22303

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 162 828 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
12.12.2001 Bulletin 2001/50

(51) Int Cl.[7]: **H04N 5/00**, H04N 7/173

(21) Application number: **01117966.0**

(22) Date of filing: **19.10.1998**

(84) Designated Contracting States:
DE FR GB NL

(30) Priority: **22.10.1997 US 956262**

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
98952358.4 / 1 025 699

(71) Applicant: **ORACLE CORPORATION**
Redwood Shores, CA 94065 (US)

(72) Inventors:
- **Weaver, Daniel**
  Redwood City, CA 94061 (US)
- **Pawson, David, J.**
  San Mateo, CA 94402 (US)

(74) Representative: **Viering, Jentschura & Partner**
Postfach 22 14 43
80504 München (DE)

Remarks:
This application was filed on 24 - 07 - 2001 as a divisional application to the application mentioned under INID code 62.

(54) Method and apparatus for implementing seamless playback of continuous media feeds

(57) A method and system for storing a continuous feed of video is provided. According to one aspect of the invention, the continuous feed is encoded in a digital video format to produce a digital data stream. A series of content files is created by repeatedly performing the steps of (1) storing the digital data stream in a current file, and (2) establishing a new file as the current file when the current file satisfies a predetermined condition. If the series of content files contains more than a predetermined amount of the continuous feed, the oldest content file in the series of content files is deleted. Tag information that indicates information about frames contained in the digital data stream is generated. The tag information includes timestamps that indicate timing of frames relative to a beginning of the digital data stream. An initial time value that indicates an absolute time that corresponds to the beginning of the digital data stream. When a request from a client for playback beginning at a specified absolute time is received, the initial time value is subtracted from the specified absolute time to determine a relative time. The tag information is used to identify a location in the digital data stream that corresponds to the relative time. The digital data stream is then transmitted to the client beginning at the location in the digital data stream that corresponds to the relative time.

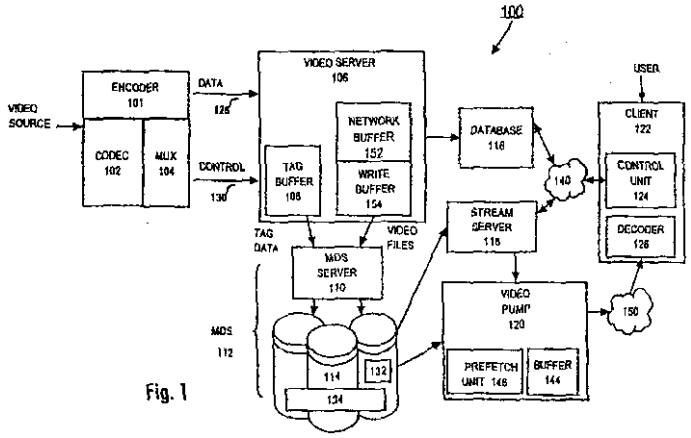

Fig. 1

EP 1 162 828 A2

Description

RELATED APPLICATIONS

[0001] The present application is related to: U.S. Patent Application No. 08/956,261, entitled "METHOD AND APPARATUS FOR CONCURRENTLY ENCODING AND TAGGING MEDIA" filed by Daniel Weaver, Mark A. Porter and David J. Pawson ", on 22 October 1997.

[0002] U.S. Patent Application No. 08/956,263, entitled "METHOD AND APPARATUS FOR NON-SEQUENTIAL ACCESS TO AN IN-PROGRESS VIDEO FEED" filed by Daniel Weaver, Mark A. Porter and David J. Pawson", on 22 October 1997.

FIELD OF THE INVENTION

[0003] The present invention relates to a method and apparatus for processing audio-visual information, and more specifically, to a method and apparatus for providing non-sequential access to audio-visual information represented in a live content stream.

BACKGROUND OF THE INVENTION

[0004] In recent years, the media industry has expanded its horizons beyond traditional analog technologies. Audio, photographs, and even feature films are now being recorded or converted into digital formats. To encourage compatibility between products, standard formats have been developed in many of the media categories.

[0005] As would be expected, the viewers of digital video desire the same functionality from the providers of digital video as they now enjoy while watching analog video tapes on video cassette recorders. For example, viewers want to be able to make the video jump ahead, jump back, fast forward, fast rewind, slow forward, slow rewind and freeze frame

[0006] Various approaches have been developed to provide non-sequential playback of digital video data. With respect to digital video data, non-sequential playback refers to any playback operation that does not play all of the encoded frames in the exact order in the sequence in which they were encoded. For example, jump ahead and fast forward operations are non-sequential in that some frames are skipped. Rewind operations at any speed are non-sequential in that during a rewind operation, frames are not played in the sequence in which they are encoded.

[0007] One approach to providing non-sequential playback of digital video data, referred to herein as the tag-based approach, is described in U.S. Patent No. 5,659,539, entitled "Method and Apparatus for Frame Accurate Access of Digital Audio-visual Information" issued to Porter et al on August 19, 1997. According to the tag-based approach, a stored digital video file is parsed to generate "tag information" about individual frames within the file.

[0008] Specifically, the tag file contains information about the state of one or more state machines that are used to decode the digital representation. The state information varies depending on the specific technique used to encode the audio-visual work. For MPEG-2 files, for example, the tag file includes information about the state of the program elementary stream state machine, the video state machine, and the transport layer state machine.

[0009] During the performance of the audio-visual work, data from the digital representation is sent from a video pump to a decoder. The information in the tag file is used to perform seek, fast forward, fast rewind, slow forward and slow rewind operations during the performance of the audio-visual work. Seek operations are performed by causing the video pump to stop transmitting data from the current position in the digital representation, and to start transmitting data from a new position in the digital representation. The information in the tag file is inspected to determine the new position from which to start transmitting data. To ensure that the data stream transmitted by the video pump maintains compliance with the applicable video format, prefix data that includes appropriate header information is transmitted by the video pump prior to transmitting data from the new position.

[0010] Fast forward, fast rewind, slow forward and slow rewind operations are performed by selecting video frames based on the information contained in the tag file and the desired presentation rate, and generating a data stream containing data that represents the selected video frames. The selection process takes into account a variety of factors, including the data transfer rate of the channel on which the data is to be sent, the frame type of the frames, a minimum padding rate, and the possibility of a buffer overflow on the decoder. Prefix and suffix data are inserted into the transmitted data stream before and after the data for each frame in order to maintain compliance with the data stream format expected by the decoder.

[0011] The tag-based approach works well when there is enough time between the creation of the original digital video stream and the viewing of the digital video stream to allow the original digital video stream to be parsed to generate tag information. However, when the digital video stream is being viewed as it is being generated, parsing the digital video stream becomes impractical. The amount of computational power required to parse the digital video stream

as it arrives would be prohibitively expensive. On the other hand, it is not considered acceptable to increase the latency between the occurrence of many types of video feeds (e.g. sporting events) and the time at which such feeds are available for audience viewing.

[0012] When a video stream is made available for viewing before generation of the stream has been completed, the video stream is said to be a "live feed". At a professional level, non-linear digital editors can be used to rapidly review footage of a live feed for a single user. However, these systems are not intended for and cannot be easily adapted to serve many users. For example, if a hundred users were watching the same live feed but wanted to rewind, pause, and fast forward the feed at different times, each would require a separate non-linear digital editor.

[0013] Another problem associated with providing non-linear access to live digital video streams is that users may attempt to fast forward into portions of the video stream that do not yet exist. For example, a viewer may attempt to fast forward a live feed to see the final score of game which, in reality, has not yet ended. It is desirable to provide techniques for handling these types of situations in a way that ensures that the decoder will not freeze nor the video stream become corrupted.

[0014] Based on the foregoing, it is clearly desirable to provide a method and apparatus for sequentially displaying non-sequential frames of a live digital video. It is further desirable to provide such non-sequential access to live digital video in a way that does not require each viewer to operate prohibitively expensive hardware. It is also desirable to provide safeguards against attempts to access portions of a live digital video stream that do not yet exist.

SUMMARY OF THE INVENTION

[0015] A method and system for storing a continuous feed of video is provided. According to one aspect of the invention, the continuous feed is encoded in a digital video format to produce a digital data stream. A series of content files is created by repeatedly performing the steps of (1) storing the digital data stream in a current file, and (2) establishing a new file as the current file when the current file satisfies a predetermined condition. If the series of content files contains more than a predetermined amount of the continuous feed, the oldest content file in the series of content files is deleted.

[0016] According to another aspect of the invention, the deletion of the oldest content file may be delayed if it is currently being accessed. The deletion may be delayed until the oldest content file is no longer accessed, or until a certain amount of time has passed since the time at which it would have been deleted but for the fact that it was being accessed.

[0017] According to another aspect of the invention, tag information that indicates information about frames contained in the digital data stream is generated. The tag information includes timestamps that indicate timing of frames relative to a beginning of the digital data stream. An initial time value that indicates an absolute time that corresponds to the beginning of the digital data stream.

[0018] When a request from a client for playback beginning at a specified absolute time is received, the initial time value is subtracted from the specified absolute time to determine a relative time. The tag information is used to identify a location in the digital data stream that corresponds to the relative time. The digital data stream is then transmitted to the client beginning at the location in the digital data stream that corresponds to the relative time.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1 is a block diagram that illustrated a video delivery system according to an embodiment of the invention;
Figure 2A is a block diagram that illustrates the format of an MPEG file;
Figure 2B is a block diagram of an exemplary tag file according to an embodiment of the invention;
Figure 2C is a block diagram illustrating the tag information generated for each frame in an MPEG-1 file according to an embodiment of the invention;
Figure 3A is a block diagram illustrating a storage system that uses RAID error correction techniques according to an embodiment of the invention;
Figure 3B is a block diagram illustrating a storage system that combines RAID error correction and disk striping according to an embodiment of the invention;
Figure 4 is a block diagram illustrating a series of content files used to store the content of a continuous feed according to an embodiment of the invention; and
Figure 5 is a block diagram illustrating the migration of tag information from an old tag file to a new tag file in response to the expiration of tag data within the old tag file.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

[0020] A method and apparatus for providing non-sequential access to a live digital video stream is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

FUNCTIONAL OVERVIEW

[0021] According to one aspect of the invention, the difficulty associated with applying the tag-based approach to live digital video feeds is addressed by eliminating the need to parse an incoming digital video stream in real time. Instead of generating tag data by parsing the digital video stream, the unit responsible for encoding the live feed retains information about how the data was encoded and transmits that information to the video server along with the encoded data. The tag information arrives at the video server along with the corresponding content, so the content itself does not have to be parsed.

[0022] According to another aspect of the invention, the video server is configured to ensure that the client cannot seek or scan past the end of the received content. Due to the fact that there will be some amount of skew between the arrival time of the content and the corresponding tags, the server is configured to make sure that tags are not used prematurely, i.e. such that they would cause the server to go past the end of the available content.

