

(19) **United States**

(12) **Patent Application Publication**
Ganesan

(10) **Pub. No.: US 2009/0007196 A1**

(43) **Pub. Date: Jan. 1, 2009**

(54) **METHOD AND APPARATUS FOR SHARING MEDIA FILES AMONG NETWORK NODES WITH RESPECT TO AVAILABLE BANDWIDTHS**

Publication Classification

(51) **Int. Cl.**
H04N 7/173 (2006.01)

(52) **U.S. Cl.** **725/87**

(75) **Inventor:** **Prasanna Ganesan**, Menlo Park, CA (US)

Correspondence Address:
SHEMWELL MAHAMED LLP
4880 STEVENS CREEK BOULEVARD, SUITE 201
SAN JOSE, CA 95129 (US)

(73) **Assignee:** **Vudu, Inc.**

(21) **Appl. No.:** **11/930,147**

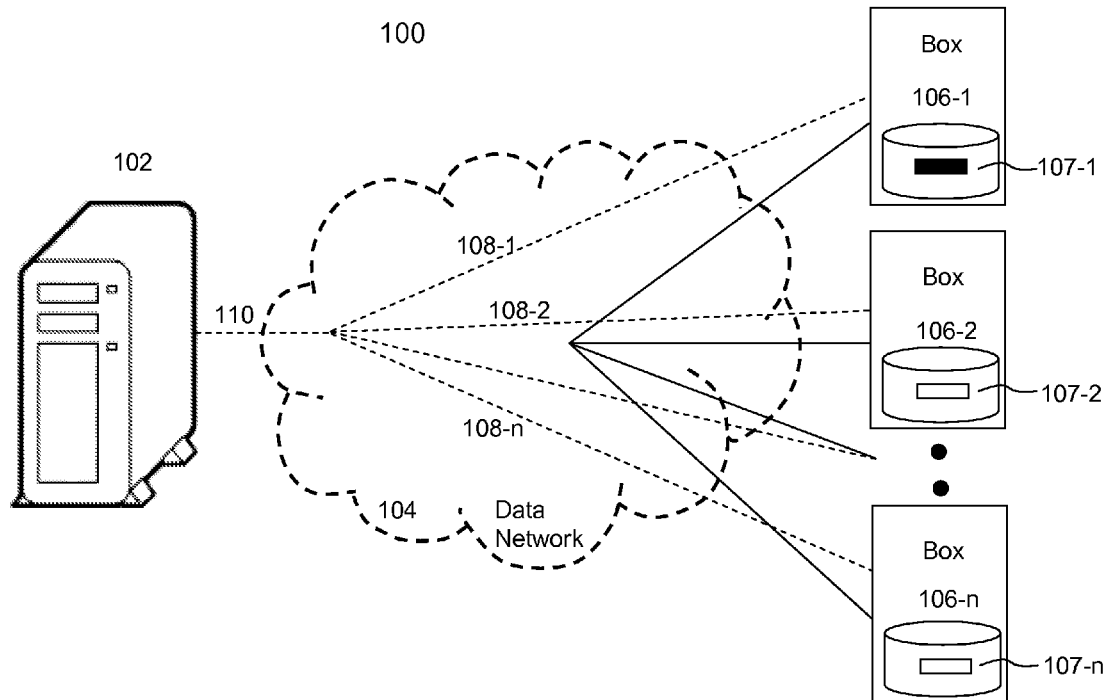
(22) **Filed:** **Oct. 31, 2007**

Related U.S. Application Data

(63) Continuation of application No. 11/278,416, filed on Apr. 1, 2006, which is a continuation-in-part of application No. 11/075,573, filed on Mar. 9, 2005.

(57) **ABSTRACT**

Techniques for sharing data among a plurality of networked devices, also referred to network nodes, are discussed. To best use an available uploading bandwidth of a device sharing data with a receiving device, a portion of the data is uploaded within the available uploading bandwidth, and another portion of the data is uploaded from another device locally caching the same data. To satisfy the minimum transmission rate of the data to the receiving device, both portions of the data are concurrently uploaded from the two devices to the receiving device. In reference to a predetermined minimum uploading bandwidth and the available uploading bandwidth, the two portions of data are interleaved and must be reassembled to recover the data for consumption in the receiving device. A size of the first portion of the data is determined by a server in view of the available uploading bandwidth a designated supplying box has.



