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

(10) **DE 603 14 106 T2** 2008.01.24

(12) # Übersetzung der europäischen Patentschrift

(97) **EP 1 488 644 B1**
(21) Deutsches Aktenzeichen: **603 14 106.4**
(86) PCT-Aktenzeichen: **PCT/GB03/01090**
(96) Europäisches Aktenzeichen: **03 712 352.8**
(87) PCT-Veröffentlichungs-Nr.: **WO 2003/084233**
(86) PCT-Anmeldetag: **14.03.2003**
(87) Veröffentlichungstag der PCT-Anmeldung: **09.10.2003**
(97) Erstveröffentlichung durch das EPA: **22.12.2004**
(97) Veröffentlichungstag der Patenterteilung beim EPA: **30.05.2007**
(47) Veröffentlichungstag im Patentblatt: **24.01.2008**

(51) Int Cl.[8]: ***H04N 7/24*** (2006.01)
H04N 7/50 (2006.01)

(30) Unionspriorität:
02252214 27.03.2002 EP

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

(73) Patentinhaber:
British Telecommunications Public Ltd. Co., London, GB

(74) Vertreter:
BEETZ & PARTNER Patentanwälte, 80538 München

(72) Erfinder:
JEBB, Timothy Ralph, Ipswich, Suffolk IP5 2YS, GB; NILSSON, Michael Erling, Ipswich, Suffolk IP5 1BY, GB

(54) Bezeichnung: **DATENSTRUKTUR FÜR EIN DATENÜBERTRAGUNGSSYSTEM**

Beschreibung

[0001] Die vorliegende Erfindung betrifft eine Datenstruktur, die geeignet ist zum Speichern von Inhalt, wie Audio- und Videoinhalt, der über IP(Internetprotokoll)-Netzwerke zu strömen (stream) ist. Insbesondere ist die vorliegende Erfindung geeignet zur Verwendung mit einem System, in dem die verfügbare Bitrate inhärent variabel ist aufgrund von physikalischen Netzwerkcharakteristiken und/oder Konkurrenz mit anderem Verkehr. Zum Beispiel ist die vorliegende Erfindung geeignet für ein Multimedia-Streaming zu mobilen handgehaltenen bzw. tragbaren Terminals, wie PDAs (Personal Digital Assistants), via GPRS (General Packet Radio Service) oder 3G-Netzwerken.

[0002] Neue Datennetzwerkzugriffstechnologien, wie Kabel und ADSL(Asymmetric Digital Subscriber Line)-Modems, zusammen mit Fortschritten bei der Komprimierung und der Verfügbarkeit von freier Client-Software treiben das Wachstum von Video-Streaming über das Internet an. Die Verwendung dieser Technologie wächst exponentiell, möglicherweise erfolgt eine Verdopplung der Größe alle sechs Monate, wobei eine geschätzte halbe Million Streams in 2000 bedient werden. Jedoch ist ein Benutzerempfang eines Internet-Streamings weiterhin getrübt durch Erfahrungen einer Überfüllung und langen Startverzögerungen.

[0003] Aktuelle IP-Netzwerke sind nicht gut geeignet für das Streaming von Videoinhalt, da sie einen Paketverlust, Verzögerung und Jitter (Verzögerungsvariation) zeigen, sowie einen variablen erzielbaren Durchsatz, all dies kann vom Vergnügen für den Endbenutzer des Multimedia-Inhalts ablenken.

[0004] Echtzeit(real time)-Videoanwendungen erfordern, dass alle Pakete auf rechtzeitige Weise ankommen. Wenn Pakete verloren gehen, dann ist die Synchronisierung zwischen dem Codierer und dem Decodierer unterbrochen und Fehler breiten sich durch das wiedergegebene Video für einige Zeit aus. Wenn Pakete übermäßig verzögert sind, werden sie für den Decodierer nutzlos, der in Echtzeit arbeiten muss, und werden als Verlust behandelt. Ein Paketverlust und sein visueller Effekt auf das wiedergegebene Video sind insbesondere signifikant in prädiktiven Video-Codiersystemen, wie H.263. Der Effekt eines Paketverlusts kann reduziert werden, aber nicht eliminiert, durch Einführung eines Fehlerschutzes in den Videostrom. Es wurde festgestellt, dass derartige Ausfallsicherheitstechniken den Effekt eines Paketverlusts nur minimieren, aber nicht eliminieren können.

[0005] In dem Fall eines erlittenen Paketverlusts, der einen Langzeit-Verlust des Durchsatzes anzeigt, muss das Streaming-System seine Langzeit-Anforderungen reduzieren können. Dies bedeutet im Allgemeinen, dass die Bitrate der gestreamten Media reduziert werden muss.

[0006] Standardmäßige Komprimierungstechnologien, wie H.263 und MPEG-4, können verwaltet werden, eine Multimedia-Quelle vorzusehen, die ihre Codierrate dynamisch ändern kann. Eine Videoquelle mit derartigen Eigenschaften wird hier als eine elastische Quelle beschrieben, d.h. eine, die sich an Langzeit-Variationen des Netzwerkdurchsatzes anpassen kann. Dies wird im Allgemeinen erreicht durch Vorsehen einer kontinuierlich adaptiven Video-Bitrate. Dies ist möglich, da, anders als Audio-Codecs, Video-Komprimierungs-Standards nicht eine absolute Betriebs-Bitrate spezifizieren.

[0007] Video-Streaming-Systeme können gestaltet werden, einen codierten Strom mit variierender Bitrate vorzusehen, wobei sich die Bitrate, als Antwort auf ein Client-Feedback, sofort an die verfügbare Netzwerkbandbreite anpasst. Ein derartiges System kann Netzwerk-freundlich ausgebildet werden durch Steuern der Übertragungsrate derart, dass sie sich in dem Fall eines Paketverlusts schnell reduziert und zu anderen Zeiten langsam zunimmt.

[0008] Jedoch ist diese Lösung aus zwei Gründen nicht praktisch. Erstens erfordert eine Echtzeit-Video-Codierung eine große Menge an Verarbeitungsleistung, wodurch eine derartige Lösung nicht skaliert, um viele Benutzer zu unterstützen. Zweitens ist die Wahrnehmung des Endbenutzers der Gesamtqualität gegenteilig beeinflusst durch schnelle Variationen der momentanen Qualität.

[0009] Für Streaming-Anwendungen in einer Richtung (unidirektional) ist die Verzögerung zwischen dem Sender und dem Empfänger nur am Beginn wahrnehmbar. Deswegen tauschen bekannte Techniken Verzögerung gegen Paketverlust und Jitter. Vorausgesetzt, die durchschnittlichen Durchsatz-Anforderungen des Video-Streams stimmen mit der durchschnittlichen verfügbaren Bandbreite überein, dann kann die Empfänger-Puffergröße dimensioniert werden, um die erwartete Variation bei der Verzögerung zu enthalten.

[0010] Von marktführenden Streaming-Systemen wird angenommen, dass sie eine signifikante Pufferung auf der Client-Seite verwenden, um die Effekte von Jitter zu reduzieren, die im Internet auftreten.

[0011] Die Verwendung eines Puffers wie oben beschrieben ermöglicht einem System, einen Paketverlust und Jitter zu überwinden. Jedoch wird nicht das Problem gelöst, dass es eine unzureichende Bitrate gibt, die von dem Netzwerk verfügbar ist. Wenn die durchschnittlichen Langzeit-Bitrate-Anforderungen des Videomaterials die durchschnittliche Bitrate über-

steigt, die von dem Netzwerk verfügbar ist, ist der Client-Puffer schließlich leer und der Video-Renderer stoppt, bis der Puffer erneut gefüllt ist. Der Grad der Nichtübereinstimmung zwischen der verfügbaren Netzwerkbitrate und der Rate, mit welcher der Inhalt codiert wurde, bestimmt die Frequenz einer Pause, um den Puffer erneut zu füllen.

[0012] Wie oben beschrieben, können die meisten Video-Komprimierungsalgorithmen, einschließlich H.263 und MPEG-4, implementiert werden, um eine kontinuierlich adaptive Bitrate zu liefern. Sobald jedoch Video und Audio komprimiert wurden, werden sie unelastisch und müssen mit der codierten Bitrate übertragen werden.

[0013] Während Netzwerk-Jitter und Kurzzeit-Variationen in dem Netzwerk-Durchsatz absorbiert werden können durch Betreiben eines Puffers an dem Empfänger, wird eine Elastizität nur erreicht, wenn Langzeit-Variationen in dem Netzwerk-Durchsatz ebenfalls absorbiert werden können.

[0014] Eine Codierung in Schichten ist eine weithin bekannte Technik zum Erzeugen von elastischen Videoquellen, wie zum Beispiel offenbart wird in dem U.S.-Patent 6,014,694, das auch die Speicherung von Schichten in einer Datei beschreibt. Eine Videokomprimierung in Schichten verwendet ein hierarchisches Codierungsschema, in dem die Qualität an dem Empfänger verbessert wird durch den Empfang und die Decodierung von höheren Schichten, die sequentiell zu der Basis-Repräsentation hinzugefügt werden. Zu jeder Zeit kann jeder Client jede Anzahl dieser Video-Schichten empfangen, abhängig von ihrer aktuellen Netzwerk-Konnektivität zu der Quelle. In ihrer einfachsten Implementierung liefert dies eine grobkörnige Adaption an die Netzwerkbedingungen, was vorteilhaft ist in Multicast-Szenarien. Eine Videokomprimierung in Schichten wurde ebenso kombiniert mit einer Pufferung an dem Client, um eine feinkörnige Adaption zu Netzwerkbedingungen hinzuzufügen. Es wurde jedoch gezeigt, dass Codiertechniken in Schichten nicht effizient sind und typischerweise signifikant mehr Verarbeitung an dem Client erfordern, was bestimmte Probleme verursacht bei der Handhabung von mobilen Vorrichtungen, die wahrscheinlich eine reduzierte Verarbeitungskapazität haben.

[0015] In der U.S.-Patentanmeldung 2002/002708A1 erzeugt ein Bandbreiten-Skalierer mehrere unabhängige Streams mit unterschiedlichen Bitraten.

[0016] Die Erfindung versucht, eine geeignete Speicherung von mehreren codierten Strömen vorzusehen.

[0017] Gemäß einem Aspekt der vorliegenden Erfindung ist vorgesehen eine Datenstruktur zum Speichern einer Datenquelle für ein Streaming-System, wobei die Datenquelle eine Vielzahl von codierten Datenströmen umfasst, wobei zumindest einige der Vielzahl von Datenströmen unabhängige Repräsentationen von Daten von der Datenquelle sind, die mit anderen Auflösungen als andere der Vielzahl von Datenströmen codiert sind, wobei die Datenstruktur einen Header, eine Strom-Datenstruktur für jeden der codierten Datenströme und ein oder mehrere Paket(e) der codierten Datenströme aufweist, wobei der Header mit einer der Strom-Datenstrukturen verbunden ist, wobei jede Strom-Datenstruktur einen Header, eine Verbindung (link) zu einer nächsten Strom-Datenstruktur und eine Verbindung zu einem ersten Paket des codierten Datenstroms umfasst.