EXEMPLARY SYSTEM

[0023] Figure 1 is a block diagram illustrating an exemplary audio-visual information delivery system 100 for delivering and providing non-sequential access to live digital video feeds. Audio-visual information delivery system 100 generally includes an encoder 101, a video server 106, a Media Data Store (MDS) 112, a database 116, a stream server 118, a video pump 120, and a client 122.

THE ENCODER

[0024] Encoder 101 receives audio visual input and generates a digital stream of data that encodes the audio visual input according to a particular format. Numerous video encoding formats have been developed and are well known in the industry. For example, the MPEG formats are described in detail in the following international standards: ISO/IEC 13818-1, 2, 3 (MPEG-2) and ISO/IEC 11172-1, 2, 3 (MPEG-1). Documents that describe these standards (hereafter referred to as the "MPEG specifications") are available from ISO/IEC Copyright Office Case Postale 56, CH 1211, Genève 20, Switzerland. While specific formats may be referenced herein for the purposes of explanation, the present invention is not restricted to any particular digital stream format.

[0025] Encoder 101 includes a Coder/Decoder (CODEC) 102 and a multiplexer (MUX) 104. CODEC 102 converts visual or audio-visual information from an input source to compressed digital data. CODEC 102 may be, for example, a fractal compressor or an MPEG compressor. For the purposes of illustration, it shall be assumed that the video source being captured by CODEC 102 is a live source and, consequently, CODEC 102 is encoding video at 1X relative to real time. However, the video source may alternatively be a stored video source which CODEC 102 encodes at any rate relative to real time.

[0026] MUX 104 multiplexes the compressed audio and visual information generated by CODEC 102 to generate a compressed video stream. In the compressed video stream, the data representing video frames and audio are merged and formatted according to the particular digital format supported by encoder 101. The specific operations performed during the merging process will vary based on the type of encoding employed. For example, the merging process may involve determining the order and placement of portions of digitized audio and video in the stream and inserting metadata at various points within the stream. The metadata may take the form, for example, of header information that identifies the starting point and content of "packets" within the stream. The stream of compressed audio-visual information constructed by MUX 104 is transmitted from the encoder 101 to the video server 106 over a communication channel 128.

CONTROL INFORMATION

[0027] According to one aspect of the invention, the encoder 101 sends control information to the video server 106 over a communication channel 130 in parallel with the video stream. The control information sent over channel 130 includes specific information about how the encoder 101 constructed the video stream. This control information includes

tag data that will be used by the stream server 118 to provide non-sequential access to the video stream. Specifically, the control information may include information about the type, length, and boundaries of the various frames encoded in the video stream as well as header information that specifies the compression ratio, the bit rate, and other types of information the video server 106 requires to determine how to process the video stream.

[0028] Significantly, the generation of the control information involves minimal additional computational power because MUX 104 generates most of the information already during the construction of the content stream. Specifically, MUX 104 arranges and encapsulates the digital video and audio data from CODEC 102. Since MUX 104 is packaging the content, MUX 104 knows the contents of and boundaries between the packages.

COMMUNICATION BETWEEN THE ENCODER AND THE VIDEO SERVER

[0029] While CODEC 102 will typically be implemented in hard-wired circuitry, MUX 104 is preferably implemented by program-controlled circuitry, such as a processor programmed to execute a particular sequence of instructions that are stored in a memory. Consequently, MUX 104 may include a processor executing a conventional multiplexing program that has been linked with and makes calls to a software library that controls the communication with the video server 106.

[0030] All data transmitted between the encoder 101 and the video server 106 is preferably sent using a reliable communication mechanism. According to one embodiment, the video content on communication channel 128 is handled as a simple bytestream and is transmitted via a lightweight, reliable protocol. For example, TCP is sufficient on lightly loaded networks. The control information and metadata on communication channel 130 contain more complicated data types and are sent via an object oriented protocol, such as the Common Object Resource Broker Architecture Interface Definition Language ("CORBA IDL").

[0031] Communication between the encoder 101 and the video server 106 occurs in sessions. According to one embodiment, a session is performed in three phases: OPEN, SEND and CLOSE. The operations performed during each of the phases is as follows:

OPEN - any provisioning that the video server 106 needs to perform for network or disk bandwidth or disk space occurs. A pipe for the video stream data (the "content") is created.

SEND TAGS and SEND DATA - these calls are made multiple times as content is encoded. The video server 106 stores all content immediately to disk and updates an end-of-file position. Tags are held in memory until the accompanying content data has been stored. Tags are held for an additional period of time to guarantee that a seek to that tag will succeed, i.e. that video pump 120 will not starve for data.

CLOSE - content pipe is torn down. Server resources are released and content services and clients are notified that the feed has become a normal static piece of content.

Encoder 101 generates content data and control data in parallel. However, the control data associated with a particular portion of content is not necessarily generated by encoder 101 at the same time as the particular portion of content. For example, encoder 101 may actually determine how it is going to line up content frames before the encoder 101 actually lines up the frames. Under these circumstances, the control data that indicates how the frames are lined up may be transmitted by encoder 101 before the content data that contains the frames.

THE VIDEO SERVER

[0032] Video server 106 receives the video stream and control data from encoder 101 and causes the data to be stored in MDS 112. In the illustrated system, the video server 106 sends an MPEG video stream to MDS server 110, and MDS server 110 stores the MPEG video stream in an MPEG file 134. In parallel, the video server 106 sends to the MDS server 110 tag information extracted from the control data received on line 130. The tag data is stored in a tag file 132 on disks 114. The video server 106 may also send information about the content, including tag data, to be stored in database 116.

[0033] Once tag data is transmitted by video server 106, any other entity in the system, including video pump 120, may use the tag data to attempt to access the content associated with the tag data. Consequently, the immediate transmission of tag data to MDS server 110 may result in errors when, for example, the tag data arrives at video server 106 before the corresponding content data. Therefore, prior to sending the tag data to MDS server 110, video server 106 buffers each tag data item in a tag buffer 108 until it is safe for entities, such as video pump 120, to access the content associated with the tag data item. The use of tag buffer 108 to avoid premature reads of content data is described in greater detail hereafter.

EXEMPLARY MPEG FILE

[0034] Digital audio-visual storage formats, whether compressed or not, use state machines and packets of various structures. The techniques described herein apply to all such storage formats. While the present invention is not limited to any particular digital audio-visual format, the MPEG-2 transport file structure shall be described for the purposes of illustration.

[0035] Referring to Figure 2a, it illustrates the structure of an MPEG-2 transport file 134 in greater detail. The data within MPEG file 134 is packaged into three layers: a program elementary stream ("PES") layer, a transport layer, and a video layer. These layers are described in detail in the MPEG-2 specifications. At the PES layer, MPEG file 134 consists of a sequence of PES packets. At the transport layer, the MPEG file 134 consists of a sequence of transport packets. At the video layer, MPEG file 134 consists of a sequence of picture packets. Each picture packet contains the data for one frame of video.

[0036] Each PES packet has a header that identifies the length and contents of the PES packet. In the illustrated example, a PES packet 250 contains a header 248 followed by a sequence of transport packets 251-262. PES packet boundaries coincide with valid transport packet boundaries. Each transport packet contains exclusively one type of data. In the illustrated example, transport packets 251, 256, 258, 259, 260 and 262 contain video data. Transport packets 252, 257 and 261 contain audio data. Transport packet 253 contains control data. Transport packet 254 contains timing data. Transport packet 255 is a padding packet.

[0037] Each transport packet has a header. The header includes a program ID ("PID") for the packet. Packets assigned PID 0 are control packets. For example, packet 253 may be assigned PID 0. Other packets, including other control packets, are referenced in the PID 0 packets. Specifically, PID 0 control packets include tables that indicate the packet types of the packets that immediately follow the PID 0 control packets. For all packets which are not PID 0 control packets, the headers contain PIDs which serve as a pointers into the table contained in the PID 0 control packet that most immediately preceded the packets. For example, the type of data contained in a packet with a PID 100 would be determined by inspecting the entry associated with PID 100 in the table of the PID 0 control packet that most recently preceded the packet.

[0038] In the video layer, the MPEG file 134 is divided according to the boundaries of frame data. As mentioned above, there in no correlation between the boundaries of the data that represent video frames and the transport packet boundaries. In the illustrated example, the frame data for one video frame "F" is located as indicated by brackets 270. Specifically, the frame data for frame "F" is located from a point 280 within video packet 251 to the end of video packet 251, in video packet 256, and from the beginning of video packet 258 to a point 282 within video packet 258. Therefore, points 280 and 282 represent the boundaries for the picture packet for frame "F". The frame data for a second video frame "G" is located as indicated by brackets 272. The boundaries for the picture packet for frame "G" are indicated by bracket 276.

[0039] Structures analogous to those described above for MPEG-2 transport streams also exist in other digital audio-visual storage formats, including MPEG-1, Quicktime, AVI, Proshare and H.261 formats. In the preferred embodiment, indicators of video access points, time stamps, file locations, etc. are stored such that multiple digital audio-visual storage formats can be accessed by the same server to simultaneously serve different clients from a wide variety of storage formats. Preferably, all of the format specific information and techniques are incorporated in the tag generator and the stream server. All of the other elements of the server are format independent.

EXEMPLARY TAG FILE

[0040] The contents of an exemplary tag file 132 shall now be described with reference to Figure 2b. In Figure 2b, the tag file 132 includes a file type identifier 202, a length indicator 204, a bit rate indicator 206, a play duration indicator 208, a frame number indicator 210, stream access information 212 and an initial MPEG time offset 213. File type identifier 202 indicates the physical wrapping on the MPEG file 134. For example, file type identifier 202 would indicate whether MPEG file 134 is a MPEG-2 or an MPEG-1 file.

[0041] Length indicator 204 indicates the length of the MPEG file 134. Bit rate indicator 206 indicates the bit rate at which the contents of the MPEG file 134 should be sent to a client during playback. The play duration indicator 208 specifies, in milliseconds, the amount of time required to play back the entire contents of MPEG file 134 during a normal playback operation. Frame number indicator 210 indicates the total number of frames represented in MPEG file 134.

[0042] Stream access information 212 is information required to access the video and audio streams stored within MPEG file 134. Stream access information 212 includes a video elementary stream ID and an audio elementary stream ID. For MPEG-2 files, stream access information 212 also includes a video PID and an audio PID. The tag file header may also contain other information that may be used to implement other features.

[0043] In addition to the general information described above, the tag file 132 contains an entry for each frame within the MPEG file 134. The entry for a video frame includes information about the state of the various MPEG layers relative

to the position of the data that represents the frame. For an MPEG-2 file, each entry includes the state of the MPEG-2 transport state machine, the state of the program elementary stream state machine and the state of the video state machine. For an MPEG-1 file, each entry includes the current state of the Pack system MPEG stream and the state of the video state machine.

[0044] Tag file entry 214 illustrates in greater detail the tag information that is stored for an individual MPEG-2 video frame "F". With respect to the state of the program elementary stream state machine, the tag entry 214 includes the information indicated in Table 1.

