(19) österreichisches patentamt

(10) **AT 509 254 A1 2011-07-15**

(12) # Österreichische Patentanmeldung

(21) Anmeldenummer: A 2001/2009
(22) Anmeldetag: 17.12.2009
(43) Veröffentlicht am: 15.07.2011

(51) Int. Cl. : ***H04L 12/18*** (2006.01)
H04L 12/58 (2006.01)
G06Q 30/00 (2006.01)

(73) Patentanmelder:
ISA AUCTIONATA AUKTIONEN AG
A-1150 WIEN (AT)

(72) Erfinder:
ZACKE ALEXANDER
WIEN (AT)
UNTERSALMBERGER GEORG
REKAWINKEL (AT)

(54) **RECHNERSYSTEM ZUM AUSTAUSCH VON NACHRICHTEN**

(57) Rechnersystem (1) zum Austausch von Nachrichten über Internet, für die Online-Abwicklung von Handels-Transaktionen, mit einer Mehrzahl von Client-Rechnern (2) mit Internet-Schnittstellen (3), und mit zumindest einem zentralen Leit-Server (7), der mit einer zentralen Datenbank (8) verbunden ist, woher zwischen den Client-Rechnern (2) und dem zumindest einen zentralen Leit-Server (7) eine Mehrzahl von Proxy-Rechnern (4) angeordnet ist, denen zumindest ein Load-Balancer-Modul (5), das zur Verteilung von Nachrichten auf vorgegebene Proxy-Rechner (4) eingerichtet ist, vorgeschaltet ist, und die je ein Relevanz-Filtermodul (11) aufweisen, das eingerichtet ist, einlangende, von Client-Rechnern (2) kommende Nachrichten auf Relevanz, nach vorgegebenen Kriterien, zu prüfen und nur relevante Nachrichten weiterzuleiten.

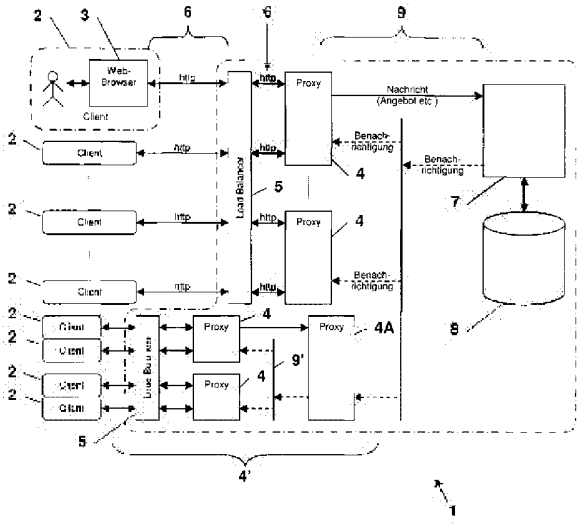


FIG. 1

AT 509 254 A1 2011-07-15

Zusammenfassung:

Rechnersystem (1) zum Austausch von Nachrichten über Internet, für die Online-Abwicklung von Handels-Transaktionen, mit einer Mehrzahl von Client-Rechnern (2) mit Internet-Schnittstellen (3), und mit zumindest einem zentralen Leit-Server (7), der mit einer zentralen Datenbank (8) verbunden ist, woher zwischen den Client-Rechnern (2) und dem zumindest einen zentralen Leit-Server (7) eine Mehrzahl von Proxy-Rechnern (4) angeordnet ist, denen zumindest ein Load-Balancer-Modul (5), das zur Verteilung von Nachrichten auf vorgegebene Proxy-Rechner (4) eingerichtet ist, vorgeschaltet ist, und die je ein Relevanz-Filtermodul (11) aufweisen, das eingerichtet ist, einlangende, von Client-Rechnern (2) kommende Nachrichten auf Relevanz, nach vorgegebenen Kriterien, zu prüfen und nur relevante Nachrichten weiterzuleiten.

(Fig. 1)

Die Erfindung betrifft ein Rechnersystem zum Austausch von Nachrichten über Internet, für die Online-Abwicklung von Handels-Transaktionen, mit einer Mehrzahl von Client-Rechnern mit Internet-Schnittstellen, und mit zumindest einem zentralen Leit-Server, der mit einer Datenbank verbunden ist.

Angestrebt wird im Einzelnen ein Echtzeit-Online-Rechnersystem zum Austausch von Nachrichten für die Abwicklung von Handelsprozessen mit einer technisch unbegrenzten Zahl von Teilnehmern auf Basis der verfügbaren Technologien des World-Wide-Web (www, kurz „Web") und unter Berücksichtigung von dessen Einschränkungen. Insbesondere soll ein System zur Verfügung gestellt werden, das es ermöglicht, im Internet Handels-Abläufe (einschließlich solche des Auktionshandels) und dabei die damit zusammenhängenden Nachrichten zwischen den am Handel beteiligten Teilnehmern in Echtzeit zu übertragen und über eine Web-Oberfläche sichtbar zu machen. Das bedeutet, dass Nachrichten von Benutzern des Systems - sowohl Käufern, Verkäufern als auch z.B. Auktionatoren - umgehend an jene Benutzer des Systems übertragen werden sollen, für die diese Nachricht relevant ist. Derzeit wird unter „umgehend" ein Zeitraum von unter 1 Sekunde verstanden. Das System soll auf Endgeräteseite nur einen modernen Web-Browser benötigen, es soll dazu keine weitere Software auf den Endgeräten der Benutzer installiert werden müssen.

Das System soll für die Abhaltung sehr großer Handelsplätze mit mehr als einer Million gleichzeitig anwesender Benutzer und mehreren (virtuellen) Handelsräumen, zwischen denen die Benutzer beliebig wechseln können, geeignet sein. Das bedeutet, dass einerseits eine sehr große Zahl von Benutzern Nachrichten senden können soll, und dass andererseits diese Nachrichten so rasch wie möglich allen betroffenen Benutzern (z.B. das Annehmen eines Gebots durch den Verkäufer oder Auktionator) sichtbar gemacht werden sollen.

Um das Verständnis der nachfolgenden Erläuterungen zu erleichtern, sollen vorab noch einige Begriffsdefinitionen erfolgen:

Handelsprozess: Ein Prozess zwischen zwei oder mehr Handelspart-

nern, die durch die Abgabe von Angeboten und deren Annahme Handelsabschlüsse tätigen.

Nachricht: Als Nachricht wird hier eine Informationseinheit bezeichnet, die von einem Absender zu einem oder mehreren Empfängern übertragen wird. Unter Nachrichten werden beim vorliegenden System alle Arten von Textnachrichten, aber auch Angebote auf Artikel, Annahmen von Angeboten sowie andere Handels- bzw. Benutzeraktionen verstanden, die jeweils eine Übertragung zu einem Handelspartner erfordern.

Nachrichtenblock: Damit wird eine Gruppe zusammengehörender Nachrichten bezeichnet, z.B. alle Angebote zu einem gehandelten Artikel.

Latenz: Die Latenz ist der Zeitraum zwischen dem Setzen einer Aktion bzw. Senden einer Nachricht durch einen Absender und deren Sichtbarmachung beim Empfänger.

Echtzeit: Unter „Echtzeit" wird hier das Verhalten des Systems verstanden, das eine durchschnittliche Latenz von unter 1 Sekunde aufweist.

Web-Oberfläche: Eine Web-Oberfläche ist die Benutzeroberfläche eines Programms, die ausschließlich mit Mitteln des World-Wide-Web und nur innerhalb eines Web-Browsers sichtbar gemacht sowie auch nur mit Mitteln des World-Wide-Web und jenen eines Web-Browsers bedient werden kann.

Benutzer: Ein über das Internet und über einen Web-Browser mit dem System verbundener Teilnehmer eines Handelsprozesses.

Aus den einleitend angegebenen Anforderungen an das vorliegende Rechnersystem ergeben sich zusammengefasst die folgenden Bedingungen, die das Rechnersystem in technischer Hinsicht erfüllen soll:

Echtzeitverhalten über das Internet ohne Software-Installation, d.h. es soll nur ein (moderner) Web-Browser erforderlich sein;

damit soll eine höchstmögliche Erreichbarkeit und Verbreitung erzielt werden.
Unbegrenzte Skalierbarkeit bei gleichbleibender Latenz, d.h. bei steigender Zahl von Benutzern soll unter der Annahme, dass die Benutzeraktivität je Benutzer konstant bleibt, die Latenz ebenfalls konstant bleiben.

Das unbeschränkte Echtzeitverhalten soll auch bei kleinsten Bandbreiten, z.B. bei Modemverbindungen oder Verbindungen über schmalbandige mobile Endgeräteverbindungen, wie z.B. GPRS, möglich sein (also Bandbreiten in der Größenordnung von 40kbit/s Downlink und 10kbit/s Uplink).

Kompatibilität mit im Internet gebräuchlichen Firewall-, Router- und Proxy-Technologien sowie insbesondere dem immer noch im Internet de facto ausschließlich genutzten IPv4-Protokoll muss gegeben sein.

Im Einzelnen bedeutet das, dass das System Benutzerfunktionen ausschließlich über das World-Wide-Web zur Verfügung stellen soll. Dabei müssen alle Applikationen (Websites) die Verbindung eines Clients (Client-Rechner) zu einem Server (Server-Rechner) über das HTTP-Protokoll (W3C: Hypertext Transfer Protocol -- HTTP/1.1, RFC 2616, http://www.w3.org/Protocols/rfc2616/rfc2616.html) herstellen, damit diese über einen Web-Browser genutzt werden können. Das HTTP-Protokoll basiert auf einem reinen Request/Response-Modell (W3C: Hypertext Transfer Protocol -- HTTP/1.1, RFC 2616, Overall Operation, Abschnitt 1.4, http://www.w3.org/Protocols/rfc2616/rfc2616-sec1.html#sec1.4), bei dem immer ein Client eine Information bei einem Server anfragt; direkte Nachrichtenübertragungen von Client zu Client sind im Web nicht vorgesehen. Wenn ein Client eine Nachricht zu einem anderen Client schicken möchte, muss er diese mittels eines HTTP-Requests zunächst an einen Server senden, der andere Client erhält bzw. die anderen Clients erhalten diese Nachricht dann auf Anfrage beim Server ebenfalls über einen HTTP-Request.

Das HTTP-Protokoll erlaubt weiters keine direkte Benachrichti-

gung der Clients durch einen Server hinsichtlich dessen, dass eine Nachricht auf dem Server bereit liegt. Damit werden gesendete Nachrichten, selbst wenn diese bereits zum Server übertragen wurden, nicht umgehend auf den Clients (für die Benutzer) sichtbar. Ein Client muss immer aktiv nachfragen. Im Regelfall erfolgt das erst durch eine Benutzeraktion und einen dadurch ausgeführten Request beim Server, daher im Regelfall mit einer erheblichen Verzögerung.

Software-Installationen auf dem Client-Computer sind in der Regel nicht möglich, nicht erwünscht bzw. nicht erlaubt oder erfordern zu hohe Computer-Kenntnisse und zu viel Zeit. Das gilt auch für Web-Browser Plug-Ins. Die erforderliche Client-Funktionalität muss daher mit den Funktionen, die ein moderner Web-Browser bereits integriert hat, umzusetzen sein. Diese sind die folgenden:

- Darstellung von HTML bzw. XHTML gemeinsam mit CSS
- DOM (W3C: Document Object Model (DOM), http://www.w3.org/DOM/) und der Zugriff über JavaScript
- Verwendung des XMLHttpRequest-Objektes (W3C: XMLHttpRequest XMLHttpRequest, W3C Working Draft, http://www.w3.org/TR/XMLHttpRequest)

Als Format zur Datenübertragung kommen XML und JSON (JavaScript Object Notation)(Network Working Group: JavaScript Object Notation, RFC 4627, http://www.ietf.org/rfc/rfc4627.txt) zur Anwendung; diese Formate sind über die JavaScript-Funktionen der Web-Browser verfügbar. Diese Kombination von Funktionen ist auch als AJAX (Asynchronous JavaScript and XML) bekannt.

Hinsichtlich Firewalls und Router kann nicht darauf vertraut werden, dass diese andere als reguläre HTTP-Requests passieren lassen. Selbst eine Tunnelung anderer Protokolle über HTTP-Tunneling (HTTP tunnel, http://en.wikipedia.org/wiki/HTTP_tunnel) kann von Übertragungseinrichtungen aus Sicherheitsgründen geblockt werden; damit scheidet ein solches Verfahren aus.

Da die Nachrichten mit dem HTTP-Protokoll nicht aktiv und direkt vom Server als Benachrichtigung auf den Client übertragen werden können, wird vom Client kontinuierlich nachgefragt:

Um unmittelbare Benachrichtigungen von Benutzeraktionen am Client sichtbar zu machen, ruft dieser Client automatisch in sehr kurzen Intervallen (ca. 1 Sekunde) eine Statusinformation vom Server auf, das sogenannte „Polling". Dazu wird die durchgehend in modernen Browsern verfügbare AJAX-Technologie genutzt. Ein solches Polling verursacht aber - ohne weitere Maßnahmen - eine hohe Serverlast, da alle verbundenen Clients nicht erst bei Benutzeraktionen auf der Web-Oberfläche Requests senden, sondern kontinuierlich. Diese hohe Zahl an Requests muss dementsprechend auf eine große Zahl von Servern verteilt werden, da mit steigender Zahl von Benutzern ein einzelner Server irgendwann nicht mehr ausreichen wird. Eine sehr große Zahl von Benutzern erfordert dann eine große Zahl von Servern.

Wenn aber mehr als ein Server Nachrichten von Benutzern empfängt, dann erhalten alle Server erst dann die Nachrichten, die von einem Benutzer an einen einzelnen Server gesendet wurden, wenn gleichzeitig eine Verbindung zwischen den Servern (sog. „Backbone") geschaffen wird, über die alle Nachrichten sofort von dem empfangenden Server an alle anderen Server weitergeleitet werden. Das bedeutet für den Backbone eine quadratisch ansteigende Systembelastung, mit der dieses System eine rasch endende Skalierbarkeit aufweist. Quadratisch steigt die Systembelastung deshalb an, weil unter der Annahme einer konstanten Zahl von Nachrichten je Zeiteinheit für jeden Benutzer, bei Weiterleitung aller Nachrichten an alle Benutzer, mit der Zahl der sendenden Benutzer auch die Zahl der empfangenden Benutzer linear ansteigt.

Wenn sehr viele Server parallel betrieben werden, muss eine zentrale Instanz (z.B. Datenbank) geschaffen werden, die die Server mit den aktuellen, also den neuen bzw. geänderten, Nachrichten versorgt. Wenn alle Nachrichten von allen Benutzern an die zentrale Instanz weitergeleitet werden, steigt deren Belastung ebenfalls quadratisch an.

Üblicherweise werden dazu die Nachrichten bei der zentralen Instanz angefragt und auf den Servern zwischengespeichert (gecacht), damit diese nicht bei jedem Polling eines Clients auf die zentrale Instanz zugreifen müssen. Damit sich dann eine signifikante Reduktion des Backbone-Netzwerkverkehrs erzielen lässt, müsste das Caching-Intervall sehr lange eingestellt werden; damit kann aber keine unmittelbare Übertragung von Nachrichten (Echtzeit) mehr erfolgen.

Der Erfindung liegt somit die technische Aufgabe zugrunde, ein Rechnersystem wie eingangs angegeben derart auszubilden, dass auch bei der angegebenen Vielzahl von Benutzern bei der Anwendung von herkömmlichen Web-Browsern die für die jeweiligen Handels-Transaktionen erforderlichen Nachrichten praktisch in Echtzeit übertragen bzw. sichtbar werden, wobei gegebenenfalls auch kleine Bandbreiten zulässig sein sollen.

Zur Lösung dieser Aufgabe sieht die Erfindung ein Rechnersystem wie in Anspruch 1 definiert vor. Vorteilhafte Ausführungsformen und Weiterbildungen sind in den abhängigen Ansprüchen angegeben.

Beim erfindungsgemäßen Rechnersystem sind somit zwischen den Client-Rechnern (nachstehend auch kurz „Clients" genannt) und dem zumindest einen zentralen Leit-Server Proxy-Rechner (die auch Zwischen- oder Verbindungs-Rechner genannt werden) angeordnet; diesen Proxy-Rechnern (nachstehend auch kurz „Proxy" genannt) sind „Lastverteilermodule", in Fachsprache „Load-Balancer"-Module oder -Rechner, zugeordnet, um die Nachrichten, die von den Clients ankommen, auf die jeweiligen Proxy-Rechner zu verteilen; die Proxy-Rechner enthalten je ein Relevanz-Filtermodul, das die von den Clients einlangenden Nachrichten nach vorgegeben Kriterien auf Relevanz prüft und nur als relevant festgestellte Nachrichten weiterleitet bzw. einer weiteren Verarbeitung zuführt. Inwieweit Nachrichten dabei relevant sind, hängt von den jeweiligen Transaktionen ab und wird dementsprechend festgelegt, wie nachstehend noch näher erläutert werden wird.

Im vorliegenden Rechnersystem ist somit eine zweistufige Opti-

mierung des Nachrichtenflusses bezüglich Benutzeraktionen vom Client zum Server und wieder zurück vorgesehen. Jede Stufe dieser Optimierung macht sich speziell optimierte Datenreduktionsverfahren und Filterverfahren zu eigen, die auf dem typischen, asymmetrischen Nachrichtenfluss innerhalb der vorliegenden Systeme beruhen.

Die Proxy-Rechner können auch eine kaskadierte Konfiguration haben, d.h. es ist von Vorteil, zur weiteren Aufgabenteilung zumindest einen Proxy-Rechner in Kaskade mit vorgeschaltenen Proxy-Rechnern anzuordnen. Dabei hat zweckmäßigerweise auch der in der Kaskade nachgeschaltete Proxy-Rechner ein Relevanz-Filtermodul, um nur relevante Nachrichten, die von den vorgeschalteten Proxy-Rechnern einlangen, weiterzuleiten.

Vorzugsweise hat auch der zentrale Leit-Server oder Leit-Rechner ein Relevanz-Filtermodul, um so nur die relevanten, von Proxy-Rechnern ankommenden Nachrichten durch Filterung zu erfassen und einer weiteren Verarbeitung zuzuführen.

Der zentrale Leit-Server kann weiters eine lokale Datenbank aufweisen bzw. mit einer lokalen Datenbank verbunden sein, um zumindest vorübergehend die als nicht-relevant erkannten Nachrichten zu speichern. In entsprechender Weise ist beim vorliegenden Rechnersystem auch mit Vorteil vorgesehen, dass zumindest einem der Proxy-Rechner eine lokale Datenbank zur zumindest vorübergehenden Speicherung von als nicht-relevant erkannten Nachrichten zugeordnet ist. In weiterer Folge ist es dann vorteilhaft, wenn eine Systemlast-Prüfeinheit vorgesehen ist, die konfiguriert ist, um zu Zeiten verminderter Last im Rechnersystem die Übertragung von in der oder den lokalen Datenbanken gespeicherten nicht-relevanten Nachrichten zur zentralen Datenbank, zur Datenkonsolidierung, zu veranlassen.

Wie dies an sich bekannt ist, können die Clients weiters zur zyklischen Anforderung von für sie bestimmten Nachrichten von den zugehörigen Proxys in vorgegebenen Intervallen eingerichtet sein (sog. Polling). Andererseits ist es zweckmäßig, wenn die Clients eingerichtet sind, um Nachrichten zu den jeweiligen Pro-

xys unmittelbar, außerhalb von den vorgegebenen Polling-Intervallen, also „out of band", zu übertragen. Demgemäß werden neue, eingehende Nachrichten, die auf einem Client generiert werden, immer zunächst an einen Proxy mittels eines Out-Of-Band-Polling-Requests gesendet. Der Proxy entscheidet dann auf Basis des Filterungsergebnisses, ob diese Nachricht sofort an den Leit-Server bzw. den nächsten Proxy in der Kaskade weitergeleitet oder aber zunächst lokal, in der lokalen Datenbank, gespeichert und erst später zur Datenkonsolidierung weitergeleitet werden soll.

Zur Bandbreitenreduktion werden vorteilhaft im vorliegenden Rechnersystem die jeweiligen übertragenen Nachrichten mit einem Zeitstempel versehen, und durch ein derartiges Zeitstempel-basiertes Datenreduktionsverfahren werden immer nur geänderte oder neue Nachrichten vom Leit-Server an die Clients im Rahmen der Polling-Antworten übertragen.

Der Leit-Server speichert alle Nachrichten in der zugeordneten zentralen Datenbank, von wo die Daten auch wieder vom Leit-Server, gegebenenfalls auch von weiteren zur Steigerung der Ausfallsicherheit parallel betriebenen Leit-Servern, ausgelesen werden können.

Der Leit-Server seinerseits entscheidet auf Basis des ins Filtermodul vorgegeben Filter-Algorithmus, welche Nachrichten an alle Proxys übertragen werden. Nur diese Nachrichten werden auch immer sofort in der zentralen Datenbank gespeichert. Die Übertragung an die Proxys erfolgt durch aktive Benachrichtigung der Proxys durch den Leit-Server. Diese aktiven Benachrichtigungen enthalten entweder bereits Informationen über die geänderten bzw. neuen Nachrichten, oder die Proxys fragen nach Erhalt der Benachrichtigung die neuen bzw. geänderten Nachrichten beim Leit-Server ab.

Auch bei Nichtvorhandensein neuer oder geänderter Nachrichten auf dem Leit-Server schickt dieser zweckmäßigerweise kontinuierlich einen „Puls" aus. Aus dem Ausbleiben dieses Pulses können die Proxys auf einen Ausfall des Leit-Servers schließen und an-

schließend neue Nachrichten auf einer parallel betriebenen Server-Instanz anfragen, die bei Einlangen von Anfragen eines Proxys die aktuellen Nachrichten aus der zentralen Datenbank ausliest.

Die Filterungen sind entsprechend den Erfordernissen der jeweiligen Handelsprozesse festgelegt, sodass nur jene für den Handelsprozess notwendigen Nachrichten an den Leit-Server übertragen bzw. die Proxys nur über solche benachrichtigt werden. Damit wird nur eine geringe Zahl an (eingehenden) Nachrichten übertragen, womit das übertragende Netzwerk und die involvierten Komponenten erheblich entlastet werden. Die Filterung ergibt sich beispielsweise dadurch, dass keine Angebote übertragen werden, die bereits auf dem Proxy oder auf dem Leit-Server durch einen anderen Anbieter überboten wurden.

Die Clients fragen durch „Polling" einen Proxy nach dem Vorliegen neuer Nachrichten, z.B. neuer Angebote, ab. Dieses Polling geschieht durch das wiederholte Senden von Polling-Requests vom jeweiligen Client an den Proxy. Die Wiederholungen der Polling-Requests erfolgen in regelmäßigen Zeitabständen, dem sog. Polling-Intervall. Der jeweilige Proxy antwortet gegebenenfalls mit Informationen über neue oder geänderte Nachrichten, die auf dem Client angezeigt werden sollen.