[0018] Ein geeignetes System und ein Verfahren zur Verwendung der Datenstruktur werden detailliert im Folgenden beschrieben. Die Komplexität der Datenstruktur ist eine Konsequenz von Paketen von potentiell vielen Strömen, die verschachtelt sind, und der Notwendigkeit, eine Umschaltung und Wiederherstellung zu unterstützen. Eine Navigation von Paket zu Paket ist notwendigerweise durch Zeiger, da im Allgemeinen Pakete, die in einem Strom aufeinander folgend sind, in der Datei nicht angrenzend aneinander gespeichert werden. Ein Schreiben von Umschaltungs- und Wiederherstellungspaketen erfordert, dass präzise Details von Quell- und Ziel-Paketen aufgezeichnet werden. Ein Umschalten zwischen Strömen während einer Wiedergabe (Playback) erfordert erstens die Identifizierung des nächsten verfügbaren Umschaltpakets, gefolgt von der Wiedergabe der verbleibenden Pakete von dem „von"-Strom, Wiedergabe der Umschaltpakete, dann die Wiedergabe von Paketen von dem „an"-Strom von dem geeigneten Punkt. Ferner ist vorzuziehen, dass es keine nennenswerte Verzögerung gibt bei der Umschaltung zwischen Strömen.

[0019] Vorzugsweise sind die Vielzahl von codierten Datenströmen Videodatenströme. Audiodaten können als ein Datenstrom codiert werden.

[0020] Die Strom-Datenstrukturen für Video- und Audio-Datenströme können Bitraten-Codier-Daten für die jeweiligen Datenströme umfassen.

[0021] Die Datenquelle kann weiter aufweisen einen Schaltstrom, der eine Vielzahl von Umschaltpunkten definiert zum Umschalten zwischen einem der Videodatenströme und einem anderen der Videodatenströme, wobei die Datenstromstruktur für den Schaltdatenstrom Daten von Videoströmen und Pakete umfasst, an die und von denen ein Umschalten möglich ist.

[0022] Der Header der Datenstruktur kann eine Verbindung (link) zu der letzten Strom-Datenstruktur um-

fassen. Der Header einer Strom-Datenstruktur kann eine Verbindung zu dem letzten Paket des codierten Datenstroms umfassen.

[0023] In dem beschriebenen System muss ein erzeugter Audio-visueller Strom nicht mit einer einzelnen festen Bitrate übertragen werden, somit muss die Datenstruktur dies unterstützen und eine Übertragung ermöglichen mit welcher Rate das Netzwerk dies augenblicklich unterstützt.

[0024] Von dem System und der Datenstruktur wurde gezeigt, dass sie gut über ein GPRS-Netzwerk arbeiten, und eine gute Ausnutzung der verfügbaren Netzwerkbandbreite liefern, um eine zufriedenstellende Multimedia-Qualität vorzusehen.

[0025] Das System und die Datenstruktur wurden gestaltet, um die Charakteristiken von IP-Netzwerken und insbesondere mobilen IP-Netzwerken zu bewältigen, um Benutzer mit Multimedia mit konsistenter Qualität mit einer minimalen Anfangsverzögerung zu versehen.

[0026] Ein Beispiel der vorliegenden Erfindung wird nun detailliert beschrieben unter Bezugnahme auf die beigefügten Zeichnungen, wobei:

[0027] Fig. 1 ein schematisches Diagramm eines Audio-visuellen Datenstromsystems ist zur Verwendung mit der vorliegenden Erfindung;

[0028] Fig. 2 ein schematisches Diagramm einer Video-Codier-Hierarchie ist, die in dem System von Fig. 1 verwendet wird;

[0029] Fig. 3 ein schematisches Diagramm einer Video-Codier-Architektur ist, die ermöglicht, dass ein Umschalten zwischen Videoströmen ohne Nicht-Übereinstimmung erreicht wird;

[0030] Fig. 4 ein schematisches Diagramm einer Client-Server-Architektur ist, die geeignet ist zur Verwendung in dem System von Fig. 1;

[0031] Fig. 5a und Fig. 5b jeweils Diagramme sind, die eine standardmäßige TKPT-Transportpaketstruktur und eine Variation dieser Struktur darstellen, implementiert für das System der Fig. 1; und

[0032] Fig. 6a-Fig. 6c schematische Diagramme sind, die Aspekte einer Datenstruktur darstellen, die einen Audio-visuellen Datenstrom gemäß einem Ausführungsbeispiel der vorliegenden Erfindung aufweist.

[0033] Fig. 1 ist ein schematisches Diagramm eines Audio-visuellen Datenstromsystems zur Verwendung mit einem Ausführungsbeispiel der vorliegenden Erfindung.

[0034] Der Server **10** empfängt einen codierten Multimedia-Inhalt entweder direkt von einem Codierer **20** oder von einer Datei **30** und liefert diesen Inhalt an einen oder mehrere Client(s) **40**–**60**. Der Server **10** skaliert, um viele Clients **40**–**60** zu unterstützen, die auf viele Teile des Inhalts unabhängig zugreifen, da er wenig Verarbeitung durchführt und nur Pakete für eine weitere Übertragung auswählt. In dem Server **10** wird keine Codierung oder Transcodierung von Media durchgeführt.

[0035] Im Prinzip arbeitet der Server **10** auf dieselbe Weise für beide Live-Ströme, von dem Codierer **20** geliefert, und für vor-codierte Ströme von der Datei **30**. In diesem bestimmten Ausführungsbeispiel wird ein Streaming von Live-Media beschrieben. Unterschiede des Strömens von Media aus vor-codierten Dateien werden in späteren Ausführungsbeispielen diskutiert.

[0036] Der Server **10** umfasst eine Anzahl von kreisförmigen Puffern **70**–**90**. Für jeden Client **40**–**60** gibt es eine Instanz eines Paket-Senders **100**. Der Paketsender **100** bestimmt, wann und von welchem Puffer **70**–**90** das nächste Paket gelesen wird, liest das gewählte Paket und sendet es an den jeweiligen Client über eine Netzwerkverbindung **110**.

[0037] Eine semi-zuverlässige Netzwerkverbindung **110** ist erforderlich von dem Server **10** zu jedem jeweiligen Client **40**–**60**, um sicherzustellen, dass fast alle gesendeten Pakete empfangen werden, wodurch Störungen der von dem Benutzer wahrgenommenen Qualität minimiert werden. Die Puffer (**120**, **130**) werden somit an den jeweiligen Enden der Netzwerkverbindung **110** verwendet, um erneute Übertragungen von verlorenen Paketen zu ermöglichen. Die Netzwerkverbindung **110** soll auch Netzwerk-freundlich sein, d.h. zu ermöglichen, dass die verwendete Bitrate erhöht wird, wenn keine Überlastung erfahren wird, und drastisch reduziert wird, wenn eine Überlastung auftritt.

[0038] Während die Systemkomponenten gezeigt und beschrieben werden als eine Kombination von integrierten und getrennten Komponenten, sollte angemerkt werden, dass andere Konfigurationen verwendet werden können. Zum Beispiel kann ein externer Codierer **20** und/oder Dateispeicher **30** verwendet werden. Genauso sind die Puffer **130** wahrscheinlich integral zu den Client-Vorrichtungen **40**–**60**.

[0039] Fig. 2 ist ein schematisches Diagramm einer Video-Codier-Hierarchie, die in dem System von Fig. 1 verwendet wird. Der Codierer **20** codiert einen Live- oder gespeicherten Multimedia-Inhalt in einer elastischen codierten Repräsentation. Audio wird mit niedriger Bitrate in einen einzelnen codierten Bitstrom codiert und ist somit an sich unelastisch. Je-

doch kann, da Audio typischerweise eine geringere Bitrate erfordert als Video, vorausgesetzt, das Video ist auf eine elastische Art codiert, die kombinierte Codierung von Audio und Video als elastisch betrachtet werden.

[0040] Audio wird codiert unter Verwendung des AMR(Adaptive Multi-Rate)-Codierers bei 4.8 kbit/s. Video wird codiert in eine elastische Repräsentation. Auf eine ähnliche Weise wie eine Schichtenbildung (layering) erzeugt der Codierer **20** eine Hierarchie von unabhängigen Videoströmen. Statt diese Hierarchie zu bilden, indem jeder Strom abhängig ist von allen Strömen, die eine niedrigere Hierarchie haben, wird jeder Strom unabhängig codiert. Eine derartige Hierarchie ist weithin bekannt und wird als „simulcast" bezeichnet.

[0041] Obwohl Audiodaten beschrieben wurde als unter Verwendung eines AMR-Schemas mit niedriger Bitrate codiert, können auch andere AMR-Codierraten und andere Codierstandards, wie MP3, unterstützt werden. Codiertes Audio mit verschiedenen Raten kann organisiert werden in eine Hierarchie von unabhängigen Strömen auf eine ähnliche Weise, wie unten für Video beschrieben wird, aber mit der Vereinfachung einer Umschaltung zwischen codierten Repräsentationen aus der Tatsache, dass jeder Audiorahmen typischerweise unabhängig codiert wird.

[0042] Die Videohierarchie, die unter Verwendung einer Erweiterung des ITU-T-Standards H.263 erzeugt wird, umfasst einen Intra-Strom **200**, um einen zufälligen Zugriff zu Videoströmen zu ermöglichen, und einen oder mehrere Abspiel-Strom/Ströme **210a**, **210b** zum gewöhnlichen Betrachten des Inhalts. Jeder Abspiel-Strom **210a**, **210b** ist mit einer anderen Bitrate codiert, wodurch ein gegebener Client **40**–**60** mit einer Rate empfangen kann, die für seine aktuelle Netzwerkverbindung **110** zu dem Server **10** geeignet ist. Die Hierarchie enthält auch Schaltströme **220**, **230**, **240**, die ein Umschalten von dem Intra-Strom **200** zu dem Abspiel-Strom **210a** mit niedrigster Rate und zwischen Abspielströmen ermöglichen.

[0043] Da die Codieralgorithmen eine Bewegungs-kompensierte Voraussage bzw. Prädiktion einsetzen, würde ein Umschalten zwischen Bitströmen an arbiträren Punkten in einem Abspiel-Strom, obwohl möglich, zu visuellen Artifakten führen aufgrund der Nicht-Übereinstimmung zwischen den rekonstruierten Rahmen in demselben Zeitmoment von unterschiedlichen Bitströmen. Die visuellen Artifakte breiten sich mit der Zeit weiter aus.

[0044] In momentanen Video-Codierstandards ist ein perfektes (Nicht-Übereinstimmungs-freies) Umschalten zwischen Bitströmen möglich nur an den Positionen, an denen die zukünftigen Rahmen/Bereiche keine Information vor der aktuellen Umschaltposition verwenden, d.h. bei Zugriffsbildern. Ferner werden durch Platzieren von Zugriffsbildern in festen (z.B. 1 sek) Intervallen VCR-Funktionalitäten, wie wahlfreier Zugriff oder „schnelles Vorwärtsspielen (Fast Forward)" oder „schnelles Zurückspielen (Fast Backward)" (erhöhte Abspielrate) für einen strömenden Videoinhalt erreicht. Ein Benutzer kann einen Teil des Videos überspringen und erneut ein Abspielen an einer Zugriffsbild-Position beginnen. Ähnlich kann eine erhöhte Abspielrate, d.h. schnelles Vorwärtsspielen (fast-forwarding), erreicht werden durch Übertragen nur von Zugriffsbildern.