TABLE 1

DATA	MEANING
PES OFFSET AT THE START OF PICTURE 217	The offset, within the PES packet that contains the frame data for frame "F" of the first byte of the frame data for frame "F".
PES OFFSET AT THE END OF PICTURE 219	The offset between the last byte in the frame data for frame "F" and the end of the PES packet in which the frame data for frame "F" resides.

[0045] With respect to the state of the video state machine, tag entry 214 includes the information indicated in Table 2.

TABLE 2

DATA	MEANING
PICTURE SIZE 220	The size of the picture packet for frame "F".
START POSITION 226	The location within the MPEG file of the first byte of the data that corresponds to frame "F"
TIME VALUE 228	The time, relative to the beginning of the movie, when frame "F" would be displayed during a normal playback of MPEG file 134.
FRAME TYPE 232	The technique used to encode the frame (e.g. I-frame, P-frame or B-frame).
TIMING BUFFER INFORMATION 238	Indicates how full the buffer of the decoder is (sent to the decoder to determine when information should be moved out of the buffer in order to receive newly arriving information).

[0046] With respect to the state of the transport layer state machine, tag entry 214 includes the information indicated in Table 3.

TABLE 3

DATA	MEANING
START OFFSET 234	The distance between the first byte in the frame data and the start of the transport packet in which the first byte resides.
# OF NON-VIDEO PACKETS 222	The number of non-video packets (i.e. audio packets, padding packets, control packets and timing packets) that are located within the picture packet for frame "F".
# OF PADDING PACKETS 224	The number of padding packets that are located within the picture packet for frame "F".
END OFFSET 236	The distance between the last byte in the frame data and the end of the packet in which the last byte resides.
CURRENT CONTINUITY COUNTER 215	The Continuity value associated with frame "F".
DISCONTINUITY FLAG 230	Indicates whether there is a discontinuity in time between frame "F" and the frame represented in the previous tag entry.

[0047] Assume, for example, that entry 214 is for the frame "F" of Figure 2b. The size 220 associated with frame "F" would be the bits encompassed by bracket 274. The number 222 of non-video packets would be five (packets 252,

253, 254, 255 and 257). The number 224 of padding packets would be one (packet 255). The start position 226 would be the distance between the beginning of MPEG file 134 and point 280. The start offset 234 would be the distance between the start of packet 251 and point 280. The end offset 236 would be the distance between point 282 and the end of packet 258.

[0048] The tag information generated for each frame in an MPEG-1 file is illustrated in Figure 2c. Referring to Figure 2c, entry 214 includes data indicating the state of three state machines: a system state machine, a pack state machine, and a video state machine. Specifically, tag entry 214 includes the information shown in Table 4.

TABLE 4

DATA	MEANING
AMOUNT OF NON-VIDEO DATA 221	The amount of non-video data (in bytes) contained within the start and end boundaries of the frame data for frame "F".
AMOUNT OF PADDING DATA 223	The amount of padding data (m bytes) contained within the start and end boundaries of the frame data for frame "F".
PACK OFFSET AT START 225	The offset between the start boundary of the frame data for frame "F" in the beginning of the pack packet that contains the start boundary for frame "F",
PACK REMAINING AT START 227	The distance between the start boundary for frame "F" and the end of the pack packet that contains the start boundary of frame "F".
PACK OFFSET AT END 229	The offset between the end boundary for frame "F" in the beginning of the packet that contains the end boundary for frame "F".
PACK REMAINING AT END 231	The distance between the end boundary for frame "F" and the end of the pack packet that contains the end boundary of frame "F".
PICTURE SIZE 233	The distance (in bytes) between the start boundary for frame "F" and the end boundary for frame "F".
PICTURE START POS 235	The distance between the start of the MPEG-1 file and the start boundary for frame "F".
PICTURE END POS 237	The position, relative to the beginning of the MPEG-1 file, of the end boundary for frame "F".
FRAME TYPE 239	The technique used to encode the data that represents frame "F".
TIME VALUE 241	The time, relative to the beginning of the movie, when frame "F" would be displayed during a normal playback of MPEG file 134.
TIMING BUFFER INFO 243	Indicates how full the decoder is (sent to the decoder to determine when information should be moved out of the buffer in order to receive newly arriving information).

[0049] The tag information includes data indicating the state of the relevant state machines at the beginning of video frames. However, the state machines employed by other digital audio-visual formats differ from those described above just as the state machines employed in the MPEG-1 format differ from those employed in MPEG-2. Consequently, the specific tag information stored for each frame of video will vary based on the digital audio-video format of the file to which it corresponds.

THE MDS

[0050] MDS 112 includes MDS server 110 and one or more non-volatile storage devices, such as disks 114. In the illustrated embodiment, MPEG file 134 is stored across numerous disks 114 to increase the fault tolerance of the system. Consider, for example, the multi-disk system 300 illustrated in Figure 3. System 300 includes N+1 disk drives. An MPEG file is stored on N of the N+1 disks. The MPEG file is divided into sections 350, 352, 354 and 356. Each section is divided into N blocks, where N is the number of disks that will be used to store the MPEG file. Each disk stores one block from a given section.

[0051] In the illustrated example, the first section 350 of the MPEG file includes blocks 310, 312 and 314 stored on disks 302, 304 and 306, respectively. The second section 352 includes blocks 316, 318 and 320 stored on disks 302,

304 and 306, respectively. The third section 354 includes blocks 322, 324 and 326 stored on disks 302, 304 and 306, respectively. The fourth section 356 includes blocks 328, 330 and 332 stored on disks 302, 304 and 306, respectively.

[0052] The disk 308 which is not used to store the MPEG file is used to store check bits. Each set of check bits corresponds to a section of the MPEG file and is constructed based on the various blocks that belong to the corresponding section. For example, check bits 334 corresponds to section 350 and is generated by performing an exclusive OR operation on all of the blocks in the first section 350. Similarly, check bits 336, 338 and 340 are the products of an exclusive OR performed on all of the blocks in the section 352, 354 and 356, respectively.

[0053] System 300 has a higher fault tolerance than a single disk system in that if any disk in the system ceases to operate correctly, the contents of the bad disk can be reconstructed based on the contents of the remaining disks. For example, if disk 304 ceases to function, the contents of block 312 can be reconstructed based on the remaining blocks in section 350 and the check bits 334 associated with section 350. Similarly, block 318 can be constructed based on the remaining blocks in section 352 and the check bits 336 associated with section 352. This error detection and correction technique is generally known as "Redundant Array of Inexpensive Disks" or RAID.

[0054] During real-time playback using RAID, video pump 120 reads and processes the MPEG file on a section by section basis so that all of the information is available to reconstruct any faulty data read from disk. Techniques for performing RAID in real time are described in U.S. Patent No. 5,623,595, entitled "METHOD AND APPARATUS FOR TRANSPARENT, REAL TIME RECONSTRUCTION OF CORRUPTED DATA IN A REDUNDANT ARRAY DATA STORAGE SYSTEM".

[0055] During normal playback operations, there is sufficient time to perform the disk accesses required to read an entire section while the data from the previous section is being transmitted in the MPEG data stream. However, during fast forward and fast rewind operations, less than all of the data in any section will be sent in the MPEG data stream. Because less data is sent, the transmission of the data will take less time. Consequently, less time will be available to read and process the subsequent section.

[0056] For example, assume that only one frame X from section 350 was selected for display during a fast forward operation. During the time it takes to transmit the segment for frame X, the data for the next selected frame Y is read and processed. Assume that the next frame Y is located in section 352. If the MPEG file is read and processed on a section by section basis (required for RAID), then all of the blocks in section 352 are read and processed during the transmission of the single frame X. Even if it were possible to read and process all of the blocks in section 352 in the allotted time, it may still be undesirable to do so because of the resources that would be consumed in performing the requisite disk accesses.

[0057] In light of the foregoing, video pump 120 does not use RAID during fast forward and fast rewind operations. Rather, video pump 120 reads, processes and transmits only the data indicated in the commands it receives from the stream server 118. Thus, in the example given above, only the frame data for frame Y would be read and processed during the transmission of the segment for frame X. By bypassing RAID during fast forward and fast rewind operations, disk bandwidth remains at the same level or below that used during normal playback operations.

[0058] Since RAID is not used during real-time fast forward and fast rewind operations, faulty data cannot be reconstructed during these operations. Consequently, when the video pump 120 detects that the data for a selected frame is corrupted or unavailable, the video pump 120 discards the entire segment associated with the problem frame. Thus, if the data associated with a frame cannot be sent, then the prefix and suffix data for the frame is not sent either. However, any padding packets that were to be sent along with the prefix or suffix data will still be sent.

[0059] By sending data in entire "segments", conformance with the digital audio-visual format is maintained. In one embodiment, the video pump 120 will send down padding packets to fill the line to maintain the correct presentation rate. In the preferred embodiment, this behavior is selectable by the client.

DATA STRIPING

[0060] The RAID techniques described above improve both throughput (because all data from all disks in an array are read in parallel) and reliability (due to error correction). To further increase throughput, RAID may be used in conjunction with data striping. Using data striping, segments of logically sequential data are written to multiple physical devices (or sets of physical devices) in a round-robin fashion. The amount of data stored at each storage element in the round-robin sequence is referred to as a "stripe". When each storage element in the round-robin sequence is an array of RAID disks, each segment of data is referred to as a RAID stripe.

[0061] Figure 3B illustrates a system in which data striping is used in conjunction with RAID. The system of Figure 3B is similar to that of Figure 3A with the exception that each of the disks in Figure 3A has been replaced by a series of M disks. Thus, disk 302 has been replaced by disks 302-1 through 302-M. The segment portions stored on disk 302 have been stored on disks 302-1 to 302-M in a sequential, round robin fashion. For example, assume that the MPEG file has been divided into 50 segments and that disk 302 has been replaced with 25 disks. Under those circumstances, disk 302-1 would store the first portion of segments 1 and 26. Disk 302-2 would store the first portion of segments 2

and 27, etc.

[0062] Data striping increases throughput because different processes can be reading from different disk arrays in parallel. For example, one data pump may be reading the first segment of an MPEG file from the RAID array that includes Disk_1,1 through Disk_1,N+1, while another data pump is concurrently reading the second segment of the same MPEG file from the RAID array that includes Disk_2,1 through Disk_2,N+1.

[0063] For throughput performance reasons, reads and writes occur in discrete chunks, typically disk RAID stripes. In a typical digital video delivery system, each access unit is 256kB or 2 megabits, and the content is 2Mb/sec MPEG. Consequently, each RAID stripe corresponds to approximately one second of video, though this could easily range from about .2 to 10 seconds per stripe depending on content bit rate and server configuration.

THE CLIENT

[0064] Audio-visual information delivery system 100 contains one or more clients, such as client 122. Client 122 generally represents devices configured to decode audio-visual information contained in a stream of digital audio-visual data. For example, client 122 may be a set top converter boxes coupled to an output display, such as television. Client 122 includes a decoder 126 for decoding a digital data stream, and a control unit 124 for communicating information to the stream server 118.