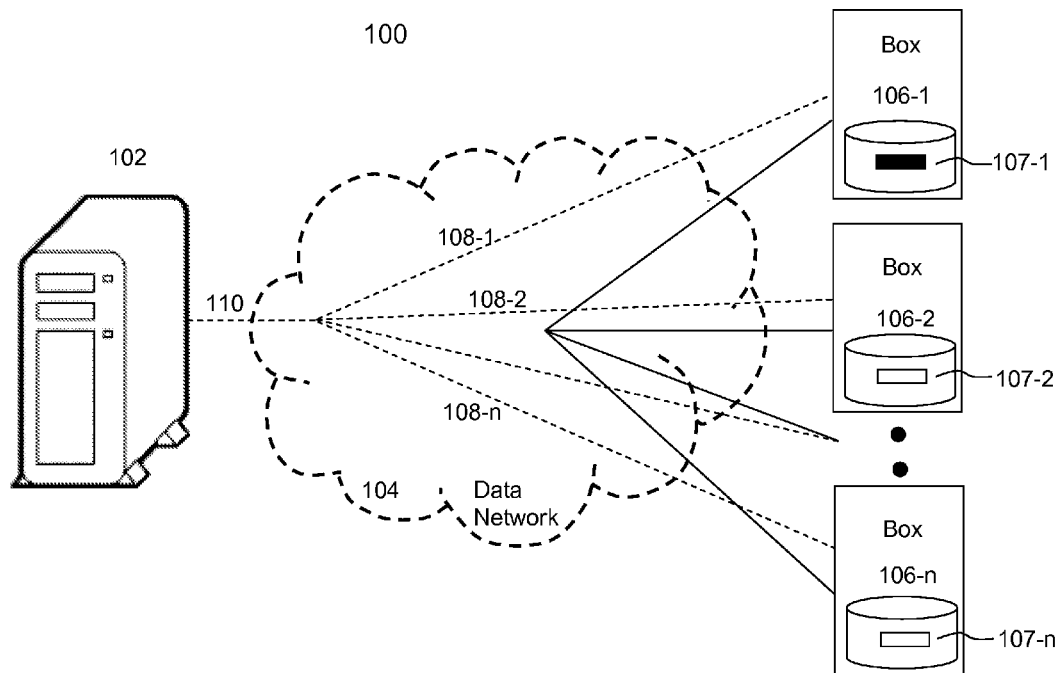


FIG. 1

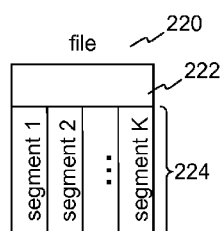


FIG. 2A

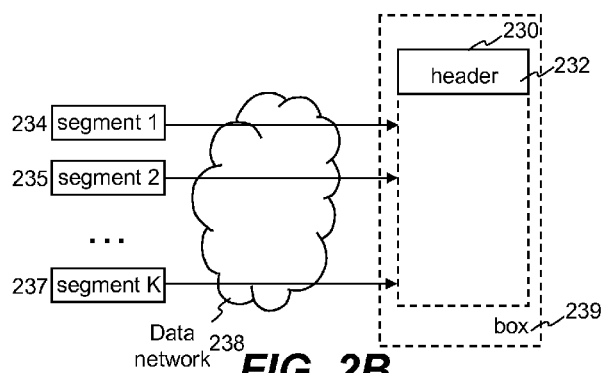


FIG. 2B

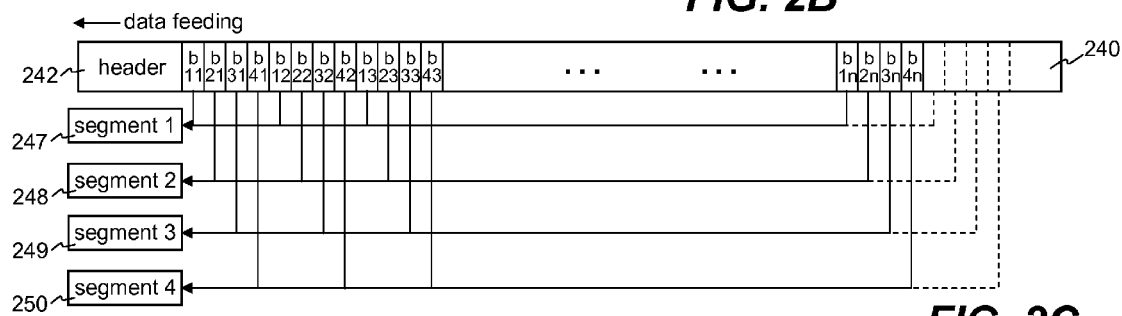


FIG. 2C

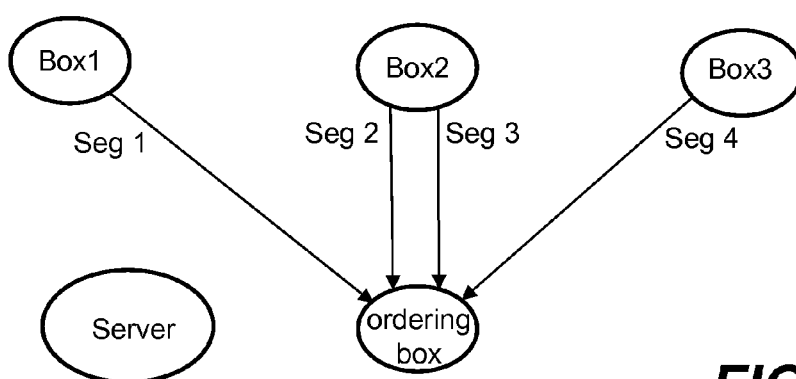


