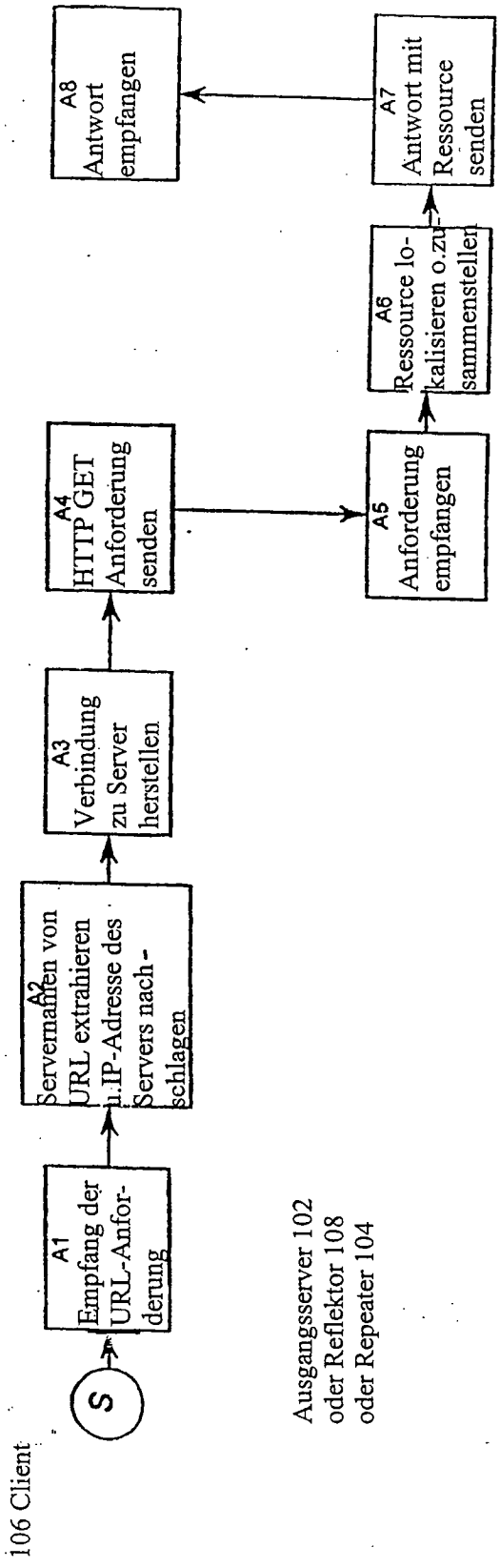
106 Client
S
A1
Empfang der URL-Anfor-derung
A2
Servernamen von URL extrahieren u.IP-Adresse des Servers nach-schlagen
A3
Verbindung zu Server herstellen
A4
HTTP GET Anforderung senden
A5
Anforderung empfangen
A6
Ressource lo-kalisieren o.zu-sammenstellen
A7
Antwort mit Ressource senden
A8
Antwort empfangen
Ausgangsserver 102
oder Reflektor 108
oder Repeater 104