Die Polling-Requests werden zweckmäßig über AJAX ausgeführt, also durch die Verwendung des XML-HTTP-Request-Objektes auf Client-Seite. AJAX wird bevorzugt benutzt, damit nicht bei jeder einzelnen Anfrage ein vollständiger Seitenaufruf des Web-Browsers erforderlich ist. Ein solcher würde zu einer Neupositionierung der Darstellung der Website führen (die Seite scrollt ganz nach oben) und darüber hinaus bewirken, dass für eine gewisse, für den Benutzer merkbare Zeitspanne, auf dem Web-Browser entweder gar keine oder nur eine unvollständige Darstellung der Seite zu sehen ist. Außerdem würden unnötig Ressourcen auf dem Client belegt werden.

Eine Polling-Response kann Informationen über mehr als eine Nachricht aus mehr als einem Nachrichtenblock beinhalten. In der

Response wird über XML bzw. JSON eine Objektstruktur übergeben, mittels derer der Web-Browser erkennen kann, welche Objekte (Nachrichtenblöcke, einzelne Nachrichten) in der Darstellung aktualisiert werden müssen. Der Web-Browser führt diese Aktualisierung dann z.B. mittels JavaScript und DOM aus.

Beispielsweise kann auf der Seite eine Liste von zuletzt eingegangenen Kauf-Angeboten zu einem Artikel angezeigt werden: diese Liste trägt z.B. die ID „Offers" (ID: s. W3C: HTML 4.01 Specification, Element identifiers: the id and class attributes, http://www.w3.org/TR/html401/struct/global.html#h-7.5.2). In der Polling-Response wird danach eine (logische) Datenstruktur übergeben.

Ein Proxy bezieht seinerseits aktuelle bzw. geänderte Nachrichten vom Leit-Server oder von einem weiteren zwischengeschalteten, kaskadierten Proxy. Ein Proxy muss aber seinerseits nicht kontinuierlich beim Leit-Server nachfragen, ob neue oder geänderte Nachrichten vorliegen, er wird aktiv vom Leit-Server darüber benachrichtigt.

Die Erfindung wird nachfolgend anhand von bevorzugten Ausführungsbeispielen, auf die sie jedoch nicht beschränkt sein soll, und unter Bezugnahme auf die beiliegende Zeichnung noch weiter erläutert. In der Zeichnung zeigen im Einzelnen:

Fig. 1 schematisch, in einem Blockschaltbild, ein Rechnersystem gemäß einer Ausführungsform der Erfindung, das für die Online-Abwicklung von Handels-Transaktionen geeignet ist;

Fig. 2 schematisch in einem Blockschaltbild mehr im Einzelnen einen im Rechnersystem gemäß Fig. 1 vorgesehenen Zwischenrechner, hier Proxy-Rechner genannt;

Fig. 3 schematisch in einem Blockschaltbild einen Proxy-Rechner in Verbindung mit einem zentralen Leit-Server, auch Leit-Rechner oder Steuerrechner genannt, wobei auch ein dem Proxy-Rechner vorgeschalteter Client-Rechner veranschaulicht ist;

die Figuren 4, 5 und 6 schematische Ablaufdiagramme zur Veranschaulichung der Vorgänge beim Senden von Polling-Anfragen (Polling-Requests) und Rücksenden von Nachrichten (Fig. 4), weiters das Senden von Polling-Requests und Rücksenden von Antworten unter Vorsehen von Zeitstempeln (Fig. 5) und das Senden von Polling-Requests sowie das Senden von „Out-Of-Band-Requests" (Fig. 6);

die Figuren 5A und 5B HTTP-Polling-Protokolle für den Fall einer Anfrage („Request", s. Fig. 5A) und für eine Antwort („Response", s. Fig. 5B);

Fig. 7 in einem Flussdiagramm den Vorgang eines Polling-Requests;

die Figuren 8 und 9 Flussdiagramme für Filtervorgänge einerseits auf einem Proxy-Rechner (Fig. 8) und andererseits auf dem Leit-Server (Fig. 9);

Fig. 10 in einem Flussdiagramm den Ablauf einer Benachrichtigung eines Proxy-Rechners;

Fig. 11 in einem schematischen Diagramm die Einrichtung bzw. den Vorgang bei der Datenkonsolidierung, wenn von den lokalen Datenbanken Daten zur zentralen Datenbank übermittelt werden;

Fig. 12 ein zur Daten-Konsolidierung gehörendes Ablaufschema (Flussdiagramm);

die Figuren 13A, 13B und 13C in einem Sequenzdiagramm (Fig. 13A) sowie in Flussdiagrammen betreffend Filtervorgänge auf dem Proxy-Rechner (Fig. 13B) und auf dem Leit-Server (Fig. 13C) eine Anwendung des vorliegenden Rechnersystems beim Verkauf von Einzelartikeln;

die Figuren 14A (Sequenzdiagramm), 14B (Filterung auf Proxy) und 14C (Filterung auf Server) ein Beispiel für den Einsatz des vorliegenden Rechnersystems für einen Live-Online-Handel;

die Figuren 15A bis 15C in entsprechenden Sequenz- und Filter-Flussdiagrammen den Vorgang beim Online-Verkauf von Artikelmengen (sog. „Teleshopping-Kanal");

die Figuren 16A, 16B und 16C wiederum in entsprechenden Diagrammen (Überblick bzw. Filterung auf Proxy bzw. Leit-Server) die Vorgangsweise bei einem sog. „englischen Bietverfahren"; und

die Figuren 17A, 17B und 17C in entsprechenden Diagramm-Darstellungen die Vorgangsweise bei einem sog. „holländischen Bietverfahren".

In Fig. 1 ist ein Rechnersystem 1 zum Austauschen von Nachrichten über Internet, für die Online-Abwicklung von Handels-Transaktionen oder Handelsprozessen, veranschaulicht, wobei eine Vielzahl von Client-Rechnern 2, die je mit einem Web-Browser 3 versehen sind, wie beim obersten Client-Rechner 2 in Fig. 1 gezeigt ist, über das Internet mit Zwischen-Rechnern oder Verbindungs-Rechnern, üblicherweise Proxy-Rechner oder kurz Proxy 4 genannt, verbunden sind. Dabei besteht auch die Möglichkeit einer kaskadierten Proxy-Konfiguration, wie in Fig. 1 mit den beiden unteren Proxys 4 und dem in Kaskade nachgeschalteten Proxy-Rechner 4A veranschaulicht ist, um so eine zusätzliche Aufgabenverteilung zu erreichen. Dem Proxy-Rechner 4 vorgeschaltet sind Lastverteiler-Module, üblicherweise Load-Balancer-Modul, Load-Balancer-Rechner oder kurz „Load Balancer" 5 genannt. Zwischen den Client-Rechnern 2 und den Proxy-Rechnern 4 bzw. den zugeordneten Load-Balancer-Modulen 5 werden Nachrichten über das Internet 6 gemäß dem HTTP-Protokoll übermittelt, wie mit den Doppelpfeilen „http" in Fig. 1 angedeutet ist. Dabei wird das Prinzip des sog. Pollings angewendet, wie in den nachstehend erläuterten Fig. 4 bis 7 veranschaulicht ist. Weiters ist im Rechnersystem 1 ein Leit-Server 7 oder Leit-Rechner, auch Controller genannt, vorgesehen, wobei diesem Leit-Server 7 eine zentrale Datenbank 8 zugeordnet ist. Diese zentrale Instanz 7(8) kann auch mehrfach vorgesehen sein, um in einem Störfall, wenn ein Leit-Server 7 ausfällt, die Funktion des gesamten Rechnersystems 1 nichtsdestoweniger sicherzustellen.

Ganz allgemein ist das Rechnersystem 1 dabei so aufgebaut, dass pro Proxy 4 eine Vielzahl von Clients 2, z.B. 10.000 Clients pro Proxy 4, vorgesehen ist bzw. sind. Andererseits sind wiederum zahlreiche Proxy-Rechner 4, z.B. 10.000 Proxy-Rechner 4, dem Leit-Server 7 zugeordnet; demgemäß ergibt sich, dass Client-Rechner 2 und damit Benutzer in Millionenhöhe am System teilnehmen können, um Handelsprozesse, in welcher Form auch immer, wie nachstehend noch näher erläutert werden wird, ablaufen zu lassen.

In Fig. 1 ist weiters noch eine sog. Backbone-Verbindung 9 zwischen den Proxy-Rechnern 4, 4A und dem Leit-Server 7 schematisch veranschaulicht, wobei eine entsprechende Backbone-Verbindung 9' im Bereich der kaskadierten Proxy-Konfiguration 4' vorgesehen ist. Über diese Backbone-Verbindungen 9, 9' werden entsprechende Benachrichtigungen von der jeweiligen höheren Stelle, z.B. vom Leit-Server 7, zu den nächst-unteren Stellen, den Proxy-Rechnern 4 bzw. dem Kaskade-Proxy-Rechner 4A, nach Filterung im jeweiligen Proxy-Rechner 4 bzw. 4A, wie noch nachfolgend erläutert werden wird, weitergeleitet.

Im vorliegenden Rechnersystem 1 ermöglichen die Proxy-Rechner 4 eine wesentliche Entlastung des zentralen Leit-Servers 7; mit anderen Worten: erst durch die hiermit gegebene Aufgabenteilung, mit der damit verbundenen zweistufigen Optimierung des Nachrichtenflusses zwischen Client-Rechnern 2 und Leit-Server 7, kann, mit zusätzlich anderen, noch näher zu erläuternden Funktionen, die gewünschte Abwicklung von Handlungsprozessen in Echtzeit (das heißt innerhalb von Zeiten von maximal 1 Sekunde) bei einer Vielzahl von Benutzern (Clients 2), z.B. in Millionenhöhe, ermöglicht werden.

Eine wesentliche Funktion, die in den Proxy-Rechnern 4 bzw. 4A, aber auch im Leit-Server 7 implementiert ist, ist die bereits angesprochene Filterfunktion, um eingehende Nachrichten auf ihre Relevanz zu überprüfen und nur relevante Nachrichten weiterzuleiten bzw. zu verarbeiten.

In Fig. 2 ist schematisch ein allgemeiner Aufbau eines Proxy-

Rechners 4 bzw. 4A gezeigt, wobei im Bereich einer CPU 10 ein Relevanz-Filtermodul 11 realisiert ist, um die Filterung der eingehenden Nachrichten auf deren Relevanz vorzunehmen. Die im Zuge der Filterung erhaltenen relevanten Nachrichten werden über eine Verbindung 12 an die nächsthöhere Stelle, z.B. zum Leit-Server 7 (Fig. 1) oder zum Kaskade-Proxy-Rechner 4A weitergeleitet; die als nicht relevant ausgefilterten Nachrichten werden in einer lokalen Datenbank 13 des Proxy-Rechners 4 bzw. 4A (selbstverständlich kann es sich dabei auch um eine gesonderte Datenbank handeln, mit der der Proxy-Rechner 4 bzw. 4A verbunden ist) gespeichert; im Zuge einer Daten-Konsolidierung, die nachstehend noch näher anhand der Fig. 11 und 12 erläutert werden soll, werden zu Zeiten einer geringeren Belastung des Rechnersystems 1 Daten über eine Verbindung 14 an die zentrale Stelle bzw. die zentrale Datenbank 8 weitergeleitet.

Der Proxy-Rechner 4 bzw. 4A enthält weiters einen Arbeitsspeicher 15, in dem ein Cache 16 realisiert ist, in dem lokal Nachrichten, die über eine Verbindung 17 beispielsweise von einem Client-Rechner 2 (oder von einem vorgeordneten Proxy-Rechner 4) einlangen, im Zuge eines Updatings, vgl. die Verbindung 18 in Fig. 2, gespeichert werden; die gespeicherten Updates werden über die Verbindung 19 für einen Vergleich im Zuge der Relevanz-Filterung herangezogen, wie nachstehend noch näher erläutert werden wird. In Fig. 2 sind weiters mit strichpunktierten Linien noch für die Verbindung mit einem Client-Rechner 2 sowie dem Leit-Rechner 7 bzw. für den Fall einer Kaskadierung von Proxy-Rechnern (4' bzw. 4, 4A in Fig. 1) einerseits eine Daten-Anfrage von einem Client 2 oder einem untergeordneten Proxy-Rechner 4 bzw. eine Daten-Anfrage bei dem Leit-Server 7 oder einem übergeordneten Proxy-Rechner 4A veranschaulicht.

In Fig. 3 ist schematisch die Anordnung eines Proxy-Rechners 4 bzw. 4A in Verbindung mit dem Leit-Server 7, im Vergleich zu Fig. 1 etwas mehr im Detail, veranschaulicht, wobei einerseits im Fall des Proxy-Rechners 4 bzw. 4A das Relevanz-Filtermodul 11 und der Cache-Speicher 16 jeweils in Verbindung mit einem beispielhaften Client-Rechner 2 für den Erhalt einer neuen Nachricht bzw. eines Polling-Requests sowie das Zurücksenden einer

Antwort (Response) veranschaulicht ist; ähnlich ist auch für den Leit-Server oder zentralen Controller 7 eine lokale Datenbank 13, weiters ein Relevanz-Filtermodul 11 und ein Cache-Speicher 16 vorgesehen. Zusätzlich ist in Fig. 3 die dem Leit-Server 7 zugeordnete zentrale Datenbank 8 veranschaulicht, ebenso wie die Busverbindung (Backbone) 9 für die in Fig. 1 mit strichliert gezeichneten Pfeilen veranschaulichten Benachrichtigungsvorgänge.

Gemäß Fig. 4 fragt der jeweilige Client-Rechner 2 in regelmäßigen Zeitabständen den zugehörigen Proxy-Rechner 4, ob eine neue Nachricht, z.B. ein neues Angebot, vorliegt (vgl. den HTTP-Request gemäß Pfeil 20). Es wird nun angenommen, dass im Fall einer Kauf-Transaktion ein neues Angebot (ein neues Gebot im Fall einer Auktion) am Proxy-Rechner 4 eingelangt ist, wobei diese Nachricht dem Client-Rechner 2 noch nicht mitgeteilt wurde, d.h. dort noch nicht „sichtbar" ist. Demgemäß wird gemäß dem strichlierten Pfeil 21 in Fig. 4 eine entsprechende Benachrichtigung (HTTP-Response) zurück an den Client-Rechner 2 gesandt, sodass der Benutzer dieses Client-Rechners 2 über das neue Angebot oder Gebot informiert wird. Nach einem fest vorgegebenen Zeitintervall, dem Polling-Intervall 22, erfolgt dann automatisch der nächste HTTP-Request (Polling-Request) 20.

Im Prinzip ist dieses Polling-Verfahren hinlänglich bekannt, sodass es keiner weiteren Erläuterung bedarf. Der Proxy-Rechner 4 erhält seine Informationen seinerseits vom zentralen Leit-Server 7 oder von einem kaskadierten Proxy-Rechner 4A (siehe Fig. 1). Zur Bandbreitenreduktion bzw. Datenreduktion werden nun im Bereich dieser Verbindung zwischen Client-Rechnern 2 und Proxy-Rechnern 4 Zeitstempel verwendet, und es werden nur geänderte oder neue Nachrichten vom Proxy-Rechner 4 an den Client-Rechner 2 im Rahmen der Polling-Antworten 21 übertragen. Dies ergibt sich beispielsweise aus dem Ablaufdiagramm von Fig. 5, wo auf die erste Anfrage 20 (Zeitstempel 00:00) hin ein Nachrichtenblock (Response 21) mit einem Zeitstempel von beispielsweise 01:00 vom Proxy-Rechner 4 an den Client-Rechner 2 zurück gesendet wird, wo dieser Nachrichtenblock empfangen und der Zeitstempel 01:00 gespeichert wird. Bei den beiden nächsten Polling-Requests 20' ergibt sich, dass der Nachrichtenblock noch unver-

ändert ist, der Zeitstempel 01:00 bleibt, und als Antwort 21' wird daher jeweils zurückgemeldet, dass es keine Änderung ergibt.

Zum Zeitpunkt 23 gemäß Fig. 5 langt von einem übergeordneten Kaskaden-Proxy-Rechner 4A oder vom Leit-Server 7 eine neue Nachricht im Proxy-Rechner 4 ein, die beispielsweise den neuen Zeitstempel 01:30 erhält. Auf den nächsten Polling-Request 20, noch mit dem Zeitstempel 01:00, wird demgemäß der gesamte neue Nachrichtenblock mit dem Zeitstempel 01:30 gemäß Pfeil 21'' zurückgesendet und beim Client-Rechner 2 mit dem Zeitstempel 01:30 gespeichert.

In den Figuren 4 und 5 ist mit einem breiten vertikalen Pfeil t der Zeitverlauf veranschaulicht.

Aus den vorstehenden Darlegungen ergibt sich, dass immer nur dann, wenn neue Nachrichten am Proxy 4 verfügbar sind, diese an die zugehörigen Clients 2 weitergeleitet werden, wobei für diese Entscheidung, ob aktuelle Nachrichten vorliegen, der jeweilige Zeitstempel von Bedeutung ist. Auf diese Weise wird das Ausmaß der Übertragung von Daten ganz wesentlich reduziert.

Neue, auf einem Client-Rechner 2 generierte Nachrichten werden an den zugehörigen Proxy-Rechner 4 mittels eines „Out-Of-Band-Polling-Requests" gesendet, wie in Fig. 6 mit einem Pfeil 24 veranschaulicht ist. Von diesem Zeitpunkt weg läuft dann das nächste Polling-Intervall 22, und nach Empfang dieser neuen Nachricht, Pfeil 24, am Proxy-Rechner 4 wird diese Nachricht zur übergeordneten Stelle, beispielsweise dem Leit-Server 7 oder dem Kaskade-Proxy 4A, gemäß Pfeil 25 weitergeleitet. Dabei entscheidet jedoch der Proxy 4 über sein Relevanz-Filtermodul 11 (s. Fig. 2), ob diese Nachricht, Pfeil 24, an den Server 7 weitergeleitet oder zunächst lokal (in der lokalen Datenbank 13) gespeichert werden soll, wobei in letzterem Falle erst später die Nachricht an den Server 7 zur Speicherung in der zentralen Datenbank 8 weitergeleitet wird. Die Nachrichten, die in der zentralen Datenbank 8 gespeichert werden, können vom Server 7 oder etwaigen parallel betriebenen Server-Instanzen, die zur Erhöhung

der Ausfallsicherheit vorgesehen sind, wieder ausgelesen werden.

Aus dem Flussdiagramm gemäß Fig. 7 ergibt sich der Ablauf eines regulären Pollings auf einem der Proxy-Rechner 4. Gemäß Feld 26 wird ein Polling-Request (20 in Fig. 4, 5 und 6) gesendet, und gemäß Block 27 langt dieser Polling-Request beim Proxy-Rechner 4 ein. Es folgt nun die Relevanz-Filterung, vgl. Filtermodul 11 in Fig. 3, mit Rückfrage beim Cache-Speicher 16, gemäß Block 28; gemäß Feld 29 wird aus dem Cache-Speicher 16 eine Antwort (Response) an den Client-Rechner 2 zurückgesendet, wie dies sich auch aus der Darstellung in Fig. 3 ergibt, und wo die neue Nachricht bzw. die Antwort entsprechend mit den Bezugszahlen 26, 29 (in Klammern) angedeutet sind.

Bevor nun mehr im Detail auf das Filterverfahren eingegangen wird, soll nochmals zur Erläuterung auf das im vorliegenden Rechnersystem 1 benützte Datenreduktionsverfahren auf Basis von Zeitstempeln, die allen Nachrichten zugeordnet werden, eingegangen werden. Diese Zeitstempel bezeichnen den letzten Änderungszeitpunkt der jeweiligen Nachricht.

In den Figuren 5A und 5B sind an sich herkömmliche HTTP-Polling-Protokolle veranschaulicht, wobei ersichtlich ist, dass nach einleitenden Protokolldaten bzw. Kopfteilen jeweils ein Zeitstempel 30 bzw. 30' vor den eigentlichen Nachrichten 31 bzw. 31' vorgesehen wird. Der Nachrichtenteil 32 der Polling-Response 21 gemäß Fig. 5B (der sog. „Response Body" 32) bleibt vollständig leer, wenn keine Änderungen in den Nachrichten gegeben sind. Im Nachrichtenblock 31' sind gemäß Fig. 5B verschiedene Nachrichten-Daten 33, wie Status 33A, Beschreibung 33B, Angebot 33C, Höchstgebot 33D und gegebenfalls andere Daten 33E enthalten.

Das Client-Proxy-Abfrage-Protokoll 20 sieht bei einer Abfrage die Übertragung eines einzigen Zeitstempels 30 vor, der dem letzten dem Client 2 mitgeteilten Änderungszeitpunkt zu den übertragenen Nachrichten entspricht. Der Proxy 4 speichert eine Kopie aller Nachrichten und Nachrichtenblöcke 31', die von den mit ihm verbundenen Clients 2 abgefragt werden, im Cache 16. Ein Nachrichtenblock ist z.B. die Liste der eingegangenen Angebote.

Der Cache 16 befindet sich ausschließlich im Arbeitsspeicher 15 des Proxys 4, s. Fig. 2.

Die gecachten Nachrichtenblöcke können vom Proxy 4 für die Anfragen mehrerer Clients 2 gemeinsam genutzt werden, wenn diese Clients 2 die jeweils gleiche Information angezeigt bekommen, was im Rahmen von Handelsprozessen meist der Fall ist: alle Teilnehmer am Handelsprozess sehen beispielsweise die gleiche Liste der höchsten Angebote. Damit kann wesentlich an Arbeitsspeicher 15, der auf einem Proxy 4 durch den Cache 16 belegt wird, gespart werden.

Weiters merkt sich der Proxy 4 in seinem Chache 16 für jeden Nachrichtenblock 31 bzw. 31' und für jede einzelne Nachricht einen Zeitstempel 30 bzw. 30' der jeweils letzten Änderung. Diese Struktur der Zeitstempel kann noch weiter verschachtelt werden, solange die Belastung durch das Vergleichen der Zeitstempel niedriger als durch die Übertragung eines ganzen Nachrichtenblocks ist.