FIG. 3A

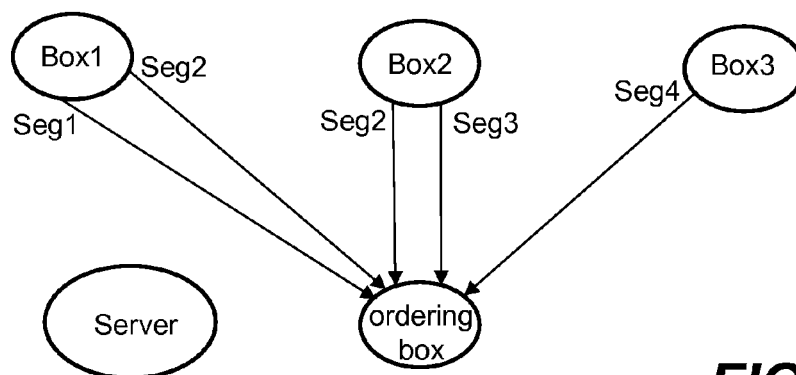
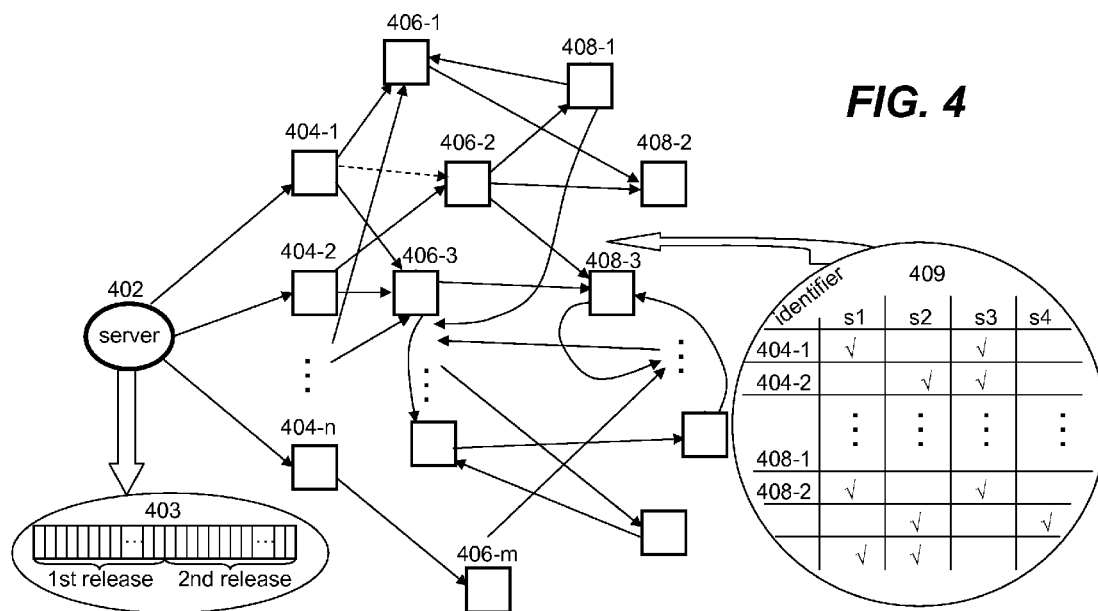
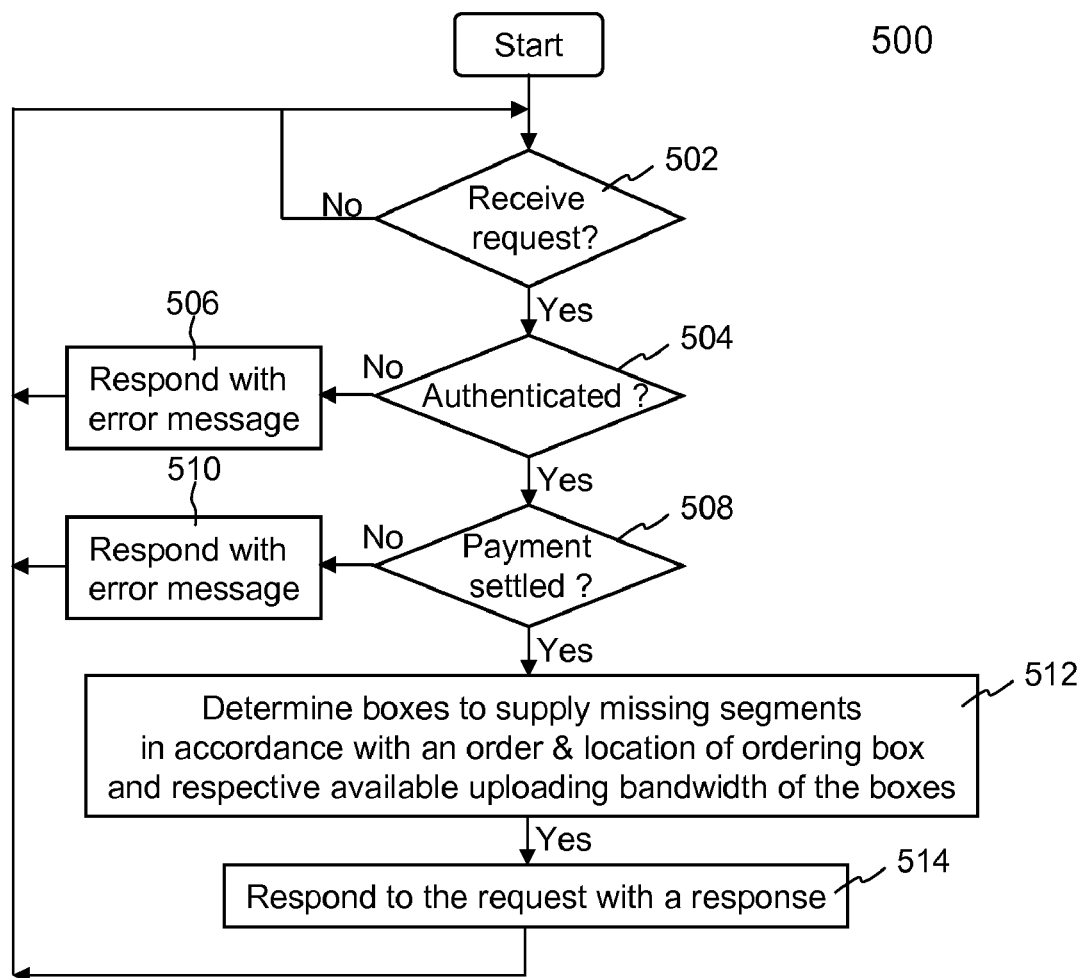
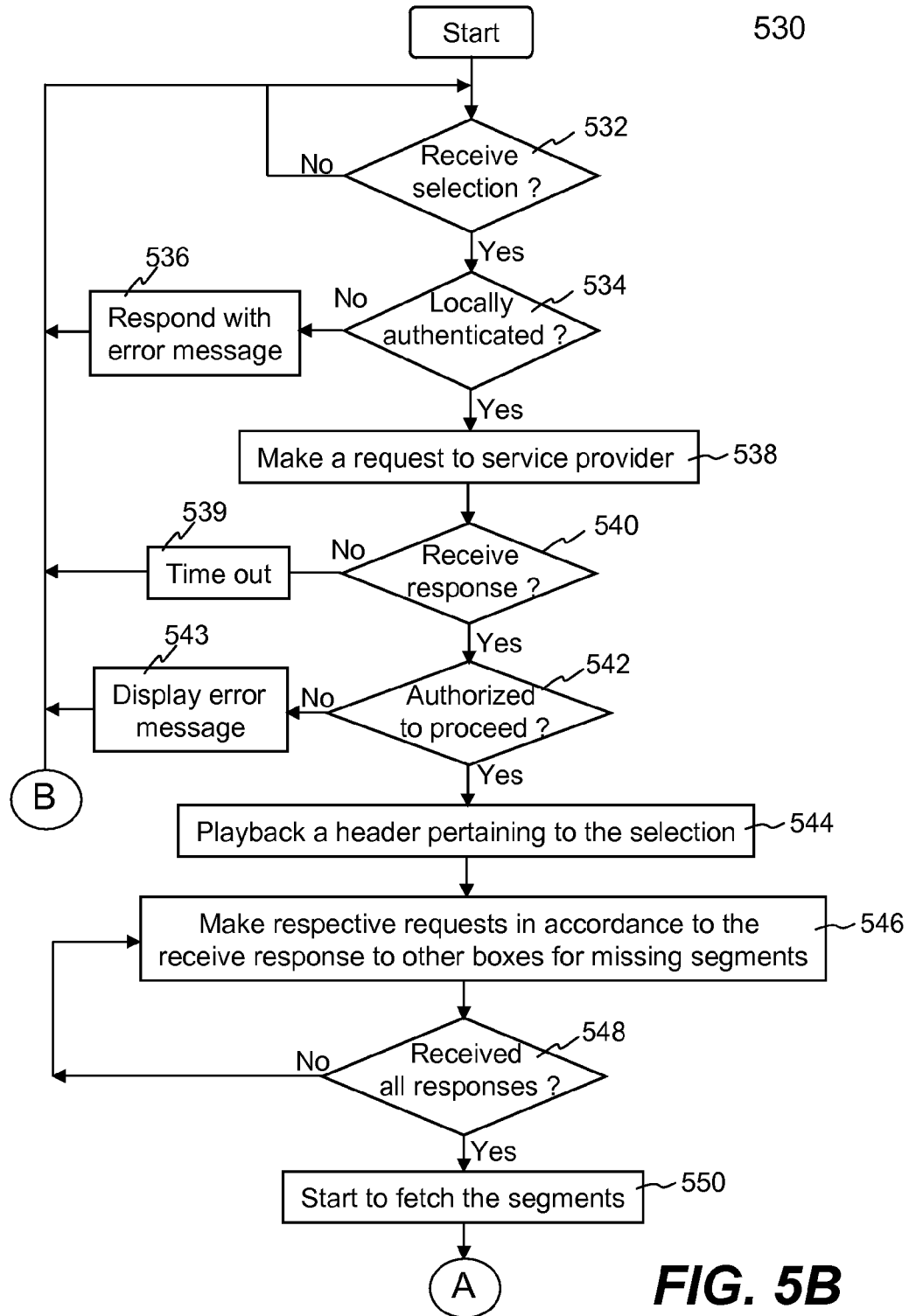
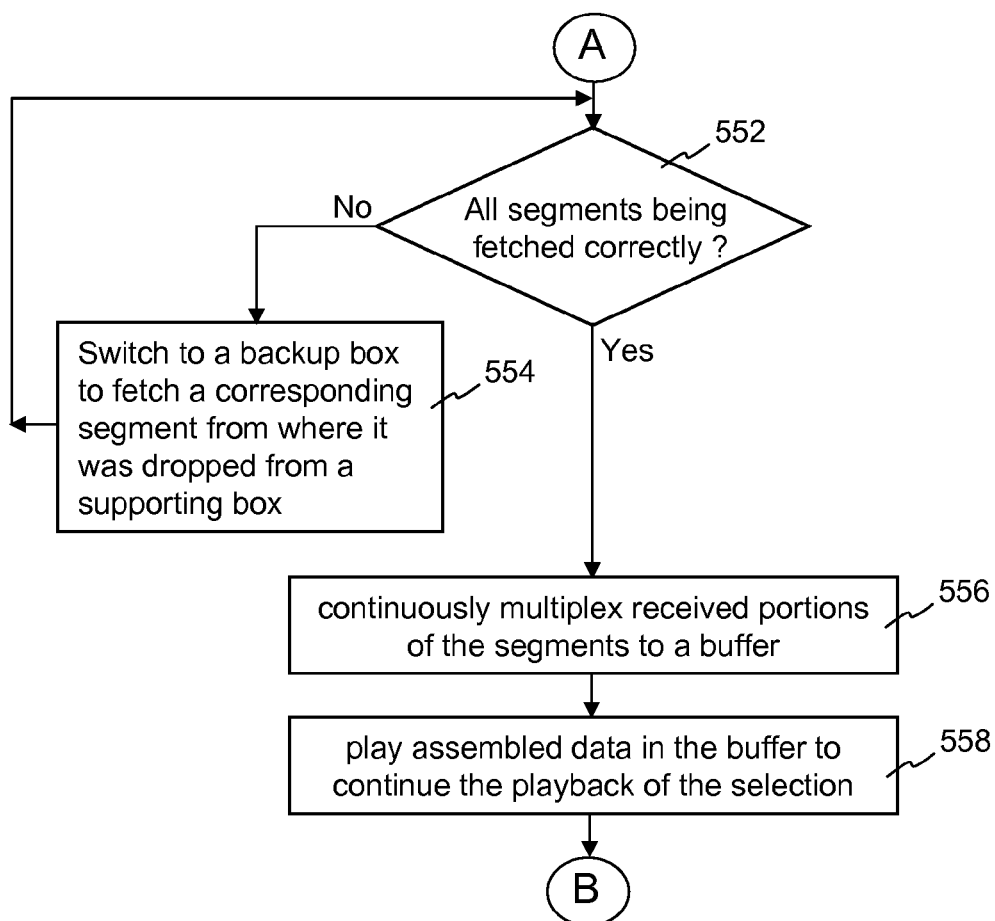