[0045] Es ist jedoch weithin bekannt, dass Zugriffsbilder mehr Bit erfordern als die Bewegungs-kompensierten vorhergesagten (prädiktiven) Rahmen. Somit werden der Intra-Strom **200** und die Schaltströme **220**, **230**, **240** verwendet. Die Haupteigenschaft von Schaltströmen ist, dass identische Bilder erlangt werden können, auch wenn unterschiedliche Referenzrahmen verwendet werden.

[0046] Der Hauptzweck der Hierarchie ist, dem Server **10** zu ermöglichen, einen Abspiel-Strom **210a** oder **210b** an einen Client **40**–**60** zu senden, um eine optimale Balance vorzusehen zwischen dem Aufbau eines Puffers von empfangenen Daten an dem Client **40**–**60**, um eine Ausfallsicherheit für Paketverlust und plötzlichen Ausfällen des Netzwerkdurchsatzes vorzusehen, und einem Vorsehen des besten Abspiel-Stroms **210a** oder **210b** für den Client **40**–**60** abhängig von der höchsten Bitrate, die seine Netzwerkverbindung **110** augenblicklich unterstützt.

[0047] Der Intra-Stream **200** ist eine Serie von Intra-codierten Bildern (**201**, **202**), die verwendet werden, um einen wahlfreien Zugriff und ein Wiederherstellen nach schwerwiegenden Fehlerbedingungen vorzusehen. Die Abspielströme **210a**, **210b** umfassen prädiktiv codierte Bilder (**211a**, **212a**, **213a**, **214a**, **215a**; **211b**, **212b**, **213b**, **214b**, **215b**), die bidirektional vorausgesagt sein können und vorausgesagt werden können aus mehreren Referenzbildern. Die Abspielströme **210a**, **210b** umfassen auch periodische Zugriffsbilder **216a**, **217a**; **216b**, **217b**. Die Schaltströme **220**, **230**, **240** bestehen aus einer Serie von Verbindungsbildern (**221**, **222**; **231**, **232**; **241**, **242**).

[0048] Die kreisförmigen Puffer **70**–**92** sind jedem Stromtyp zugewiesen, einer für jeden Intra-(**70**), Abspiel-(**80**, **85**) und Schalt-(**90**, **91**, **92**) Strom für jedes Stück Inhalt.

[0049] Wenn ein Client **40** das erste Mal mit dem Server **10** verbindet, lokalisiert der Server **10** ein geeignetes Intrabild (zum Beispiel das Intrabild **201**) aus dem kreisförmigen Puffer **70**, der den Intra-Strom speichert, und sendet dieses an den Client **40**. Der Server **10** wählt dann das Verbindungsbild (**221**), um

von Intrastrom **220** zu dem Abspiel-Strom **210a** mit der niedrigsten Codier-Bitrate zu schalten, und dann weiter von diesem Abspiel-Strom (**213a** und weiter) zu beliefern.

[0050] Die Übertragung von Paketen an den Client **40** ist ein unabhängiger Prozess, wobei die Rate der Übertragung abhängig ist von dem Zustand des Netzwerks und dem verwendeten Übertragungsprotokoll. Jedoch ist die Absicht, dass anfangs die Übertragungsrate größer ist als die Codier-Bitrate des Abspiel-Stroms **210a** mit der niedrigsten Codier-Bitrate. Dies ermöglicht dem Client **40**, mit dem Decodieren zu beginnen und die Media dem Benutzer zu präsentieren unmittelbar an dem Punkt, an dem die Daten empfangen und decodiert werden, während es dem Client **40** auch ermöglicht, überschüssige komprimierte Mediadaten in seinem Decodierpuffer aufzubauen.

[0051] An dem Punkt, an dem ein Zugriffsbild (wie das Zugriffsbild **217a** in dem obigen Beispiel), kann der Client **40** und/oder der Server **10** bestimmen, dass ein anderer Abspiel-Strom geeigneter ist (zum Beispiel aufgrund einer erhöhten oder verringerten Netzwerkkapazität). In dem obigen Beispiel wird ein Umschalten von dem Abspiel-Strom **210a** mit niedriger Rate zu dem Abspiel-Strom **210b** mit höherer Rate erreicht, indem der Server **10** das Verbindungsbild **232** statt das Zugriffsbild **217a** überträgt. Das Verbindungsbild **232** verbindet zu dem Abspiel-Strombild **215b** des Abspiel-Stroms **210b** mit höherer Rate, was dem Client **40** ermöglicht, diesen Abspiel-Strom zu empfangen. Ein Umschalten zu einem Abspiel-Strom mit einer verringerten Bitrate wird auf ähnliche Weise erreicht.

[0052] Drei Verfahren zum Codieren von Verbindungsbildern wurden untersucht. Jedes Verfahren liefert andere Kompromisse zwischen der Akkumulierung von Abweichung bzw. Drift des Umschaltens, den Kosten im Hinblick auf die Bitrate der tatsächlichen Umschaltung und dem Einfluss auf die Qualität der einzelnen Abspielströme, verursacht durch Codieren regulärer Bilder eines Typs, der eine Abweichungs-freie Umschaltung mit geringer Bitrate ermöglicht.

1. Prädiktiv codierte Verbindungsbilder

[0053] In dem ersten Verfahren werden Verbindungsbilder als vorhergesagte Bilder erzeugt. Sie werden codiert auf eine Weise, dass, wenn sie rekonstruiert werden, sie ähnlich sind in dem Sinn, dass sie zum Beispiel eine geringe mittlere quadratische Differenz haben, zu der Rekonstruktion des simultanen Zugriffsbilds in dem Ziel-Abspiel-Strom. Zugriffsbilder können als vorhergesagte Bilder codiert werden. Die Anzahl von verwendeten Bits, um die Verbindungsbilder zu codieren, bestimmen, wie gut das rekonstruierte Verbindungsbild mit dem rekonstruierten Zugriffsbild übereinstimmt, und somit die Menge an Abweichung, die als ein Ergebnis des Umschaltens auftritt. Jedoch akkumuliert sich eine Abweichung bei jedem Auftreten eines Umschaltens.

2. Intra-codierte Verbindungsbilder

[0054] In dem zweiten Verfahren werden Verbindungsbilder als Intra-Bilder erzeugt. Sie werden codiert auf eine Weise, dass, wenn sie rekonstruiert werden, sie ähnlich sind in dem Sinn, dass sie zum Beispiel eine geringe mittlere quadratische Differenz haben, zu der Rekonstruktion des simultanen Zugriffsbilds in dem Ziel-Abspiel-Strom. Zugriffsbilder können als vorhergesagte Bilder codiert werden. Die Anzahl von verwendeten Bits, um die Verbindungsbilder zu codieren, bestimmen, wie gut das rekonstruierte Verbindungsbild mit dem rekonstruierten Zugriffsbild übereinstimmt, und somit die Menge an Abweichung, die als ein Ergebnis des Umschaltens auftritt. Für ein gegebenes Maß an Nicht-Übereinstimmung erfordert jedoch ein Intracodiertes Verbindungsbild normalerweise viel mehr Bits als ein prädiktiv codiertes Verbindungsbild. Die Verwendung von Intra-Codierung für Verbindungsbilder verhindert die Akkumulation von Abweichung.

3. Quantisierte-Quelle-codierte Verbindungsbilder

[0055] In dem dritten Verfahren werden Verbindungsbilder codiert mit einer Technik basierend auf dem Konzept, das beschrieben wird in „VCEG-L27, A proposal for SP-frames", eingereicht von Marta Karczewicz und Ragip Kurceren bei ITU-Telecommunications Standardization Sector Video Coding Experts Group's Twelfth Meeting: Eibsee, Germany, 9.-12. Januar 2001, verfügbar unter ftp://standard.pictel.com/video-site/, die hier als Quantisierte-Quelle-Bilder (Quantised-Source pictures) bezeichnet werden.

[0056] Die Codierungs-Architektur für Quantisierte-Quelle-Bilder wird in **Fig. 3** gezeigt. Das Quellen-Bild und die Bewegungs-kompensierte Prädiktion werden jeweils in den Schritten **300** und **310** mit demselben Quantisiererindex unabhängig quantisiert und transformiert, bevor sie in Schritt **320** subtrahiert werden und in Schritt **330** variable-Länge-codiert werden. Das rekonstruierte Bild wird gebildet durch Hinzufügen, in Schritt **340**, der Ausgabe des Subtrahierers **320** und der Ausgabe der Quantisierung und Transformation **310** und ein inverses Transformieren und inverses Quantisieren des Ergebnisses in Schritt **350**. Das rekonstruierte Bild wird in dem Bildspeicher **360** gespeichert. Das Ergebnis ist, dass das rekonstruierte Bild einfach das quantisierte Quellenbild ist und unabhängig ist von der Bewegungskompensierten Prädiktion. Somit kann ein gegebenes Quellen-Bild identisch rekonstruiert werden, wenn von un-

terschiedlichen Referenzbildern vorausgesagt, und somit ist ein Abweichungs-freies Umschalten möglich. Die Bewegung-kompensierte Prädiktion ist nicht irrelevant, da sie die Entropie des mittels variabler Länge zu codierenden Signals reduziert und somit die Anzahl von Bits reduziert, die durch Codieren eines Bilds erzeugt werden.

[0057] Zugriffsbilder werden auch als Quantisierte-Quelle-Bilder codiert, mit einer identischen Auswahl von Codier-Modi, intra oder inter, und Quantisierer-Auswahl, wie das Verbindungsbild. Dies stellt sicher, dass das Verbindungsbild identisch zu dem simultanen Zugriffsbild in dem Ziel-Abspiel-Strom rekonstruiert wird.

[0058] Die Anzahl der erforderlichen Bits, um das Verbindungsbild zu codieren, wird bestimmt durch die Codierung des entsprechenden Zugriffsbilds. Die Anzahl der verwendeten Bits, um das Zugriffsbild zu codieren, hängt davon ab, wie die Quantisierung durchgeführt wird, ist aber im Allgemeinen mehr als die Anzahl von verwendeten Bits, um prädiktive Bilder zu codieren, und weniger als die Anzahl von verwendeten Bits, um Intra-Bilder zu codieren. Dies ist aufgrund dessen, da eine Codierung effizienter ist als eine Intra-Codierung aufgrund der Verwendung von Prädiktion, aber nicht so effizient wie eine normale Prädiktion aufgrund der Quantisierung des Prädiktionsfehlers. Somit ermöglicht die Verwendung von Quantisierte-Quelle-Bilder ein Abweichungs-freies Umschalten, aber auf Kosten einer weniger effizienten Codierung des Abspiel-Stroms.