Fig. 2

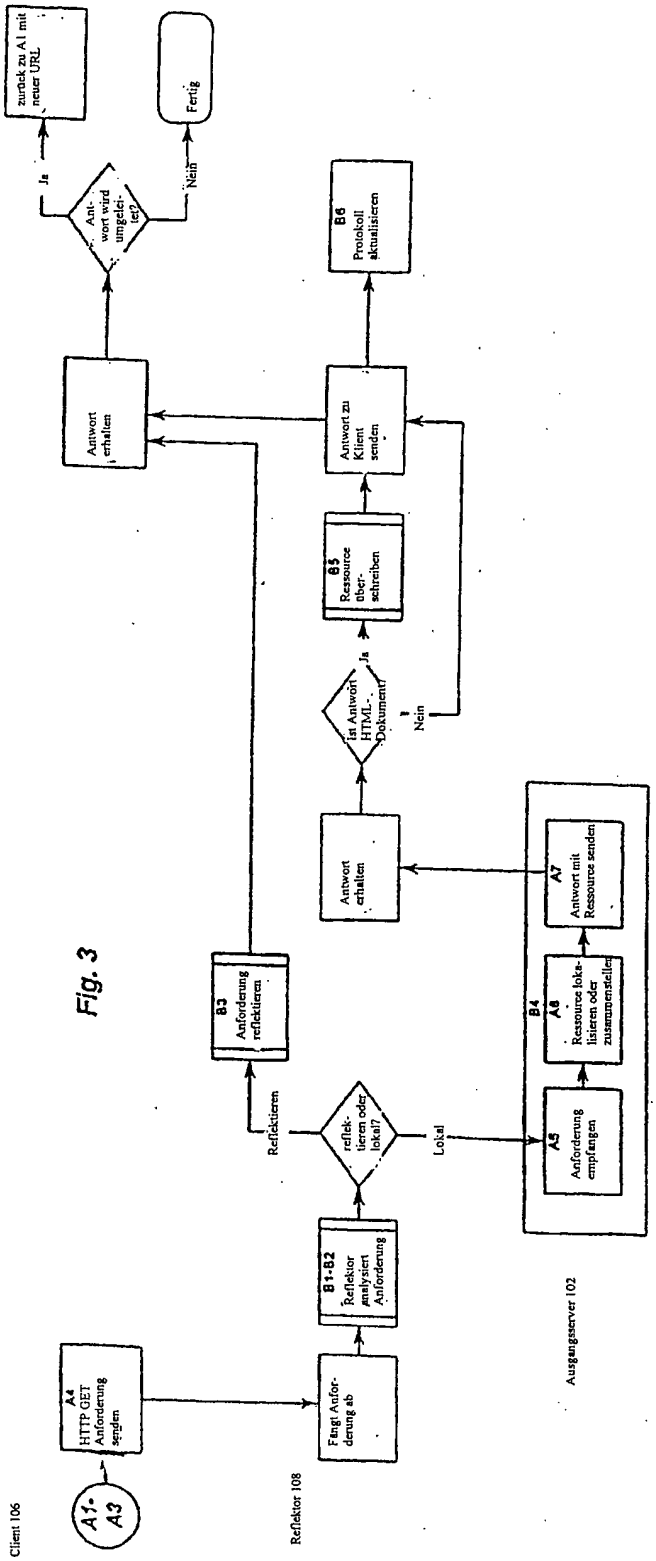
Client 106
A1-A3
A4 HTTP GET Anforderung senden
Fängt Anforderung ab
Reflektor 108
B1-B2 Reflektor analysiert Anforderung
reflektieren oder lokal?
Reflektieren
B3 Anforderung reflektieren
Lokal
Ausgangsserver 102
A5 Anforderung empfangen
B4 A6 Ressource lokalisieren oder zusammenstellen
A7 Antwort mit Ressource senden
Antwort erhalten
Ist Antwort HTML-Dokument?
Ja
Nein
B5 Ressource überschreiben
Antwort zu Klient senden
B6 Protokoll aktualisieren
Antwort erhalten
Antwort wird umgeleitet?
Ja
zurück zu A1 mit neuer URL
Nein
Fertig