FIG. 3B



**FIG. 5A**



**FIG. 5C**

METHOD AND APPARATUS FOR SHARING MEDIA FILES AMONG NETWORK NODES WITH RESPECT TO AVAILABLE BANDWIDTHS

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This is a continuation of co-pending U.S. application Ser. No. 11/278,416 which is continuation-in-part of co-pending U.S. application Ser. No. 11/075,573, entitled "Continuous data feeding in a distributed environment" and filed Mar. 9, 2005, and by at least one of the co-inventors herein.

BACKGROUND

[0002] 1. Technical Field

[0003] The present invention is generally related to multimedia data delivery over the Internet. Particularly, the present invention is related to techniques for sharing media files among network nodes.

[0004] 2. Description of the Related Art

[0005] It is common to share files among computing devices across a network. However, when data to be shared is of huge size and the network bandwidth between two devices sharing the data is limited, the utility or application of the data could suffer. For example, one device is supplying data pertaining to a movie to another device over a network with a limited bandwidth. If the data does not arrive at a required transmission rate, the playback of the movie at the receiving device could demonstrate annoying jittery effects.

[0006] One possible solution to resolve jittery effects is to design a lower transmission rate. However, there are applications that require a minimum transmission rate, otherwise the applications could not perform or be conducted as expected.

[0007] There, thus, is a great need for techniques that maximize the use of available bandwidths while sharing data among computing devices across one or more networks.

SUMMARY

[0008] This section is for the purpose of summarizing some aspects of embodiments of the present invention and to briefly introduce some preferred embodiments. Simplifications or omissions in this section as well as the title and the abstract of this disclosure may be made to avoid obscuring the purpose of the section, the title and the abstract. Such simplifications or omissions are not intended to limit the scope of the present invention.

[0009] Broadly speaking, the invention is related to techniques for sharing data among a plurality of networked devices, also referred to network nodes. To best use an available uploading bandwidth of a device sharing data with a receiving device, a portion of the data is uploaded within the available uploading bandwidth, and another portion of the data is uploaded from another device locally caching the same data. To satisfy the minimum transmission rate of the data to the receiving device, both portions of the data are concurrently uploaded from the two devices to the receiving device.

[0010] In reference to a predetermined minimum uploading bandwidth and the available uploading bandwidth, the two portions of data are interleaved and must be reassembled to recover the data for consumption in the receiving device. A

size of the first portion of the data is determined by a server in view of the available uploading bandwidth a designated supplying box has.

[0011] According to one aspect, a media-on-demand system is provided. Significantly different from the prior art systems in which media services are rendered at a central server or servers, the present invention utilizes individual devices on a network to supply each other with needed sources in pieces to render required services. As a result, the loading pressure on a server is distributed into the network.

[0012] The invention may be implemented in numerous ways, including a method, system, device, or a computer readable medium. Several embodiments of the invention are discussed below. In one embodiment, the present invention is a method for sharing a file, the method comprises dividing at least some of the file into a sequence of data blocks, forming k segments, each of the k segments including some of the data blocks, wherein k is a finite integer greater than 1 and determined in accordance with a predefined minimum uploading bandwidth, causing at least a first device and a second device to locally cache at least one of the k segments; and designating the first device to supply a first portion of the one of the k segments and the second device to supply a second portion of the one of the k segments to a third device at substantially similar time, wherein the first and second portions are interleaved and multiplexed to reassemble the one of the k segments in the third device. In general, the file is a collection of data together with auxiliary data, if there is such auxiliary data. The designating of the first device to supply the first portion of the one of the k segments and the second device to supply the second portion of the one of the k segments to the third device happens when an uploading bandwidth of either one of the first and second devices is below the predefined minimum uploading bandwidth.

[0013] According to another embodiment, the present invention is a method for sharing a file, the method comprises dividing a file pertaining to a title into a sequence of data chunks, wherein the title is included in a release to update the library in each of boxes in service; designating an initial set of seeding boxes to receive the data chunks collectively, each of the seeding boxes receiving at least some of the data chunks; and causing each of the seeding boxes to propagate at least some or all of the received data chunks to a set of the boxes, wherein one of the boxes receives one of the data chunks from a first one and a second one of the seeding boxes concurrently, and data from the first one and the second one is interleaved and multiplexed to recover the one of the data chunks.

[0014] According to yet another embodiment, the present invention is a system for sharing a file, the system comprises at least a first device and a second device, each locally caching a data segment, an ordering device configured to receive the data segment; and a server configured to designate the first device to upload a first portion of the data segment to the ordering device, and the second device to upload a second portion of the data segment to the ordering device in accordance with respective uploading bandwidths for the first and second devices, wherein the first and second portions are interleaved determine. The data from the first and second portions must be multiplexed to recover the data segment in the ordering device.

[0015] Accordingly one of the objects of the present inventions is to provide techniques for sharing data or a file among a number of network nodes.

[0016] Other objects, features, and advantages of the present invention will become apparent upon examining the following detailed description of an embodiment thereof, taken in conjunction with the attached drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0017] The invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

[0018] FIG. 1 shows a video delivery system based on a distributed network system, in accordance with one embodiment of the present invention;

[0019] FIG. 2A, according to one embodiment, shows a file organized or fragmented into a header and K segments;

[0020] FIG. 2B shows a file pertaining to a title comprising one header and k segments, assuming a situation in which a box locally stores the header and receives the k segments from no more than k boxes to continue a playback of the title;

[0021] FIG. 2C shows a data stream representing a file, a beginning portion of which being allocated as a header and a remaining portion being decimated into four individual segments ($k=4$);

[0022] FIG. 3A shows that three boxes, box1, box2 and box3, are designated to supply four segments to an ordering box;

[0023] FIG. 3B shows a configuration in which box1 and box2 each have a bandwidth greater than the minimum uploading bandwidth (perhaps less than twice the minimum uploading bandwidth) so that box1 and box2 are concurrently uploading a common segment;

[0024] FIG. 4 shows a diagram of updating a library of all boxes in service using what is referred to as gossip protocol;

[0025] FIG. 5A shows a flowchart or process of starting a playback of a selection (i.e., a title) in a library; and

[0026] FIG. 5B and FIG. 5C collectively show a flowchart or process of starting a playback of a selection (e.g., a title).

DETAILED DESCRIPTION OF THE INVENTION

[0027] In the following description, numerous specific details are set forth to provide a thorough understanding of the present invention. The present invention may be practiced without these specific details. The description and representation herein are the means used by those experienced or skilled in the art to effectively convey the substance of their work to others skilled in the art. In other instances, well-known methods, procedures, components, and circuitry have not been described in detail since they are already well understood and to avoid unnecessarily obscuring aspects of the present invention.