[0059] Quantisierte-Quelle-Bilder werden codiert mit derselben H.263-Syntax wie vorhergesagte Bilder, mit der Ausnahme, dass sie unterschieden werden von vorhergesagten Bildern durch Setzen der ersten drei Bits von MPPTYPE auf den reservierten Wert von „110".

[0060] Die periodische Codierung von Quantisierte-Quelle-Bildern kann einen pulsierenden (beating) Effekt in stationären Bereichen von Bildern verursachen. Dies wird wie folgt erläutert. Bei einer normalen prädiktiven Codierung werden stationäre Bereiche des Bilds, die bereits als eine angemessene Repräsentation des Quellen-Bilds codiert wurden, nicht modifiziert. Bei der Codierung solcher Bereiche in Quantisierte-Quelle-Bildern muss die Prädiktion quantisiert werden und, wenn mit dem Quantisierer-Index durchgeführt, der für nichtstationäre Bereiche des Bilds verwendet wird, verändert sie den Bereich, möglicherweise zum schlechteren, aber auf jeden Fall wird er verändert. Diese Änderung ist der pulsierende Effekt.

[0061] Dies wird gelöst durch Anmerken, dass, wenn die Prädiktion für einen Bereich des Bildes eine ausreichend gute Repräsentation der Quelle bietet, es keine Notwendigkeit gibt, eine Information zu übertragen und folglich den Bereich zu ändern. Wenn somit ein Zugriffsbild als ein Quantisierte-Quelle-Bild codiert wird, wird ein Test durchgeführt, um zu bestimmen, ob eine Information über den Bereich übertragen würde, wenn das Bild als ein prädiktives Bild codiert worden wäre statt als ein Quantisierte-Quelle-Bild. Wenn keine Information übertragen worden wäre, wird der Quantisierer-Index, der durch die Quantisierung der Schritte **300** und **310** und die inverse Quantisierung des Schrittes **350** verwendet wird, auf einen geringen Wert gesetzt, die Ausgabe des Subtrahierers **320**, im Allgemeinen als der Prädiktionsfehler bekannt, wird auf Null gesetzt, somit ist dieser Bereich des neu rekonstruierten Bilds gleich zu dem entsprechenden Bereich des vorher rekonstruierten Bildes, das mit einem feinen Quantisierer quantisiert wurde. In dem H.263- und anderen Standards ist der Bereich des Quantisierer-Indexes von 1 (fein) bis 31 (grob). Durch Bezugnahme auf einen kleinen Index wird typischerweise ein Wert von 8 oder weniger bezeichnet. Dies minimiert unnötige Änderungen an dem rekonstruierten Bild, während die Menge an Information minimiert wird, die übertragen werden muss. Es gibt jedoch Kosten hinsichtlich der Bitrate in dem entsprechenden Verbindungsbild, wo der Prädiktionsfehler wahrscheinlich nicht null ist, aber derselbe feine Quantisierer verwendet werden muss.

[0062] Fig. 4 ist ein schematisches Diagramm einer Client-Server-Architektur, die geeignet ist zur Verwendung in dem System von Fig. 1.

[0063] Der Client **40** umfasst einen Netzwerkpuffer **130**, einen Decodier-Puffer **41** und einen Decodierer **42**. Der Server **10** umfasst kreisförmige Puffer **70**, **80**, **90**, wie oben diskutiert, und einen Pakettransmitter **100** und einen Netzwerkpuffer **120** für jeden Client.

[0064] Der Client **40** informiert den Server **10** über die Menge an Information in seinem Decodier-Puffer **41** und die Rate, mit der er Daten empfängt. Der Server **10** verwendet diese Information, um zu bestimmen, wann er zwischen Abspielströmen umschalten soll. Wenn zum Beispiel der Client **40** mehr als eine Schwelle von Daten akkumuliert hat, zum Beispiel 15 Sekunden von Daten in seinem Decodier-Puffer **41**, und der Client **40** empfängt mit einer Rate, die höher oder gleich zu der Codierrate des nächst höheren Abspiel-Stroms in der Hierarchie ist, kann der Server **10** den Pakettransmitter **100** des Clients zu dem nächst höheren Abspiel-Strom an dem nächsten Verbindungsbild schalten.

[0065] Ähnlich, wenn die Menge von Daten, die von dem Client **40** in seinem Decodier-Puffer **41** unter eine Schwelle fällt, kann der Server **10** den Pakettransmitter **100** des Clients zu dem nächst niedrigen Abspiel-Strom an dem nächsten Verbindungsbild

schalten.

[0066] Der Gesamteffekt ist, dass die Übertragungsrate auf eine Netzwerkfreundliche Weise variiert gemäß dem Status der Überlastung in dem Netzwerk, aber aufgrund der Akkumulation von Daten in dem Decodier-Puffer **41** des Clients nimmt der Benutzer keine Änderung der Qualität als Ergebnis von Kurzzeit-Änderungen der Übertragungsrate wahr. Langzeit-Änderungen der Übertragungsrate werden gehandhabt durch Schalten zu einem Strom mit einer anderen Codierrate, um eine verbesserte Qualität zu ermöglichen, wenn es das Netzwerk erlaubt, und um eine Qualität zu reduzieren, ohne eine Präsentation anzuhalten oder beschädigte Media dem Benutzer zu präsentieren, wenn der Netzwerk-Durchsatz abnimmt.

[0067] Der Decodier-Puffer **41** an dem Client wird verwendet, um den Einfluss von Netzwerkleistungsvariationen auf die Qualität von Media zu reduzieren, die dem Benutzer präsentiert wird. Die Netzwerk-Charakteristiken, zu deren Handhabung der Puffer gestaltet ist, fallen in drei Kategorien: Paket-Jitter, Paketverlust und variabler Durchsatz. In der Praxis sind diese drei Netzwerkcharakteristiken nicht unabhängig, sie gehören alle zu einer Netzwerküberlastung und in dem Fall von mobilen Netzwerken zu einer Verschlechterung an der physikalischen Ebene.

[0068] Durch Entkoppeln der Übertragungsrate von der Media-Codierrate kann der Decodier-Puffer **41** des Clients gefüllt werden, wenn Netzwerkbedingungen günstig sind, um eine Ausfallsicherheit für Zeiten vorzusehen, wenn Netzwerkbedingungen nicht so gut sind.

[0069] Die Akkumulation von zehn zu hunderten von Sekunden von Daten in dem Decodier-Puffer **41** ermöglicht, dass ein Paket-Jitter (Verzögerungsvariationen) derselben Größe vor dem Benutzer versteckt werden. In der Praxis maskiert dies den gesamten Paket-Jitter, da große Mengen von Jitter besser klassifiziert werden als temporäre Verbindungsausfälle, die von dem im Folgenden beschriebenen Fehlerwiederherstellungsprozess gehandhabt werden.

[0070] Durch eine Akkumulation von Daten in dem Decodier-Puffer **41** ist Zeit verfügbar für die erneute Übertragung von verlorenen Paketen, bevor sie für eine Decodierung erforderlich sind. Wiederum ist durch Dimensionierung des Decodier-Puffers **41**, mehr Daten zu enthalten als ein Mehrfaches der Hin- und Rückverzögerung, Zeit vorhanden für eine geringe Anzahl von erneuten Übertragungsversuchen, um nach einem Paketverlust wiederherzustellen. Dies ermöglicht eine Wiederherstellung von den meisten Instanzen eines Paketverlusts, ohne die decodierte Mediaqualität zu beeinflussen, und macht die Verbindung semi-zuverlässig.

[0071] Schließlich kann, wiederum durch eine Akkumulation von Daten in dem Decodier-Puffer **41**, der Client **40** eine gleich bleibende Mediaqualität für einige Zeit aufrechterhalten, wenn die Empfangs-Bitrate geringer ist als die Codier-Bitrate, und für einige Zeit, wenn die Empfangsrate auf Null gefallen ist.

[0072] Wenn die Daten an den Client **40** geströmt werden mit einer Rate, die unabhängig ist von der Codierrate, und in dem Decodier-Puffer **41** gespeichert wird, ist es erforderlich, zum Decodieren von Daten das richtige Timing zu verwenden, anstatt einfach so schnell wie möglich zu decodieren und zu präsentieren. Es werden Zeitstempel verwendet für diesen Zweck sowie für die Synchronisierung von Audio und Video.

[0073] Aufgrund von Netzwerkvariationen kann die Menge von Daten in dem Decodier-Puffer **41** des Clients, gemessen in Bytes, über die Zeit Variieren. Zusätzlich würde die Menge von Daten in dem Decodier-Puffer **41**, gemessen hinsichtlich der Länge der Media-Präsentationszeit, die sie repräsentieren, ebenso über die Zeit variieren. Dies hat Implikationen für das Streaming von Live-Inhalt: es ist nicht möglich, Daten in dem Decodier-Puffer **41** anzusammeln, wenn die ersten Daten, die an den Client **40** gesendet werden, mit minimaler Verzögerung zu der Zeit, an der sie erfasst und codiert wurden, gesendet werden. Somit müssen die ersten Daten, die an den Client **40** gesendet werden, alte Daten sein, das heißt, Daten, die Ereignisse repräsentieren, die stattgefunden haben, bevor der Client **40** mit dem Server **10** verbunden wurde. Dann werden, wenn sich der Decodier-Puffer **41** füllt, die neuesten Daten darin neuer und neuer, während die dem Benutzer präsentierten Media bei einer konstanten Verzögerung zu der tatsächlichen Zeit des Auftretens bleibt.

[0074] Der Server puffert codierte Daten in seinen kreisförmigen Puffern **70**, **80**, **90** für eine konstante Zeitdauer nach der Codierung, so dass, wenn ein Client **40** mit dem Server **10** verbunden wird, „alte" Daten verfügbar sind zum Strömen an den Client **40**. Wenn sich der Decodier-Puffer **41** des Clients füllt, kommen die Lesepunkte von den kreisförmigen Puffern **70**, **80**, **90** näher zu den neuesten Daten in diesen Puffern.

[0075] Die optimale Größe der kreisförmigen Puffer **70**, **80**, **90** und des Decodier-Puffers **41** des Clients ist vorzugsweise derart, dass jeder dieselbe Menge an Daten enthalten kann, gemessen hinsichtlich der Media- Präsentationszeit, die sie repräsentieren.

[0076] Die Netzwerk-Puffer **120**, **130** jeweils in dem Server **10** und dem Client **40** werden verwendet durch ein Transportprotokoll, das eine semi-zuverläs-

sige Datenverbindung implementiert. Typischerweise werden Daten in dem Netzwerkpuffer **120** des Servers gehalten, bis sie und alle früheren Daten bestätigt wurden, dass sie an dem Client **40** empfangen wurden. Ähnlich werden Daten aus dem Netzwerkpuffer **130** des Servers entfernt, wenn sie und alle früheren Daten erfolgreich empfangen wurden und an den Decodier-Puffer **41** geleitet wurden. Folglich weiß der Server **10**, durch Kenntnis der Daten, die in seinem eigenen Netzwerkpuffer **120** bleiben, welche Daten erfolgreich durch den Client **40** empfangen wurden, innerhalb von Grenzen, die durch die unidirektionale Übertragungsverzögerung gegeben sind.

