US 20250267320A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2025/0267320 A1**

PRESSNELL et al. (43) **Pub. Date: Aug. 21, 2025**

(54) **MEDIA STREAMING SYSTEMS AND METHODS WITH A PREFETCHING FUNCTION**

(71) Applicant: **Penthera Partners, Inc.**, New York, NY (US)

(72) Inventors: **Joshua PRESSNELL**, Spring Valley, OH (US); **Scott HALPERT**, Santa Monica, CA (US)

(73) Assignee: **Penthera Partners, Inc.**, New York, NY (US)

(21) Appl. No.: **18/663,993**

(22) Filed: **May 14, 2024**

Related U.S. Application Data

(60) Provisional application No. 63/554,573, filed on Feb. 16, 2024.

Publication Classification

(51) **Int. Cl.**

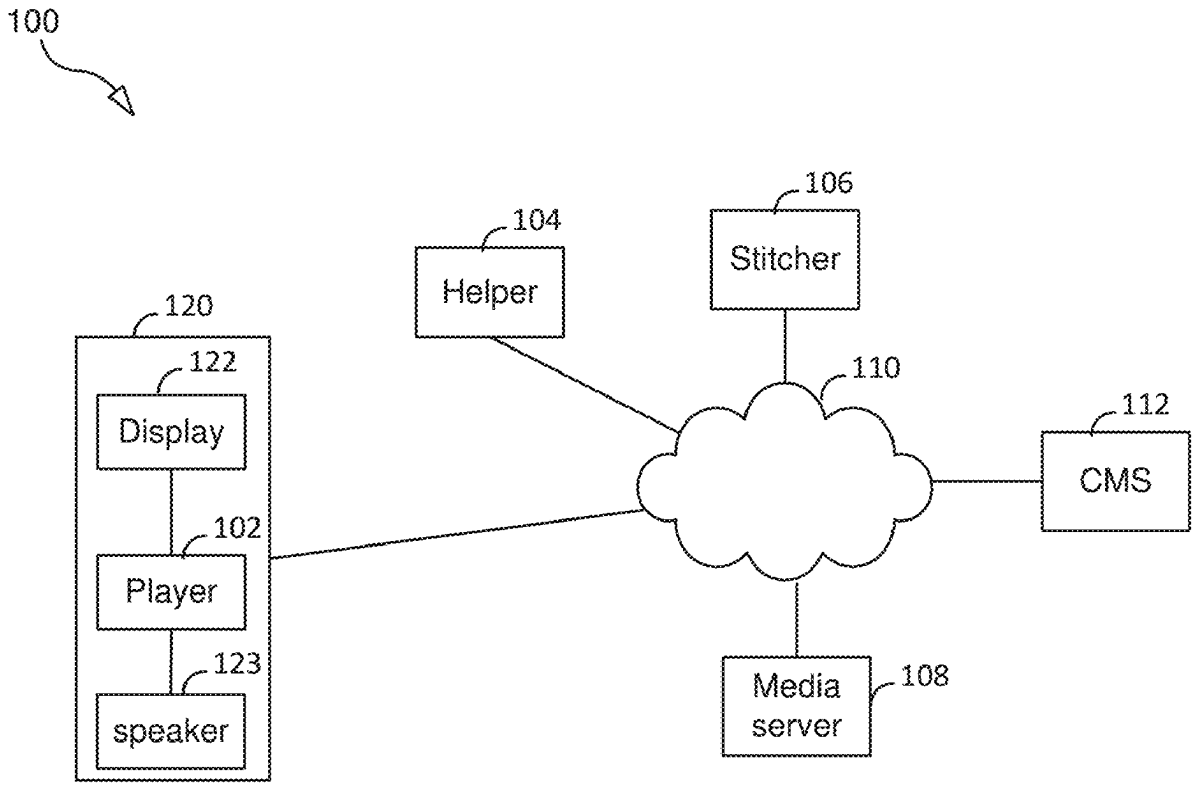
H04N 21/234	(2011.01)
H04N 21/239	(2011.01)
H04N 21/262	(2011.01)
H04N 21/81	(2011.01)