[0028] Reference herein to “one embodiment” or “an embodiment” means that a particular feature, structure, or characteristic described in connection with the embodiment can be included in at least one implementation of the invention. The appearances of the phrase “in one embodiment” in various places in the specification are not necessarily all referring to the same embodiment, nor are separate or alternative embodiments mutually exclusive of other embodiments. Further, the order of blocks in process, flowcharts or functional diagrams representing one or more embodiments do not inherently indicate any particular order nor imply limitations in the invention.

[0029] Embodiments of the present invention are discussed herein with reference to FIGS. 1-8. However, those skilled in the art will readily appreciate that the detailed description given herein with respect to these figures is for explanatory purposes only as the invention extends beyond these limited embodiments.

[0030] FIG. 1 shows an exemplary configuration 100 of a distributed network system 100, in which the present invention may be practiced. It will be appreciated that the entire network may comprise a number of such network systems 100, for example, one for each box of a particular type, size, content, and etc.

[0031] A server 102, presumably managed and/or populated by a service provider, is configured to handle the delivery of video (or multimedia) services to users via local machines or boxes 106-1, 106-2, . . . 106-n. Different from a prior art video delivery system that delivers video data to a subscriber upon receiving a request therefrom, the server 102 is not responsible for delivering the content in response to a request from a user, and instead is configured to provide source information as to where and how to retrieve at least some of the content from other client machines, also referred to as boxes. The server 202 does not need a huge media storage device to provide the content. Instead, some of the boxes 106-1, 106-2, . . . 106-n are respectively configured to supply part or all of the content to each other.

[0032] According to one embodiment, when fulfilling a request from a local machine or a box (e.g., 106-1), communication between the server 102 and the box 106-1 over the network paths 108-1 and 110 may be limited to small-scale requests and responses (e.g., of small size and very short). A server response to a request from a box may include source information (e.g., identifiers), authorization information and security information. Using the response from the server 102, the box may be activated to begin playback of a title (e.g., 107-1). Substantially at the same time, the box may initiate one or more requests to other boxes (e.g., 106-2 and 106-n) in accordance with the source identifiers to request subsequent portions of the title (e.g., 107-2 and 107-n). Assuming proper authorization, the requesting box receives the subsequent portions of the data concurrently from the other boxes. Because of box-to-box communication of content, the bandwidth requirement for box-to-server communications over the network paths 108-1 and 110 is kept low and typically short in duration. In the event there are a large number of user boxes issuing playback requests substantially at the same time, the bandwidth of the backbone path 110 should be sufficient to avoid noticeable or burdensome delay.

[0033] The contents available in a library offered in any of the boxes 106-1, 106-2, . . . 106-n are originally provided by one or more content providers. Examples of the content providers include satellite receivers, television relay stations, analog or digital broadcasting station, movie studios and Internet sites. Depending on implementation, the contents may be initially received or originated in the server 102. Instead of maintaining and managing the content in a large storage device, the server 102 is configured to distribute the content or files to a plurality of local machines registered with the server 102. The boxes 106-1, 106-2, . . . 106-n shown in FIG. 1 are examples of local machines in service. Unless there is a need for a backup copy, the server 102 at any time has no need to keep a copy of the content. On the other hand, unless there is a special need to keep a complete copy of an

extremely high-demand title in a box, none of the boxes in service has a complete copy of a title until an order is placed.

[0034] For convenience, it is assumed herein that a file pertaining to a title is played back when the title is selected and ordered by a user. When an order for a title is placed, a corresponding file must start to be available for playback. According to one embodiment, where a file is 840 Mbytes on average and a box includes a storage capacity of 300 Gbytes, a system may offer a large library of titles (e.g., 5000) for access at any time instantly. In the prior art, if the files for the titles must be stored in advance to offer instantaneous playback, the local storage of a box would have to have a capacity of 4,000 Gbytes, consequently, rendering VOD economically impractical.

[0035] According to one embodiment of the present invention, only one or more tail segments of a file are locally cached in a box. In some case, the locally cached may also include a beginning portion (referred to as a “header”) for instant access. Such locally cached segments are referred to as residing objects or segments, while segments not residing locally are referred to as distributed objects or segments. If a beginning portion is locally cached, when a title is selected, the locally cached beginning portion or the header of the corresponding file is instantly played back. During the time the header is being played, the distributed segments corresponding to the title are retrieved simultaneously from other designated boxes. When the header is finished, the received parts of the distributed segments being streamed in from other boxes is combined with the residing segments for the title, if any, to enable the continuous playback of the title.

[0036] Depending on the popularity and concurrent demand for a particular title, the number of residing segments may be increased or decreased to control the dependency of each box on other boxes for playback. Typically, the more residing segments for a title a box has, the more distributed copies of the title there are in the entire system and thus the less dependency of the ordering box on the other boxes.

[0037] For example, in FIG. 1, when a user selects a title for playback from a box 106-1, a header 107-1 of the corresponding file residing in the box 106-1 is instantly accessed (provided that the user has been authenticated and/or payment is settled). In this example, there may be four segments for the video file, two of which are distributed in other boxes (e.g., 106-2 and 106-n). During the playback of the header, two distributed segments are streamed in from the other two boxes and locally buffered with the resident segments as continuing content. When the header is done, the continuing content is played back. As a result, instantaneous VOD may be realized.

[0038] Referring back to FIG. 2A, as an example, a file 220 is being organized or fragmented in terms of a header portion 222 and a tail portion comprising four segments 224. In general, the file 220 may be divided into any number of header and segment portions in consideration of a required transmission rate (e.g., related to the encoding and decoding rates for successful playback), and the minimum uploading and downloading capabilities of a network. According to one embodiment, given a required transmission rate (e.g., 1 megabit per second or 1 Mbps), the minimum uploading and downloading speeds of a network are considered to determine a number that defines the segmentation, and thus the dependency on other boxes and the support for concurrent demands of a particular title. It is assumed that a minimum uploading speed is U and a required transmission rate is D , and $D/U = K < k$, where k is the smallest integer greater than K . In

one embodiment, a file is preferably divided into a header and k segments to optimally utilize the uploading speed of U , assuming that the downloading speed is at least k times faster than the uploading speed. For example, in a POTS-based DSL network for residential areas, the required transmission is about 1.0 Mbps while the uploading speed is about 30 kbps. Hence, $k=4$. In another embodiment, the number of segments is chosen to be some value k' larger than k such that the uploading rate corresponding to each of k' segments matches some pre-determined value.

[0039] As shown in FIG. 2B, a file 230 comprises one header 232 and K segments 234-237. FIG. 2B assumes a situation in which a local box stores a header 232 and depends on K other boxes to supply the K segments 234-237. It is assumed that the local box 239 has a downloading speed N times the uploading speed of the other boxes. While the header 232 or a time-fill program is being played back, the K segments can be downloaded concurrently across the network 238 as streaming into the local box 239 around the same time.

[0040] It should be noted that a time-fill program is preferably provided when a file pertaining to a title is fragmented without a header. The program may be played before an ordered movie is started. Similar to watching a movie in a movie theater, the time-fill program provides a number of previews of upcoming new releases. The time-fill program may be locally configured. In one embodiment, the time-fill program is used to stabilize data being fetched from one or more other boxes. In another embodiment, the time-fill program provides a platform for sponsors that hope to display their respective programs to audience. Orders or slot positions for these programs in a time-fill program may be auctioned.