[0065] Stream server 118 is able to receive information from client 122 over a control network 140. Control network 140 may be any network that allows communication between two or more devices. For example, control network 140 may be a high bandwidth network, an X.25 circuit or an electronic industry association (EIA) 232 (RS - 232) serial line.

[0066] The client 122 communicates with the stream server 118 and database 116 via the control network 140. For example, client 122 may send a query to database 116 requesting information about what is currently available for viewing. The database 116 responds by sending the requested information back to the client 122. The user of client 122 may then select to view a particular audio-visual work beginning at a particular location and at a particular speed. Client 122 transmits requests to initiate the transmission of audio-visual data streams and control information to affect the playback of ongoing digital audio-visual transmissions through network 140 to stream server 118.

THE VIDEO PUMP AND STREAM SERVER

[0067] The video pump 120 is coupled to the stream server 118 and receives commands from the stream server 118. The video pump 120 is coupled to the disks 114 such that the video pump 120 stores and retrieves data from the disks 114.

[0068] In addition to communicating with the stream server 118, the client 122 receives information from the video pump 120 through a high bandwidth network 150. The high bandwidth network 150 may be any of type of circuit-style network link capable of transferring large amounts of data. A circuit-style network link is configured such that the destination of the data is guaranteed by the underlying network, not by the transmission protocol. For example, the high bandwidth network 150 may be an asynchronous transfer mode (ATM) circuit or a physical type of line, such as a T1 or E1 line. In addition, the high bandwidth network 150 may utilize a fiber optic cable, twisted pair conductors, coaxial cable, or a wireless communication system, such as a microwave communication system.

[0069] Network 150 may alternatively be a relatively low bandwidth network, or a combination of high bandwidth and low bandwidth communication mediums. For example, a portion of network 150 may comprise a relatively high bandwidth ATM circuit, while a relatively low bandwidth device such as a 28.8K modem is used downstream to deliver the video information from the network to the client 122.

[0070] The audio-visual information delivery system 100 permits a server, such as the video pump 120, to transfer large amounts of data from the disks 114 over the high bandwidth network 150 to the client 122 with minimal overhead. In addition, the audio-visual information delivery system 100 permits the client 122 to transmit requests to the stream server 118 using a standard network protocol via the control network 140. In a preferred embodiment, the underlying protocol for the high bandwidth network 150 and the control network 140 is the same. The stream server 118 may consist of a single computer system, or may consist of a plurality of computing devices configured as servers. Similarly, the video pump 120 may consist of a single server device, or may include a plurality of such servers.

[0071] To receive a digital audio-visual data stream from a particular digital audio-visual file, client 122 transmits a request to the stream server 118. In response to the request, the stream server 118 transmits commands to the video pump 120 to cause video pump 120 to transmit the requested digital audio-visual data stream to the client that requested the digital audio-visual data stream. For live feeds, the video server 106 will be storing the video stream into the video file 134 at the same time the video pump 120 is sending a video stream from the file 134 to the client 122.

[0072] The commands sent to the video pump 120 from the stream server 118 include control information specific to the client request. For example, the control information identifies the desired digital audio-visual file, the beginning offset of the desired data within the digital audio-visual file, and the address of the client. In order to create a valid

digital audio-visual stream at the specified offset, the stream server 118 also sends "prefix data" to the video pump 120 and requests the video pump 120 to send the prefix data to the client. As shall be described in greater detail hereafter, prefix data is data that prepares the client to receive digital audio-visual data from the specified location in the digital audio-visual file.

[0073] The video pump 120, after receiving the commands and control information from the stream server 118, begins to retrieve digital audio-visual data from the specified location in the specified digital audio-visual file on the disks 114. For the purpose of explanation, it shall be assumed that audio-visual information delivery system 100 delivers audio-visual information in accordance with one or more of the MPEG formats. Consequently, video pump 120 will retrieve the audio-visual data from an MPEG file 134 on the disks 114.

[0074] The video pump 120 transmits the prefix data to the client, and then seamlessly transmits MPEG data retrieved from the disks 114 beginning at the specified location to the client. The prefix data includes a packet header which, when followed by the MPEG data located at the specified position, creates an MPEG compliant transition packet. The data that follows the first packet is retrieved sequentially from the MPEG file 134, and will therefore constitute a series of MPEG compliant packets. The video pump 120 transmits these packets to the requesting client via the high bandwidth network 150.

[0075] The requesting client receives the MPEG data stream, beginning with the prefix data. The client decodes the MPEG data stream to reproduce the audio-visual sequence represented in the MPEG data stream.

PREMATURE READ AVOIDANCE

[0076] When client 122 is playing an MPEG stream at the same time the MPEG stream is being generated by encoder 101, safeguards should be taken to ensure that client 122 does not stall (because it has reached the end of valid content data) or play bad data (because it has read beyond the end of the currently available content data). If the video pump 120 prematurely reads a stripe of disks 114, video pump 120 will send invalid data to the client 122, resulting in the display of unintended content or garbage (dirty content). Such a premature read will occur if, for example, a user requests display of a portion of the video stream that has not yet been stored on disks 114. To prevent this, end-of-file information for MPEG file 134 is maintained to indicate the current end-of-file 134. As more content data is added to file 134, the end-of-file information is updated so that the new data may be accessed.

[0077] One approach to avoid premature reads is to repeatedly update a table of contents on disks 114 with a new end-of-file value, and have the video pump 120 check this value before reading stripes from disks 114. The MDS server 110 updates the end-of-file to indicate that the content file 134 includes new content only after it has been verified that the new content has been successfully stored to disks 114. Unfortunately, unless this end-of-file information is guaranteed to be held in dynamic memory, this technique leads to a jitter in the latency period of updates that is difficult to predict.

[0078] Another approach to avoid premature reads is for the MDS server 110 to actively transmit the new end-of-file information to all processes that are reading the content. Thus, MDS server 110 stores content data into file 134 on disks 114, waits for verification that the content has been stored, and then transmits messages indicating the existence of the newly stored content to all processes reading the content data (e.g. video pump 120). The MDS server 110 may make such end-of-file notification messages periodically (e.g. after every 5 seconds) or after a predetermined amount of new content data has been successfully stored (e.g. after every 1 Megabyte). Unfortunately, notification times will also jitter due to variations in the content arrival times, which is a function of the encoder 101 and the network between the encoder 101 and the video server 106.

[0079] According to one embodiment, the tag information is used to indicate the current end-of-file. Specifically, video server 106 effectively updates the end-of-file of file 134 by sending tag information from tag buffer 108 for storage by MDS 112. As soon as the tag information that corresponds to a particular portion of content has been transmitted by video server 106, the video pump 120 is free to perform a seek to that particular portion of video. Until the tag information that corresponds to a particular portion of video is released, video pump 120 may not perform a seek to the corresponding portion of video. Because the newest tag information indicates the current end-of-file, newly connected users may simply seek to the content associated with the newest tag information, and begin playing the feed at the real-time rate.

MINIMUM TAG DELAY PERIOD

[0080] To prevent client 122 from stalling or playing bad data, the transmission of tag data from tag buffer 108 to MDS 112 is delayed. Preferably, the duration of the delay is long enough to ensure that the associated content data will not be prematurely accessed. On the other hand, delaying the tag data longer than necessary increases the latency between when content is encoded and when users can seek or scan to the content. Consequently, it is desirable to determine a minimum tag delay period, and to buffer tag data in tag buffer 108 for the minimum tag delay period. The

minimum tag delay period for a tag data item is determined by the maximum latency associated with delivering the corresponding content data from encoder 101 to video pump 120.

[0081] Video server 106 includes a network buffer 152 and a write buffer 154. Typically, the video server 106 will be reading content data from channel 128 into network buffer 152 at the same time that video server 106 is writing content data from write buffer 154 to disks 114. In embodiments that use RAID storage techniques, content data is received and buffered within video server 106 in units that correspond to one RAID stripe.

[0082] Video pump 120 includes a prefetch unit 146 and a buffer 144. Video pump 120 reads content data from disks 114 asynchronously. To read content data, prefetch unit 146 requests the transmission of a particular portion of content data, and disks 114 respond by either sending the requested content data or by indicating that the requested data cannot be sent. Some latency occurs between the time the prefetch unit 146 requests data, and the time the data is received by video pump 120.

[0083] When content data from file 134 arrives at video pump 120, video pump 120 stores the content data from file 134 into the buffer 144. As bandwidth becomes available on network 150, video pump 120 transmits content data from the buffer 144 over network 150 to client 122. As with the video server 106, content data is pre-fetched and buffered within video pump 120 in units that correspond to one RAID stripe when RAID storage techniques are used.

[0084] As explained above, the video pump 120 is typically copying data from one RAID stripe into network buffers and prefetching the following stripe. Likewise, the video server 106 is typically writing one RAID stripe of content to the data store and receiving data from the network into a second memory buffer. Consequently, there are typically four RAID stripes " in transit", so the latency between when any content data is generated and when it is available to be played is approximately the time it takes to deliver four RAID stripes worth of data.

[0085] RAID stripes are usually 128K bits or 256K bits per disk. The combined total of all disks in a RAID stripe is therefore 1 to 2 Megabits. For typical MPEG files, each raid stripe will correspond to approximately one second of video. Consequently, having four RAID stripes in transit results in a minimum latency of approximately 4 seconds.

[0086] The implication for tag data is that a given tag may only be released by the video server 106 for use by other entities when the corresponding content is available to be played (i.e. has been successfully stored on disk for two seconds). Therefore, in a video delivery system where the content delivery has a four second latency, the tag data retained in tag buffer 108 is transmitted no earlier than four seconds after the generation of the corresponding content.

[0087] According to one embodiment, jitter and stalling are both avoided by transmitting a batch of tag data from tag buffer 108 to MDS 112 every twelve seconds. The tag data batch transmitted at every twelve second interval includes all tag information in tag buffer 108 that is at least twelve seconds old. Tag data that is less than twelve seconds old is retained in tag buffer 108 and transmitted to MDS 112 in a batch at the end of the next twelve second interval. MDS server 110 sends the tag data to the various entities (e.g. video pump 120) that are reading video file 134, and then stores the tag information on disks 114.

DIGITAL CHANNELS

[0088] Video files generated for specific audio-visual works, such as sporting events, will be of finite length. Consequently, their corresponding content files will also consume a finite amount of storage making it practical to perpetually store the entire content file for later viewing. However, a traditional television "channel" is composed of a never-ending sequence of audio-visual works. Perpetually retaining all of the content of a digital channel would continuously consume storage at an unacceptably high rate. On the other hand, it is desirable to allow users to view programs that they may not have been able to view at the time the programs were originally broadcast. For example, it would be desirable for a viewer to have access to the last 24 hours of programming that was broadcast over a digital channel. According to an embodiment of the invention, historical viewing for an infinite feed is provided through the use of a continuous finite buffer, where older data "expires" and is overwritten with new data.