[0077] Dies impliziert, dass keine Rückmeldung bzw. Feedback von dem Client **40** an den Server **10** für den Server **10** erforderlich ist, außer der von dem Transportprotokoll selbst erforderlichen, um zu wissen, wie viele Daten von dem Client **40** empfangen wurden, so dass er Entscheidungen treffen kann über ein Umschalten zwischen Abspielströmen.

[0078] Das Vorhandensein einer Akkumulation von Daten in dem Decodier-Puffer **41** des Clients liefert eine Ausfallsicherheit für eine Anzahl von Netzwerkbeeinträchtigungen, wie Jitter, Paketverlust und variabler Durchsatz. Offensichtlich ist es nicht möglich, die gesamten Netzwerkbeeinträchtigungen auszugleichen, außer der Decodier-Puffer **41** ist dimensioniert, den gesamten Media-Inhalt zu enthalten, und eine Präsentation wird verzögert, bis alle Daten empfangen sind. Da dieser Fall kein Strömen, sondern ein Herunterladen ist, ist eine Strategie zur Wiederherstellung nach ernsten Netzwerkbeeinträchtigungen erforderlich.

[0079] Zu Zeiten, wenn der Netzwerkdurchsatz auf einen Pegel unter der Codierrate des Abspiel-Stroms der niedrigsten Rate für eine beträchtliche Zeitdauer fällt, reduziert sich die Menge von Daten in dem Decodier-Puffer **41** und wird schließlich Null. An diesem Zeitpunkt endet die Präsentation an den Benutzer. Jedoch geht die Füllung des kreisförmigen Puffers an dem Server **10** weiter. Folglich, wenn das Netzwerk auf einen Status wiederhergestellt wird, in dem eine Übertragung des Abspiel-Stroms mit niedrigster Rate wieder möglich ist, sind die nächsten Daten, die von dem Client **40** erforderlich sind, wahrscheinlich nicht in dem kreisförmigen Puffer **70**, **80**, **90** des Servers, da sie durch neuere Daten überschrieben wurden.

[0080] Um sich aus dieser Situation wieder zu erholen, muss der Server **10** ein Strömen wieder beginnen, als ob eine neue Verbindung von dem Client gemacht worden wäre: er muss einen Punkt in dem Intra-Strom finden und von dort zu strömen beginnen und dann durch den Verbindungsstrom in den Abspiel-Stroms mit niedrigster Rate schalten. Der Effekt für den Benutzer ist der Verlust von Media von dem Zeitpunkt, an dem der Decodier-Puffer **41** leer wurde, bis zu dem Zeitpunkt, an dem der Server beginnt, den Intra-Strom zu senden.

[0081] Der Server **10** weiß, dass der Decodier-Puffer **41** des Clients leer wird, genauso wie er weiß, wann der Client mit dem Decodieren begonnen hat und wie viele Daten erfolgreich empfangen wurden. Er kann somit an einem Intra-Strom-Bild erneut starten, ohne der Notwendigkeit für eine spezifische Nachricht von dem Client. Um jedoch eine Ausfallsicherheit für das System vorzusehen, zum Beispiel, um sich von dem Effekt von unterschiedlichen Taktgeschwindigkeiten in dem Server und dem Client zu erholen, wird in dieser Situation eine Steuernachricht von dem Client **40** an den Server **10** gesendet.

[0082] In Prinzip ist ein Strömen aus der Datei identisch zu einem Live-Strömen. In der Praxis ist es etwas einfacher. Es gibt keine Notwendigkeit für die kreisförmigen Puffer **70**, **80**, **90**, da Daten aus der Datei gelesen werden können, wie und wann erforderlich. Der Server **10** verwendet jedoch dieselben Techniken, um den Decodier-Puffer **41** an dem Client **40** zu füllen und zwischen Abspiel-Strömen zu schalten. In dem Fall, in dem der Decodier-Puffer **41** leer wird, gibt es keine Notwendigkeit, an einem späteren Punkt in dem Inhalt mit einem Intra-Strom-Bild erneut zu starten, da eine Präsentation wiederaufgenommen werden kann, wenn der Netzwerkdurchsatz wieder ausreichend wird: der Benutzer nimmt einfach eine Zeitspanne wahr, in der keine Media präsentiert werden.

[0083] Trickmodi, wie schnelles Vorwärtsspielen, schnelles Rückspielen und wahlfreier Zugriff werden durch die Verwendung des Intra-Stroms möglich.

[0084] Durch Schreiben von „alten" Daten in den kreisförmigen Puffern **70**, **80**, **90** in eine Datei, kurz bevor sie überschrieben werden, kann das Problem, das oben beschrieben wird, dass der Decodier-Puffer **41** leer wird und der Benutzer einen Inhalt verpasst, bis eine Wiederherstellung mit einem Intra-Strom-Bild stattfindet, vermieden werden, da Daten für ein Strömen an den Client immer verfügbar sind: sie müssen aus der Datei statt aus den kreisförmigen Puffern **70**, **80**, **90** gelesen werden.

[0085] Eine derartige Funktionalität ermöglicht einem Client auch, die präsentierten Media für eine unbefristete Zeitdauer anzuhalten und danach mit dem Strömen fortzufahren. Sie ermöglicht einem Benutzer auch, schnell vorwärts zu spielen nach einer derartigen Pause, um den Live-Strom einzuholen.

[0086] Eine Implementierung des Transportprotokolls, das in der oben erwähnten Client-Server-Architektur getestet wird, basiert auf dem ISO-TCP-Transportprotokoll TPKT, das in RFC-2126 von Y. Pouffary „ISO Transport Service an top of TCP (ITOT)" detail-

liert beschrieben wird.

[0087] Das standardmäßige TPKT-Protokoll definiert einen Header, der in Fig. 5a dargestellt wird, gefolgt von einer Nutzlast bzw. Nutzdaten (payload). Die Paketlänge zeigt die kombinierte Länge von Header und Nutzlast in Oktetten an.

[0088] In der für das oben beschriebene System verwendeten Implementierung ist TPKT erweitert, um einen Header aufzuweisen, von dem ein Beispiel in Fig. 5b dargestellt wird, gefolgt von einer Nutzlast. Die Paketlänge zeigt die kombinierte Länge von Header, Zeitstempel, wenn vorhanden, und Nutzlast in Oktetten an. T ist ein Bit, das anzeigt, ob der Zeitstempel vorhanden ist, und M ist ein Bit, das anzeigt, ob die Nutzlast eine Audio- oder Video-Information enthält.

[0089] Wie oben angeführt, sind Zeitstempel erforderlich für das richtige Timing der Decodierung von Daten. Eine Information, die in Paket-Header eingebettet ist, umfasst die Länge des Pakets, einen Zeitstempel für die Daten in dem Paket und einen Strom-Identifizierer.

[0090] Der Strom-Identifizierer ist vorgesehen, damit Audio und Video in eine einzelne TCP-Verbindung gemultiplext werden können. Dies dient, um eine Synchronisierung einer Audio- und Video-Übertragung sicherzustellen. Wenn getrennte TCP-Verbindungen verwendet werden, ist es möglich, dass sie etwas unterschiedlich auf Netzwerkcharakteristiken reagieren und unterschiedliche Durchsätze erzielen, was letztendlich zu stark unterschiedlichen Mengen von Daten in den Decodier-Puffern der Clients führt, gemessen hinsichtlich der Präsentationszeit. Obwohl diese Unterschiede verwaltet werden können, wird das Problem überhaupt vermieden durch die Verwendung einer einzelnen TCP-Verbindung und ein Multiplexen von Audio und Video mit derselben Präsentationszeit in benachbarten Paketen. Tatsächlich erfordert ein Hinzufügen von Audio zu einem nur-Video-System einfach das Senden von Audiopaketen zur selben Zeit wie das zugehörige Video: es ist keine weitere Steuerung erforderlich.

[0091] Der Server **10** versucht, Pakete so schnell wie möglich zu senden. Anfangs wird eine Anzahl von Paketen aufeinander folgend gesendet, ungeachtet der Netzwerkkapazität, da sie sich einfach in dem Netzwerkpuffer **120** des Servers ansammeln. Wenn der Netzwerkpuffer **120** voll wird, stimmt die Rate, mit der Pakete an den Netzwerkpuffer **120** gesendet werden können, mit der Rate einer Übertragung über das Netzwerk überein, wobei der Übertragungsprozess begrenzt wird durch Blockieren von Aufrufen an die Socket-Sende-Funktion.

[0092] Die Übertragungsrate wird auch begrenzt, wenn die Menge von Daten, die an dem Client gepuffert werden, eine Schwelle erreichen, zum Beispiel 30 Sekunden. Wenn der Decodier-Puffer **41** des Clients so viele Daten hat, schränkt der Server **10** die Übertragungsrate ein, um diesen Füllpegel beizubehalten.

[0093] Ein Netzwerkdurchsatz wird geschätzt durch Zählen von Bytes, die an den Netzwerkpuffer **120** gesendet wurden, Subtrahieren davon der Größe des Netzwerkpuffers und Teilen durch die Zeit seit dem Start der Übertragung. Kürzere Schätzungen des Netzwerkdurchsatzes werden berechnet unter Verwendung von zwei Zählungen von übertragenen Bytes und zwei Messungen der Zeit, die deren Senden beanspruchte, Berechnen des Durchsatzes aus einem Paar und regelmäßiges Umschalten zwischen ihnen, Zurücksetzen des Paares, das nicht länger verwendet wird, auf null. Wenn zum Beispiel ein Zurücksetzen alle 200 Sekunden stattfindet, wird der Netzwerkdurchsatz über eine Zeitdauer geschätzt, die von 200 Sekunden unmittelbar nach dem Zurücksetzen bis zu 40 Sekunden gerade vor dem nächsten Zurücksetzen variiert.

[0094] Diese Technik arbeitet zufriedenstellend, vorausgesetzt, der Server **10** versucht, so schnell wie möglich zu strömen. Aber wie oben erwähnt, wenn die Menge von Daten in dem Decodier-Puffer **41** eine Schwelle überschreitet, schränkt der Server **10** seine Übertragungsrate ein, um eine konstante Pufferfüllung beizubehalten. In diesem Fall wird der Netzwerkdurchsatz geschätzt als die Codier-Bitrate des aktuellen Abspiel-Stroms. Wenn in diesem Zustand, kann das Netzwerk möglicherweise einen Abspiel-Strom mit höherer Rate übertragen als der aktuell geströmte, aber der Server **10** schaltet nicht um, da er keine genaue Schätzung des Netzwerkdurchsatzes machen kann aufgrund seiner eigenen Raten-Begrenzung. Um aus diesem Zustand herauszukommen, ignoriert der Server regelmäßig die Füllschwelle des Decodier-Puffers des Clients, um mit voller Rate für eine gegebene Zeitdauer oder eine gegebene Menge von Daten zu strömen. Er zeichnet die Anzahl von Bytes, die an den Netzwerkpuffer **120** gesendet werden, und die in Anspruch genommene Zeit auf, beginnend, wenn der Netzwerkpuffer **120** voll wird, wie durch einen Blockieraufruf an die Sende-Funktion erfasst wird. Er schätzt dann den erzielbaren Durchsatz und verwendet dies, um zu bestimmen, ob zu einem Abspiel-Strom mit höherer Rate zu schalten ist.