[0041] As also shown in FIG. 2B, a header 232 is the beginning portion of a file while each segment is a decimated portion of the remainder of the file. In this embodiment, the data in the header is continuous, meaning the header itself can be played back (e.g., the first 15 minutes of the title), while the segment portions 234-237 must be present together before the tail portion of the file can be played. FIG. 2C shows a data stream 240 representing a file. As a special case, $K=4$, the beginning portion of the file 240 is allocated as a header 242 and the remaining portion is divided into four “vertical” segments 247-250. The segments 247-250 are created or formed by respectively sampling, in a decimated manner, the remaining portion of the file.

[0042] Depending on an exact data length of the remaining portion, the n -th data block in each of the segments 247-250 is four successive data blocks in the remaining portion of the file. In one embodiment, a data block comprises a chunk of data, for example, 256 Kbytes or 1 Mbyte. As shown in FIG. 2C, the remaining portion of the data stream 240 is expressed in data blocks as follows: $b_{11}, b_{21}, b_{31}, b_{41}, b_{12}, b_{22}, b_{32}, b_{42}, b_{13}, b_{23}, b_{33}, b_{43}, \dots, b_{1n}, b_{2n}, b_{3n}, b_{4n}$. With the decimated sampling, the four segments 247-250 obtained from the remaining portion can be respectively expressed as follows:

[0043] Segment 1={ $b_{11}, b_{12}, b_{13}, b_{14} \dots$ };

[0044] Segment 2={ $b_{21}, b_{22}, b_{23}, b_{24} \dots$ };

[0045] Segment 3={ $b_{31}, b_{32}, b_{33}, b_{34} \dots$ }; and

[0046] Segment 4={ $b_{41}, b_{42}, b_{43}, b_{44} \dots$ }.

[0047] FIG. 2C shows one exemplary embodiment of fragmenting a file into a header 242 and four segments 247-250. There can be other ways to fragment a file. For example, besides fragmenting a file into several “vertical” segments

representing a tail portion of a file, one or more segments may be allocated to represent audio portions of the file. Typically, a movie includes several audio tracks, each for a language (e.g., English, French or Spanish). As a result, all segments are not necessarily equal in length but need to be available at the same time to support playback. This particular example illustrates that not all segments for a title must be fetched to play back the title (e.g., all segments for the video data and only one segment for one selected audio track).

[0048] One of the features in the present invention is to best use the available uploading bandwidth. When an ordering box is requesting to fetching the distributed segments from other supplying boxes, these supplying boxes may be in the middle of a task or in an area that has an insufficient bandwidth. As a result, the supplying boxes may not be able to deliver what the ordering box has.

[0049] It is appreciated from the above description that each segment is formed in conjunction with a minimum uploading speed of a network. When a supplying box has an uploading bandwidth twice the minimum uploading bandwidth, the supplying box may be designed to supply two segments at the same time. Referring now to FIG. 3A, it shows that three boxes, box1, box2 and box3, are designated to supply four segments to an ordering box. Both box1 and box3 have a respective bandwidth equal to or greater than the minimum uploading bandwidth while box2 has a bandwidth twice the minimum uploading bandwidth. Accordingly, the server designates box2 to supply segments 2 and 3 at the same time while box1 and box3 supply segments 1 and 4, respectively. Logically, if a box has a bandwidth that may accommodate n segments, such a box may be designed to supply 1 to n segments at the same time to at least one ordering box.

[0050] Quite often, an uploading bandwidth of a supplying box may not have an uploading bandwidth exactly twice but greater than the minimum uploading bandwidth. According to another embodiment, such a bandwidth may be optimally used by uploading part of a segment. Referring to FIG. 3B, there shows a configuration in which box1 and box 2 each have a bandwidth greater than the minimum uploading bandwidth (perhaps less than twice the minimum uploading bandwidth). Accordingly, the server designates both box1 and box 2 to supply one complete segment and one partial segment. As shown in FIG. 3B, in addition to uploading Seg1, box1 is also at the same time uploading a portion of Seg2. Similar to box1, box2 is uploading another portion of Seg2 and at the same time uploading Seg3. Box3 is designed to upload only one segment Seg4. On the receiving side, the ordering box is receiving concurrently four segments Seg1, Seg2, Seg3 and Seg4.

[0051] According to one embodiment, If a supplying box has a (remaining) bandwidth less than a predefined minimum uploading bandwidth (e.g. 384 kbps), it can be configured to stream out only a portion of a segment, and another supplying box is designed to stream out the "left-over" portion. Specifically, if a segment needs to be streamed out at 384 kbps and a supplying box has only an available bandwidth of 256 kbps, thus supplying box transmits a portion of the segment at 256 kbps and another designated supplying box will be designed to transmit the remaining portion of the segment at 384-256=128 kbps. It can be appreciated that the another designated supplying box locally caches the segment and the two transmissions happen concurrently and must be interleaved for reconstruction in the receiver in order to enable a playback. For example, one can transmit every 2 out of 3 blocks in

the segment for some fixed block size, e.g., 512 bytes, while another one transmits the remaining blocks.

[0052] One embodiment may enable a dynamically updated library with a large number of titles offered to users. Each of the titles may be selected and ordered for playback. Given a large library, for example, of 5000 titles that are updated periodically (e.g., daily) and can be accessed any-time, some of the titles may be more popular and thus demanded more often and by more users than others. To minimize possible bandwidth problems or unavailability of boxes for sourcing popular titles, the seeding of residing objects and distributed objects should be performed intelligently, e.g., according to popularity, geography, demographics, and/or like criteria.

[0053] Referring now to FIG. 4, there shows a diagram 400 of updating a library of all boxes in service. When a server (e.g., the server 102 of FIG. 1) updates the library, the library in all the boxes is updated accordingly. According to one embodiment, the updating process is carried out synchronously and/or asynchronously.

[0054] The server 402 is configured to prepare a file pertaining to the release of a title into segments or in addition to a header, if needed. For convenience, it is assumed that that file is segmented into K segments. As described above, these K segments are distributed into boxes in service. In principle, each of the boxes receives none or at least one but not all of the K segments. Accordingly, there may be a plurality of release packages in reference to what a box is supposed to receive. For example:

[0055] Release package 0=(); meaning a box receives none of the segments;

[0056] Release package 1=(segment 1);

[0057] Release package 2=(segment 1, segment 2);

[0058] Release package 3=(segment 1, segment 2, segment 3);

[0059] . . .

[0060] Release package N=(segment 1, segment 2, . . . segment N);

[0061] Release package N+1=(segment 2);

[0062] Release package N+2=(segment 2, segment 3);

[0063] . . .

N is a positive integer and must be smaller than K for the invention to work efficiently. For example, K=4, though there may be 24 (4×3×2) release packages, only a fractional portion of the release packages will make the invention work efficiently.

[0064] In operation, each of the boxes in service is configured to receive one release package. Initially, the server prepares a release instruction including metadata about a release, a least popular title to be discarded from the library and/or title transfers. The instruction may describe which box gets which of the release packages and how the package is to be received (i.e., from which other boxes). For example, the instruction could specify that boxes identified by identifiers that demonstrate certain characteristics are to receive a release package X. There may be a default instruction that applies in general to all releases, and there may be instructions tailored to specific releases as well. One of the objectives of assigning release packages to boxes may be to distribute the packages fairly evenly among all boxes in service to ensure uniform distribution of different segments.