(52) **U.S. Cl.**

CPC ... ***H04N 21/23424*** (2013.01); ***H04N 21/2393*** (2013.01); ***H04N 21/26258*** (2013.01); ***H04N 21/812*** (2013.01)

(57) **ABSTRACT**

A method that includes providing to a player a manifest that comprises a segment identifier for identifying a segment of first media content or template information for generating the segment identifier. The method also includes receiving a segment request comprising the segment identifier and, in response to receiving the segment request: i) providing to the player a response message responsive to the segment request; and ii) performing a first process for enabling the player to obtain second media content to be played by the player during a break in the first media content. The first process for enabling the player to obtain the second media content comprises transmitting to a PF a prefetch request for triggering the PF to send to an ad server a metadata request. The ad server responds to the request by providing to the PF metadata for the second media content.



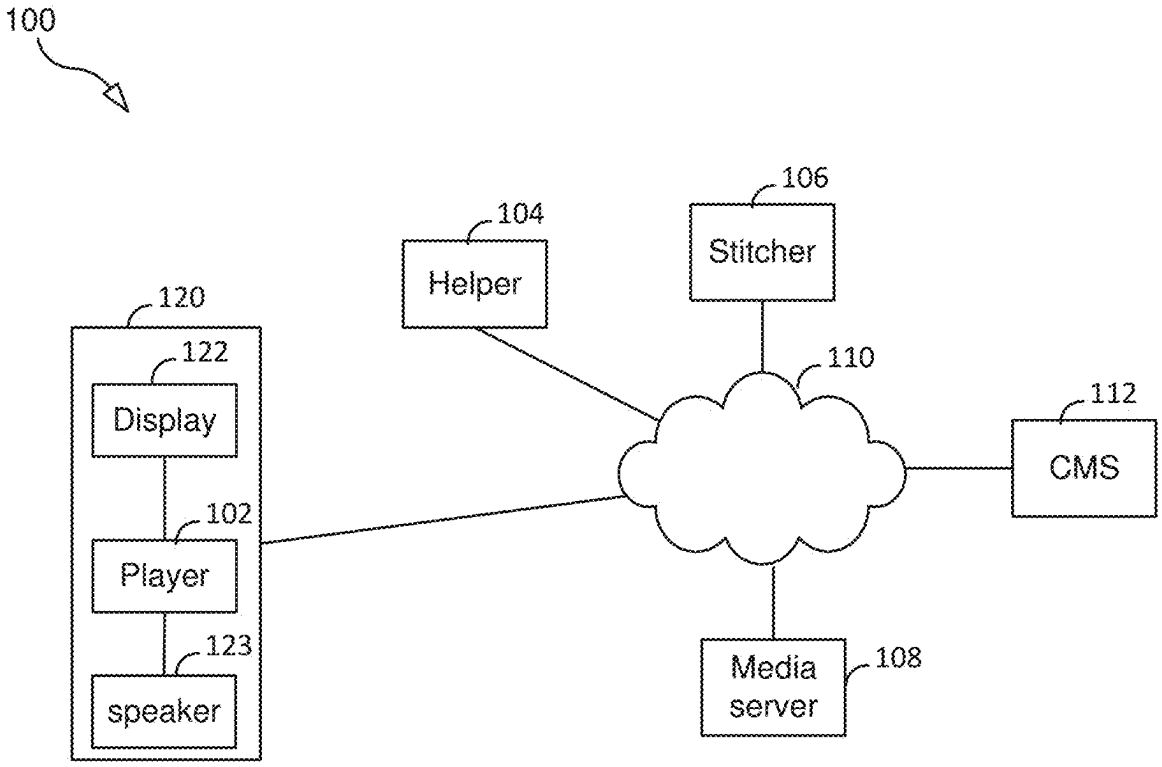

100
106
Stitcher
104
Helper
120
122
Display
110
112
CMS
102
Player
123
speaker
Media
server
108