Bei eingehenden Abfragen eines Clients 2 (Polling) wird der Zeitstempel 30 der eingehenden Abfrage mit den Zeitstempeln 30' der im Cache 16 des Proxys 4 gespeicherten Nachrichtenblöcke 31' verglichen, und wenn dort eine Abweichung vorliegt, schließlich die Zeitstempel der einzelnen Nachrichten 33. An den Client 2 werden nur jene Nachrichten aus jenen Nachrichtenblöcken übertragen, die auf dem Proxy 4 einen neueren Zeitstempel 30' aufweisen als jener Zeitstempel 30, der vom Client 2 mitgesendet wurde. Nachrichtenblöcke mit älteren oder gleich alten Zeitstempeln werden gar nicht, aus solchen mit jüngeren Zeitstempeln werden nur jene Nachrichten übertragen, die ihrerseits jüngere Zeitstempel besitzen. Im Idealfall wird eine leere Antwort zurückgeschickt, wenn alle Zeitstempel 30' aller Nachrichtenblöcke 33 im Cache 16 nicht jünger als der vom Client 2 mitgesendete Zeitstempel 30 sind.

Bei Ausfall eines Proxys 4 kann es notwendig sein, den für die Bandbreitenoptimierung notwendigen Inhalt des Cache 16 auf einem anderen Proxy 4 wiederherzustellen. Da generell nur Nachrichten,

die auch zum Server 7 übertragen wurden, auf den Clients 2 angezeigt werden, kann diese Wiederherstellung durch Abfrage der aktuellen Nachrichten beim Server 7 erfolgen. Das gleiche Prinzip ist auch anzuwenden, wenn der Inhalt des Cache 16 aus anderen Gründen auf dem angefragten Proxy 4 nicht verfügbar sein sollte.

Die im Cache 16 befindlichen Nachrichtenblöcke können aus dem Cache 16 gelöscht werden, sobald kein Client 2 diese Nachrichtenblöcke mehr abfragt. Im Prinzip kann dies bereits nach dem Verstreichen weniger Polling-Intervalle, in denen der betreffende Nachrichtenblock nicht mehr angefragt wurde, durchgeführt werden, da davon auszugehen ist, dass in jedem Polling-Intervall zumindest einer der verbundenen Clients 2 diesen Nachrichtenblock angefragt hätte.

Neue Nachrichten, die von einem Client 2 zum Server 7 übertragen werden sollen, werden zunächst zu einem Proxy 4 übertragen, der dann die Weiterleitung zum Server 7 (evtl. über einen oder mehrere kaskadierte Proxy 4A) übernimmt. Der Server 7 speichert die Nachrichten, sofern erforderlich, in der zentralen Datenbank 8. Diese Nachrichten stehen so parallel betriebenen Controller-Instanzen bei Ausfall der ersten Controller-Instanz, d.h. dem Leit-Server, 7 sofort zur Verfügung.

Die Nachrichten werden bei der Übertragung vom Client 2 zum Proxy 4 in einem Polling-Request 20 eingebettet, sodass unmittelbare Ergebnisse der übertragenen Nachricht bereits in der Response 21 zu diesem Request an den Client 2 übertragen werden können.

Zur Verkürzung der Zeitspanne, die vom Erstellen der Nachricht auf dem Client 2 bis zum Empfang auf dem Proxy 4 vergeht, werden Nachrichten, die auf einem Client 2 erstellt wurden, unmittelbar an den Proxy 4 übertragen, ohne auf den Ablauf des Polling-Intervalls zu warten. Ein solcher Request wird Out-Of-Band-Request (vgl. 24 in Fig. 6) genannt. Er unterscheidet sich von gewöhnlichen Polling-Requests 20 nur dadurch, dass zuvor nicht auf den Ablauf des normalen Polling-Intervalls 22 gewartet wird und er im Regelfall eine neue Nachricht zur Übertragung vom Client 2 zum Server 7 beinhaltet.

Wenn sich zum Zeitpunkt der Initiierung eines Out-Of-Band-Requests 24 ein Polling-Request auf dem Weg zwischen Client 2 und Proxy 4 befindet, wird dieser Request vom Client 2 mittels XMLHttpRequest.Abort() (W3C: XMLHttpRequest, W3C Working Draft 20 August 2009, 4.6.5 The abort() method; http://www.w3.org/TR/XMLHttpRequest/#the-abort-method) abgebrochen; der neue Request wird mit der neuen Nachricht und dem gleichen Zeitstempel wie zuvor gesendet.

Eingehende Nachrichten werden in den Filtermodulen 11 durch Filter-Algorithmen bewertet, die entsprechend den Erfordernissen des jeweiligen Handelsprozesses kalibriert sind, sodass nur die für den Handelsprozess relevanten Nachrichten unmittelbar übertragen werden.

Durch diese Relevanz-Filterung wird die Zahl der zu übertragenden Nachrichten erheblich verringert. Insbesondere steigt die Zahl der zu übertragenden Nachrichten nicht mit der Zahl der Teilnehmer an den Handelsprozessen, sondern nur noch mit der Zahl der Handelsprozesse. Dieses gilt, wenn man davon ausgehen kann, dass jeder Handelsprozess nur einer bestimmten, maximalen Zahl von Nachrichten bedarf, die unabhängig von der Zahl der beteiligten Teilnehmer ist. Im einfachsten Fall genügt bei einem Festpreis beispielsweise der erste Kaufauftrag, alle weiteren Aufträge sind bereits irrelevant.

Nachrichten, die bei einem Proxy 4 oder einem Server 7 eingehen, werden durch die Filterung in folgende zwei Kategorien unterteilt:

- Nachrichten, die für andere Benutzer (die über andere Proxys 4 mit dem System 1 verbunden sind) unmittelbar relevant sind; bzw.
- Nachrichten, die für andere Benutzer nicht unmittelbar relevant sind

Die Relevanz der Nachrichten wird hinsichtlich ihrer Bedeutung für den Handelsprozess beurteilt, d.h. dass Nachrichten, die keine Auswirkung auf eine Entscheidung eines Handelspartners ha-

ben, als nicht unmittelbar relevant eingestuft werden. Das sind zum Beispiel einlangende Angebote mit einem niedrigeren Preis als bereits zuvor eingelangte Angebote.

Unmittelbar relevante Nachrichten werden sofort von einem Proxy 4 an den Server 7 (bzw. an einen dazwischen liegenden kaskadierten Proxy 4A) bzw. vom Server 7 an die zentrale Datenbank 8 übertragen.

Nicht unmittelbar relevante Nachrichten werden nicht weiter geleitet, sondern zunächst in einer jeweiligen lokalen Datenbank 13 zwischengespeichert. In Zeiträumen verminderter Last des Rechnersystems 1 werden diese Daten an die zentrale Datenbank 8 übertragen (Offload-Daten-Konsolidierung) und stehen damit für spätere Anfragen ebenfalls in der zentralen Datenbank 8 zur Verfügung. Damit wird vor allem auch die Last auf der zentralen Datenbank 8 vermindert.

Über die Aussendung von Benachrichtigungen werden die Proxys 4 über das Vorliegen neuer oder geänderter Nachrichten auf dem Server 7 informiert. Die Proxys 4 müssen also nicht regelmäßig nachfragen, ob neue oder geänderte Nachrichten vorliegen, sondern werden aktiv vom Server 7 darüber informiert.

Dabei werden wiederum nur Benachrichtigungen im Fall von unmittelbar relevanten Nachrichten ausgesendet, also jene, die auf Basis der Filterung als unmittelbar relevant eingestuft werden. Damit erfahren die Proxys 4 vom Vorliegen neuer oder geänderter Nachrichten und können diese vom Server 7 (oder von einem kaskadierten Proxy 4A) abholen. Den Clients 2 werden diese Nachrichten anschließend über das Polling übertragen.

Die Benachrichtigungen werden z.B. über UDP (J.Postel, User Datagram Protocol, RFC 768, http://www.ietf.org/rfc/rfc768) versendet. Wenn die Nachrichten für alle Proxys 4 relevant sind, dann vorzugsweise über IP-Multicast (Network Working Group, Internet Group Management Protocol, Version 3, RFC 3376, http://www.ietf.org/rfc/rfc3376) bzw. innerhalb eines Netzwerk-Segments über einen IP-Broadcast (Network Working Group, Broad-

casting Internet Datagrams in the Presence of Subnets, RFC 922, http://www.ietf.org/rfc/rfc922.txt). Damit wird der Overhead durch die Benachrichtigungen so weit wie möglich minimiert.

Die Benachrichtigungen können ihrerseits neben der Information, dass eine neue oder geänderte Nachricht vorliegt, auch schon eine einfache Nachricht selbst übertragen. Komplexe Nachrichten oder ganze Nachrichtenblöcke werden aber von den Proxys 4 beim Server 7 angefragt.

Jeder Server (bzw. Controller) 7 kann seinerseits ausfallen, sei es wegen eines Software- oder eines Hardware-Fehlers oder aus Gründen, die in der Umgebung liegen. Neben konventionellen Redundanz-Maßnahmen zur Absicherung der Verfügbarkeit eines Controllers (Fail-Over Cluster, Spiegelung, etc.) kann innerhalb des Rechnersystems 1 das folgende Setup zur Erzielung redundanter Controller-Instanzen genutzt werden:

Da von einem Proxy 4 nicht erkannt werden kann, ob deshalb keine Benachrichtigung eingelangt ist, weil keine neuen Nachrichten vorliegen, oder weil der Controller 7 ausgefallen ist, wird von jedem Controller 7 ein „Puls" in Form von UDP-Paketen ausgesendet. Damit wird außerdem vermieden, dass alle Proxys 4 kontinuierlich beim Controller 7 anfragen und damit den Netzwerkverkehr ansteigen lassen. Das Intervall richtet sich nach der Zeitspanne, wie rasch das Rechnersystem 1 über einen Ausfall Bescheid wissen sollte, damit entsprechende alternative Ressourcen (andere Server) aktiviert werden können.

Sobald ein Proxy 4 keine Benachrichtigungen von einem Server 7 mehr erhält, muss der Proxy 4 den letzten Stand der vorhandenen Nachrichtenblöcke bei einem alternativen Server abfragen, damit auch in der Zwischenzeit eingelangte und verarbeitete Nachrichten an diesen Proxy 4 weitergeleitet werden. Dieser alternative Server wird dann für den betroffenen Handelsprozess zur zentralen Controller-Instanz und lädt bei der ersten Abfrage die erforderlichen Daten aus der zentralen Datenbank 8.

Ein Proxy 4 kann ebenfalls wegen eines Software- oder eines

Hardware-Fehlers oder aus Umgebungsgründen ausfallen. Da das System 1 zur Bandbreitenoptimierung die Polling-Requests eines Clients 2 zunächst bevorzugt immer an den gleichen Proxy 4 senden würde, würde bei Ausfall des betroffenen Proxys 4 die Verbindung der über diesen Proxy 4 zugreifenden Clients 2 unterbrochen werden.

Wie erwähnt werden jedoch vor die Proxys 4 die Load-Balancer-Module bzw. -Rechner 5 geschaltet, die die Anfragen vieler Clients 2 gleichmäßig auf alle Proxys 4 verteilen. Bei Ausfall eines Proxys 4 werden dadurch die folgenden Polling-Requests an einen anderen Proxy 4 weitergeleitet. Da dieser Proxy 4 eventuell den angefragten Nachrichtenblock noch nicht im Arbeitsspeicher 15 bereit hält (also noch nicht im Cache 16 hat), fragt dieser Proxy 4 die entsprechenden Informationen beim Server 7 an und befüllt damit seinen Cache 16. Wenn die Informationen zur letzten Änderung von Nachrichten und Nachrichtenblöcken auch in der zentralen Datenbank 8 gespeichert werden und damit über den Server 7 verfügbar sind, kann sogar bei dieser Wiederherstellung des Cache-Inhaltes auf einem Proxy 4 bereits in der ersten Response die genau richtige Differenzinformation an den Client 2 übertragen werden.

Soweit die meisten Nachrichtenblöcke von vielen bzw. von allen Clients 2 genutzt werden, können ohne signifikanten Bandbreiten- oder Performanceverlust die Polling-Requests der Clients 2 sogar beliebig auf alle Proxys 4 verteilt werden.

Im Betrieb empfängt ein jeweiliger Proxy 4 bzw. der Leit-Server 7 von einer untergeordneten Instanz eingehende Nachrichten, also z.B. der Proxy 4 von einem Client 2 bzw. der Controller 7 von einem Proxy 4. Diese Nachrichten werden durch den Relevanz-Filter hinsichtlich der Relevanz-Kriterien bewertet; das erfolgt durch Vergleich mit Schwellwerten, die aus dem Cache 16 ausgelesen werden. Diese Schwellwerte sind ihrerseits Nachrichten, die im Cache 16 gespeichert wurden, z.B. bereits eingelangte Gebote auf den gleichen Artikel.

Nicht (unmittelbar) relevante Nachrichten werden in der lokalen

Datenbank 13 („Offload") zwischengespeichert und später in die zentrale Datenbank 8 konsolidiert.

(Unmittelbar) relevante Nachrichten werden zur nächsthöheren Instanz weitergeleitet, das ist für einen Proxy 4 der Controller 7 oder ein kaskadierter Proxy 4A, für den Controller 7 ist das die zentrale Datenbank 8.

Außerdem wird der lokale Zwischenspeicher (Cache 16) unmittelbar von einer eingegangenen, relevanten Nachricht informiert. Damit erhalten untergeordnete Instanzen, z.B. der Client 2, sofort Feedback über die Weiterleitung oder Ausfilterung einer Nachricht. Außerdem basiert jeder folgende Schwellwertvergleich noch vor der (möglichen) Aktualisierung durch die übergeordnete Instanz (4 oder 4A oder 7 oder 8) bereits auf diesem lokal aktualisierten Schwellwert.

Der Server 7 setzt darüber hinaus eine Benachrichtigung über das Einlangen einer relevanten Nachricht über den Benachrichtigungs-Bus 9 ab, über den alle abhängigen Proxys 4 bzw. 4A vom Vorliegen einer neuen, relevanten Nachricht informiert werden. Die Proxys 4, 4A veranlassen damit ihren Cache 16, bei der nächsten Anfrage diese neue Nachricht von der übergeordneten Instanz (4A/7) abzuholen.

Der Proxy 4 sendet keine Benachrichtigungen aus, der Server 7 muss keine empfangen.
In Fig. 8 ist die Filterung auf einem Proxy-Rechner 4 in einem Flussdiagramm mehr im Detail veranschaulicht. Dabei wird gemäß Feld 35 eine neue Nachricht von einem Client-Rechner 2 oder einem vorgeordneten Proxy-Rechner 4 (wenn es sich um einen Kaskaden-Proxy 4A handelt) empfangen, und es wird gemäß Prüffeld 36 sodann geprüft, ob die Nachricht relevant ist, d.h. ob der Schwellwert wie beschrieben überschritten wird. Wenn ja, wird der Status und die Nachricht im zugehörigen Cache 16 aktualisiert, s. Block 37, und die Nachricht wird an den Leit-Server 7 weitergeleitet und verarbeitet, vgl. Block 38 in Fig. 8. Gemäß Block 39 wird der Cache 16 mit der zuletzt vom Server 7 erhaltenen Nachricht aktualisiert; danach wird eine entsprechende Ant-

wort aus dem Cache 16 an den jeweiligen Client 2 zurückgesendet, s. Feld 40.

Wenn im Feld 36 die Nachricht als nicht relevant ausgefiltert wird, wird die Nachricht gemäß Block 41 in der örtlichen Datenbank 13 des Proxys 4 gespeichert; gemäß Block 42 wird weiters der Status bzw. die letzte Nachricht aus dem Cache 16 ausgelesen und gemäß Feld 40 als Antwort an den Client 2 gesendet.

Beim in Fig. 9 dargestellten, am Leit-Server 7 stattfindenden Filtervorgang wird in entsprechender Weise gemäß Feld 45 eine von einem Proxy-Rechner 4 bzw. 4A kommende neue Nachricht empfangen, und gemäß Prüffeld 46 wird diese Nachricht auf Relevanz überprüft. Wenn die Nachricht relevant ist, wird sie gemäß Block 47 in der zentralen Datenbank 8 gespeichert, und der Cache-Speicher 16 des Leit-Servers 7 wird mit dieser Nachricht gemäß Block 48 aktualisiert; sodann werden gemäß Block 49 alle Proxys 4 bzw. 4A benachrichtigt, was gemäß Feld 50 aus dem Cache-Speicher 16 des Servers 7 an die Proxys 4 erfolgt (Response zurückgeben).

Wenn die Nachricht nicht relevant ist, s. Prüffeld 46, so wird die Nachricht vorübergehend in der lokalen Datenbank 13 des Servers 7 gespeichert, s. Block 51 in Fig. 9, und es wird der Status bzw. die letzte Nachricht auf dem Cache-Speicher 16 des Servers 7 ausgelesen, Block 52, und gemäß Feld 50 in der Antwort an den Proxy 4 bzw. 4A zurückgegeben.

In Fig. 10 ist in einem Flussdiagramm der Vorgang bei der Benachrichtigung eines Proxy-Rechners 4 bzw. 4A veranschaulicht. Gemäß Feld 55 erfolgt eine Benachrichtigung durch den Leit-Server 7. In einem Prüffeld 56 wird im jeweiligen Proxy 4 bzw. 4A geprüft, ob die Nachricht bereits vorhanden ist; wenn ja, ist der Cache-Speicher 16 gemäß Feld 57 aktuell. Wenn die Nachricht jedoch noch nicht vorhanden ist, wird gemäß Block 58 der Cache-Speicher 16 aktualisiert. Sofern untergeordnete Proxys 4 vorhanden sind, werden sodann diese - gemäß dem strichliert gezeichneten Block 49 - benachrichtigt. Beim nächsten Polling wird der aktuelle Status bzw. die letzte Nachricht bereits zurückgegeben.

Auch dann wird das End-Feld 57 erreicht, gemäß dem der Cache-Speicher 16 als aktualisiert festgestellt wird.

Die Konsolidierung der Offload-Datenbanken 13 erfolgt zu einem Zeitpunkt, zu dem sowohl die betreffende lokale Datenbank 13 als auch die zentrale Datenbank 8 signifikant unterhalb von Vollauslastung betrieben werden. Dazu wird in regelmäßigen Abständen die Auslastung der lokalen und der zentralen Datenbank 13 bzw. 8 mittels einer Systemlast-Prüfeinheit 60 (s. Fig. 11) abgefragt; sinkt diese Last unter einen bestimmten Schwellwert, wird mit der Übertragung der Daten begonnen, s. Übertragungskanal 61 in Fig. 11; auch während der Übertragung der Daten wird die Auslastung weiter gemessen und die Übertragung bei Überschreitung eines Schwellwertes unterbrochen.

Für die Konsolidierung werden die in der lokalen Datenbank 13 gespeicherten Nachrichten, also jene, die noch nicht zum Zeitpunkt des Einlangens dieser Nachrichten an die zentrale Datenbank 8 weitergeleitet wurden, eine nach der anderen in die zentrale Datenbank 8 übertragen und nach erfolgreicher Übertragung aus der lokalen Datenbank 13 gelöscht; dies ist im Ablaufdiagramm gemäß Fig. 12 im Einzelnen gezeigt.

Gemäß Fig. 12 folgt nach einem Startschritt 65 für die Konsolidierung eine Abfrage zwecks Prüfung der Belastung der lokalen (Offload-)Datenbanken 13 und der zentralen Datenbank 8 gemäß Block 66. In Prüffeld 67 wird daraufhin überprüft, ob die abgefragten Lastzustände unterhalb eines vorgegebenen Schwellwerts liegen, was mit der Prüfeinheit 60 (die durch einen Konsolidierungsprozess gebildet sein kann) geschieht. Wenn dies zutrifft, d.h. die Belastung des Systems 1 ausreichend gering ist, wird gemäß Block 68 die nächste Nachricht aus einem lokalen Datenspeicher 13 geholt und gemäß Block 69 in die zentrale Datenbank 8 kopiert. In einem Prüffeld 70 wird sodann abgefragt, ob der Kopiervorgang erfolgreich war, und wenn ja, wird die Nachricht in der betreffenden lokalen Datenbank 13 gelöscht, s. Block 71; danach wird gemäß Feld 72 abgefragt, ob weitere Nachrichten vorliegen, und wenn ja, wird zum Block 66 im Flussdiagramm gemäß

Fig. 12 zurückgekehrt. Wenn keine weiteren Nachrichten in der betreffenden lokalen Datenbank 13 vorliegen, wird zum Feld 73 übergegangen und der Konsolidierungsvorgang beendet.

Dies geschieht auch, wenn bei der Abfrage gemäß Feld 67 das Ergebnis ist, dass die Lastzustände oberhalb des Schwellwerts liegen; auch dann wird der Konsolidierungsvorgang zumindest vorläufig beendet, s. Feld 73.

Wenn gemäß Abfragefeld 70 das Kopieren der Nachricht in die zentrale Datenbank 8 nicht erfolgreich war, wird gemäß Block 74 der Fehler aufgezeichnet, und es wird vermerkt, die Konsolidierung zu einem späteren Zeitpunkt noch einmal zu starten. Danach wird auch hier das Konsolidierungs-Ende 73 erreicht.

Die Konsolidierung kann wie somit ersichtlich jederzeit bei Anstieg der Auslastung der lokalen Datenbank 13 oder der zentralen Datenbank 8 durch laufende Handelsprozesse unterbrochen und später fortgesetzt werden.

Zur Vermeidung einer Duplizierung von Nachrichten werden alle Nachrichten durch eine UUID (Network Working Group: A Universally Unique IDentifier (UUID) URN Namespace, RFC 4122, http://www.ietf.org/rfc/rfc4122.txt) gekennzeichnet.

Abschließend soll nun die Betriebsweise des vorliegenden Rechnersystems 1 anhand von verschiedenen beispielhaften Anwendungen unter Bezugnahme auf die Figuren 13 bis 17 auch zusätzlich erläutert werden.

Die Figuren 13A bis 13C beziehen sich auf den Vorgang eines Sofortverkaufs von Einzelartikeln.

Einzelne Artikel können zum angegebenen Preis sofort gekauft werden; an den ersten Kaufauftrag wird sofort verkauft, alle anderen interessierten Käufer können den Verkauf live mitverfolgen. Sowohl Käufer als auch Verkäufer können den Verkaufsstatus eines Artikels in Echtzeit verfolgen.

Nur der erste Kaufauftrag zu einem Artikel, der auf einem Proxy 4 einlangt, wird weitergeleitet, alle anderen werden aufgrund der Relevanz-Filterung sofort zurückgewiesen und nur in der lokalen Datenbank 13 des jeweiligen Proxys 4 gespeichert. Mit dem ersten auf einem Proxy 4 einlangenden Kaufauftrag gilt der Artikel auf jeden Fall als verkauft, vom Leit-Server 7 muss noch angefragt werden, an welchen Käufer der Artikel verkauft wurde: an den, dessen Kaufauftrag auf diesem oder auf einem anderen Proxy 4 zuerst eingelangt ist.

Der erste auf dem Server 7 einlangende Kaufauftrag bewirkt den Verkauf des Artikels und wird in der zentralen Datenbank 8 gespeichert, der Server 7 benachrichtigt umgehend alle Proxys 4 vom abgeschlossenen Verkauf. Alle anderen Kaufaufträge werden sofort zurückgewiesen und nur in den lokalen Datenbanken 13 gespeichert.