[0065] The release instruction, once prepared by the server, is propagated to boxes in service either via direct communication between the server and a box, or by box-to-box propa-

gation of the instruction via a gossip protocol, which will be described in detail below. In any case, it is assumed that each box recognizes the need for it to receive a particular release package.

[0066] The release is transformed into a sequence of data chunks **403** representing segments for the release. A data chunk is an atomic unit of data transfer from the server to the boxes, or between two boxes. For example, each of the data chunks may be 1 Mbyte in size and uniquely identified. The sequence of data chunks **403** may represent two separate titles being propagated to the boxes to update the library. In general, each box desires a specific subset of the data chunks that make up a proper release package corresponding to that box. In addition, the release instruction itself may be represented as one or more data chunks that are to be propagated to all boxes.

[0067] In operation, the server **402** initiates respective communications with a set of boxes **404-1**, **404-2**, . . . **404-n** and provides each of them with some of the data chunks required by that box. Preferably, each data chunk is seeded into at least one of the boxes by the server **402**. The exact number of the boxes **404-1**, **404-2**, . . . **404-n** initially to receive the data chunks does not constrain distribution. In one embodiment, the designation of the boxes **404-1**, **404-2**, . . . **404-n** is fairly random. In another embodiment, the designation of the boxes **404-1**, **404-2**, . . . **404-n** is based on one or more of a time zone, the geographical location, an available network bandwidth, and the latency thereof, the Internet service provider for the box, etc. In any case, whenever the server **402** is idle, the server **402** can always designate different boxes to receive data chunks.

[0068] Each of the boxes **404-1**, **404-2**, . . . **404-n** is configured to spread data chunks to other boxes in service based on what is commonly referred to as a “gossip protocol,” an application-layer multicast-like protocol. It should be noted that not all of the boxes **404-1**, **404-2**, . . . and **404-n** have received identical data chunks. Any of the boxes **404-1**, **404-2**, . . . and **404-n** may start to spread a data chunk to other boxes as soon as it has received a data chunk in its entirety. In operation, the box **404-1** is assigned to propagate at least some of its received data chunks to boxes **406-1**, **406-2** and **406-3**, communicating with one or more of these boxes simultaneously. The box **404-2** is assigned to propagate at least some of its received data chunks to boxes **406-2** and **406-3**. The box **406-2** is configured to know exactly what data chunks to get from the box **404-1**, the box **404-2**, and any other boxes configured to feed it chunks of data. Further, the box **406-2** is assigned to propagate at least some of its received data chunks to boxes **408-1**, **408-2** and **408-3**. Note that the propagation of data is not necessarily hierarchical. For example, box **408-1** might send data chunks “backward” to **406-1**, as seen in the figure.

[0069] In one embodiment, data chunks are propagated only to boxes that actually desire those particular chunks in order to avoid wasteful data transmission. Moreover, wasteful data transmissions may be avoided by ensuring that a data chunk is propagated to a box only if it does not already possess that chunk and is not in the process of downloading that chunk from elsewhere. The propagation of chunks may be through synchronous protocols where all boxes participate at the same time in a coordinated fashion, or asynchronous protocols in which each box may choose when to participate and for how long in a flexible manner. For example, a box could cease participating in the downloading and propagation

of chunks whenever it is busy serving a movie for an ordering box, or when the network is detected to be in a period of intense usage. The box may continuously monitor the network conditions, and adaptively rejoin the gossiping propagation when there is sufficient bandwidth available.

[0070] In operation, if any one of the boxes, for whatever reason, fails to accept data chunks, the box could be dropped as a supplier or a substitute box could be configured to receive and supply the data chunk. The box that missed the release may fetch the data later from one or more updated boxes. By repeatedly and recursively propagating data chunks via boxes after boxes (i.e., by pulling or pushing synchronously and/or asynchronously), eventually all boxes in service will be populated with each release (the header and designated segments of all titles to be added and the identification of titles to be removed).

[0071] In seeding the releases in a box from multiple other boxes, a data chunk may not be necessarily from a single box. Depending on the available bandwidth, the data chunk may be from two other boxes. Similar to the operation as shown in FIG. 2E, the two other boxes may each send a portion of the data chunk due to the limited available uploading bandwidth.

[0072] After updating is finished, a map **409** identifying which box has which segments can be developed. By the map **409**, whenever an order is received from an ordering box, the server can designate appropriate boxes to supply the non-locally cached segments to the ordering box. Alternatively, the map **409** enables a box to obtain source information to fetch needed segments to fulfill an order.

[0073] When the release is not for the mostly demanded category, the determination of which box to retain what segment(s) may be based on several factors such as geographic locations, time zone, viewing behaviors or a preferred language so as to maximize the efficiency of transporting the segments among the boxes when needed.

[0074] Referring now to FIG. 5A, there shows a flowchart or process **500** of starting a playback of a selection (i.e., a title) in a library. The process **500** may be implemented in software, hardware or a combination of both as a method, a process, and/or a system. Preferably, the process **500** is executed in a computing device designated as a server facilitating the instant playback of a selected title from a box associated with a user. In one embodiment, the process **500** is employed in a media on demand system. At **502**, the process **500** is awaiting a request from an ordering box associated with a user. Typically, a user selects a title and then places an order. As will be further described below, the ordering box generates a request to be transported to the server. The process **500** is activated when such request including the order is received from the ordering box. In general, the request includes an identifier and an IP address of the ordering box, the user account information (e.g., user name), and the order information. Before anything happens in the ordering box, the process **500** proceeds with authentication of the user. If the user is not registered, the process **500** goes from **504** to **506** wherein a response including an error message is generated and returned to the ordering box. Depending on implementation, the error message may activate a local module in the ordering box to display an error message or ask the user to register with the system.

[0075] After the user is authenticated, the process **500** goes to **508** to determine if payment for the order is settled. In one embodiment, in a registration process, a user may provide credit card information for charges on any order the user

places with the system. In another embodiment, the user may receive a monthly statement for collective settlement of charges. If payment is not settled (e.g., the user has a large outstanding balance in his/her account), the process 500 goes to 510 where a response including an error message is generated and returned to the ordering box. The error message may be displayed locally to the user for payment.

[0076] After a payment is settled, the process 500 goes to 512 to determine a number of boxes that are designated to supply the segments to the ordering box. According to one of the features in the present invention, after a number of candidate boxes are selected, their respective uploading bandwidths are considered. If a remaining uploading bandwidth for a box is less than what is needed for uploading a whole segment to an ordering box, depending on implementation, the server may be configured to look for another box with a sufficient uploading bandwidth or utilize what is available. It is assumed that the server is configured to use the remaining available uploading bandwidth. In operation, the remaining available uploading bandwidth has to exceed a threshold otherwise it would be meaningless to use such a supplying box.

[0077] Once the supplying box is determined to upload only a portion of a segment, another supplying box that locally caches the same segment is determined to complement the uploading of the segment. As a result, a segment may be supplied from two boxes. But in any case, the exact number of boxes depends on the number of segments the ordering box needs to continue the playback of the selected title. At 514, a response is generated in accordance with the received request. In general, the response includes source information, authentication information and security information. The source information instructs where and how the ordering box can get the needed segment(s) to continue the playback of the selected title. The authentication information allows the ordering box to conduct respective secured communications with boxes designated to supply the needed segment(s). The security information facilitates decryption of any data for the playback of the ordered title.