CONTENT EXPIRATION

[0089] In order to have continuous buffer of data, for instance the last 24 hours of Lifetime, Television for Women, older content needs to be deleted along with the corresponding tags. Various approaches may be used to implement such a continuous buffer.

[0090] With respect to the content data, the simplest approach to implement a continuous buffer is to create a single file large enough to hold 24 hours of footage. The file is then treated as a circular buffer. Specifically, after the creation of the initial 24 hour file, the MDS server 110 would establish the beginning of the file as the current "insertion point". The MDS server 110 would then store new content data over the old data at the insertion point, and move the insertion point to the end of the new data. When the insertion point hits the end of the file, it wraps around again to the beginning of the file.

[0091] Unfortunately, this single-file circular buffer approach makes it difficult to grow or shrink the time of the file.

For example, assume that the insertion point is in the middle of the file, and a decision is made to expand the file to cover 48 hours. Under these circumstances, the MDS server 110 could not begin to extend the time covered for another 12 hours, when the insertion point would have reached the end of the file. Using the single circular buffer approach, it is also difficult to detect if a client has paused and had the "horizon" move over their position, such that when they resume the content they were watching has been overwritten.

[0092] Figure 4 illustrates an alternative, more flexible approach to buffering a predetermined amount of an infinite video feed. Referring to Figure 4, the content data is stored in a group of smaller files 402-414. Each of the smaller files stores a subset of the buffered content data. In the illustrated embodiment, each of files 402-412 store two hours worth of content. File 414 currently stores one hour of content. The current insertion point is at the end of file 414. When file 414 reaches two hours of content, file 414 will be closed and a new content file will be created. As content files age, the older content files are deleted to free up disk space for new files. During playback, files are joined together seamlessly by the video pump as the content data is sent to the client.

[0093] When the buffering technique illustrated in Figure 4 is used, a lenient expiration policy can be set. Specifically, a policy may be established that a file is not deleted until all clients have finished with the (file and any files that precede the file). For example, assume that users are allowed to access the last 12 hours of a feed. When file 414 is completed, files 404-414 will contain the most recent 12 hours, so file 402 is no longer required. However, a client may currently be viewing the contents of file 402. Consequently, file 402 is not immediately deleted. Rather, new clients are prevented from accessing file 402, but the client currently accessing file 402 is allowed to finish playing file 402. When the last client has finished playing file 402, the file 402 is deleted.

[0094] To put a cap on the number of existing files, a time limit may be established for clients to finish playing old files. For example, when file 414 is completed, not only are new clients prevented from accessing file 402, but the clients currently accessing file 402 are given two hours to finish playing file 402. At the end of two hours, file 402 is then deleted to free up disk space without regard to whether any clients were still reading file 402.

TAG EXPIRATION

[0095] When a content file (e.g. file 402) is deleted, the tags that correspond to the deleted content file are considered "expired", and therefore can also be deleted. Ideally, tags are stored in a format, such as a database table, that allows easy deletion of old tags as well as the addition of new ones. Unfortunately, the overhead associated with storing and retrieving tags from a database table may be too expensive to be practical under live feed conditions. For ease and speed of access, therefore, tags are typically stored in a flat file.

[0096] Referring to Figure 5, it illustrates a flat tag file 500. The flat tag file 500 includes a header 502 followed by a set of tags 504. The header 502 indicates information about the contents of tag file 500, including the set of content files to which the tags within tag file 500 correspond.

[0097] As new tags arrive, the tags are appended to tag file 500. Because tag file 500 is associated with a continuous feed, tag file 500 will grow indefinitely unless a mechanism is provided for deleting expired tags. However, tag file 500 itself should remain valid even after the expiration of some tags (e.g. tags 510) within the tag file 500, since clients may continue to access and use the tags 512 within tag file 500 that have not yet expired. Therefore, the expiration mechanism cannot simply delete the expired tags 510 from the tag file 500.

[0098] Rather than directly delete the expired tags from within tag file 500, a temporary tag file 514 is created by constructing a new header 506 and appending to the new header 506 a copy of the unexpired tags 512 from the old tag file 500. The new header 506 includes the same information as the old header 502, except that data within header 502 indicates that tag file 500 includes tags for the deleted content file, while data within header 506 does not.

[0099] While new tag file 514 is being created, new tag data is appended to both the new tag file 514 and the old tag file 500. After the new tag file 514 is created, new tag data is appended to the new tag file 514 rather than the old tag file 500. To ensure that the new tag data is appended after tag data 512, storage for the to-be-copied tags 512 is preallocated in the new tag file 514, and the new tags are appended after the preallocated storage while the existing tags 512 are copied into the preallocated storage.

[0100] When all of the unexpired tags 512 have been copied to the new tag file 514, the old tag file 500 is closed and the new tag file 514 is renamed over the top of the old tag file 500. After the new tag file 514 has been renamed, the tag file readers (e.g. stream server 118) that were using the old tag file 500 are reset based on the information contained in the header of the new tag file 514. According to one embodiment (the "push model"), messages are sent to the tag file readers to expressly inform them that the tag file has been modified, and that they should update themselves based on header information in the new tag file 514.

[0101] According to an alternative "pull model" embodiment, the tag file readers are not expressly informed. Rather, they are configured to read and update themselves based on the header information of the new tag file if they ever fail in an attempt to read a tag. The pull model approach has the benefit that it avoids the transmission of messages which, under many circumstances, are not necessary.

[0102] When tags associated with a particular content segment are deleted, clients may continue to view the content segment. However, the clients will not be able to perform non-sequential access operations that require the deleted tag information, such as fast forward and rewind.

TIMESTAMPING

[0103] Tag information includes timestamp information for each of the frames in the corresponding content data. For the purposes of decoding, the timestamp information typically represents time relative to the beginning of a feed (i.e. the "presentation time"), and is mapped to the byte offset in the content file of the frame that corresponds to that presentation time. However, for continuous feeds, such relative time values are not meaningful. For example, a user would want to request playback beginning at Jan 21, 1997 16:30:23, rather than beginning at 5,345,789.76 seconds from the time a station began broadcasting.

[0104] According to one embodiment of the invention, absolute time values are supported by storing an absolute time value that corresponds to the "zero" relative time value. Therefore, when a client specifies playback from an absolute time, the absolute time value associated with "zero" is subtracted from the specified absolute time value to yield a relative time value. The relative time value is then used by stream server 118 to identify the appropriate tag information, and the tag information is used by stream server 118 to cause video pump 120 to begin sending content from the appropriate location in the content file 134.

[0105] Typically, the transport formats of digital video provide a fixed number of bits (e.g. 33 bits) to represent timestamps. For continuous feed environments, the relative timestamp values will inevitably reach numbers that cannot be represented by the number of bits available in the transport format. When this occurs, the timestamp values " wrap" and begin again at zero.

[0106] To address the wrapping problem, a higher-precision wrap value (e.g. 64 bits) is maintained. When performing a seek or other non-sequential access, the stream server 118 uses the higher-precision timestamp values. When transmitting content to a client, the video pump 120 sends the lower-precision timestamps.

[0107] The video encoding and delivery techniques described herein empower users with control of functions that were previously in the exclusive domain of program providers. For example, program providers currently determine which plays of a SuperBowl will be replayed to viewers, the speed at which they will be replayed, and how many times they will be replayed.

[0108] However, viewers may have strongly differing opinions as to which plays merit multiple viewings. For example, two viewers may dispute the accuracy of a particular call. However, the program provider may not consider the play that gave rise to the call to be significant enough to replay the play. Using the techniques provided herein, viewers may determine for themselves which plays should be immediately replayed, at what speed they are replayed, and how many times they are replayed.

[0109] In the foregoing specification, the invention has been described with reference to specific embodiments tereof. It will, however, be evident that various modifications and changes may be made thereto. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

Claims

1. A method for providing non-sequential access to video from a continuous feed, the method including receiving at a server (106) a digital data stream produced by encoding said continuous feed in a digital video format, the method being **characterized by**:

   generating, at said server (106), tag information that indicates information about frames contained in said digital data stream, said tag information including timestamps that indicate timing of frames relative to a beginning of said digital data stream;
   storing an initial time value at said server (106) that indicates an absolute time that corresponds to said beginning of said digital data stream;
   receiving a request from a client (122) for playback beginning at a specified absolute time;
   subtracting said initial time value from said specified absolute time to determine a relative time;
   using said tag information to identify a location in said digital data stream that corresponds to said relative time; and
   transmitting said digital data stream to said client (122) beginning at said location in said digital data stream that corresponds to said relative time.

2. The method of Claim 1 wherein:

said server (106) uses a first precision timestamp when performing a seek or other non-sequential access; and said server (106) uses a second precision timestamp when transmitting content to a client (122), wherein the first precision is higher than said second precision.