Es werden pro Artikel maximal so viele Kaufaufträge an den Server 7 gesendet wie Proxys 4 mit diesem Server 7 verbunden sind. Beispielsweise ist gemäß Fig. 13A Client A mit Proxy Nr. 1 verbunden; Client B und C sind mit Proxy Nr. 2 verbunden. Client B und C versuchen kurz hintereinander, einen Artikel durch Senden eines Kaufauftrags zu kaufen; Client A beobachtet nur den Vorgang. Der erste, der das Angebot gesendet hat, erhält den Artikel, in diesem Fall Client B. Der Kaufauftrag von Client C wird sofort zurückgewiesen und nicht zum Server 7 weitergeleitet, da auf dem gleichen Proxy 4 durch Client B bereits ein Kaufauftrag eingelangt war.

Dieser Ablauf ergibt sich unmittelbar aus der Darstellung im Sequenzdiagramm gemäß Fig. 13A.

In Fig. 13B ist der zugehörige Filtervorgang auf einem jeweiligen Proxy-Rechner 4 in einem Flussdiagramm veranschaulicht. Mit einem Feld 75 ist dabei ein einlangender Kaufauftrag veranschaulicht, und es wird sodann gemäß Prüffeld 76 überprüft, ob der Artikel bereits - an einen beliebigen anderen Benutzer - verkauft wurde. Wenn nein, wird gemäß Block 77 der zugehörige Cache-Speicher 16 des betreffenden Proxy-Rechners 4 für weitere

Requests auf „verkauft" gesetzt, und gemäß Block 78 wird eine entsprechende Nachricht an den Leit-Server 7 weitergeleitet und dort verarbeitet; gemäß Block 79 wird weiters der Cache-Speicher 16 mit der Response vom Leit-Server 7 aktualisiert, und gemäß Feld 80 wird dann die entsprechende Antwort aus dem Cache 16 an den jeweiligen Client 2, hier z.B. an den Client B, zurückgegeben.

Wenn sich bei der Abfrage gemäß Feld 76 ergibt, dass der Artikel bereits verkauft wurde, was im Beispiel gemäß Fig. 13A für den Client C zutrifft, so wird gemäß Block 81 der Kaufantrag unmittelbar (von Proxy Nr. 2) zurückgewiesen und in der lokalen Datenbank 13 dieses Proxy-Rechners 4 (also im gezeigten Beispiel gemäß Fig. 13A Proxy Nr. 2) gespeichert. Gemäß Block 82 wird sodann der Verkaufsstatus aus dem Cache-Speicher 16 ausgelesen und in der Antwort an den Client 2 (hier Client C) zurückgegeben (Response „verkauft"), s. Feld 80 in Fig. 13B.

Beim im Leit-Server 7 erfolgenden Filtervorgang, gemäß dem Flussdiagramm nach Fig. 13C, langt von einem der Proxy-Rechner 4, hier vom Proxy Nr. 2, gemäß Feld 85 der Kaufauftrag ein, wonach gemäß Prüffeld 86 überprüft wird, ob der Artikel bereits verkauft wurde. Wenn nein, wird gemäß Block 87 der Kaufauftrag in der zentralen Datenbank 8 gespeichert; gemäß Block 88 wird der Cache-Speicher 16 des Leit-Servers 7 aktualisiert („verkauft an Benutzer - Client-B"); danach werden alle Proxys 4 entsprechend benachrichtigt, s. Block 89, und die entsprechende Antwort wird aus dem Cache-Speicher 16 an den jeweiligen Proxy (hier Proxy Nr. 1) zurückgegeben, s. Feld 90 in Fig. 13C.

Wenn sich hingegen bei der Abfrage gemäß Feld 86 ergibt, dass der Artikel bereits verkauft ist, wird der gerade eingelangte Kaufauftrag gemäß Block 91 zurückgewiesen und in der lokalen Datenbank 13 des Leit-Servers 7 gespeichert. Gemäß Block 92 wird weiters der Verkaufsstatus aus dem Cache-Speicher 16 ausgelesen und gemäß Feld 90 in der Response aus dem Cache 16 an den entsprechenden Proxy 4 zurückgegeben.

Ein weiteres Anwendungsbeispiel ist der sog. Live-Online-Handel,

bei dem auf Artikel Angebote platziert werden können; der Verkäufer kann nach seinem Belieben entscheiden, wann er das - beste - Angebot annimmt. Im Einzelnen wird dabei nur ein jeweils höheres Angebot (d.h. höher als alle vorher eingelangten Angebote) zu einem Artikel, das auf einem Proxy 4 einlangt, an den Leit-Server 7 weitergeleitet. Alle anderen Angebote werden sofort zurückgewiesen und in der lokalen Datenbank 13 des jeweiligen Proxys 4 gespeichert.

Auch vom Leit-Server 7 wird immer nur ein jeweils besseres Angebot unmittelbar in der zentralen Datenbank 8 gespeichert und werden umgehend alle Proxys 4 von diesem eingelangten Angebot benachrichtigt. Alle anderen Angebote werden sofort zurückgewiesen und nur in der lokalen Datenbank 13 gespeichert.

Der Verkäufer kann das höchste Angebot annehmen; diese Angebotsannahme langt zunächst bei einem Proxy 4 ein, dieser leitet diese sofort an den Server 7 weiter.

Nach Einlangen der Angebotsannahme auf dem Server 7 wird diese unmittelbar in der zentralen Datenbank 8 gespeichert, und alle Proxys 4 werden vom erfolgten Verkauf benachrichtigt.

Beispielsweise sind - gemäß Fig. 14A - Client A und B (beide sind Käufer) mit Proxy Nr. 1 verbunden, Client C (Käufer) und Client D (Verkäufer) sind mit Proxy Nr. 2 verbunden. Client A und B senden kurz hintereinander Angebote, von Proxy Nr. 1 wird bereits nur das höhere Angebot von Client A weitergeleitet und jenes von Client B sofort zurückgewiesen. Danach sendet Client B ein noch höheres Angebot und Client C ein niedrigeres als Client B. Das Angebot von Client B wird weitergeleitet; das Angebot von Client C, weil auf einem anderen Proxy 4, nämlich Proxy Nr. 2, eingelangt, wird erst vom Server 7 zurückgewiesen.

Im an sich aus sich heraus verständlichen Beispiel gemäß Fig. 14A lag das erste Angebot von Client A bei 100, das danach von Client B übermittelte Angebot lag bei 50; dieses Angebot wurde von Client B jedoch nachfolgend auf 200 erhöht; das danach von Client C übermittelte Angebot von 150 lag demgemäß unter diesem

vorherigen Angebot von 200 und war somit nicht erfolgreich. Im Anschluss daran hat der Verkäufer, also Client D, entschieden, nicht weiter auf Angebote zu warten, und er hat das Angebot des Clients B mit 200 akzeptiert, sodass der Artikel an B verkauft wird.

Weitere Details in Zusammenhang mit diesem beispielhaften Ablauf können direkt der Fig. 14A entnommen werden.

In Fig. 14B ist in einem Flussdiagramm der allgemeine Filtervorgang auf einem der Proxys (Proxy Nr. 1 bzw. Proxy Nr. 2 gemäß Fig. 14A) dargestellt. Dabei wird ein gemäß Feld 95 einlangendes Angebot gemäß Prüffeld 96 dahingehend überprüft, ob - im Cache 16 dieses Proxys - ein höheres Angebot vorliegt oder nicht. Wenn nein, wird gemäß Block 97 der Cache 16 für weitere Requests auf das aktuelle Höchstgebot gesetzt, und das Angebot wird an den Leit-Server 7 weitergeleitet und dort verarbeitet, vgl. Block 98; in der Folge wird der Cache 16 entsprechend der Antwort vom Server 7 gemäß Block 99 aktualisiert, und gemäß Feld 100 wird eine Response aus dem Cache 16 an den jeweiligen Client 2 zurückgegeben.

Liegt jedoch bei der Überprüfung gemäß Feld 96 bereits ein höheres Angebot vor, so wird gemäß Block 101 das betroffene Angebot zurückgewiesen und in der lokalen Datenbank 13 gespeichert; gemäß Block 102 wird das Höchstgebot aus dem Cache 16 ausgelesen und in der Response an den betroffenen Client 2 zurückgegeben, s. Feld 100.

Bei der Filterung auf dem Leit-Server 7 gemäß dem Flussdiagramm in Fig. 14C wird ein von einem Proxy 4 einlangendes Angebot, siehe Feld 105, gemäß Prüffeld 106 daraufhin überprüft, ob ein höheres Angebot vorhanden ist. Wenn dies nicht zutrifft, wird gemäß Block 107 das eingelangte Angebot in der zentralen Datenbank 8 gespeichert, und der Cache-Speicher 16 des Servers 7 wird auf das aktuelle Höchstgebot gesetzt, s. Block 108. Danach werden alle Proxys 4 gemäß Block 109 von diesem nunmehrigen Höchst-Angebot benachrichtigt, und eine entsprechende Response wird aus dem Cache 16 an die Proxys 4 zurückgegeben, s. Feld 110.

Lag jedoch bereits ein höheres Angebot vor (s. Prüffeld 106), so wird gemäß Block 111 das erhaltene Angebot zurückgewiesen und in der lokalen Datenbank 13 des Leit-Servers 7 gespeichert. Gemäß Block 112 wird ferner das existierende Höchstgebot aus dem Cache-Speicher 16 ausgelesen, und es wird eine entsprechende Response aus dem Cache 16 an die Proxys 4 zurückgegeben, s. Feld 110.

Das nächste Beispiel, das anhand der Figuren 15A bis 15C erläutert werden soll, bezieht sich auf den Online-Verkauf von Artikelmengen (sog. „Teleshopping-Kanal"). Im Einzelnen wird nacheinander oder parallel jeweils eine bestimmte Stückzahl von gleichwertigen Artikelangeboten verkauft. Jeder einlangende Kaufauftrag wird automatisch angenommen, bis alle Artikel verkauft sind. Es können in einem Kaufauftrag auch mehrere Artikel (z.B. 2 Stück) abgenommen werden.

Jeder Kaufauftrag zu einem Artikelangebot, der auf einem Proxy 4 einlangt, wird weitergeleitet, sofern die aktuell bekannte Zahl von Artikeln eines zu verkaufenden Artikelangebotes noch nicht erschöpft ist. Alle anderen Kaufaufträge werden sofort zurückgewiesen und nur in der lokalen Datenbank 13 des betreffenden Proxy 4 gespeichert. Jeder einzelne Verkauf muss aber vom Server 7 bestätigt werden, da auf anderen Proxys 4 zeitgleich ebenfalls Artikel des gleichen Artikelangebots gewünscht worden sein konnten.

Auf dem Server 7 einlangende Kaufaufträge bewirken den Verkauf der Stückzahl des Artikels, bis die verfügbare Stückzahl dieses Artikelangebotes erreicht ist. Jeder Verkauf wird in der zentralen Datenbank 8 gespeichert. Der Server 7 benachrichtigt umgehend alle Proxys 4 von jedem abgeschlossenen Verkauf und von der noch verfügbaren Stückzahl dieses Artikelangebotes. Alle weiteren Kaufaufträge werden sofort zurückgewiesen und nur in der lokalen Datenbank 13 des Servers 7 gespeichert.

Die Zahl der Kaufaufträge pro Artikel, die an den Server 7 gesendet werden, ist mit dem Produkt der Anzahl der Proxys 4 mal

der Stückzahl des jeweiligen Artikelangebotes begrenzt.

Im konkret gezeigten Beispiel von Fig. 15A sind Client A, B und C mit Proxy Nr. 1 verbunden, Client D mit Proxy Nr. 2. Vom Artikelangebot sind 3 Stück vorhanden. A kauft 1 Stück, anschließend kauft B 2 Stücke. Die unmittelbar auf den Kaufauftrag von B folgenden Kaufaufträge von C und D werden zurückgewiesen. Der Kaufauftrag von Client C wird bereits unmittelbar von Proxy 1 (keine Weiterleitung an den Controller 7) zurückgewiesen, jener von Client D erst vom Controller 7, weil die Benachrichtigung über die erschöpfte Stückzahl noch nicht auf Proxy 2 eingelangt war.

Der vorstehend kurz beschriebene Ablauf ergibt sich unmittelbar aus der Darstellung in Fig. 15A, sodass sich eine weitere Erläuterung hiervon erübrigen kann.

In den Figuren 15B und 15C sind wiederum Filtervorgänge einerseits auf einen Proxy (Fig. 15B) und andererseits auf dem Leit-Server 7 (Fig. 15C) in Flussdiagrammen veranschaulicht.

Gemäß Fig. 15B langt entsprechend Feld 115 ein Kaufauftrag bei einem Proxy 4 (z.B. Proxy Nr. 1) ein, und es wird dort gemäß Feld 116 sofort überprüft, ob nach Stand an diesem Proxy 4 genügend Stück von dem Artikel vorhanden sind. Wenn ja, wird gemäß Block 117 die Stückzahl im Cache 16 dieses Proxys für weitere Requests reduziert, und gemäß Block 118 erfolgt eine Weiterleitung des Kaufauftrags an den Server 7 und eine entsprechende Verarbeitung auf diesem Server 7. Gemäß Block 119 wird sodann der Cache-Speicher 16 mit der Response von Server 7 aktualisiert, und eine entsprechende Response wird aus dem Cache 16 an den beauftragenden Client 2 zurückgegeben, s. Feld 120.

Eine derartige Vorgangsweise ergibt sich bei den ersten beiden Kaufaufträgen, nämlich von Client A und Client B, gemäß Fig. 15A.

Wenn jedoch im Prüffeld 116 festgestellt wird, dass keine ausreichende Stückzahl mehr vorhanden ist (vgl. den Auftrag zum Kauf von 1 Stück von Client C bzw. zum Kauf von 2 Stück von

Client D), so wird gemäß Block 121 in Fig. 15B der Kaufauftrag vom zugehörigen Proxy (Proxy Nr. 1 im ersten Fall und Proxy Nr. 2 im zweiten Fall) zurückgewiesen und in der zugehörigen lokalen Datenbank 13 gespeichert. Gemäß Block 121 wird die noch verfügbare Stückzahl aus dem Cache 16 ausgelesen. Danach erfolgt wieder eine entsprechende Antwort aus dem Cache-Speicher 16 an den jeweiligen Client 2, gemäß Feld 120.

Was den Server 7 betrifft, s. Fig. 15C, so wird nach einem gemäß Feld 125 in Fig. 15C von einem Proxy 4 einlangenden Kaufauftrag gemäß Prüffeld 126 ebenfalls überprüft, ob die Stückzahl ausreicht, und wenn ja, wird der betreffende Kaufauftrag gemäß Block 127 in der zentralen Datenbank 8 gespeichert. Der Cache 16 des Servers 7 wird mit der neuen Stückzahl aktualisiert, Block 128, und alle Proxys (die Proxys Nr. 1 und Nr. 2 in Fig. 15A) werden hierüber benachrichtigt; eine entsprechende Response aus dem Cache 16 wird an die Proxys 4 zurückgegeben.

Sind jedoch nicht mehr genügend Stück des Artikels vorhanden (Prüffeld 126), so wird gemäß Block 131 der Kaufauftrag zurückgewiesen und in der lokalen Datenbank 13 des Servers 7 gespeichert. Gemäß Block 132 wird die noch verfügbare Stückzahl (gegebenenfalls auch eine Stückzahl 0) aus dem Cache 16 des Servers 7 ausgelesen, und eine entsprechende Response wird aus dem Cache 16 an die Proxys 4 zurückgegeben, s. Feld 130 in Fig. 15C.

Die letzten beiden Beispiele, gemäß Fig. 16 und gemäß Fig. 17, beziehen sich auf verschiedene Bietverfahren oder Auktionen, die mit dem vorliegenden Rechnersystem 1 ebenfalls in Echtzeit und online, auch bei einer sehr großen Anzahl von Teilnehmern, möglich sind.

Im Einzelnen ist in den Figuren 16A, 16B und 16C die Vorgangsweise bei einem „englischen" Bietverfahren veranschaulicht, bei dem Bieter in der Auktion live auf einen Artikel bieten; der Auktionator nimmt jeweils das erste Gebot, das höher als das vorhergehende ist, an. Wenn für eine bestimmte Zeit keine höheren Angebote mehr einlangen, erfolgt der Zuschlag.

Jeweils nur das erste Gebot in der Höhe des nächsten Bietschrittes (der vom Auktionator vorgegeben wird), das auf einem Proxy 4 einlangt, wird an den Server 7 weitergeleitet. Alle anderen Gebote werden sofort zurückgewiesen und in der lokalen Datenbank 13 des betreffenden Proxys 4 gespeichert.

Auch der Server 7 berücksichtigt nur das erste Gebot in der Höhe des nächsten Bietschrittes, speichert dieses unmittelbar in der zentralen Datenbank 8 und benachrichtigt umgehend alle Proxys 4 von diesem Gebot. Alle anderen Gebote werden sofort zurückgewiesen und nur in der lokalen Datenbank 13 gespeichert.

Der Auktionator wartet einen bestimmten Zeitablauf ab und schlägt den Artikel dann zum höchsten eingelangten Gebot zu. Dieses wurde über einen Proxy 4 ebenfalls zum Server 7 weitergeleitet, dieser speichert den Zuschlag und benachrichtigt alle anderen Proxys 4 vom Zuschlag.

Beispielsweise sind gemäß Fig. 16A Client A und B (Käufer) mit Proxy Nr. 1 verbunden, Client C (Käufer) und Client D (Auktionator) mit Proxy Nr. 2. Client A und B bieten kurz hintereinander mit dem gleichen Bietschritt, danach B und C. Das erste Gebot von Client B kann sofort vom Proxy Nr. 1 ohne Weiterleitung an den Controller 7 zurückgewiesen werden, das Gebot von Client C wird erst vom Server 7 zurückgewiesen, da dieses am Proxy Nr. 2 eingelangt war, der noch nicht vom früheren Einlangen des Gebotes von Client A (am Proxy Nr. 1) benachrichtigt worden war. Der Auktionator, Client D, sieht immer nur das jeweilige Höchstgebot und schlägt dieses zu gegebener Zeit zu.

Im Detail ergibt sich der konkrete Ablauf bei diesem englischen Bietverfahren, bei dem zuerst zeitlich hintereinander zwei gleichlautende Gebote von A und B und danach wiederum gleichlautende, erhöhte Gebote zeitlich aufeinanderfolgend von B und C bei den jeweiligen Proxys einlangen, wobei das erhöhte Gebot von Client B letztlich den Zuschlag erhält, unmittelbar aus dem Sequenzdiagramm von Fig. 16A, sodass sich eine detaillierte Beschreibung erübrigen kann.

Es soll allerdings nachfolgend wiederum im Einzelnen auf die Flussdiagramme von Fig. 16B und 16C in Zusammenhang mit den Filtervorgängen am jeweiligen Proxy 4 bzw. am Server 7 eingegangen werden.

In Fig. 16B ist wieder der Filtervorgang auf einem Proxy 4 in einem Flussdiagramm veranschaulicht. Gemäß Feld 135 langt ein Gebot am Proxy 4 (z.B. Proxy Nr. 1 in Fig. 16A) ein, und im Zuge der Relevanz-Prüfung wird gemäß dem Prüffeld 136 abgefragt, ob es sich hierbei um ein erstes Gebot in dieser Höhe handelt, wobei wie vorstehend anhand Fig. 2 erläutert ein Vergleich mit dem Inhalt des Cache-Speichers 16 dieses Proxys 4 durchgeführt wird. Im Fall dass es sich tatsächlich um ein erstes Gebot in der gegebenen Höhe handelt, wird gemäß Block 137 der Cache-Speicher 16 des Proxys 4 für weitere Requests auf diese Gebotshöhe gesetzt, und das Gebot wird gemäß Block 138 an den Leit-Server 7 weitergeleitet und dort verarbeitet. Gemäß Block 139 wird dann der Cache-Speicher 16 mit der Response vom Leit-Server 7 aktualisiert (vgl. die Zeile „Cache auf Gebot Client A = 100 setzen" in Fig. 16A), und eine entsprechende Response wird aus dem Cache-Speicher 16 an den jeweiligen Client, z.B. den Client A gemäß Fig. 16A, zurückgegeben.

Im Fall dass bereits ein besseres Gebot vorliegt, was gemäß Prüffeld 136 festgestellt wird, wird das Gebot gemäß Block 141 zurückgewiesen und in der lokalen Datenbank 13 des jeweiligen Proxys 4 gespeichert. Gemäß Block 142 wird das Höchstgebot aus dem Cache 16 ausgelesen, und gemäß Feld 140 wird eine Antwort aus dem Cache 16 an den betroffenen Client 2 zurückgegeben, wie etwa die Antwort „bereits überboten" in Fig. 16A.

Was den Server 7 betrifft, der wiederum dann die Relevanz-Prüfung durchführt, wenn Gebote über verschiedene Proxys 4 einlangen, so wird ein gemäß Feld 145 einlangendes Gebot, das von einem Proxy 4 kommt, gemäß Prüffeld 146 ebenfalls dahingehend überprüft, ob es sich um ein erstes Gebot dieser Höhe handelt. Wenn ja, wird dieses Gebot in der zentralen Datenbank 8 gespeichert, s. Block 147, und gemäß Block 148 wird der Cache-Speicher 16 auf dieses Gebot gesetzt, und alle Proxys 4 werden gemäß

Block 149 entsprechend benachrichtigt, vgl. beispielsweise die Benachrichtigung „Client A = 100" an Proxy Nr. 2 in Fig. 16A. Das Feld 160 in Fig. 16C betrifft auch diese Zurückgabe der Response aus dem Cache-Speicher 16 an den jeweiligen Proxy 4.

Abschließend soll nun noch der Vorgang bei einem sog. „holländischen" Bietverfahren anhand der Fig. 17 erläutert werden, wobei hier der Preis für einen bestimmten Artikel kontinuierlich verringert wird, bis der erste Kaufauftrag zum angezeigten Preis einlangt.

Nur der allererste auf einem Proxy 4 einlangende Kaufauftrag des vom Auktionator laufend gesenkten Preises wird an den Server 7 weitergeleitet. Alle anderen werden sofort zurückgewiesen und in der lokalen Datenbank 13 des Proxys gespeichert.

Auch der Server 7 speichert nur den allerersten Kaufauftrag in der Datenbank 8 und benachrichtigt umgehend alle Proxys 4 vom erfolgten Verkauf. Alle anderen Kaufaufträge werden sofort zurückgewiesen und nur in der lokalen Datenbank 13 gespeichert.