Fig. 3

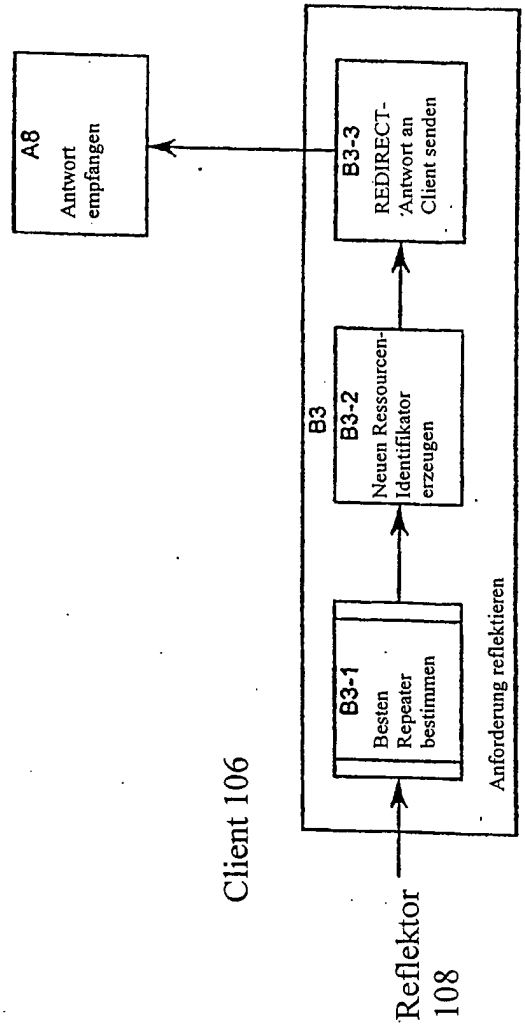
Reflektor
108
Client 106
Anforderung reflektieren
B3
B3-1
Besten
Repeater
bestimmen
B3-2
Neuen Ressourcen-
Identifikator
erzeugen
B3-3
REDIRECT-
'Antwort an
Client senden
A8
Antwort
empfangen

Fig. 4

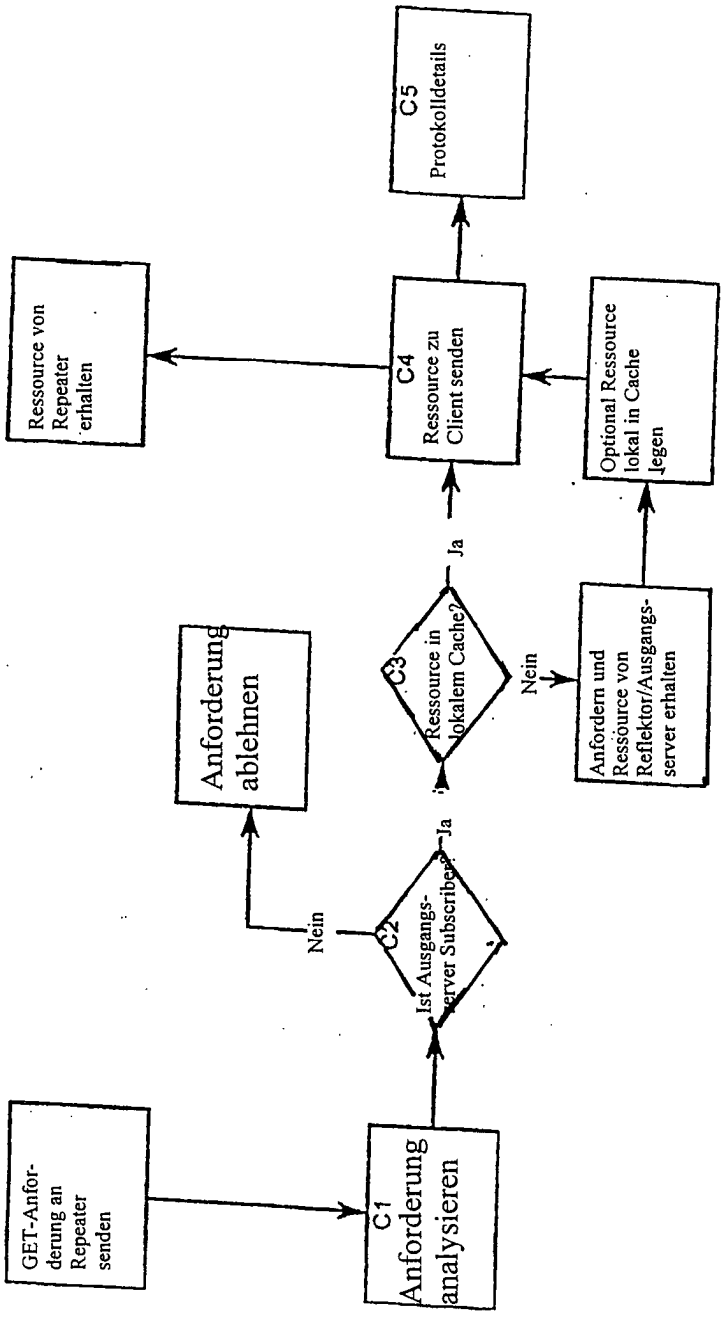
Client 106
Reflektor 108
GET-Anfor-derung an Repeater senden
C1 Anforderung analysieren
C2 Ist Ausgangs-server Subscriber?
Ja
Nein
Anforderung ablehnen
C3 Ressource in lokalem Cache?
Ja
Nein
Anfordern und Ressource von Reflektor/Ausgangs-server erhalten
Optional Ressource lokal in Cache legen
C4 Ressource zu Client senden
Ressource von Repeater erhalten
C5 Protokolldetails