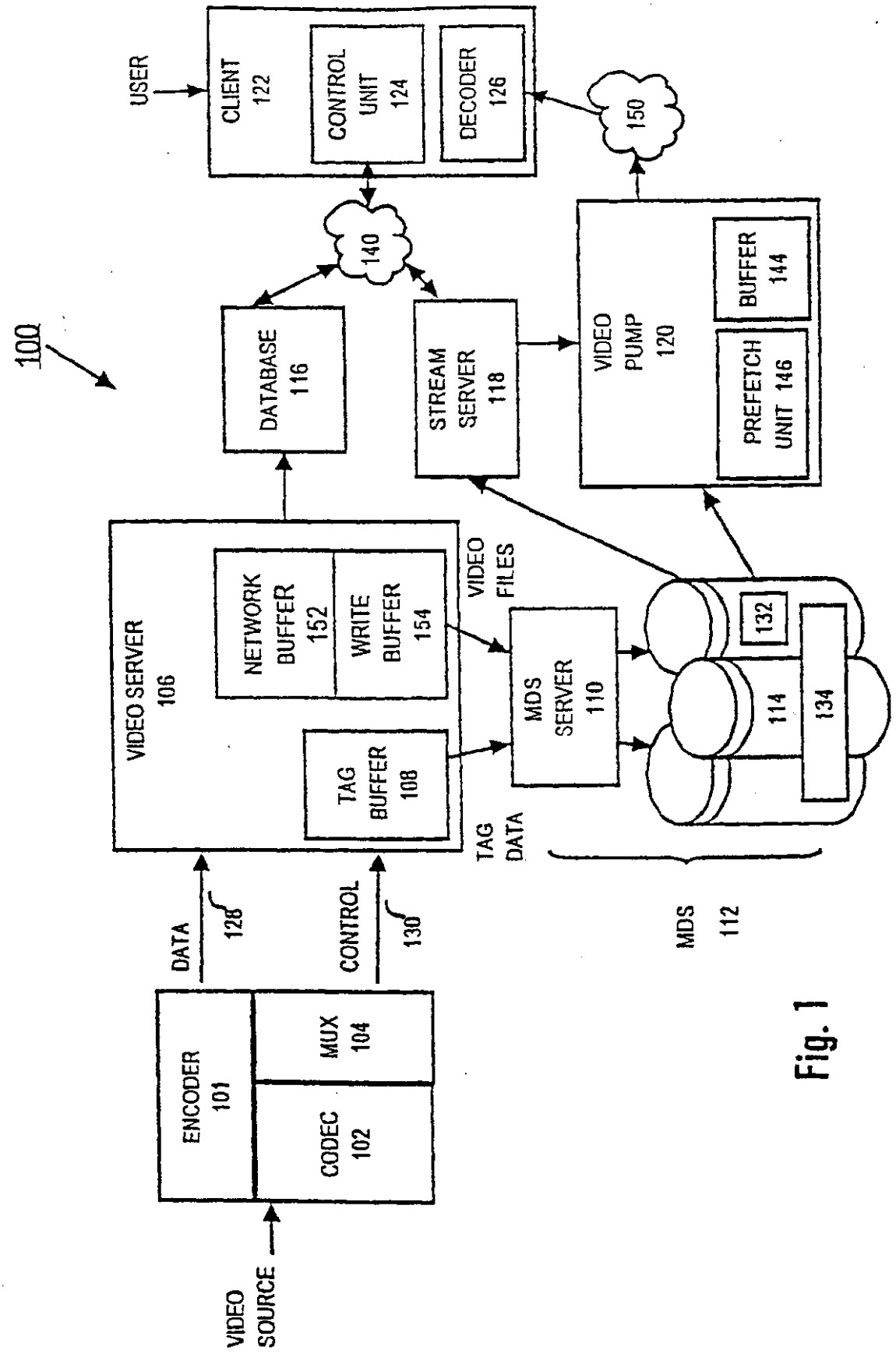
100
USER
CLIENT 122
CONTROL UNIT 124
DECODER 126
150
140
DATABASE 116
STREAM SERVER 118
VIDEO PUMP 120
BUFFER 144
PREFETCH UNIT 146
VIDEO SERVER 106
NETWORK BUFFER 152
WRITE BUFFER 154
VIDEO FILES
TAG BUFFER 108
MDS SERVER 110
132
114
134
TAG DATA
MDS 112
DATA 128
CONTROL 130
ENCODER 101
MUX 104
CODEC 102
VIDEO SOURCE

Fig. 1

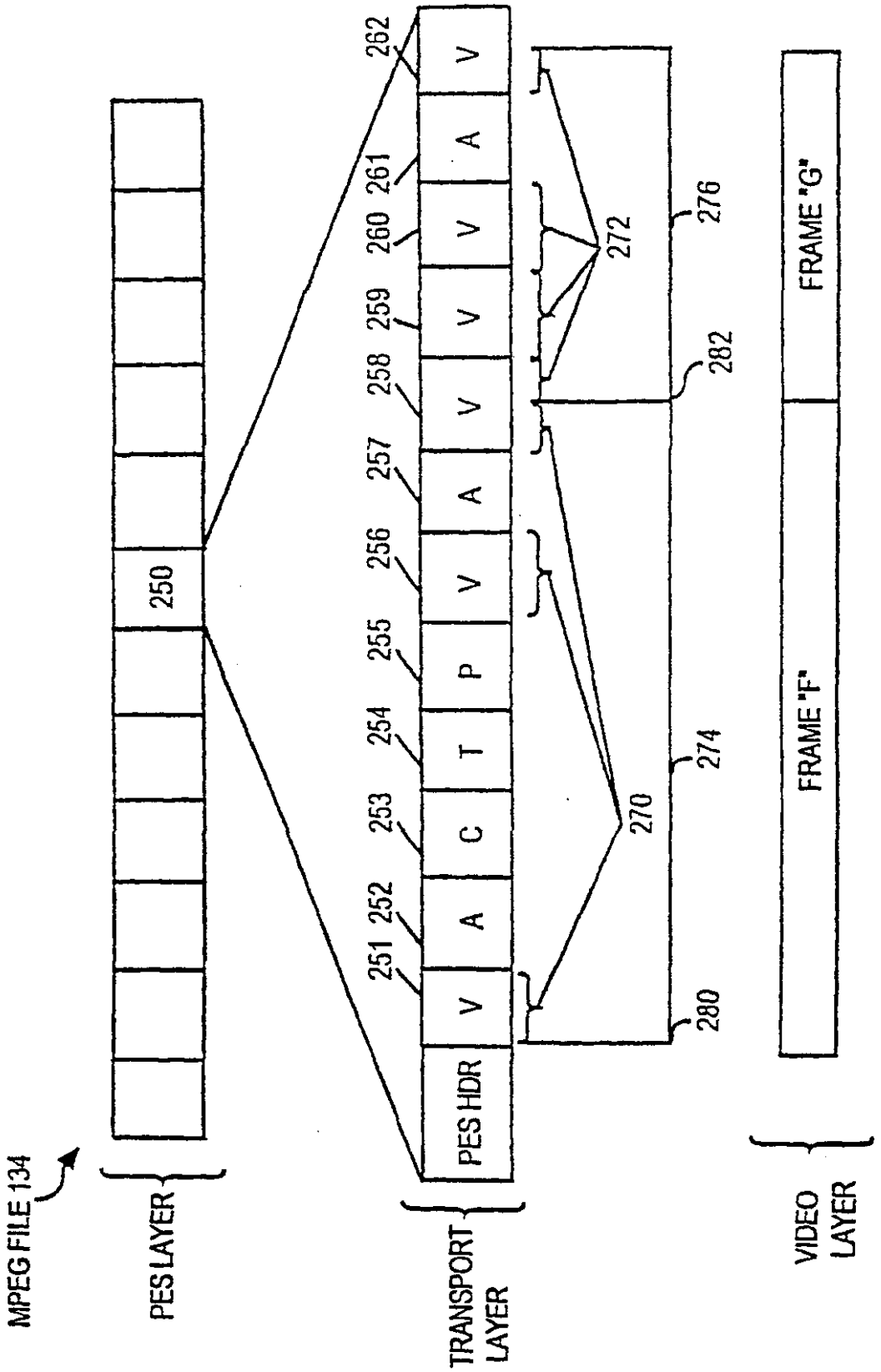
MPEG FILE 134
PES LAYER
250
TRANSPORT LAYER
PES HDR
V
A
C
T
P
V
A
V
V
V
A
V
V
V
251
252
253
254
255
256
257
258
259
260
261
262
270
272
274
276
280
282
VIDEO LAYER
FRAME "F"
FRAME "G"

Fig. 2A

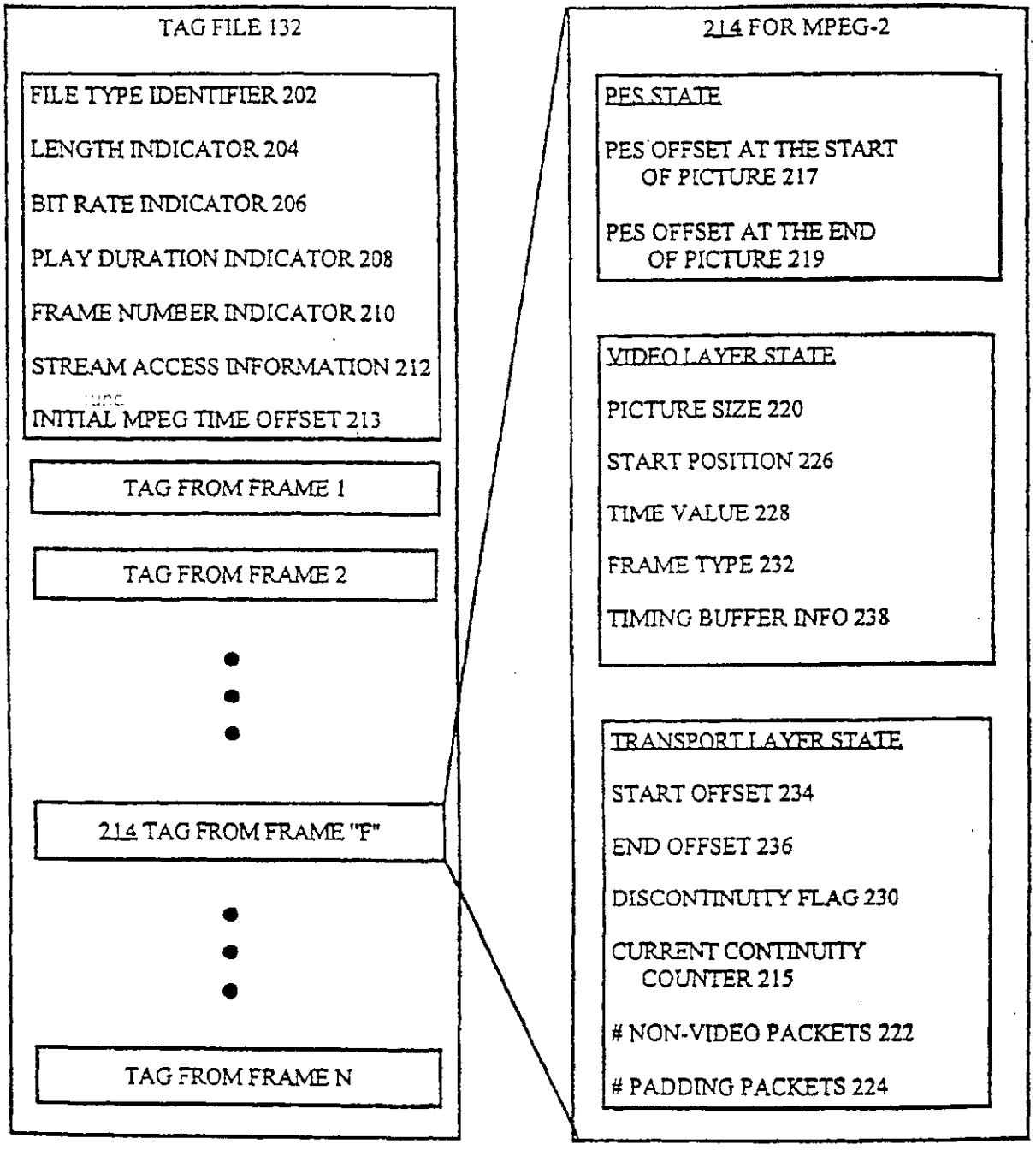
TAG FILE 132
FILE TYPE IDENTIFIER 202
LENGTH INDICATOR 204
BIT RATE INDICATOR 206
PLAY DURATION INDICATOR 208
FRAME NUMBER INDICATOR 210
STREAM ACCESS INFORMATION 212
INITIAL MPEG TIME OFFSET 213
TAG FROM FRAME 1
TAG FROM FRAME 2
214 TAG FROM FRAME "F"
TAG FROM FRAME N
214 FOR MPEG-2
PES STATE
PES OFFSET AT THE START OF PICTURE 217
PES OFFSET AT THE END OF PICTURE 219
VIDEO LAYER STATE
PICTURE SIZE 220
START POSITION 226
TIME VALUE 228
FRAME TYPE 232
TIMING BUFFER INFO 238
TRANSPORT LAYER STATE
START OFFSET 234
END OFFSET 236
DISCONTINUITY FLAG 230
CURRENT CONTINUITY COUNTER 215
NON-VIDEO PACKETS 222
PADDING PACKETS 224