[0095] Wie oben angeführt, weiß der Server **10** implizit, durch Kenntnis der in seinem Netzwerkpuffer **120** gespeicherten Daten, welche Daten von dem Client **40** empfangen wurden und an seinen Decodier-Puffer **41** geliefert wurden. Diese Information kann verwendet werden, um zu bestimmen, wann zwischen Abspiel-Strömen zu schalten ist und wann

die Fehler-Wiederherstellungs-Verfahren aufzurufen sind. Jedoch wird eine Sichtbarkeit der Inhalte und Füllung des Netzwerkpuffers **120** des Servers in den meisten Socket-Implementierungen nicht unterstützt. Um die Inhalte des Netzwerkpuffers **120** zu überwachen, wird ein Spiegelpuffer **120a** implementiert. Der Spiegelpuffer **120a** speichert die tatsächlichen Daten nicht, die an den Netzwerkpuffer **120** gesendet werden, sondern speichert nur die Anzahl von gesendeten Bytes und den Zeitstempel der Daten. Durch Kenntnis der Größe des Netzwerkpuffers **120** und unter der Annahme, dass dieser immer voll ist, hat der Server **10** Zugriff auf den Zeitstempel der ältesten Daten in dem Netzwerkpuffer **120** über den Spiegelpuffer **120a**, der ungefähr derselbe ist wie der Zeitstempel der neuesten Daten in dem Decodier-Puffer **41** des Clients.

[0096] Beim Testen wurde beobachtet, dass die Annahme, dass der Netzwerkpuffer **120** an dem Server **10** immer voll ist, meistens richtig ist. Dies aufgrund dessen, da der Übertragungsprozess gesteuert wird, so schnell wie möglich an den Netzwerkpuffer **120** zu senden. Wenn der Netwerkpuffer **120** leerer als voll wird, tritt der Effekt auf, dass die Menge von Daten an dem Client **40** unterschätzt wird, was in den meisten Fällen ungefährlich ist, da als das Hauptproblem eine Erschöpfung von Daten an dem Client **40** statt eine Überfüllung gesehen wird. In der Praxis kann der Decodier-Puffer **41** dimensioniert werden, größer als die größte Menge von Daten zu sein, die er speichern muss. In jedem Fall stoppt, wenn der Decodier-Puffer **41** voll ist, der Client **40** das Lesen aus dem Netzwerkpuffer **130**, was wiederum den Server-Netzwerkpuffer **120** vor einer Entleerung stoppt, und die Übertragung endet.

[0097] Um die exakte Menge von Daten in dem Decodier-Puffer **41** zu bestimmen, muss der Server auch den Zeitstempel des Datenpakets kennen, das der Client momentan decodiert und präsentiert. Der Server **10** berechnet dies unter Verwendung von zwei Annahmen: erstens, dass der Client **40** mit der Decodierung beginnt, unmittelbar nachdem der Server **10** das erste Paket sendet; und zweitens, dass der Takt des Clients nicht signifikant von dem Takt des Servers während der Dauer der Strömung abweicht.

[0098] In der Praxis wurden beide Annahmen als gültig befunden. Der Client **40** ist ausgebildet, unmittelbar bei Empfang von Daten mit einer Decodierung zu beginnen, und so führt jeder Fehler für die geschätzte Präsentationszeit des Servers zu einer Unterschätzung der Menge von Daten in dem Decodier-Puffer **41**, was kein Problem ist, wie oben erläutert wird. Eine Abweichung zwischen den Takten des Clients und des Servers während einer typischen Strömungssitzung kann wahrscheinlich vernachlässigt werden im Vergleich zu den Mengen von Daten, die gepuffert werden. Zum Beispiel würde es bei einer Differenz von 100 Teilen pro Million 10000 Sekunden oder fast drei Stunden dauern, bis eine Abweichung von einer Sekunde auftritt. In dem seltenen Fall, dass eine große Menge von Abweichung akkumuliert, kann der Client **40** den Server **10** alarmieren über die Verwendung einer Steuernachricht, wie die oben beschriebene, die für eine Unterschreiten des Decodier-Puffers gesendet wird.

[0099] Der Server **10** strömt anfangs den Abspiel-Strom mit der niedrigsten Bitrate, um dem Client **40** zu ermöglichen, unmittelbar zu decodieren und Media dem Benutzer zu präsentieren, während ebenfalls die Menge von Daten in dem Decodier-Puffer **41** aufgefüllt wird, um eine Ausfallsicherheit für Netzwerkbeeinträchtigungen vorzusehen. Wenn das Netzwerk eine ausreichende Kapazität hat, um eine Übertragung eines Abspiel-Stroms mit höherer Rate zu unterstützen, sollte der Server **10** an einem geeigneten Zeitmoment umschalten zum Strömen eines Abspiel-Stroms mit höherer Rate.

[0100] Es gibt viele mögliche Strategien, die verwendet werden können, um zu bestimmen, wann zu einem Abspiel-Strom mit höherer Rate umzuschalten ist. Vorzugsweise sollte der Client **40** ausreichend Daten in seinem Decodier-Puffer **41** haben, um fortfahren zu können mit der Decodierung und Präsentation von Media für eine vorgegebene Zeitdauer, beispielsweise 15 Sekunden. Es ist auch vorzuziehen, dass ein Netzwerkdurchsatz, der vor kurzem erreicht wurde, gemessen über beispielsweise die letzten 60 Sekunden, ausreichend sein sollte, um ein Strömen des Abspiel-Stroms, zu dem geschaltet wird, unbefristet aufrechtzuerhalten; das heißt, die vor kurzem erreichte Netzwerkdurchsatzrate sollte größer oder gleich zu der Bitrate des Abspiel-Stroms sein. Das Ziel ist, ein häufiges Schalten zwischen Strömen zu vermeiden, da dies für den Benutzer störender sein kann als eine konstante Qualität mit niedrigerer Rate.

[0101] Um dieses Ziel zu erreichen, ist vorzuziehen, dass die Entscheidung zum Herunterschalten eine Hysterese relativ zu der Entscheidung zu Hinaufschalten umfasst. Zum Beispiel kann ein Herunterschalten zu dem Abspiel-Strom mit der nächst niedrigeren Bitrate ausgelöst werden, wenn der Client **40** nicht länger ausreichend Daten in seinem Decodier-Puffer **41** hat, um weiterhin zu decodieren und Media zu präsentieren für eine spezifizierte Zeitdauer, beispielsweise 8 Sekunden. In dem Fall einer Konfiguration mit drei oder mehr Abspiel-Strömen, und wobei der aktuell geströmte Abspiel-Strom der Abspiel-Strom mit der dritten oder sogar höheren Rate ist, führt diese Strategie nicht zu einem unmittelbaren Abfall an das untere Ende der Hierarchie, da Zugriffsbilder nur periodisch auftreten, und es ist zu hoffen, dass die Füllung des Decodier-Puffers wiederhergestellt wird nach einem ersten Herunterschalten, so dass ein zweites Herunterschalten nicht erfor-

derlich ist.

[0102] Die Fig. 6a–Fig. 6c sind schematische Diagramme von Aspekten einer Datenstruktur zum Speichern einer audiovisuellen Datenquelle gemäß einem Ausführungsbeispiel der vorliegenden Erfindung.

[0103] Die in Fig. 6a gezeigte Hauptdatenstruktur ermöglicht die Speicherung in einer einzelnen Datei von mehreren Audio-Abspiel-Strömen, einem Intra-Video-Strom und mehreren Video-Abspiel- und Schalt-Strömen.

[0104] Da die audio-visuelle Datenquelle, die in der vorliegenden Erfindung erzeugt und verwendet wird, eine Anzahl von codierten Strömen hat, die zu jeder Zeit an einen Client übertragen werden können, ist eine Speicherung in einer herkömmlichen sequentiellen Datei nicht möglich. Zum Beispiel kann in dem Fall von Video ein bestimmtes Quellen-Bild in jedem Abspiel-Strom codiert werden und kann auch in dem Intra-Strom und in einigen oder allen der Schalt-Ströme codiert sein.

[0105] Die Datei enthält eine Datenstruktur, wobei ein Beispiel daraus in Fig. 6a dargestellt wird, gefolgt von Strom-Daten. Die Datenstruktur umfasst einen Header **600**, der eine Information über die Anzahl und den Typ von Strömen (Audio, Video, Schalten, usw.) enthält. Für die erste und letzte Instanz jedes Typs von Strom umfasst sie auch Zeiger **610**–**680** (ausgedrückt als Offsets von dem Beginn der Datei) zu dem Header für den jeweiligen Strom.

[0106] Jeder Zeiger **620**–**680** zeigt zu einer Stromdatenstruktur, die einen Strom-Header **700** umfasst, der einen Zeiger **710** zu dem nächsten Strom-Header desselben Typs, einen Zeiger **720**, **730** zu den jeweils ersten und letzten Paketen des Stroms enthält. Jeder Stromtyp verwendet einen spezifischen Strom-Header-Typ, jedoch sind bestimmte Elemente allen Strom-Header-Typen gemeinsam: Eine Strom-Identifizierungsnummer **705**, ein Zeiger **710** zu dem nächsten Strom-Header desselben Typs und die Zeiger **720**, **730** zu den jeweils ersten und letzten Paketen des Stroms. Ein beispielhafter Strom-Header, der nur diese gemeinsamen Elemente enthält, wird in der Fig. 6b gezeigt. Abspiel- und Audio-Strom-Header enthalten zusätzlich die Bitrate, mit welcher der Strom codiert wurde. Schalt-Strom-Header enthalten die Strom-Identifizierer der Abspiel-Ströme von und zu denen der Schalt-Strom ein Schalten ermöglicht.

[0107] Jeder Strom besteht aus einer Sequenz von Paketen, die jeweils durch eine Paketdatenstruktur repräsentiert wird, wobei ein Beispiel dafür in der Fig. 6c dargestellt wird. Jede Paketdatenstruktur umfasst einen Paket-Header **800** und eine Nutzlast **810**. Der Header umfasst Daten, einschließlich einen Zeiger **801** zu dem nächsten Paket in dem Strom, einen Zeitstempel **802**, eine Paketsequenznummer **803**, eine Paketgröße **804** und eine Rahmennummer **805** (d.h. die Sequenznummer des Videobilds oder Audiorahmens, die das Paket möglicherweise zusammen mit anderen Paketen repräsentiert). Schaltpakete enthalten zusätzlich die Sequenznummern von Paketen in von- und zu-Abspiel-Strömen, zwischen denen sie eine Bit-Raten-Schaltung ermöglichen. Der Schalt-Strom-Paket-Header definiert effektiv einen Schaltpunkt und enthält die Sequenznummer des letzten Pakets, das von dem „von"-Strom zu spielen ist vor dem Schalten, und das erste zu spielende von dem „an"-Strom nach dem Schalten. Die Sequenznummern beginnen bei 0 und sind niemals negativ. Die Verwendung von Zeigern, um eine Navigation zwischen Strömen zu unterstützen, wenn ein Schalten möglich ist, obwohl diesem Ansatz in diesem bestimmten Ausführungsbeispiel nicht gefolgt wird.