Die Festlegung eines abgesenkten Preises durch den Auktionator wird zunächst ebenfalls an einen Proxy 4 und dann an den Server 7 geleitet. Dies erfolgt ohne sofortige Aktualisierung des Caches 16 auf dem Proxy 4 - die Proxys 4 erfahren erst durch die Benachrichtigung vom Server 7 vom neuen Preis, damit dies auf allen Proxys 4 möglichst zeitgleich passiert.

Die Zahl der Kaufaufträge pro Artikel, die an den Server 7 gesendet werden, ist mit der Anzahl der Proxys 4 begrenzt.

Im Beispiel gemäß Fig. 17 A sind beispielsweise Client A und B (Käufer) mit Proxy Nr. 1 verbunden, Client C (Käufer) und Client D (Auktionator) mit Proxy Nr. 2. Zunächst verringert Client D (Auktionator) den Preis von 500 auf 400, dann senden alle Käufer ihre Kaufaufträge kurz hintereinander, der Kaufauftrag von Client A schließt - als erster bei diesem Preis - erfolgreich ab. Der Kaufauftrag von Client B kann bereits sofort von Proxy Nr. 1 zurückgewiesen werden, jener von Client C erst vom Server

7, da auf dem Proxy Nr. 2 die Benachrichtigung über den erfolgten Verkauf noch nicht eingelangt ist.

Auch hier ist wiederum das Sequenzdiagramm gemäß Fig. 17A, ähnlich wie die Diagramme gemäß Fig. 16A, 15A oder auch 14A, aus sich heraus verständlich, sodass keine weitere Erläuterung erforderlich ist.

In Fig. 17B ist wiederum in einem Flussdiagramm der Ablauf bei einem Filtervorgang, d.h. bei einer Relevanz-Prüfung, auf einem Proxy veranschaulicht, wobei ein gemäß Feld 155 von einem Client 2 einlangender Kaufauftrag gemäß Relevanz-Prüfungsfeld 156 auf Relevanz prüft, d.h. es wird abgefragt, ob der betreffende Gegenstand bereits verkauft ist. Wenn nein, wird gemäß Block 157 der betreffende Cache-Speicher 16 für weitere Requests von zugehörigen Clients auf „verkauft" gesetzt, gemäß Block 158 wird der Kaufauftrag an den Leit-Server 7 weitergeleitet und verarbeitet und gemäß Block 159 wird der Cache-Speicher 16 mit der Response aktualisiert, die vom Leit-Server 7 einlangt.

Gemäß Feld 160 wird dann eine entsprechende Antwort aus dem Cache-Speicher 16 an den Client 2 zurückgegeben.

Wenn sich der Artikel gemäß Prüfung, Feld 156, bereits als verkauft erweist, wird gemäß Block 161 der Kaufauftrag des betreffenden Clients 2 zurückgewiesen und in der lokalen Datenbank 13 des betreffenden Proxys 4 gespeichert. Gemäß Block 162 wird der Verkaufsstatus aus dem Cache-Speicher 16 ausgelesen, und eine entsprechende Response wird aus dem Cache-Speicher 16 an den Client 2 zurückgegeben, s. Feld 160.

Was die in Fig. 17C in einem Flussdiagramm veranschaulichte Relevanz-Prüfung (-Filterung) auf dem Leit-Server 7 betrifft, so wird der gemäß Feld 165 von einem jeweiligen Proxy 4 einlangende Kaufauftrag gemäß Prüffeld 166 daraufhin untersucht, ob der Artikel bereits verkauft ist. Wenn nein, wird der Kaufauftrag in der zentralen Datenbank 8 gespeichert, s. Block 167, und gemäß Block 168 wird der Cache-Speicher 16 aktualisiert (Eintrag: „verkauft an Benutzer"). Danach werden gemäß Block 169 alle Pro-

xys 4, also in der vereinfachten Darstellung gemäß Fig. 17A die Proxys Nr. 1 und Nr. 2, benachrichtigt; eine entsprechende Antwort wird aus dem Cache 16 an die Proxys 4 zurückgegeben, s. Feld 170 in Fig. 17C.

Wenn jedoch der betreffende Artikel bereits verkauft ist (s. Prüffeld 166), so wird wiederum gemäß Block 171 der Kaufauftrag zurückgewiesen bzw. in der lokalen Datenbank 13 gespeichert. Gemäß Block 172 wird der Verkaufsstatus aus dem Cache-Speicher 16 ausgelesen, und gemäß Feld 170 wird eine Response aus dem Cache-Speicher 16 an den Proxy 4 zurückgegeben.

Aus den vorstehenden Darlegungen ist ersichtlich, dass mit dem vorliegenden Rechnersystem 1 durch die spezifische Aufgabenteilung, Lastverteilung und Filterung eine Online-Abwicklung von Prozessen (Handelsprozessen) allgemein in Echtzeit ermöglicht wird, wobei ein besonderer Aspekt dabei die spezielle Relevanz-Filterung in Proxy-Rechnern 4 ist, da somit bereits an dieser Zwischen-Stelle vielfach Fragen bzw. Aufträge gestoppt werden und nur wirklich relevante Requests zum Leit-Server 7 weitergeleitet werden. Mit Hilfe des beschriebenen Zeitstempelverfahrens wird zusätzlich eine Bandbreitenreduktion bzw. Reduktion der notwendigen Datenübertragungen erreicht, da nur Differenzdaten, d.h. Daten mit jüngerem (späterem) Zeitstempel, als relevante Daten oder Nachrichten akzeptiert werden.

Wenn die Erfindung vorstehend anhand von besonders bevorzugten Ausführungsbeispielen im Einzelnen erläutert wurde, so sind doch selbstverständlich Abwandlungen und Modifikationen denkbar, ohne dass der Rahmen der Erfindung verlassen wird. So ist es beispielsweise denkbar, ein Rechnersystem 1 ohne Kaskadierung 4' von Proxy-Rechnern 4, 4A vorzusehen. Auch kann beispielsweise ein einziges Load-Balancer-Rechnermodul 5 für alle Proxys 4 vorhanden sein, um die entsprechende Lastverteilung zu erzielen.

Patentansprüche:

1. Rechnersystem (1) zum Austausch von Nachrichten über Internet, für die Online-Abwicklung von Handels-Transaktionen, mit einer Mehrzahl von Client-Rechnern (2) mit Internet-Schnittstellen (3), und mit zumindest einem zentralen Leit-Server (7), der mit einer zentralen Datenbank (8) verbunden ist, dadurch gekennzeichnet, dass zwischen den Client-Rechnern (2) und dem zumindest einen zentralen Leit-Server (7) eine Mehrzahl von Proxy-Rechnern (4) angeordnet ist, denen zumindest ein Load-Balancer-Modul (5), das zur Verteilung von Nachrichten auf vorgegebene Proxy-Rechner (4) eingerichtet ist, vorgeschaltet ist, und die je ein Relevanz-Filtermodul (11) aufweisen, das eingerichtet ist, einlangende, von Client-Rechnern (2) kommende Nachrichten auf Relevanz, nach vorgegebenen Kriterien, zu prüfen und nur relevante Nachrichten weiterzuleiten.

2. Rechnersystem nach Anspruch 1, dadurch gekennzeichnet, dass zumindest ein Proxy-Rechner (4A) in Kaskade mit vorgeschalteten Proxy-Rechnern (4) angeordnet ist.

3. Rechnersystem nach Anspruch 2, dadurch gekennzeichnet, dass der Kaskade-Proxy-Rechner (4A) ebenfalls ein Relevanz-Filtermodul (11) aufweist, das relevante, von den vorgeschalteten Proxy-Rechnern (4) einlangende Nachrichten weiterleitet.

4. Rechnersystem nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, dass auch der zentrale Leit-Server (7) ein Relevanz-Filtermodul (11) aufweist, das relevante, von Proxy-Rechnern (4) einlangende Nachrichten zur weiteren Verarbeitung erfasst.

5. Rechnersystem nach Anspruch 4, dadurch gekennzeichnet, dass dem zentralen Leit-Server (7) weiters eine lokale Datenbank (13) zur zumindest vorübergehenden Speicherung von als nicht-relevant erkannten Nachrichten zugeordnet ist.

6. Rechnersystem nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, dass zumindest einem der Proxy-Rechner (4) eine

lokale Datenbank (13) zur zumindest vorübergehenden Speicherung von als nicht-relevant erkannten Nachrichten zugeordnet ist.

7. Rechnersystem nach Anspruch 5 oder 6, dadurch gekennzeichnet, dass eine Systemlast-Prüfeinheit (60) vorgesehen ist, die konfiguriert ist, um zu Zeiten verminderter Last im Rechnersystem die Übertragung von in der oder den lokalen Datenbank(en) (13) gespeicherten nicht-relevanten Nachrichten zur zentralen Datenbank (8), zur Datenkonsolidierung, zu veranlassen.

8. Rechnersystem nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, dass die Client-Rechner (2) zur zyklischen Anforderung von für sie bestimmten Nachrichten von den zugehörigen Proxy-Rechnern (4) in vorgegebenen Polling-Intervallen eingerichtet sind.

9. Rechnersystem nach einem der Ansprüche 1 bis 8, dadurch gekennzeichnet, dass die Client-Rechner (2) eingerichtet sind, Nachrichten zu den jeweiligen Proxy-Rechnern (4) unmittelbar, außerhalb von vorgegebenen Polling-Intervallen, zu übertragen.

10. Rechnersystem nach einem der Ansprüche 1 bis 9, dadurch gekennzeichnet, dass die Client- (2) und Proxy-Rechner (4) eingerichtet sind, die zwischen ihnen übertragenen Nachrichten mit Zeitstempeln (30; 30') zu versehen, und die Proxy-Rechner (4) eingerichtet sind, immer nur Nachrichten mit einem jüngerem Zeitstempel als jenem des Client-Rechner (2) an den Client-Rechner (2) zu versenden.

AW/pr

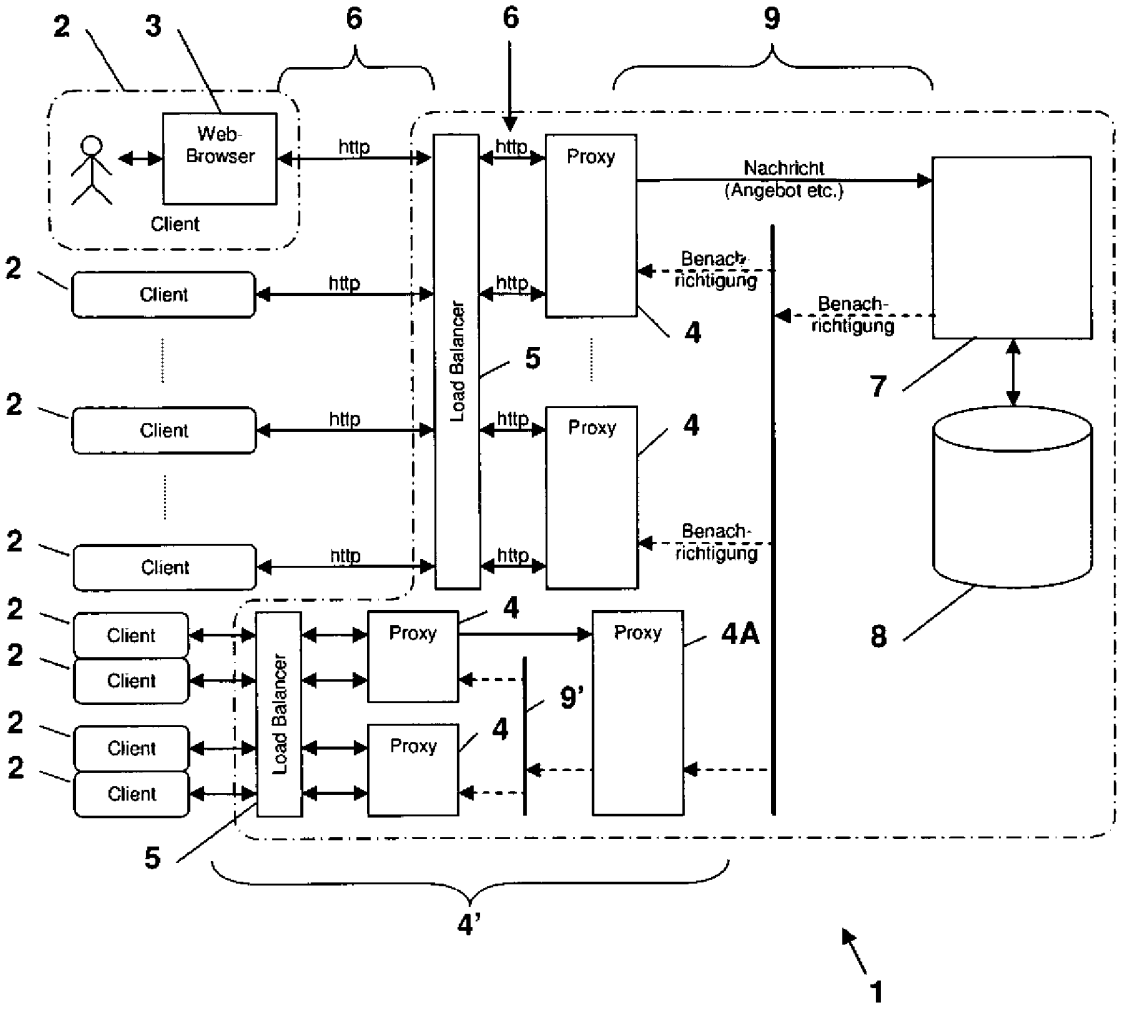

2
3
6
6
9
Web-
Browser
Client
http
Proxy
Nachricht
(Angebot etc.)
Benach-
richtigung
Client
Load Balancer
5
4
7
8
4A
9'
4'
1

FIG. 1

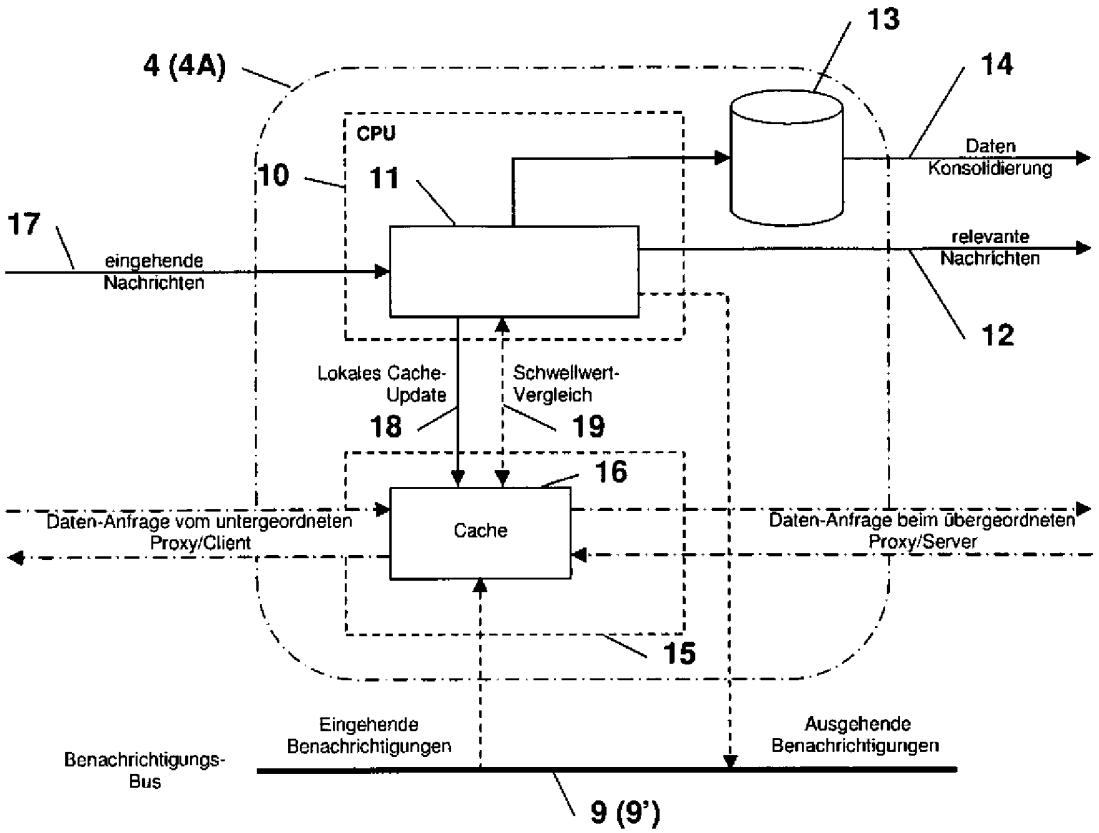

4 (4A)
13
14
CPU
10
11
Daten
Konsolidierung
17
eingehende
Nachrichten
relevante
Nachrichten
12
Lokales Cache-
Update
Schwellwert-
Vergleich
18
19
16
Daten-Anfrage vom untergeordneten
Proxy/Client
Cache
Daten-Anfrage beim übergeordneten
Proxy/Server
15
Eingehende
Benachrichtigungen
Ausgehende
Benachrichtigungen
Benachrichtigungs-
Bus
9 (9')

FIG. 2

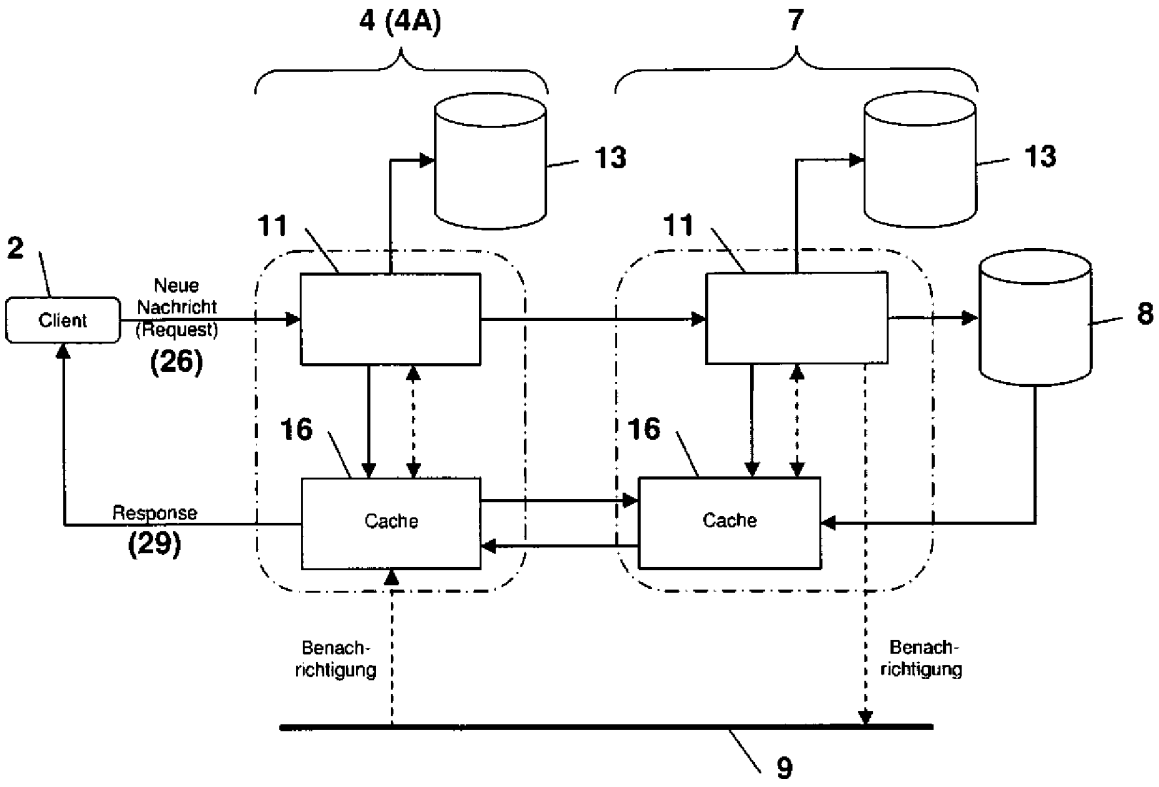

4 (4A)
7
13
13
11
11
2
Client
Neue
Nachricht
(Request)
(26)
8
16
16
Response
(29)
Cache
Cache
Benach-
richtigung
Benach-
richtigung
9

FIG. 3

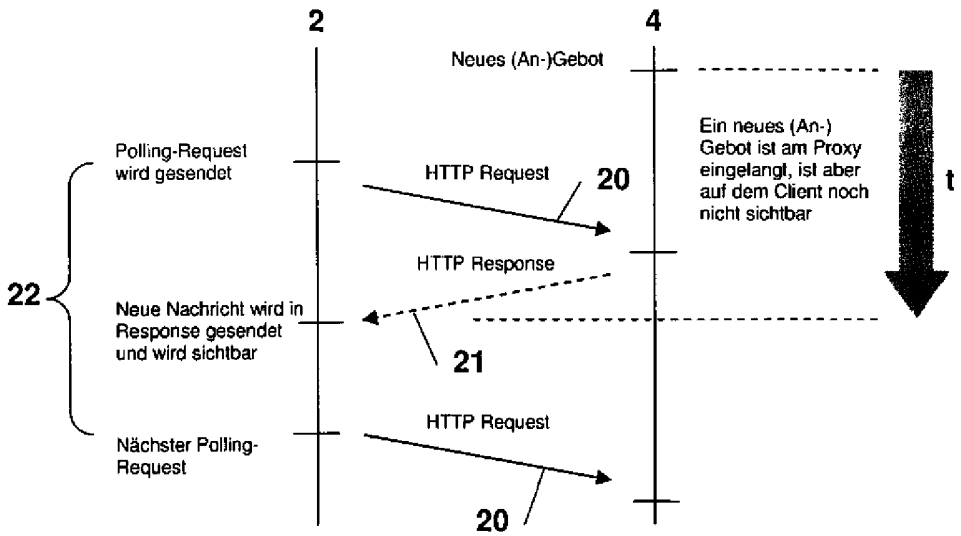

2
4
Neues (An-)Gebot
Ein neues (An-) Gebot ist am Proxy eingelangt, ist aber auf dem Client noch nicht sichtbar
t
Polling-Request wird gesendet
HTTP Request
20
HTTP Response
22
Neue Nachricht wird in Response gesendet und wird sichtbar
21
HTTP Request
Nächster Polling-Request
20