Fig. 2B

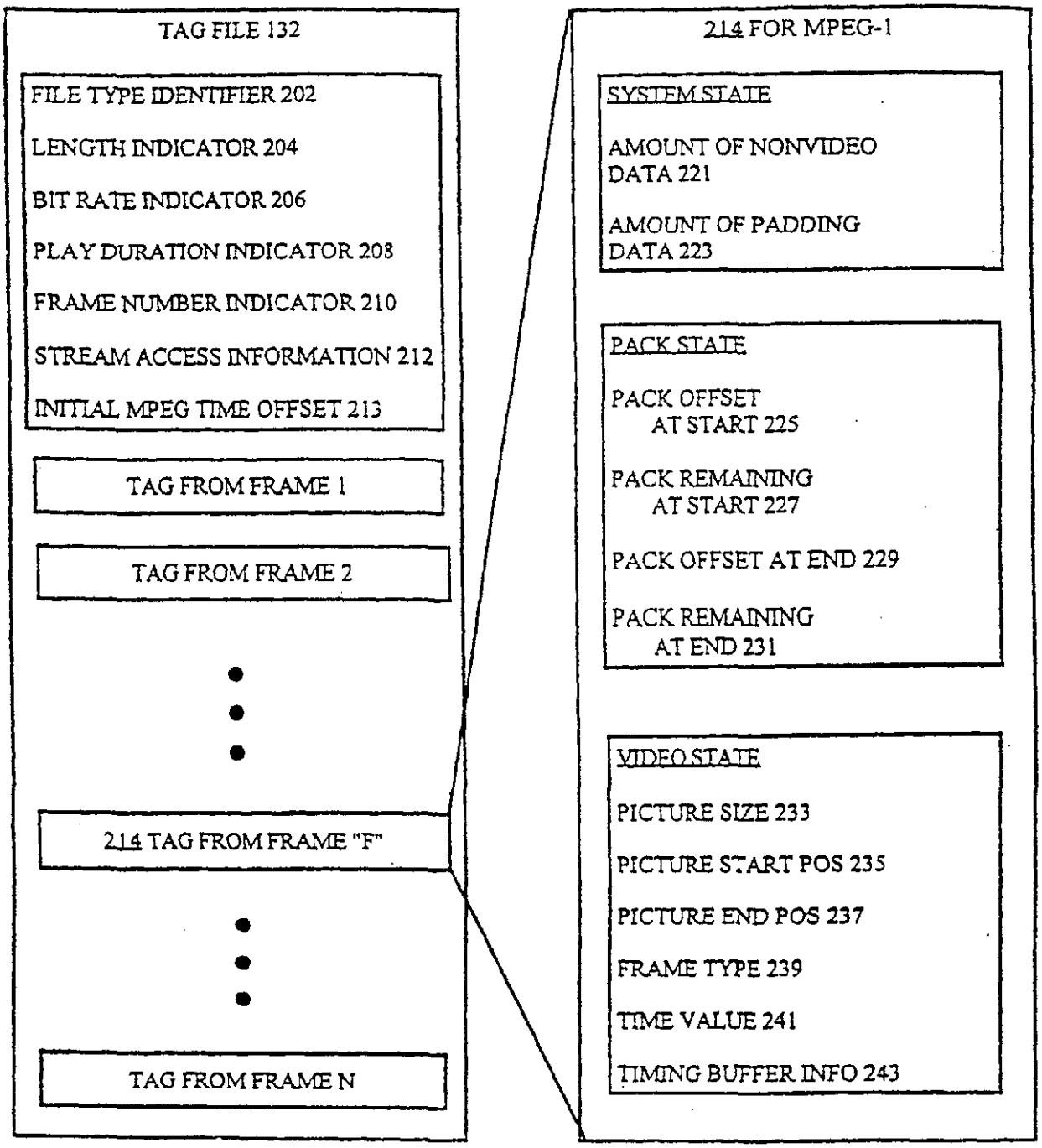
TAG FILE 132
FILE TYPE IDENTIFIER 202
LENGTH INDICATOR 204
BIT RATE INDICATOR 206
PLAY DURATION INDICATOR 208
FRAME NUMBER INDICATOR 210
STREAM ACCESS INFORMATION 212
INITIAL MPEG TIME OFFSET 213
TAG FROM FRAME 1
TAG FROM FRAME 2
214 TAG FROM FRAME "F"
TAG FROM FRAME N
214 FOR MPEG-1
SYSTEM STATE
AMOUNT OF NONVIDEO DATA 221
AMOUNT OF PADDING DATA 223
PACK STATE
PACK OFFSET AT START 225
PACK REMAINING AT START 227
PACK OFFSET AT END 229
PACK REMAINING AT END 231
VIDEO STATE
PICTURE SIZE 233
PICTURE START POS 235
PICTURE END POS 237
FRAME TYPE 239
TIME VALUE 241
TIMING BUFFER INFO 243

Fig. 2C

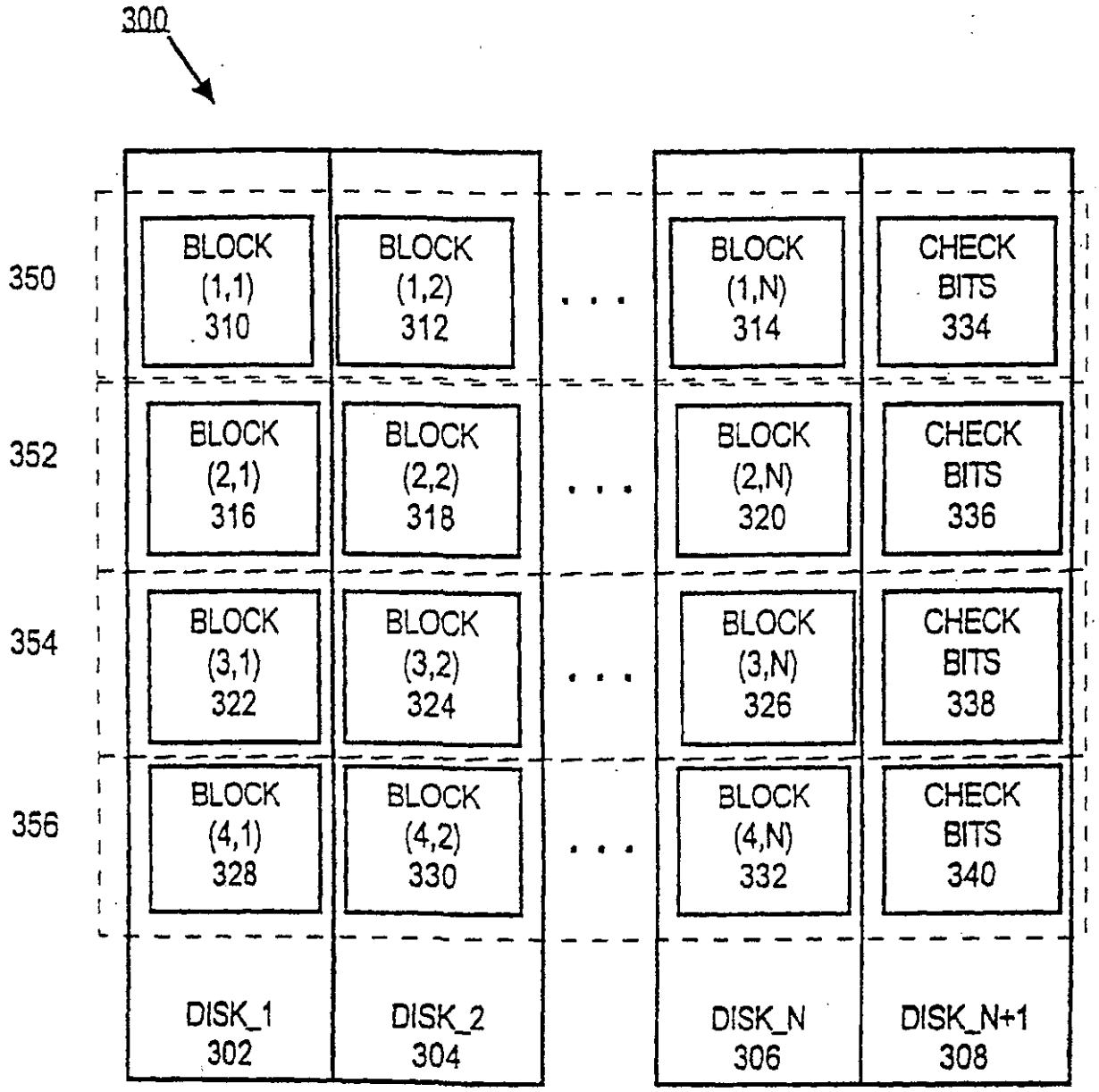
300
350
352
354
356
BLOCK (1,1) 310
BLOCK (1,2) 312
BLOCK (1,N) 314
CHECK BITS 334
BLOCK (2,1) 316
BLOCK (2,2) 318
BLOCK (2,N) 320
CHECK BITS 336
BLOCK (3,1) 322
BLOCK (3,2) 324
BLOCK (3,N) 326
CHECK BITS 338
BLOCK (4,1) 328
BLOCK (4,2) 330
BLOCK (4,N) 332
CHECK BITS 340
DISK_1 302
DISK_2 304
DISK_N 306
DISK_N+1 308