[0108] Die Zeiger zu der letzten Strom-Datenstruktur und dem letzten Paket sind nützlich beim Anhängen an eine Datei, da sie einen unmittelbaren Zugriff zu den Punkten vorsehen, an denen die Datei erweitert werden muss, ohne die Notwendigkeit, durch die gesamte Datei zu suchen.

[0109] Die Komplexität der Datenstruktur ist eine Folge der Verschachtelung von Paketen von potentiell vielen Strömen und der Notwendigkeit, ein Umschalten und eine Wiederherstellung zu unterstützen. Eine Navigation von Paket zu Paket erfolgt notwendigerweise durch Zeiger, da im Allgemeinen Pakete, die in einem Strom aufeinander folgend sind, in der Datei nicht angrenzend gespeichert werden. Ein Schreiben von Schalt- und Wiederherstellungs-Paketen erfordert, dass präzise Details von Quell- und Ziel-Paketen aufgezeichnet werden. Ein Schalten zwischen Strömen während einer Wiedergabe erfordert erstens die Identifizierung des nächsten verfügbaren Schaltpakets, gefolgt von der Wiedergabe der verbleibenden Pakete von dem „von"-Strom, Wiedergabe der Schalt-Pakete, dann die Wiedergabe der Pakete von dem „an"-Strom von dem geeigneten Punkt. Ferner darf es keine nennenswerte Verzögerung geben beim Schalten zwischen Strömen.

[0110] In Tests wurden sowohl Datei-basierte als auch Live-Streaming-Szenarien untersucht unter Verwendung des BTCellnet™ GPRS-Netzwerks. Ein Desktop-Pentium-PC wurde verwendet, um den Codierer und Server laufen zu lassen. Der Client ist ein Compaq iPaq™, der über eine Infrarotverbindung mit einem mobilen Motorola Timeport™ GPRS-Telefon verbunden ist.

[0111] In einer Nur-Video-Konfiguration wurden zwei Schalt-Ströme verwendet, mit Bitraten von 6 kbit/s und 12 kbit/s.

[0112] Das System funktionierte wie erwartet. Eine Übertragung startet mit dem Intra-Strom und schaltet dann zu dem Abspiel-Strom mit 6 kbit/s, wo sie eine Weile bleibt, wodurch Daten in dem Client als ein Ergebnis der tatsächlichen Übertragung schneller als 6 kbit/s akkumulieren. Wenn dann ausreichend Daten akkumuliert sind und die durchschnittliche Kurzzeit-Empfangsrate größer als 12 kbit/s ist, schaltet es zu dem Abspiel-Strom mit höherer Rate.

[0113] Manchmal während einer längeren Sitzung treten gelegentliche Umschaltungen zurück zu dem Abspiel-Strom mit niedriger Rate auf als ein Ergebnis eines reduzierten Netzwerkdurchsatzes. Und sehr selten wird eine Media-Präsentation unterbrochen aufgrund einer signifikanten Periode, während der das Netzwerk keine Daten an den Client liefern konnte.

[0114] Der Gesamteffekt für die meisten Sitzungen ist, dass der Benutzer eine kontinuierliche Media-Präsentation sehen kann mit gelegentlichen Änderungen der Qualität, aber ohne Verzerrungen des Typs, die normalerweise zu Bitfehlern und Paketverlust gehören. Nur sehr selten werden vollständige Pausen in der Media-Präsentation beobachtet als ein Ergebnis von schwerwiegenden Netzwerkbeeinträchtigungen und Verlust eines Durchsatzes.

Patentansprüche

1. Datenstruktur zum Speichern einer Datenquelle für ein Streaming-System, wobei die Datenquelle eine Vielzahl von codierten Datenströmen umfasst, wobei zumindest einige der Vielzahl von Datenströmen unabhängige Repräsentationen von Daten von der Datenquelle sind, die mit anderen Auflösungen als andere der Vielzahl von Datenströmen codiert sind, wobei die Datenstruktur einen Header (**600**–**680**), eine Strom-Datenstruktur (**700**) für jeden der codierten Datenströme und ein oder mehrere Paket(e) (**800**) der codierten Datenströme aufweist, wobei der Header (**600**–**680**) mit einer der Strom-Datenstrukturen (**700**) verbunden ist, wobei jede Strom-Datenstruktur (**700**) einen Header (**705**, **740**, **750**), eine Verbindung (**710**) zu einer nächsten Strom-Datenstruktur und eine Verbindung (**720**) zu einem ersten Paket des codierten Datenstroms umfasst.

2. Datenstruktur gemäß Anspruch 1, einschließlich Audiodaten, die als ein Datenstrom codiert sind.

3. Datenstruktur gemäß Anspruch 1 oder 2, wobei die Vielzahl von codierten Datenströmen Videodatenströme sind.

4. Datenstruktur gemäß Anspruch 3, wobei die Datenquelle weiter aufweist einen Schaltstrom, der eine Vielzahl von Schaltpunkten definiert zum Schalten zwischen einem der Videodatenströme und einem anderen der Videodatenströme, wobei die Datenstromstruktur für den Schaltdatenstrom Daten von Videoströmen und Pakete umfasst, an die und von denen ein Schalten möglich ist.

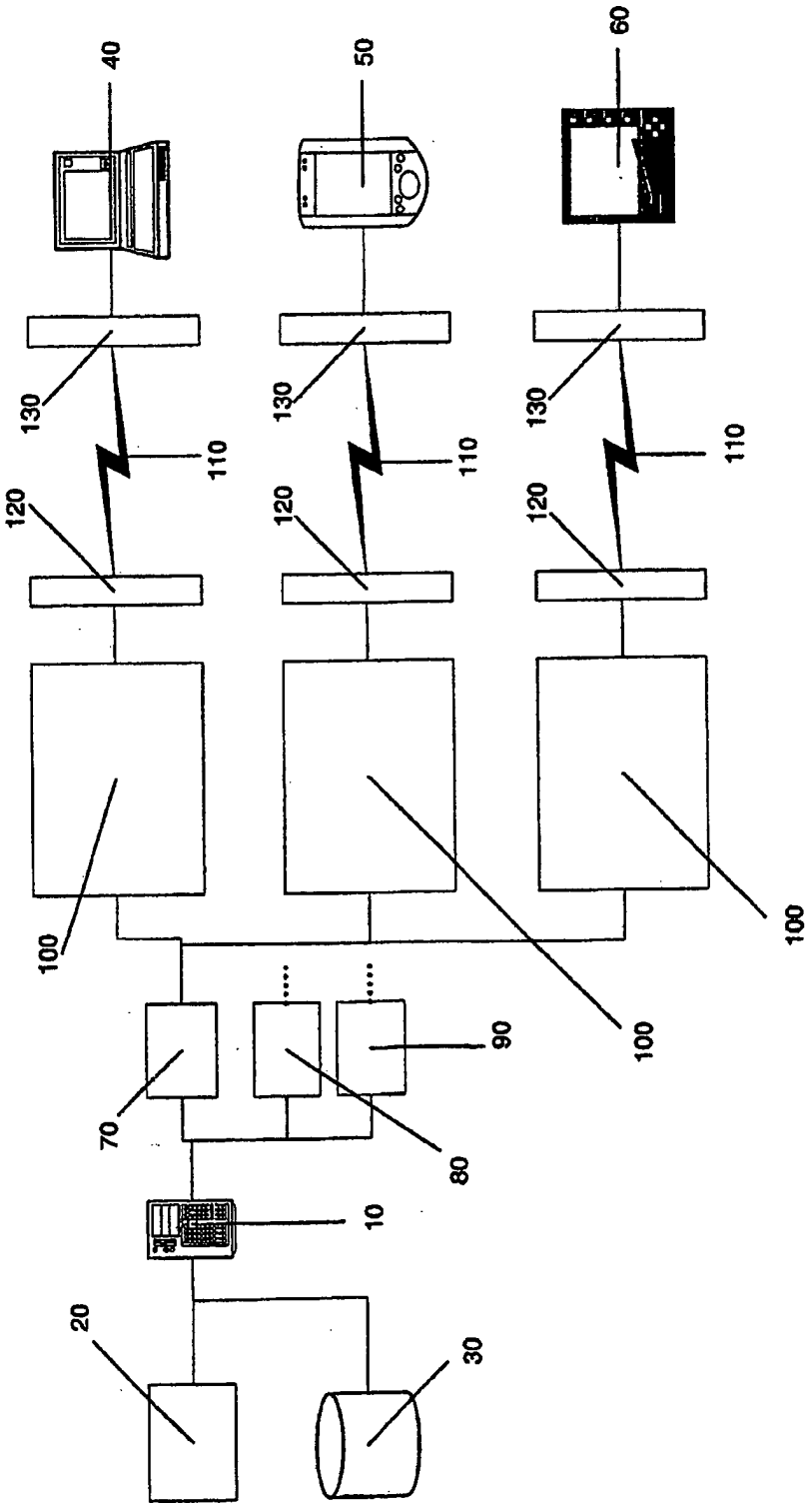
5. Datenstruktur gemäß einem vorhergehenden Anspruch, wobei der Header der Datenstruktur eine Verbindung (**620**) zu einer letzten Strom-Datenstruktur umfasst.

6. Datenstruktur gemäß einem vorhergehenden Anspruch, wobei der Header einer Strom-Datenstruktur (**700**) eine Verbindung (**730**) zu dem letzten Paket des codierten Datenstroms umfasst.