FIG. 4

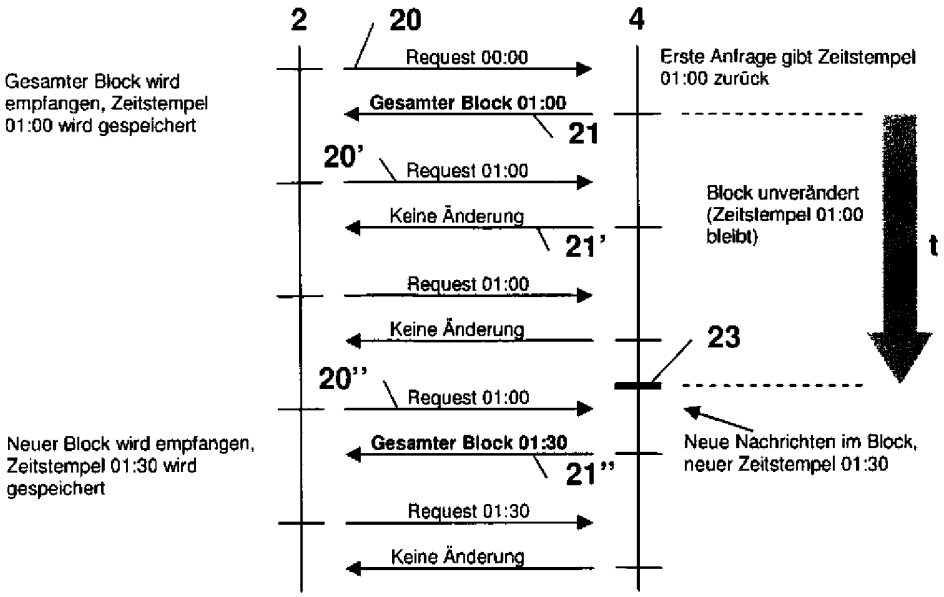

2
20
4
Request 00:00
Erste Anfrage gibt Zeitstempel 01:00 zurück
Gesamter Block wird empfangen, Zeitstempel 01:00 wird gespeichert
Gesamter Block 01:00
21
20'
Request 01:00
Block unverändert (Zeitstempel 01:00 bleibt)
Keine Änderung
21'
t
Request 01:00
Keine Änderung
23
20''
Request 01:00
Neuer Block wird empfangen, Zeitstempel 01:30 wird gespeichert
Gesamter Block 01:30
Neue Nachrichten im Block, neuer Zeitstempel 01:30
21''
Request 01:30
Keine Änderung

FIG. 5

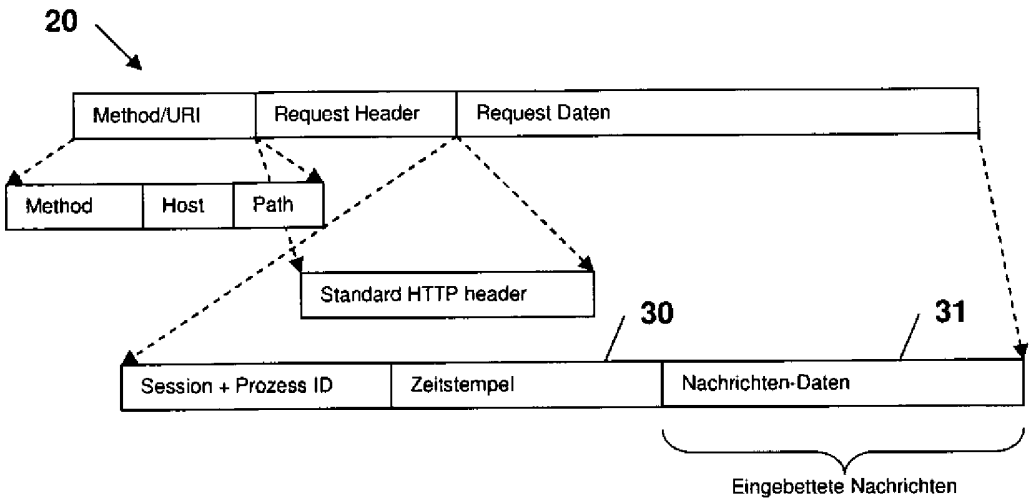

20
Method/URI
Request Header
Request Daten
Method
Host
Path
Standard HTTP header
30
31
Session + Prozess ID
Zeitstempel
Nachrichten-Daten
Eingebettete Nachrichten

FIG. 5A

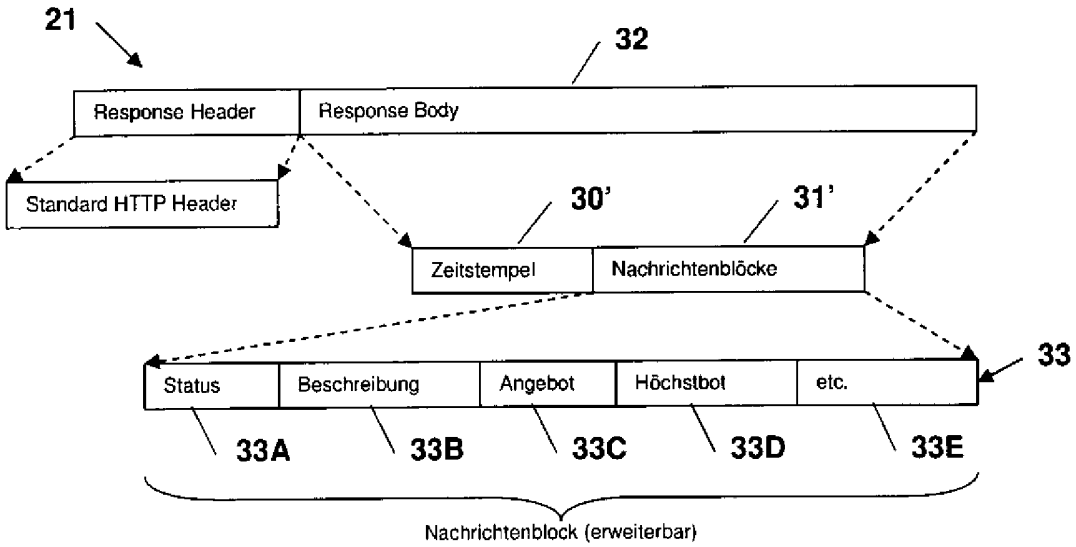

21
32
Response Header
Response Body
Standard HTTP Header
30'
31'
Zeitstempel
Nachrichtenblöcke
33
Status
Beschreibung
Angebot
Höchstbot
etc.
33A
33B
33C
33D
33E
Nachrichtenblock (erweiterbar)

FIG. 5B

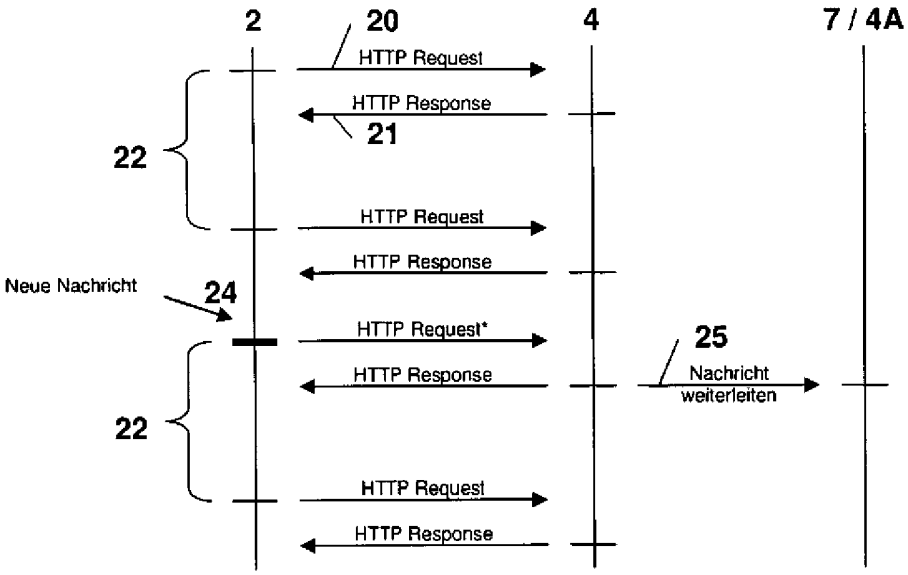
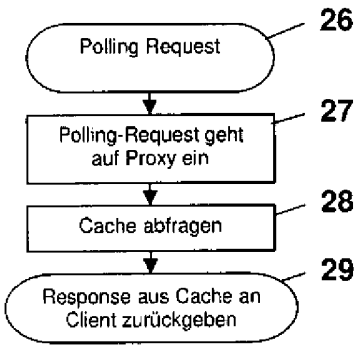

2
20
4
7 / 4A
HTTP Request
HTTP Response
21
22
HTTP Request
HTTP Response
Neue Nachricht
24
HTTP Request*
25
HTTP Response
Nachricht
weiterleiten
22
HTTP Request
HTTP Response

FIG. 6

FIG. 7


Polling Request
26
Polling-Request geht
auf Proxy ein
27
Cache abfragen
28
Response aus Cache an
Client zurückgeben
29

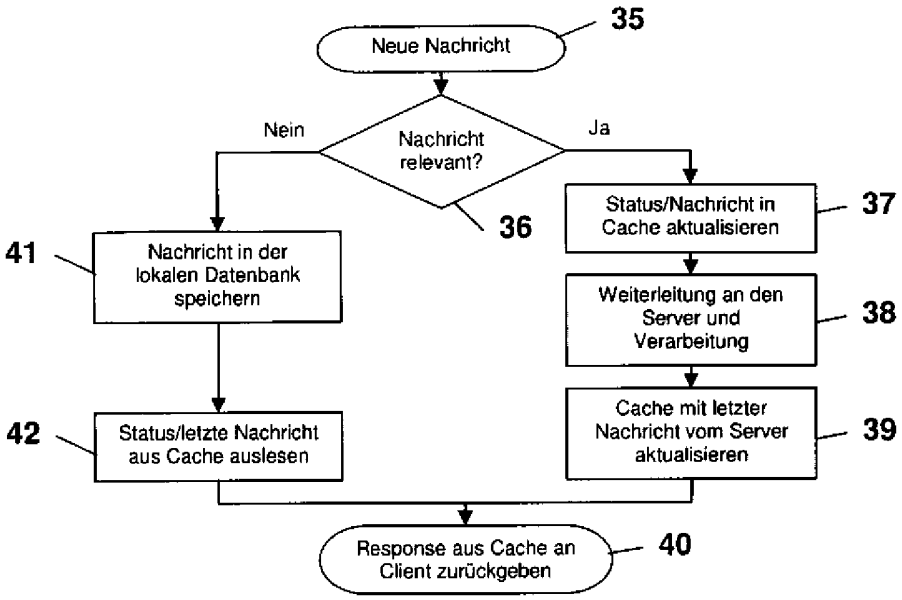

Neue Nachricht
35
Nachricht relevant?
Nein
Ja
36
Status/Nachricht in Cache aktualisieren
37
Weiterleitung an den Server und Verarbeitung
38
Cache mit letzter Nachricht vom Server aktualisieren
39
Nachricht in der lokalen Datenbank speichern
41
Status/letzte Nachricht aus Cache auslesen
42
Response aus Cache an Client zurückgeben
40

FIG. 8

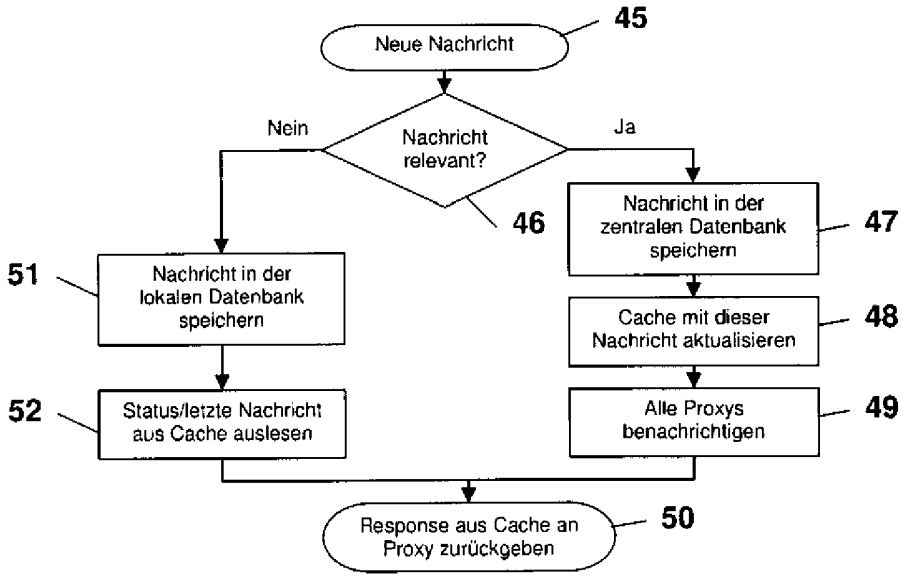

Neue Nachricht
45
Nachricht relevant?
Nein
Ja
46
Nachricht in der zentralen Datenbank speichern
47
Cache mit dieser Nachricht aktualisieren
48
Alle Proxys benachrichtigen
49
Nachricht in der lokalen Datenbank speichern
51
Status/letzte Nachricht aus Cache auslesen
52
Response aus Cache an Proxy zurückgeben
50

FIG. 9

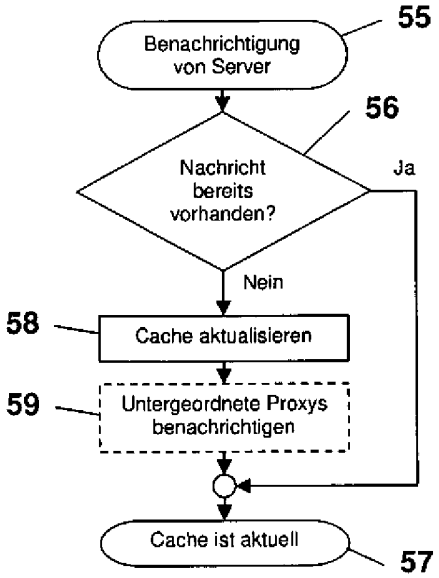

55
Benachrichtigung von Server
56
Nachricht bereits vorhanden?
Ja
Nein
58
Cache aktualisieren
59
Untergeordnete Proxys benachrichtigen
Cache ist aktuell
57

FIG. 10

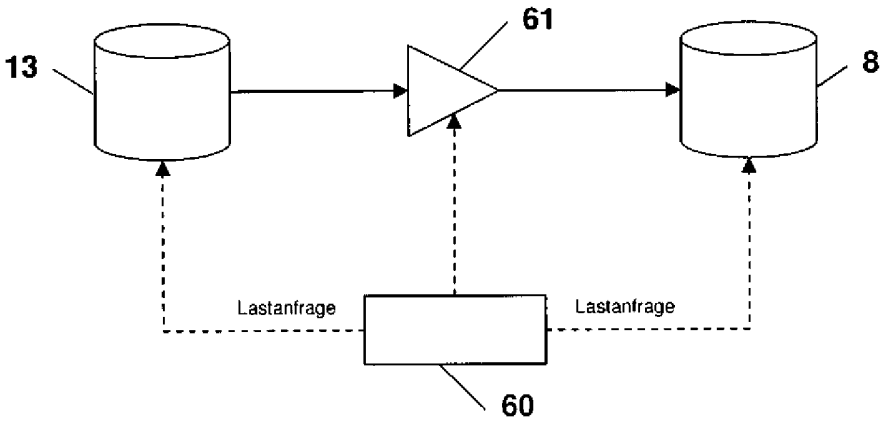

13
61
8
Lastanfrage
Lastanfrage
60

FIG. 11

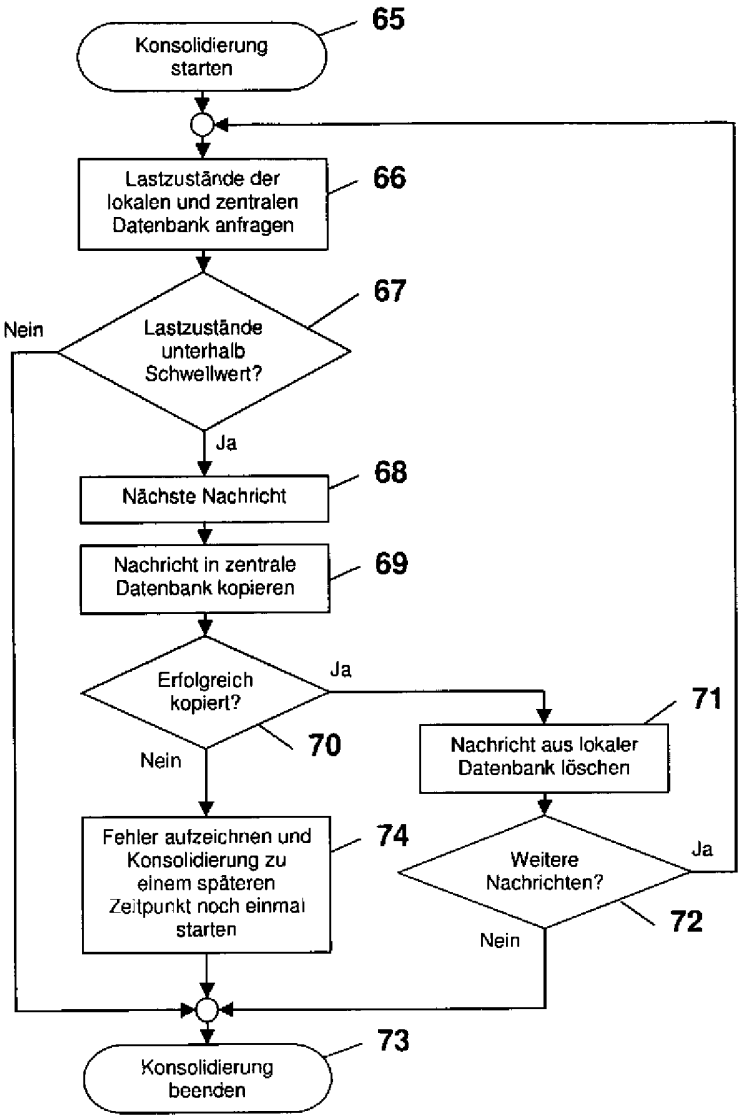

Konsolidierung starten
65
Lastzustände der lokalen und zentralen Datenbank anfragen
66
Lastzustände unterhalb Schwellwert?
67
Nein
Ja
Nächste Nachricht
68
Nachricht in zentrale Datenbank kopieren
69
Erfolgreich kopiert?
Ja
70
Nein
71
Nachricht aus lokaler Datenbank löschen
Weitere Nachrichten?
Ja
72
Nein
Fehler aufzeichnen und Konsolidierung zu einem späteren Zeitpunkt noch einmal starten
74
Konsolidierung beenden
73

FIG. 12

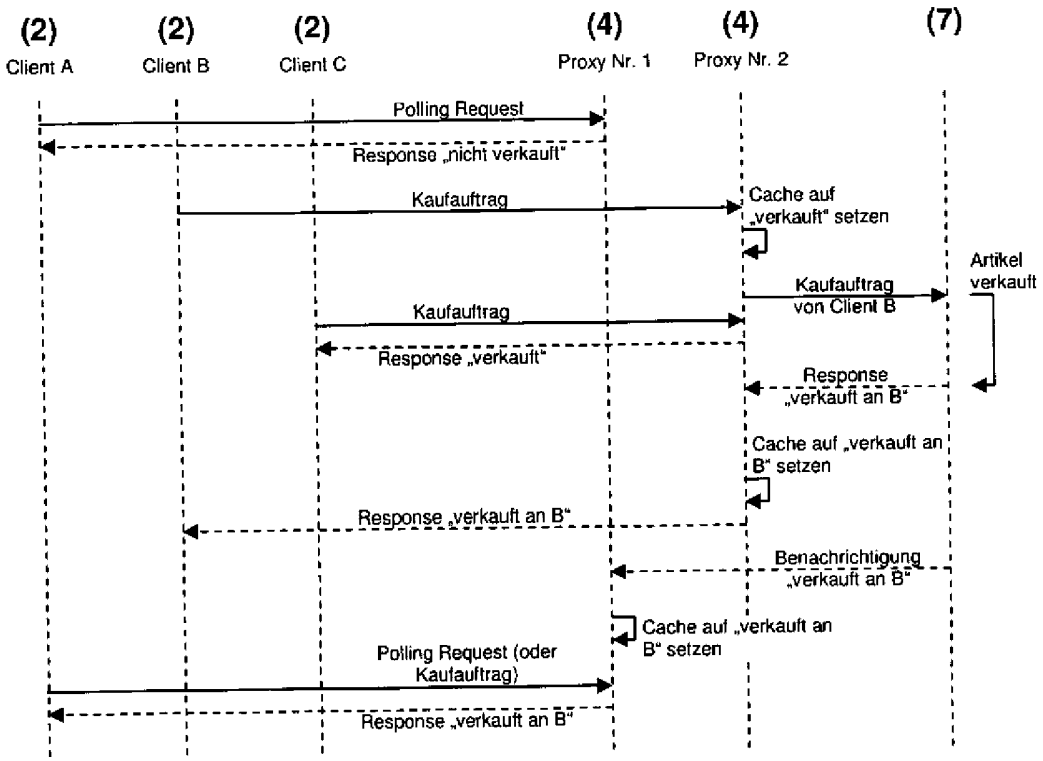

(2)
Client A
(2)
Client B
(2)
Client C
(4)
Proxy Nr. 1
(4)
Proxy Nr. 2
(7)
Polling Request
Response „nicht verkauft"
Kaufauftrag
Cache auf „verkauft" setzen
Artikel verkauft
Kaufauftrag von Client B
Kaufauftrag
Response „verkauft"
Response „verkauft an B"
Cache auf „verkauft an B" setzen
Response „verkauft an B"
Benachrichtigung „verkauft an B"
Cache auf „verkauft an B" setzen
Polling Request (oder Kaufauftrag)
Response „verkauft an B"