Fig. 5

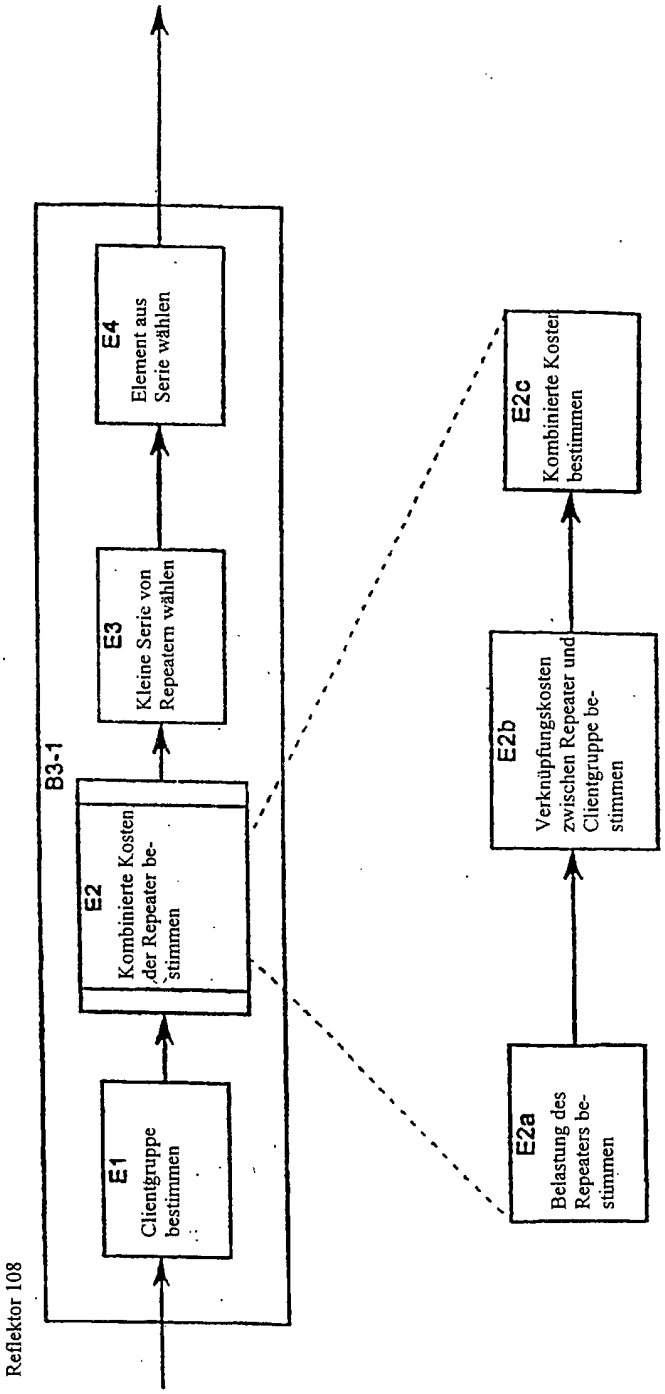
Reflektor 108
B3-1
E1
Clientgruppe bestimmen
E2
Kombinierte Kosten der Repeater bestimmen
E3
Kleine Serie von Repeatern wählen
E4
Element aus Serie wählen
E2a
Belastung des Repeaters bestimmen
E2b
Verknüpfungskosten zwischen Repeater und Clientgruppe bestimmen
E2c
Kombinierte Kosten bestimmen

Fig. 6