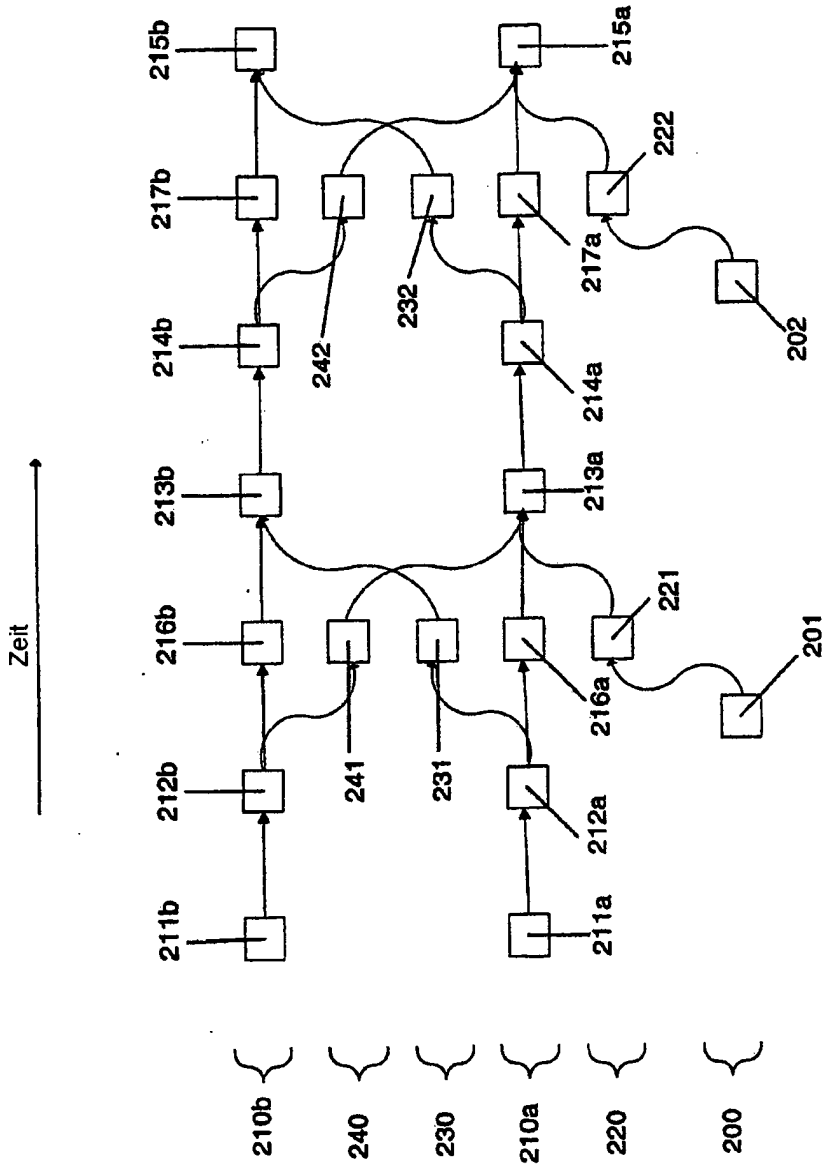
7. Datenstruktur gemäß einem der Ansprüche 1 bis 6, die auf einem Computer-lesbaren Medium codiert ist.

Es folgen 6 Blatt Zeichnungen

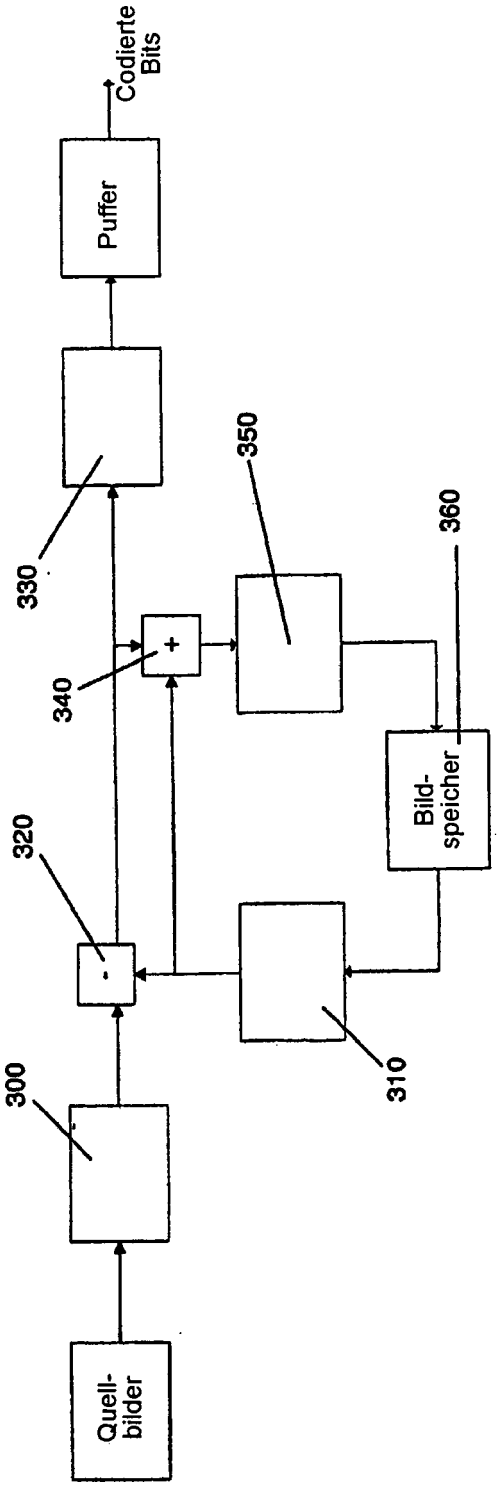
Anhängende Zeichnungen



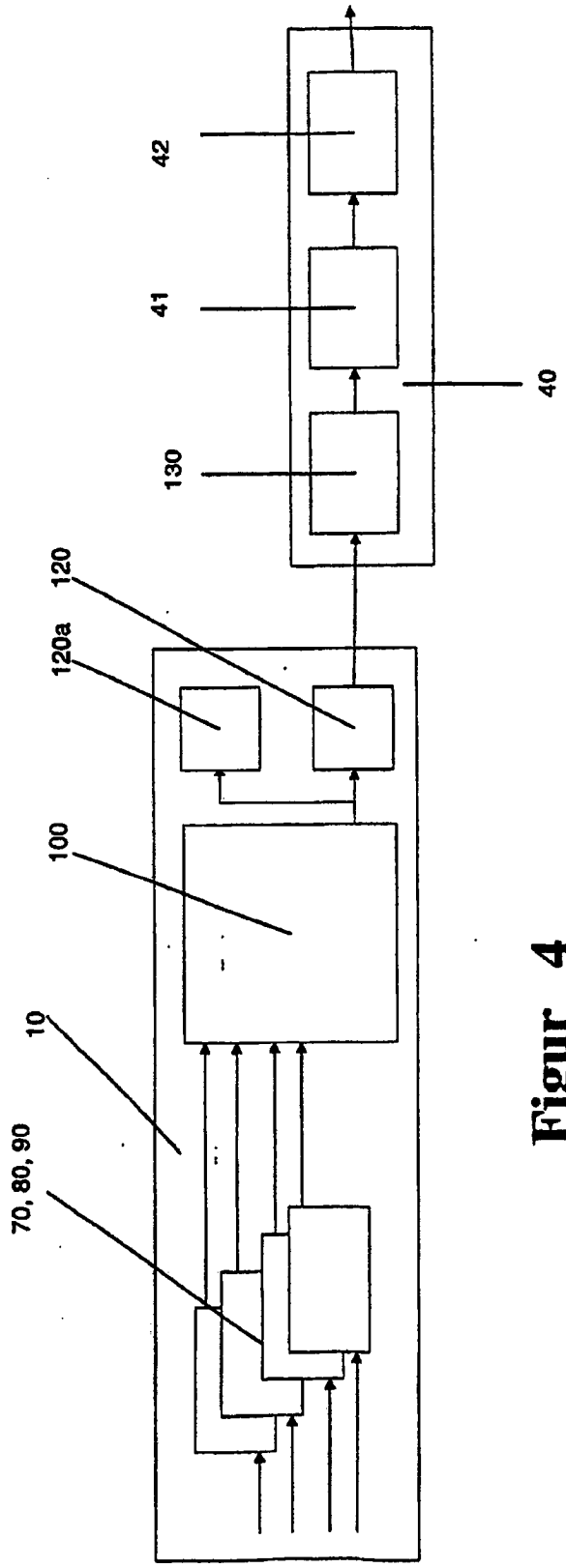
Figur 1


Zeit
211b
212b
216b
213b
214b
217b
215b
211a
212a
216a
213a
214a
217a
215a
241
231
242
232
221
222
201
202
210b
240
230
210a
220
200

Figur 2


Codierte Bits
Puffer
330
350
360
340
+
Bild-speicher
320
-
310
300
Quell-bilder

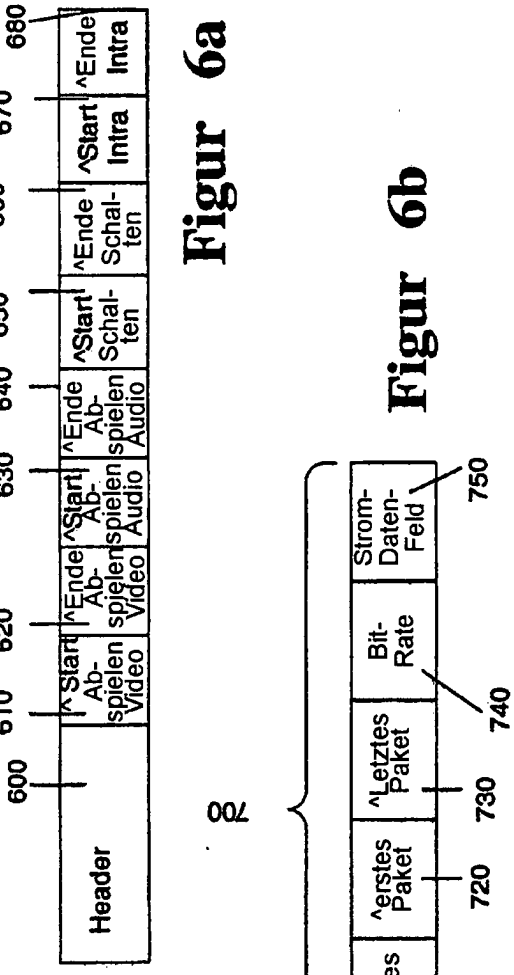
Figur 3



Figur 4

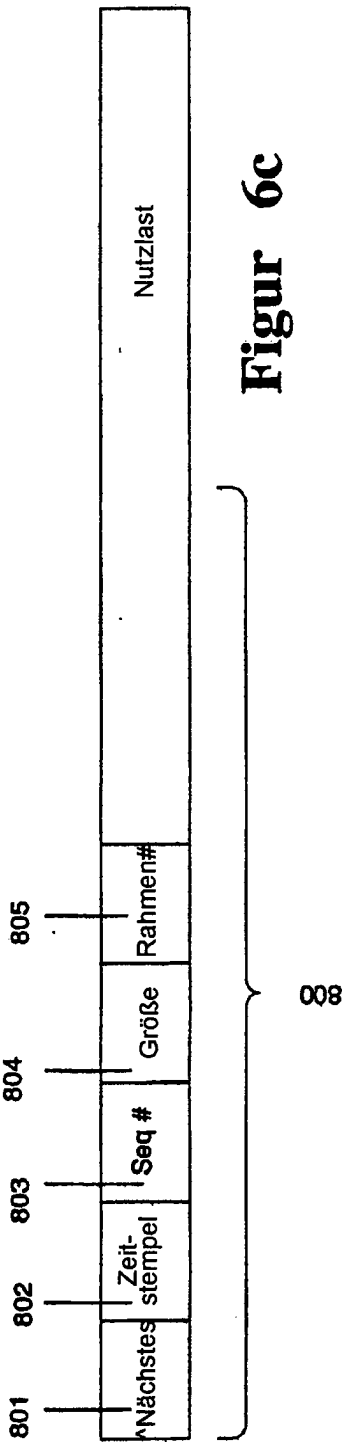
Figur 5a

Figur 5b



Figur 6a

Figur 6b



Figur 6c