FIG. 13A

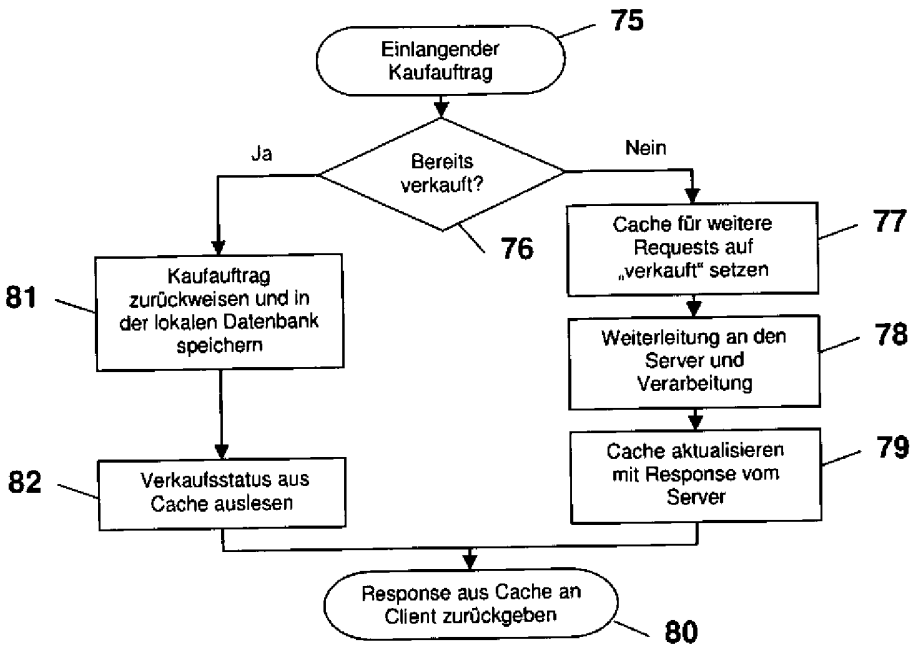


FIG. 13B

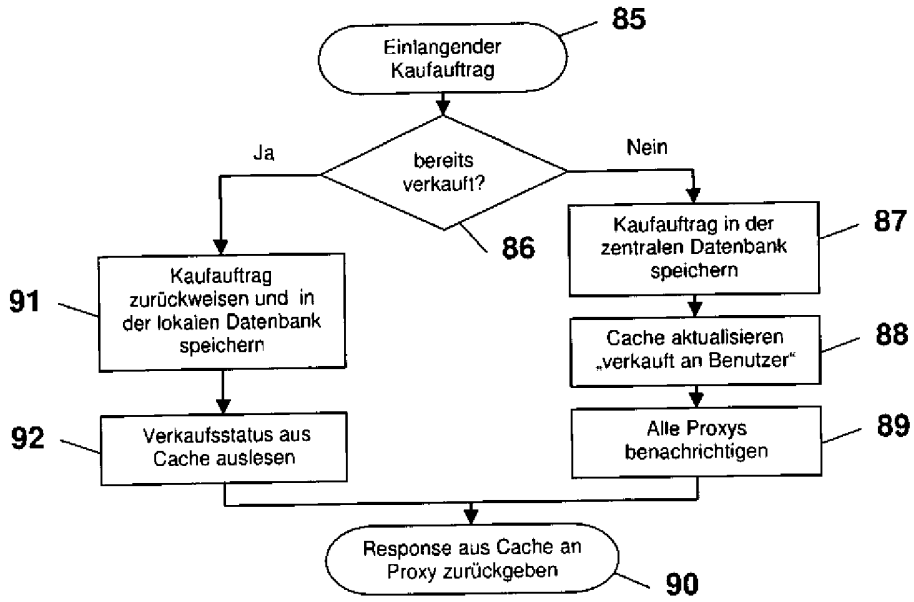


FIG. 13C

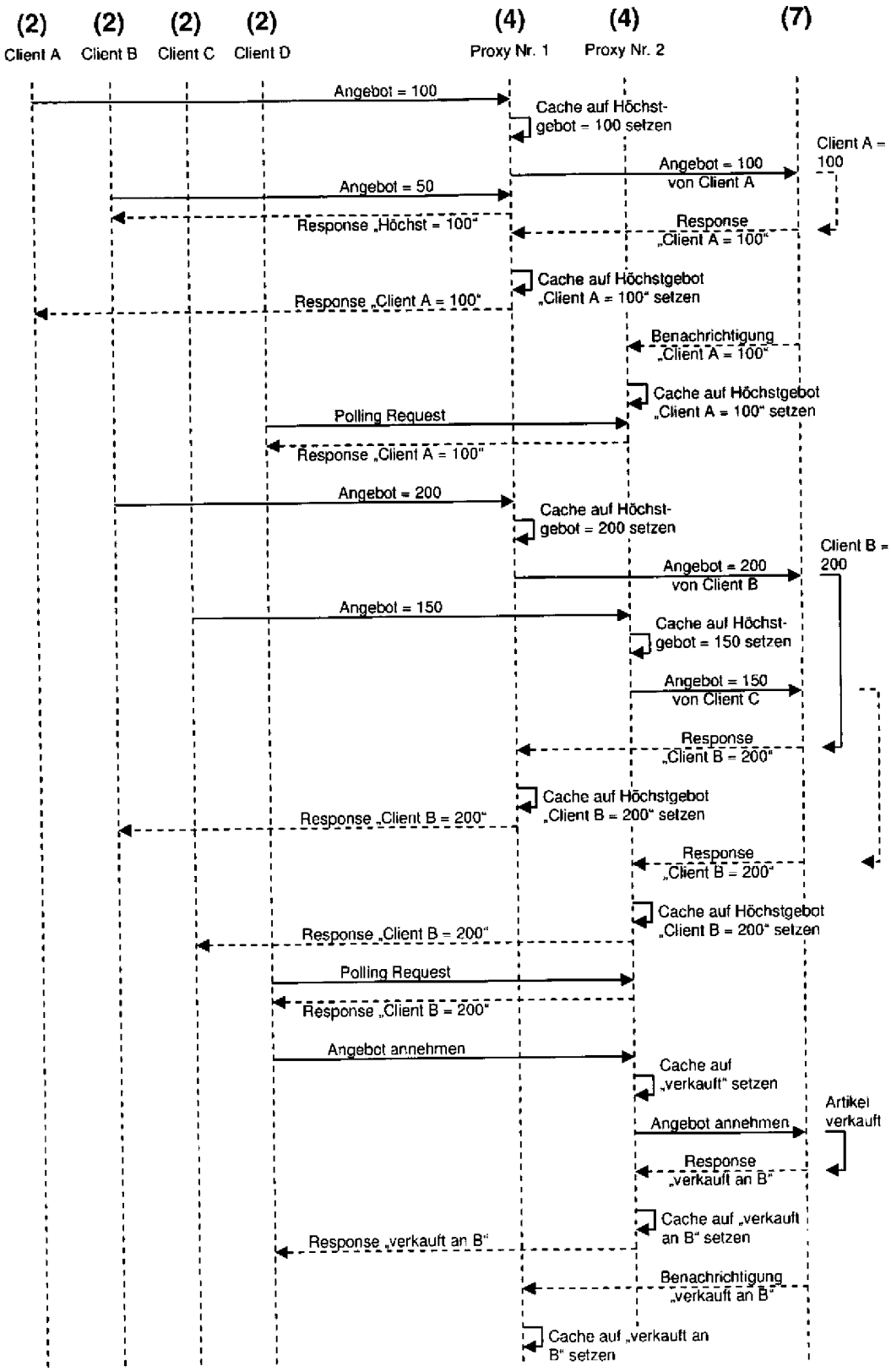

(2)
Client A
(2)
Client B
(2)
Client C
(2)
Client D
(4)
Proxy Nr. 1
(4)
Proxy Nr. 2
(7)
Angebot = 100
Cache auf Höchst-gebot = 100 setzen
Client A = 100
Angebot = 100 von Client A
Angebot = 50
Response „Höchst = 100"
Response „Client A = 100"
Cache auf Höchstgebot „Client A = 100" setzen
Response „Client A = 100"
Benachrichtigung „Client A = 100"
Cache auf Höchstgebot „Client A = 100" setzen
Polling Request
Response „Client A = 100"
Angebot = 200
Cache auf Höchst-gebot = 200 setzen
Client B = 200
Angebot = 200 von Client B
Angebot = 150
Cache auf Höchst-gebot = 150 setzen
Angebot = 150 von Client C
Response „Client B = 200"
Cache auf Höchstgebot „Client B = 200" setzen
Response „Client B = 200"
Response „Client B = 200"
Cache auf Höchstgebot „Client B = 200" setzen
Response „Client B = 200"
Polling Request
Response „Client B = 200"
Angebot annehmen
Cache auf „verkauft" setzen
Artikel verkauft
Angebot annehmen
Response „verkauft an B"
Cache auf „verkauft an B" setzen
Response „verkauft an B"
Benachrichtigung „verkauft an B"
Cache auf „verkauft an B" setzen

FIG. 14A

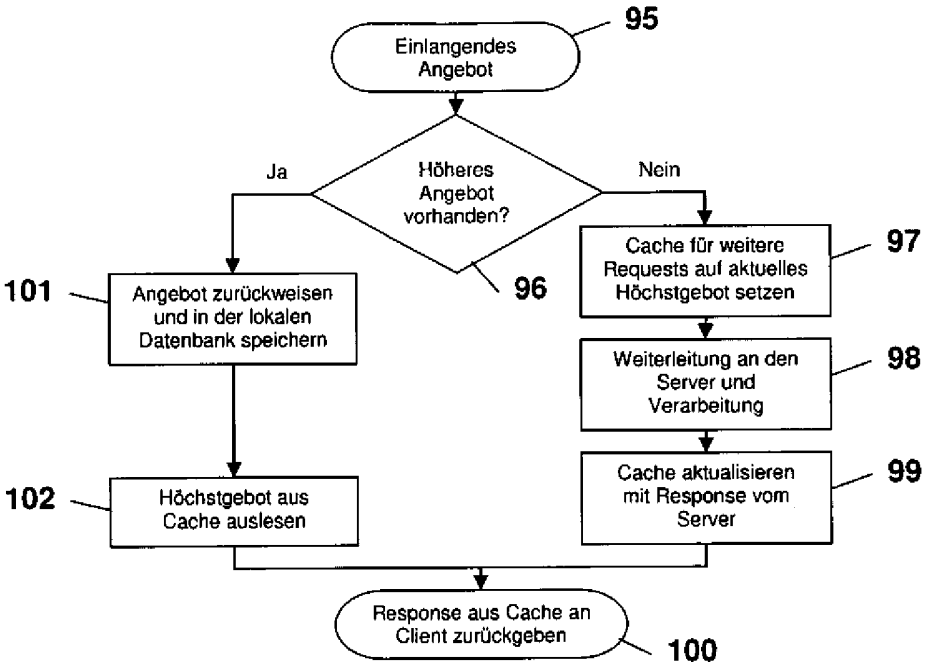


FIG. 14B

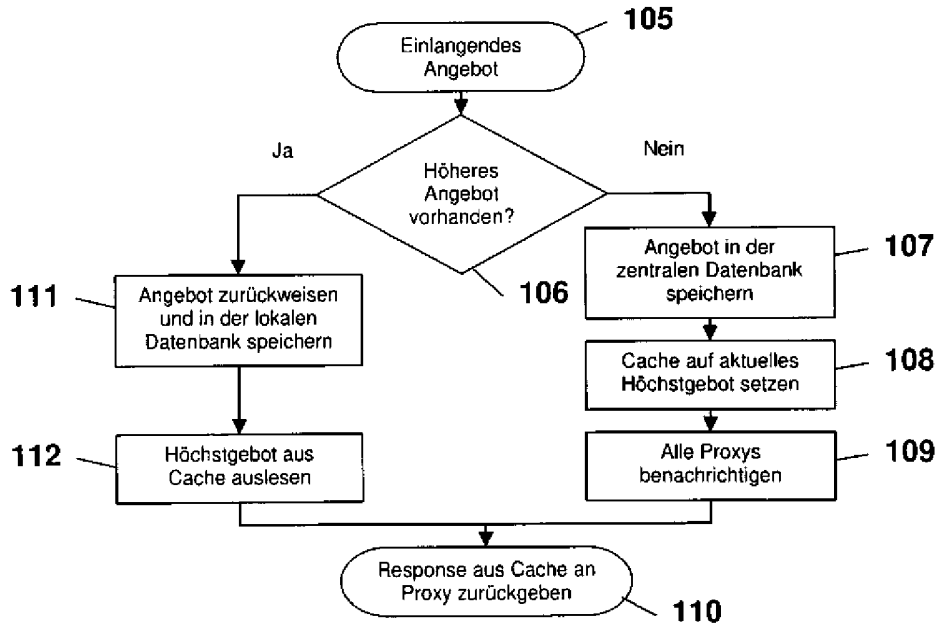


FIG. 14C

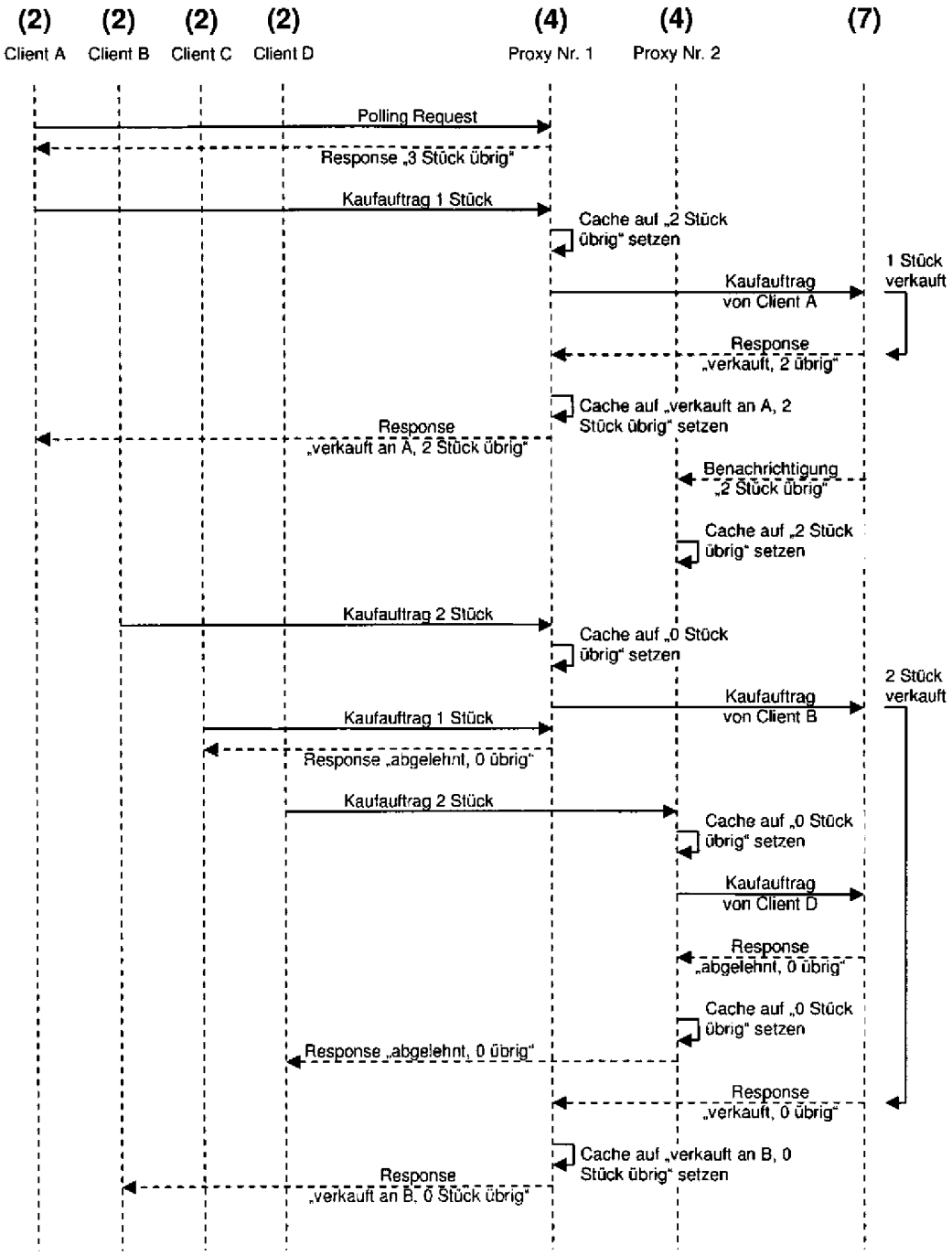

(2)
Client A
(2)
Client B
(2)
Client C
(2)
Client D
(4)
Proxy Nr. 1
(4)
Proxy Nr. 2
(7)
Polling Request
Response „3 Stück übrig"
Kaufauftrag 1 Stück
Cache auf „2 Stück übrig" setzen
1 Stück verkauft
Kaufauftrag von Client A
Response „verkauft, 2 übrig"
Cache auf „verkauft an A, 2 Stück übrig" setzen
Response „verkauft an A, 2 Stück übrig"
Benachrichtigung „2 Stück übrig"
Cache auf „2 Stück übrig" setzen
Kaufauftrag 2 Stück
Cache auf „0 Stück übrig" setzen
2 Stück verkauft
Kaufauftrag von Client B
Kaufauftrag 1 Stück
Response „abgelehnt, 0 übrig"
Kaufauftrag 2 Stück
Cache auf „0 Stück übrig" setzen
Kaufauftrag von Client D
Response „abgelehnt, 0 übrig"
Cache auf „0 Stück übrig" setzen
Response „abgelehnt, 0 übrig"
Response „verkauft, 0 übrig"
Cache auf „verkauft an B, 0 Stück übrig" setzen
Response „verkauft an B, 0 Stück übrig"

FIG. 15A

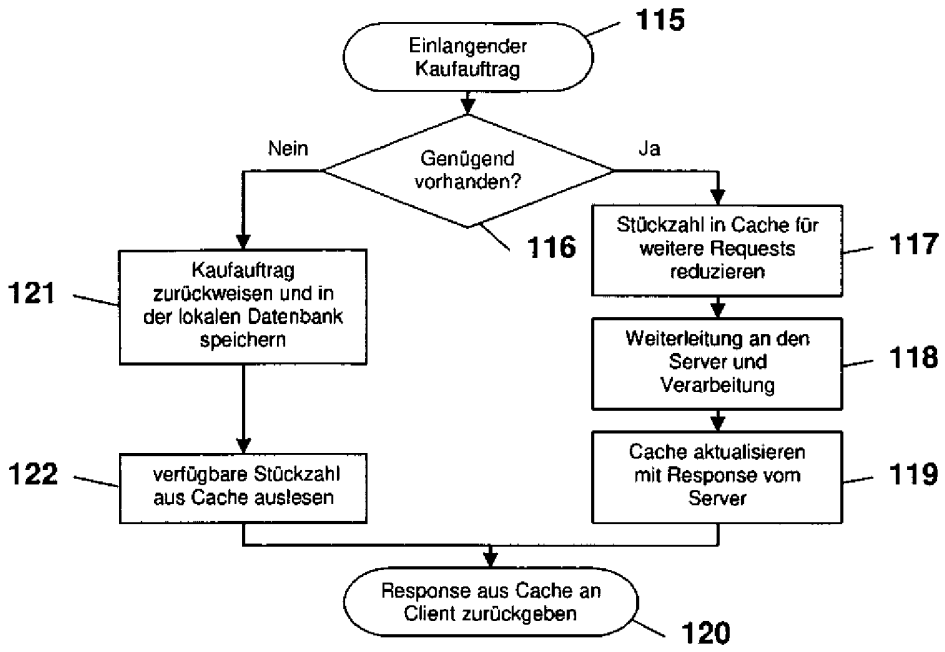


FIG. 15B

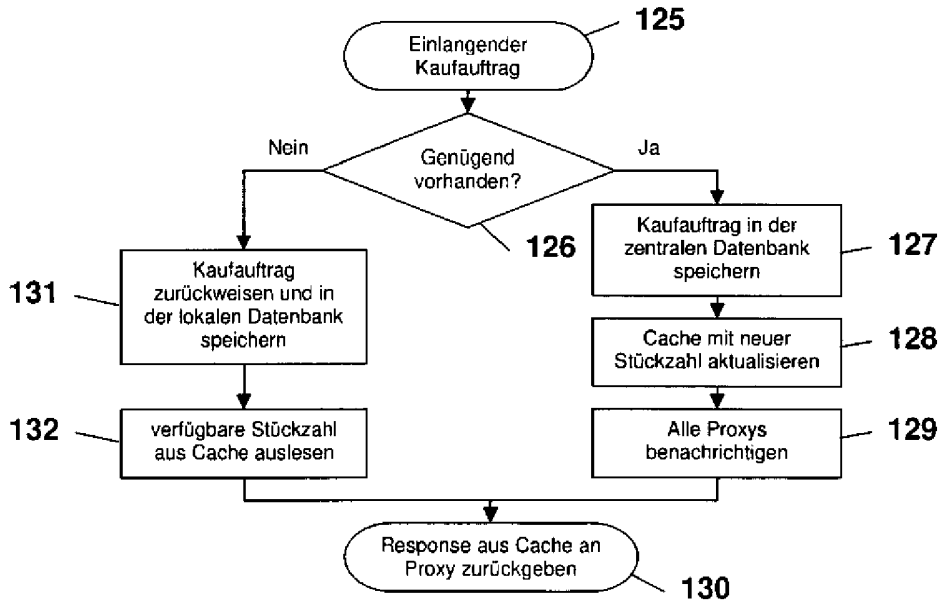


FIG. 15C

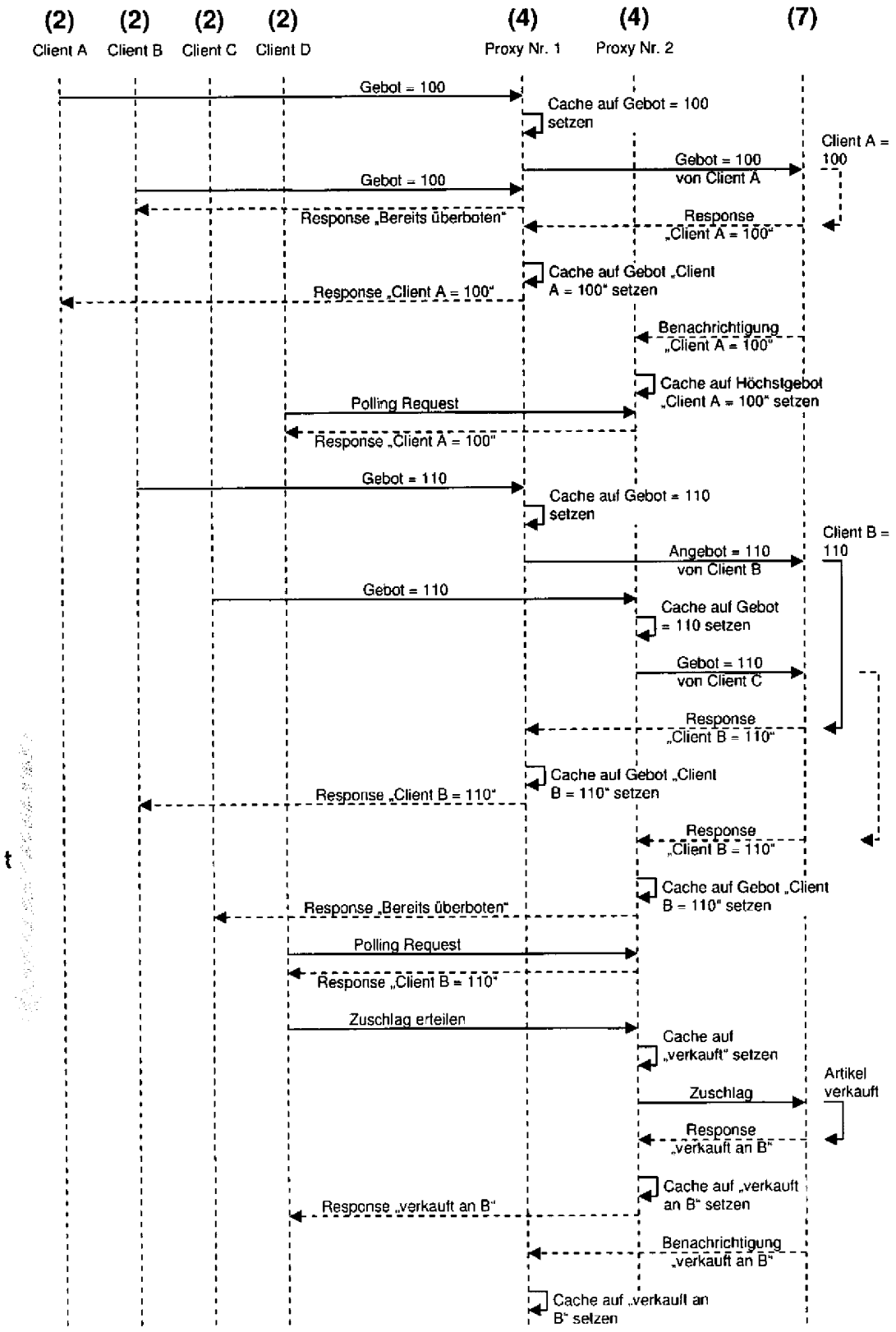

(2)
Client A
(2)
Client B
(2)
Client C
(2)
Client D
(4)
Proxy Nr. 1
(4)
Proxy Nr. 2
(7)
Gebot = 100
Cache auf Gebot = 100 setzen
Client A = 100
Gebot = 100 von Client A
Gebot = 100
Response „Bereits überboten"
Response „Client A = 100"
Cache auf Gebot „Client A = 100" setzen
Response „Client A = 100"
Benachrichtigung „Client A = 100"
Cache auf Höchstgebot „Client A = 100" setzen
Polling Request
Response „Client A = 100"
Gebot = 110
Cache auf Gebot = 110 setzen
Client B = 110
Angebot = 110 von Client B
Gebot = 110
Cache auf Gebot = 110 setzen
Gebot = 110 von Client C
Response „Client B = 110"
Cache auf Gebot „Client B = 110" setzen
Response „Client B = 110"
Response „Client B = 110"
Cache auf Gebot „Client B = 110" setzen
Response „Bereits überboten"
Polling Request
Response „Client B = 110"
Zuschlag erteilen
Cache auf „verkauft" setzen
Artikel verkauft
Zuschlag
Response „verkauft an B"
Cache auf „verkauft an B" setzen
Response „verkauft an B"
Benachrichtigung „verkauft an B"
Cache auf „verkauft an B" setzen
t