[0078] In determining one or more boxes to supply the needed segments among others, one or more several factors may be considered depending on implementation. These factors include, but may not be limited to, respective available bandwidths, geographic locations, histories of availability of these supplying boxes, and the Internet service provider of each box. In addition, whether the ordered title is popular or not, a supplying box is new or not, and a supplying box is busy or not may be considered as well. In any case, the response is transported back to the ordering box or causes the ordering box to start the playback while receiving the needed segments. The process 500 then returns to 502 to await another request.

[0079] Referring now to FIG. 5B and FIG. 5C, both figures collectively show a flowchart or process 530 of starting a playback of a selection (i.e., a title). The process 530 may be implemented in software, hardware or a combination of both as a method, a process, and/or a system. Preferably, the process 530 is executed in a computing device that may correspond to a box as used herein.

[0080] At 532, the process 530 awaits a selection from a user. In one case, a user views a display with a plurality of titles from which the user may activate a key (e.g., on a remote control or keyboard) to choose one of the titles. The process 530 is activated when a selection is made by the user. The process 530 goes to 534 to determine whether the user and/or

box is properly authenticated. In one embodiment, a registered user is required to input a username and a password for authentication. In another embodiment, a registered user is required to enter a code for authentication. There may be other ways to authenticate a user. In any case, the process 530 needs to ensure that a user and a box are legitimate. If not, the user is sent an error message at 536 that may recommend that the user register with the system.

[0081] After a registered user has been authenticated at 534, the box sends a request at 538 in accordance with the selection. The request includes information about the order and the user. The request is transported to the server by a service provider. Upon receiving the request, the server proceeds with process 500 of FIG. 5A. Meanwhile, the box awaits a response from the server at 540. The request may be re-sent if a response is not received within a predefined time (e.g., 5 seconds). However, if the response is not received beyond a certain time (e.g., the network is down), an error message will be displayed at 539.

[0082] At 542, the response is received from the server. For a proper reason, the response may restrict the user from using the system. If the user is restricted, the process 530 goes to 543 to display an error message to the user. Upon authorization, the process 530 goes to 544 where a header of the file pertaining to the selected title is played and may be displayed via a display utility.

[0083] At 546, in accordance with the response from the server, the box makes respective requests to other boxes for missing segments. As described above, the response includes source information indicating where the box can fetch the missing segments. For example, if there are eight segments for a file and the box stores two of the segments locally, then six segments must be fetched from other boxes. In one embodiment, the six segments are provided from four other designated boxes as follows:

[0084] Box 1: {segment 1, $\frac{1}{3}$ segment 2}

[0085] Box 2: { $\frac{2}{3}$ segment 2}

[0086] Box 3: {segment 4, segment 5, $\frac{1}{2}$ segment 6}

[0087] Box 4: { $\frac{1}{2}$ segment 6}

where the respective uploading bandwidths for the four designated boxes are best used without interrupting the uploading of the needed segments to the ordering box.

[0088] At 548, the box awaits a response from the boxes being requested to supply the missing segments. If one of the boxes is unable to respond to the request, a backup box may be called upon to supply the segment. If the backup box is also unable to respond to the request, the box will send a request to the server for additional backup boxes. In any case, after the designated boxes respond to the requests from the ordering box, the ordering box at 550 starts to fetch the missing segments from the designated and responded boxes.

[0089] As described above, the missing segments are expected to arrive at a predetermined speed. If, for some reason, a portion of the network is congested or the box itself is malfunctioning, causing a significant slowdown of the segment being fetched, the process 530 goes to 554 where a backup box is called in to continue supplying the segment being interrupted.

[0090] If all segments are streaming at predetermined minimum speeds, then, at 556, portions of the segments locally cached and the portions of the segments being streamed in are multiplexed into a buffer to continue the playback of the ordered title. If there is a segment being supplied from two boxes, the data streams from the two boxes are multiplexed to

recover the original order of the segment before the data from the segment is joined to be multiplexed with that from other segments.

[0091] The process 530 goes to 558 to continue the playback of the assembled data in the buffer until the entire file for the ordered title is played. The process 530 then goes back to 532 to await another order from the user.

[0092] One skilled in the art will recognize that elements of the system may be implemented in software, but can be implemented in hardware or a combination of hardware and software. The invention can also be embodied as computer-readable code on a computer-readable medium. The computer-readable medium can be any data-storage device that can store data which can be thereafter be read by a computer system. Examples of the computer-readable medium may include, but not be limited to, read-only memory, random-access memory, CD-ROMs, DVDs, magnetic tape, hard disks, optical data-storage devices, or carrier wave. The computer-readable media can also be distributed over network-coupled computer systems so that the computer-readable code is stored and executed in a distributed fashion.

[0093] The foregoing description of embodiments is illustrative of various aspects/embodiments of the present invention. Various modifications to the present invention can be made to the preferred embodiments by those skilled in the art without departing from the true spirit and scope of the invention as defined by the appended claims. For example, in one embodiment, the size of a header for a file may be reduced to zero, in other words, a file is fragmented into a plurality of segments that can be distributed into boxes. Also, when an order of a title is placed, the server may identify the sources that will supply data for the order, and then contact these suppliers itself to initiate data transfers, instead of requiring the ordering box to initiate communication with the sources. In fact, an ordering box could even dynamically obtain the source information from boxes that cache respective segments for the title instead of requiring the server to identify the supplying boxes. Accordingly, the scope of the present invention is defined by the appended claims rather than the foregoing description of embodiments.

I claim:

1. A method for providing media-on-demand services, the method comprising:

preparing media data pertaining to a title into a sequence of data blocks, each of the data blocks including a chunk of data;

fragmenting the media data into K segments, each of the K segments including data blocks interleaved from the media data, thus the data blocks in each of the K segments are non-consecutive;

distributing the K segments into boxes in service, and causing an i-th segment of the K segments to be cached in at least a first box and a second box;

measuring an available uploading bandwidth of the first and second boxes when receiving an order for the title from an ordering box;

designating the first and second boxes to supply collectively the i-th segment to the ordering box, wherein the first device is configured to supply a first portion of the i-th segment not exceeding the available uploading bandwidth of the first device.

2. The method as recited in claim 1, further comprising: designating the second box to supply another one of the segment in addition to a second portion of the i-th segment to the ordering device without exceeding the available uploading bandwidth of the second device.

3. The method as recited in claim 2, wherein the ordering box is configured to multiplex the first and second portions of the i-th segment being streamed in to recover the i-th segment being streamed in, and at the same time to multiplex the i-th segment being streamed in and remaining K-1 segments some or all of which are being streamed in for a playback of the title on the ordering box.

4. A system for providing media-on-demand services, the system comprising

a plurality of boxes including an ordering box, a first device and a second device, the ordering box associated with a user being placed an order of a title offered in a library;

a server configured to perform operations of:

preparing media data pertaining to a title into a sequence of data blocks, each of the data blocks including a chunk of data;

fragmenting the media data into K segments, each of the K segments including data blocks interleaved from the media data, thus the data blocks in each of the K segments are non-consecutive;

distributing the K segments into the boxes ahead of the order, and causing an i-th segment of the K segments to be cached in at least the first box and the second box;

measuring an available uploading bandwidth of the first and second boxes when receiving the order for the title from the ordering box; and

designating the first and second boxes to supply collectively the i-th segment to the ordering box, wherein the first device is configured to supply a first portion of the i-th segment not exceeding the available uploading bandwidth of the first device.

5. The system as recited in claim 4, wherein the uploading bandwidth of the first box is below a predefined minimum uploading bandwidth so that only the first portion of the data segment is uploaded to the ordering box, and thus the second box is configured to upload a remaining portion of the i-th segment to the ordering box.

* * * * *