Fig. 3A

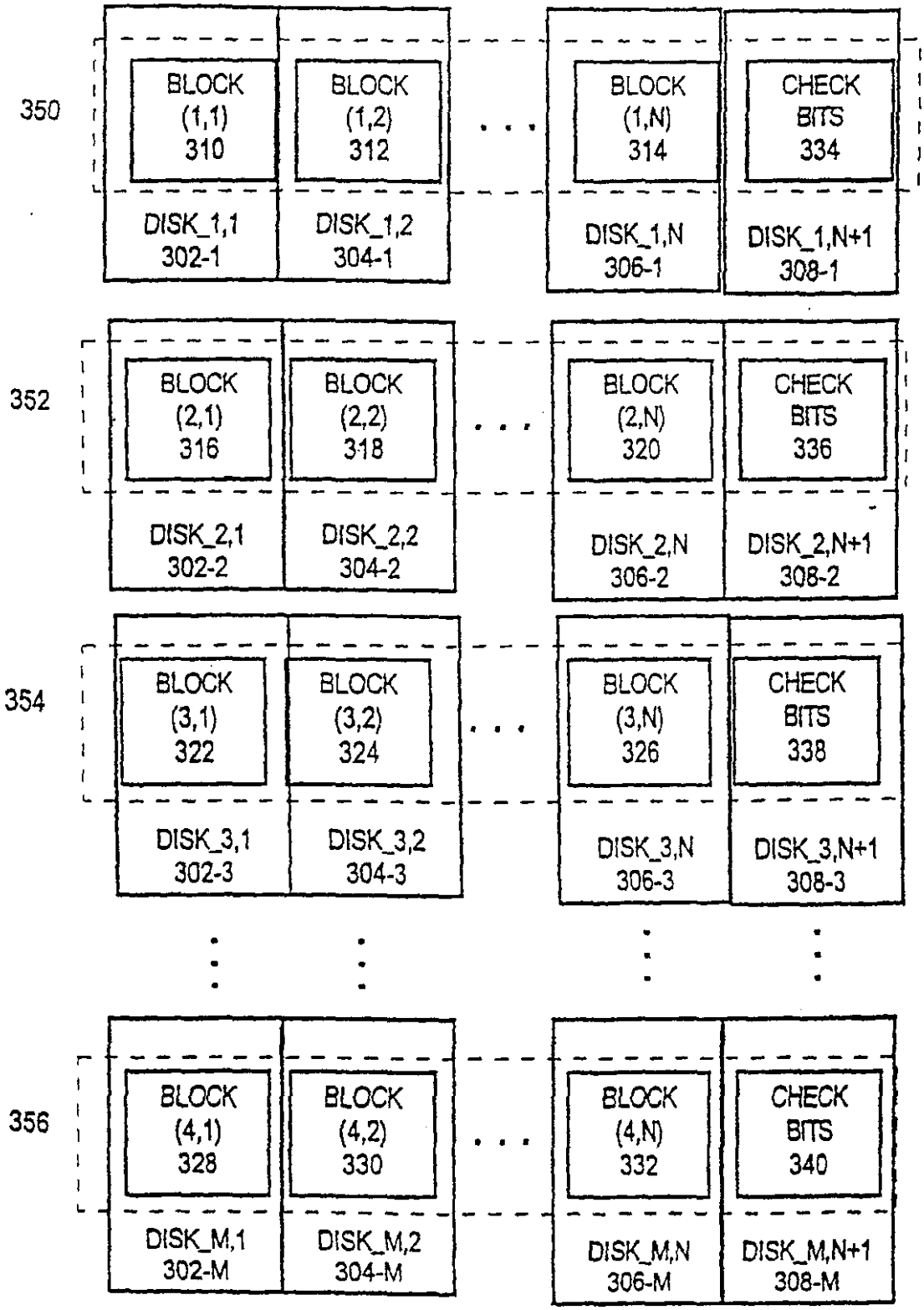
350
BLOCK (1,1) 310
BLOCK (1,2) 312
BLOCK (1,N) 314
CHECK BITS 334
DISK_1,1 302-1
DISK_1,2 304-1
DISK_1,N 306-1
DISK_1,N+1 308-1
352
BLOCK (2,1) 316
BLOCK (2,2) 318
BLOCK (2,N) 320
CHECK BITS 336
DISK_2,1 302-2
DISK_2,2 304-2
DISK_2,N 306-2
DISK_2,N+1 308-2
354
BLOCK (3,1) 322
BLOCK (3,2) 324
BLOCK (3,N) 326
CHECK BITS 338
DISK_3,1 302-3
DISK_3,2 304-3
DISK_3,N 306-3
DISK_3,N+1 308-3
356
BLOCK (4,1) 328
BLOCK (4,2) 330
BLOCK (4,N) 332
CHECK BITS 340
DISK_M,1 302-M
DISK_M,2 304-M
DISK_M,N 306-M
DISK_M,N+1 308-M

Fig. 3B

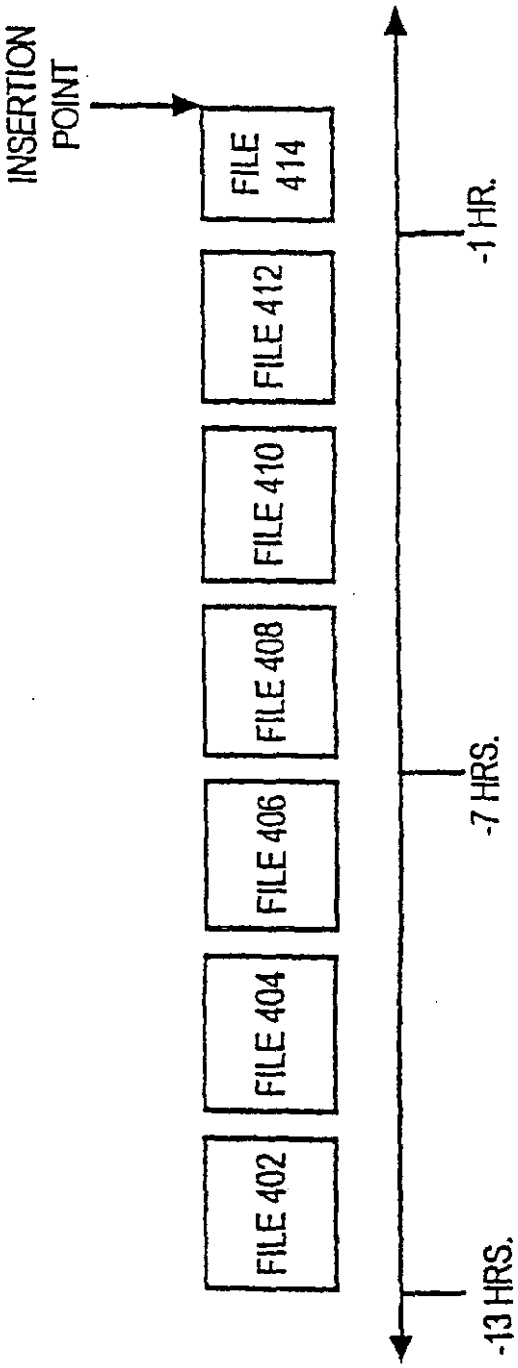
INSERTION POINT
FILE 414
FILE 412
FILE 410
FILE 408
FILE 406
FILE 404
FILE 402
-1 HR.
-7 HRS.
-13 HRS.

Fig. 4

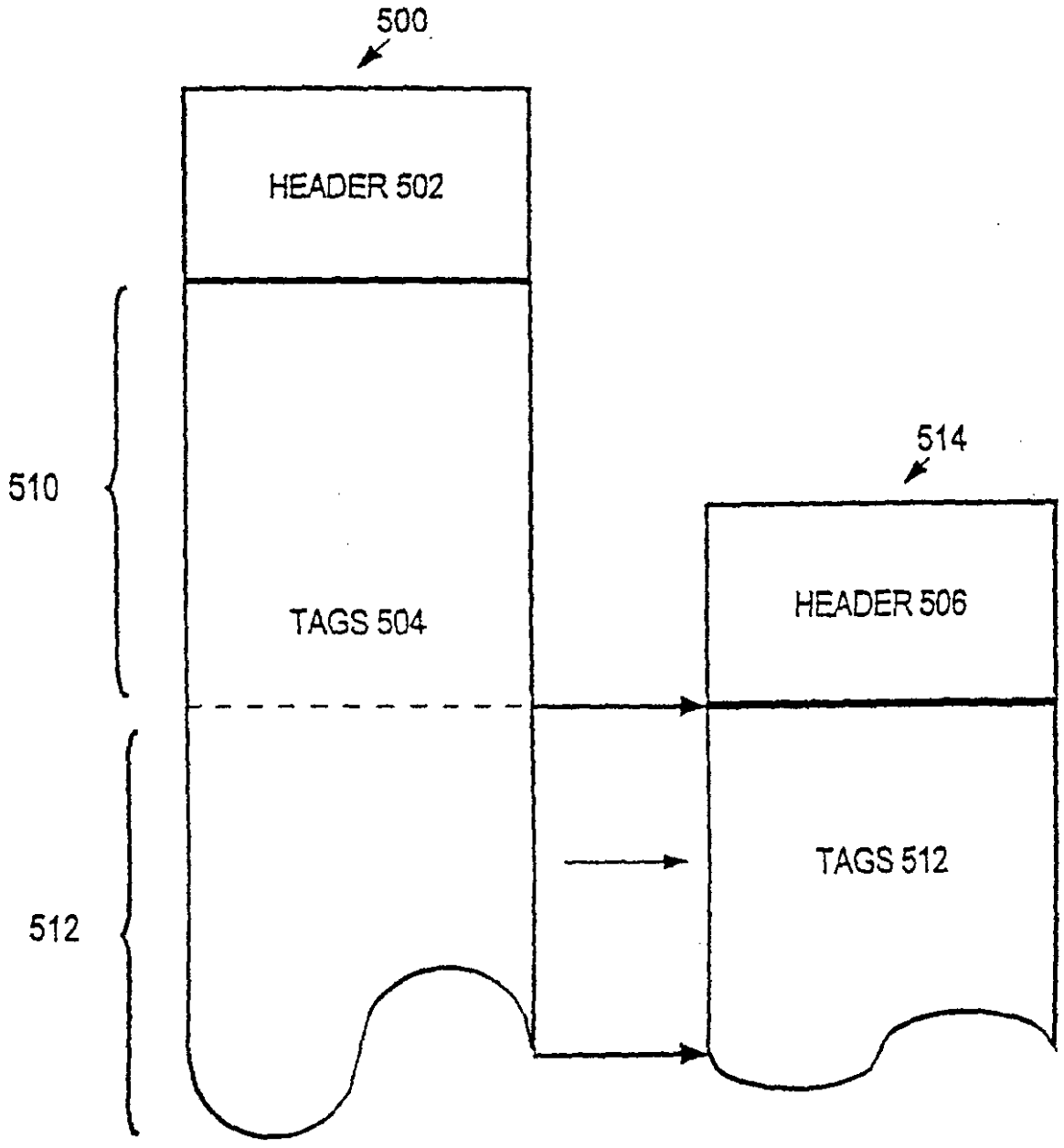

Fig. 5

VV-020470

用於執行連續媒體饋送的無縫隙回放的方法及裝置

提供一種存儲視頻的連續饋送的方法及裝置。根據本發明的一個方面，連續饋送以數字視頻的格式編碼來產生數字數據流。通過反復地執行在當前文件中存儲數字數據流的步驟(1)及(2)，當當前文件滿足一預定條件時建立一個新文件作爲當前文件來生成一系列內容文件。如果該系列內容文件包括多於預定數量的連續饋送時，系列中最老的內容文件被刪除。生成指示關於包括在數字數據流中的幀的信息的標識信息。標識信息包括指示幀相對於數字數據流的開始的時序的時間標記。指示對應於數字數據流的開始的絕對時間的初始時間值。當接收到來自客戶的在一特定的絕對時間回放的請求時，初始時間值被從特定絕對時間中減掉來確定相對時間。標識信息用於確定對應於該相對時間的數字數據流中的一位置。數字數據流於是從對應於相對時間的數字數據流中的一個位置開始傳送到客戶。