FIG. 16A

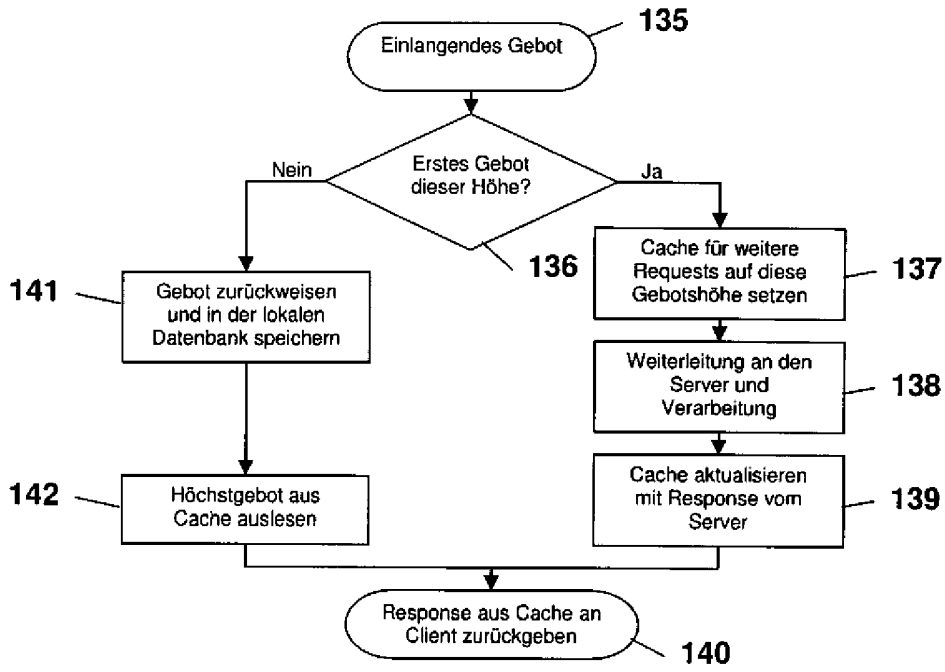


FIG. 16B

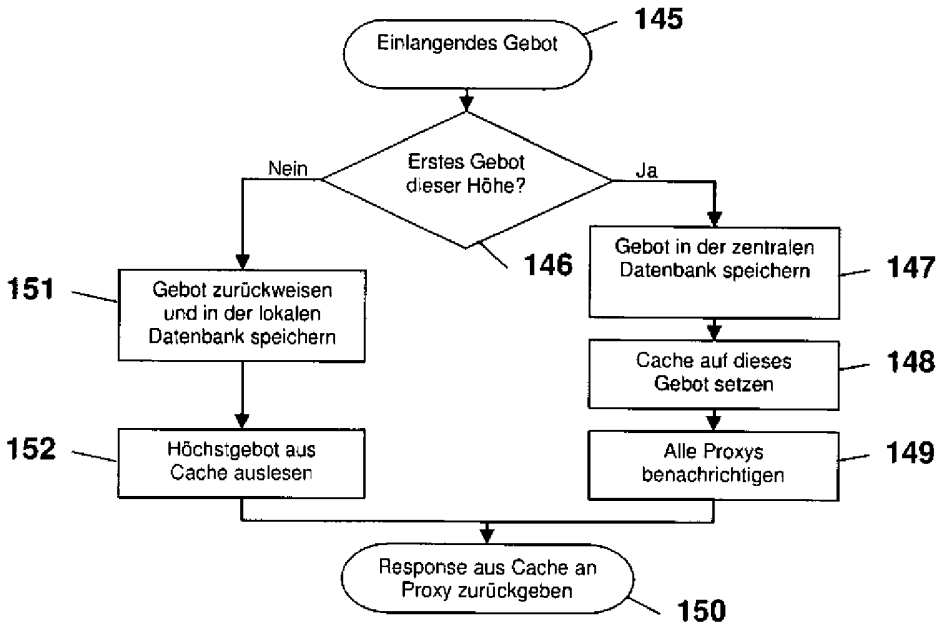


FIG. 16C

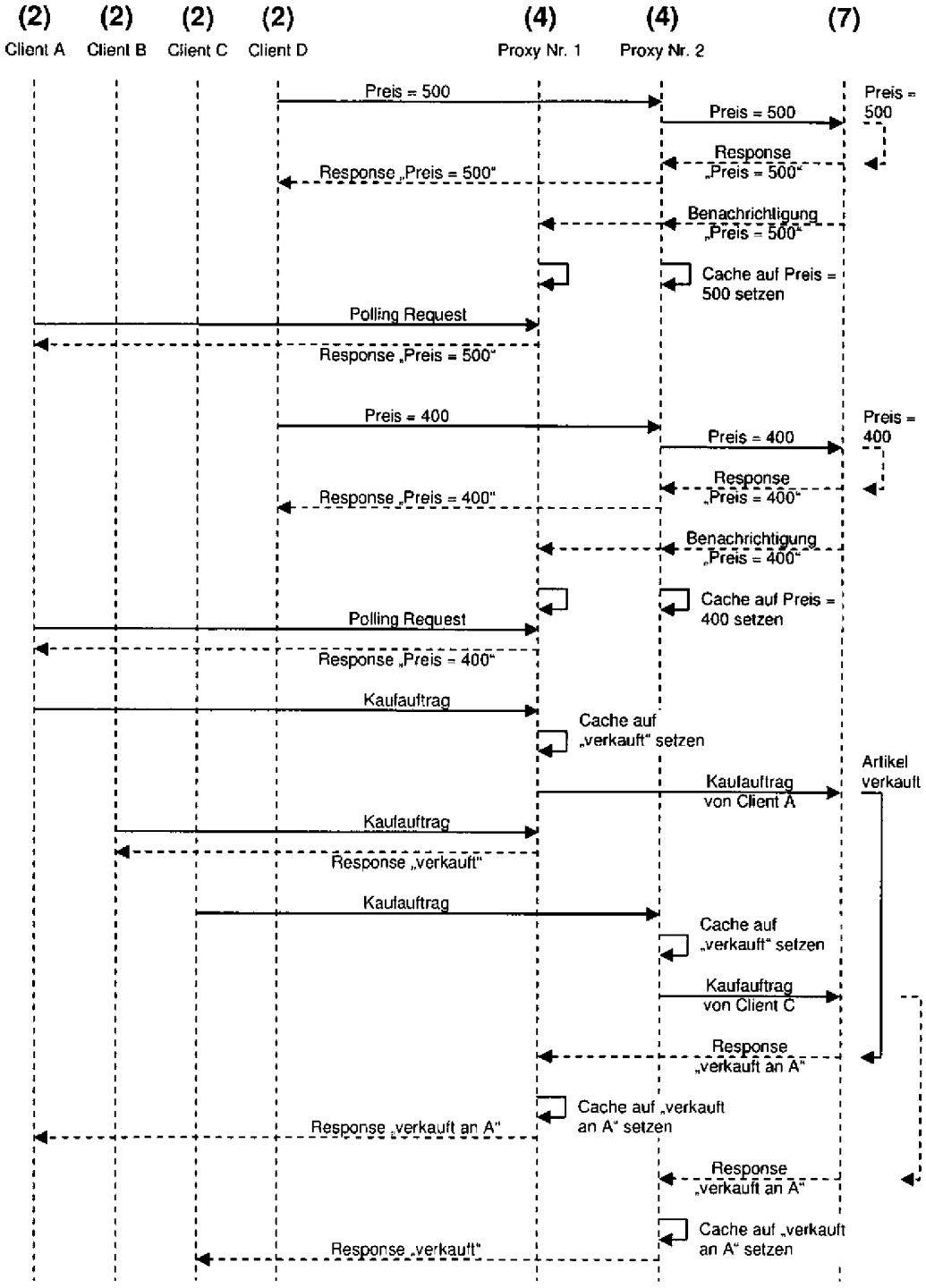

(2)
Client A
(2)
Client B
(2)
Client C
(2)
Client D
(4)
Proxy Nr. 1
(4)
Proxy Nr. 2
(7)
Preis = 500
Preis = 500
Preis = 500
Response „Preis = 500"
Response „Preis = 500"
Benachrichtigung „Preis = 500"
Cache auf Preis = 500 setzen
Polling Request
Response „Preis = 500"
Preis = 400
Preis = 400
Preis = 400
Response „Preis = 400"
Response „Preis = 400"
Benachrichtigung „Preis = 400"
Cache auf Preis = 400 setzen
Polling Request
Response „Preis = 400"
Kaufauftrag
Cache auf „verkauft" setzen
Artikel verkauft
Kaufauftrag von Client A
Kaufauftrag
Response „verkauft"
Kaufauftrag
Cache auf „verkauft" setzen
Kaufauftrag von Client C
Response „verkauft an A"
Cache auf „verkauft an A" setzen
Response „verkauft an A"
Response „verkauft an A"
Cache auf „verkauft an A" setzen
Response „verkauft"

FIG. 17A

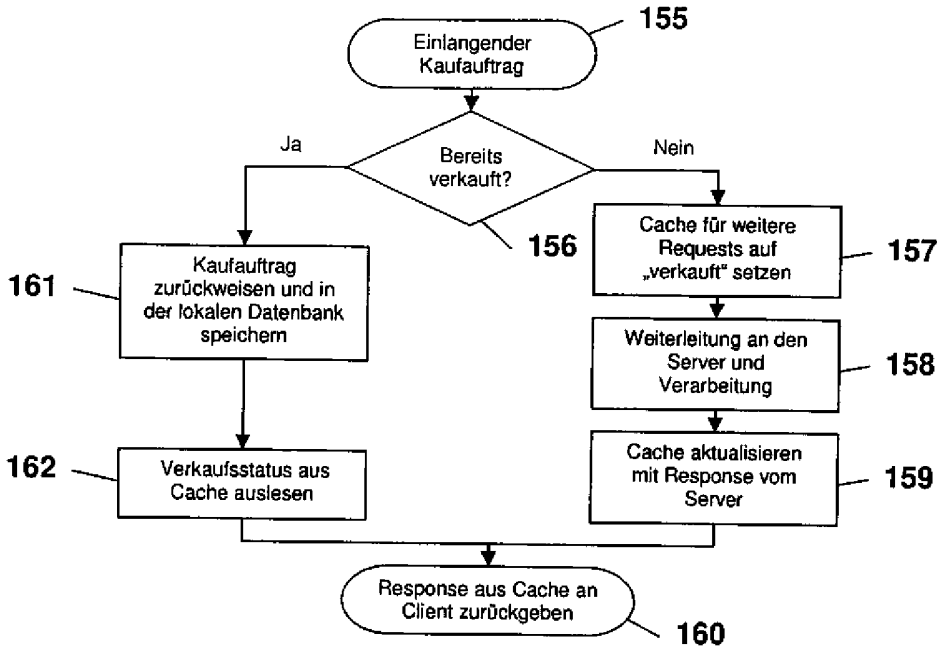


FIG. 17B

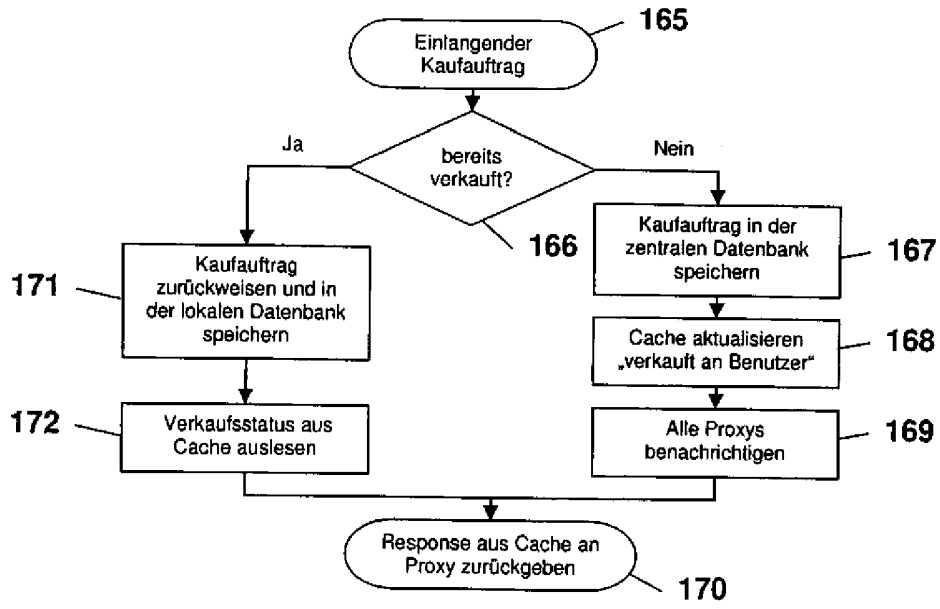


FIG. 17C

Patentansprüche:

1. Rechnersystem (1) zum Austausch von Nachrichten über Internet, für die Online-Abwicklung von Handels-Transaktionen, mit einer Mehrzahl von Client-Rechnern (2) mit Internet-Schnittstellen (3), mit zumindest einem zentralen Leit-Server (7), der mit einer zentralen Datenbank (8) verbunden ist, und mit eine Filterfunktion aufweisenden Verteilstellen zwischen den Client-Rechnern (2) und dem zumindest einen Leit-Server (7) dadurch gekennzeichnet, dass als Verteilstellen zwischen den Client-Rechnern (2) und dem zumindest einen zentralen Leit-Server (7) eine Mehrzahl von Proxy-Rechnern (4) vorgesehen ist, denen zumindest ein Load-Balancer-Modul (5), das zur Verteilung von Nachrichten auf vorgegebene Proxy-Rechner (4) eingerichtet ist, vorgeschaltet ist, und die je ein Relevanz-Filtermodul (11) aufweisen, das eingerichtet ist, einlangende, von Client-Rechnern (2) kommende Nachrichten auf Relevanz, nach vorgegebenen Kriterien, zu prüfen und nur relevante Nachrichten weiterzuleiten, und dass die Kommunikation zwischen den Client-Rechnern (2) und den Proxy-Rechnern (4) auf dem HTTP-Protokoll basiert.

2. Rechnersystem nach Anspruch 1, dadurch gekennzeichnet, dass zumindest ein Proxy-Rechner (4A) in Kaskade mit vorgeschalteten Proxy-Rechnern (4) angeordnet ist.

3. Rechnersystem nach Anspruch 2, dadurch gekennzeichnet, dass der Kaskade-Proxy-Rechner (4A) ebenfalls ein Relevanz-Filtermodul (11) aufweist, das relevante, von den vorgeschalteten Proxy-Rechnern (4) einlangende Nachrichten weiterleitet.

4. Rechnersystem nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, dass auch der zentrale Leit-Server (7) ein Relevanz-Filtermodul (11) aufweist, das relevante, von Proxy-Rechnern (4) einlangende Nachrichten zur weiteren Verarbeitung erfasst.

5. Rechnersystem nach Anspruch 4, dadurch gekennzeichnet, dass dem zentralen Leit-Server (7) weiters eine lokale Datenbank (13) zur zumindest vorübergehenden Speicherung von als nicht-relevant

erkannten Nachrichten zugeordnet ist.

6. Rechnersystem nach einem der Ansprüche 1 bis 5, dadurch gekennzeichnet, dass zumindest einem der Proxy-Rechner (4) eine lokale Datenbank (13) zur zumindest vorübergehenden Speicherung von als nicht-relevant erkannten Nachrichten zugeordnet ist.

7. Rechnersystem nach Anspruch 5 oder 6, dadurch gekennzeichnet, dass eine Systemlast-Prüfeinheit (60) vorgesehen ist, die konfiguriert ist, um zu Zeiten verminderter Last im Rechnersystem die Übertragung von in der oder den lokalen Datenbank(en) (13) gespeicherten nicht-relevanten Nachrichten zur zentralen Datenbank (8), zur Datenkonsolidierung, zu veranlassen.

8. Rechnersystem nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, dass die Client-Rechner (2) zur zyklischen Anforderung von für sie bestimmten Nachrichten von den zugehörigen Proxy-Rechnern (4) in vorgegebenen Polling-Intervallen eingerichtet sind.

9. Rechnersystem nach einem der Ansprüche 1 bis 8, dadurch gekennzeichnet, dass die Client-Rechner (2) eingerichtet sind, Nachrichten zu den jeweiligen Proxy-Rechnern (4) unmittelbar, außerhalb von vorgegebenen Polling-Intervallen, zu übertragen.

10. Rechnersystem nach einem der Ansprüche 1 bis 9, dadurch gekennzeichnet, dass die Client- (2) und Proxy-Rechner (4) eingerichtet sind, die zwischen ihnen übertragenen Nachrichten mit Zeitstempeln (30; 30') zu versehen, und die Proxy-Rechner (4) eingerichtet sind, immer nur Nachrichten mit einem jüngeren Zeitstempel als jenem des Client-Rechner (2) an den Client-Rechner (2) zu versenden.

AW/pr/clu

Recherchenbericht zu **A 2001/2009**
Technische Abteilung 3A

österreichisches patentamt

Klassifikation des Anmeldungsgegenstands gemäß IPC[8]: ***H04L 12/18*** (2006.01); ***H04L 12/58*** (2006.01); ***G06Q 30/00*** (2006.01)
Klassifikation des Anmeldungsgegenstands gemäß ECLA: H04L 12/18C; H04L 12/58G; G06Q 30/00C4
Recherchierter Prüfstoff (Klassifikation): G06F, G06Q, H04L
Konsultierte Online-Datenbank: EPODOC, WPI, TXTDE, TXTEN
Dieser Recherchenbericht wurde zu den am **17. Dezember 2009** eingereichten Ansprüchen **1-10** erstellt.

Kategorie*)	Bezeichnung der Veröffentlichung: Ländercode, Veröffentlichungsnummer, Dokumentart (Anmelder), Veröffentlichungsdatum, Textstelle oder Figur soweit erforderlich	Betreffend Anspruch
X Y	US 2007/0214074 A1 (FRIEDLAND ET AL.) 13. September 2007 (13.09.2007) *Zusammenfassung; Figuren 3-4, 8, 10-11, 14-15, 17; Beschreibung der Figuren; Ansprüche 1-3, 6-7, 12-13;* *(D1 in Zusammenschau mit D2)* --	1-4, 9 5-8, 10
Y A	US 2009/0063360 A1 (CALLAWAY ET AL.) 5. März 2009 (05.03.2009) *Zusammenfassung; Figuren 1-5; Beschreibung der Figuren; Ansprüche 1, 3-8, 10-13;* *(D2 in Zusammenschau mit D1)* --	5-8, 10 1-4, 9
A	WO 2007/107727 A2 (BROCA COMMUNICATIONS LIMITED) 27. September 2007 (27.09.2007) *Zusammenfassung; Figur 2; Beschreibung der Figur; Ansprüche 1, 3-4, 7-9, 14;* ----	1-10

Datum der Beendigung der Recherche:	☐ Fortsetzung siehe Folgeblatt	Prüfer(in):
4. November 2010		Mag. STOLL

*) **Kategorien** der angeführten Dokumente:

- **X** Veröffentlichung **von besonderer Bedeutung**: der Anmeldungsgegenstand kann allein aufgrund dieser Druckschrift nicht als neu bzw. auf erfinderischer Tätigkeit beruhend betrachtet werden.
- **Y** Veröffentlichung **von Bedeutung**: der Anmeldungsgegenstand kann nicht als auf erfinderischer Tätigkeit beruhend betrachtet werden, wenn die Veröffentlichung mit einer oder mehreren weiteren Veröffentlichungen dieser Kategorie in Verbindung gebracht wird und diese **Verbindung für einen Fachmann naheliegend** ist.
- **A** Veröffentlichung, die den **allgemeinen Stand der Technik** definiert.
- **P** Dokument, das **von Bedeutung** ist (Kategorien **X** oder **Y**), jedoch **nach dem Prioritätstag** der Anmeldung **veröffentlicht** wurde.
- **E** Dokument, das **von besonderer Bedeutung** ist (Kategorie **X**), aus dem ein **älteres Recht** hervorgehen könnte (früheres Anmeldedatum, jedoch nachveröffentlicht, Schutz ist in Österreich möglich, würde Neuheit in Frage stellen).
- **&** Veröffentlichung, die Mitglied der selben **Patentfamilie** ist.