FIG. 1

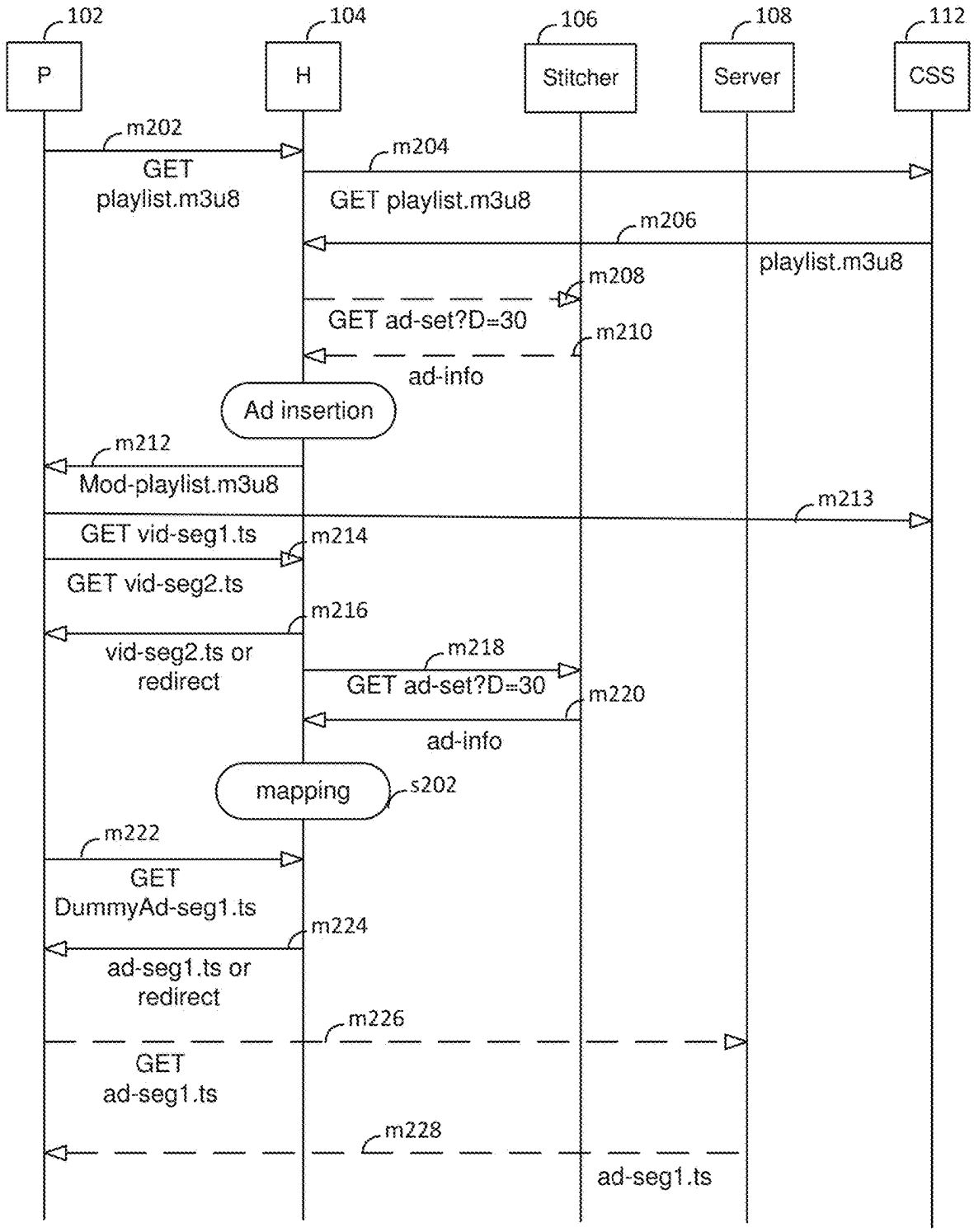

102
104
106
108
112
P
H
Stitcher
Server
CSS
m202
GET
playlist.m3u8
m204
GET playlist.m3u8
m206
playlist.m3u8
m208
GET ad-set?D=30
m210
ad-info
Ad insertion
m212
Mod-playlist.m3u8
m213
GET vid-seg1.ts
m214
GET vid-seg2.ts
m216
vid-seg2.ts or
redirect
m218
GET ad-set?D=30
m220
ad-info
mapping
s202
m222
GET
DummyAd-seg1.ts
m224
ad-seg1.ts or
redirect
m226
GET
ad-seg1.ts
m228
ad-seg1.ts

FIG. 2

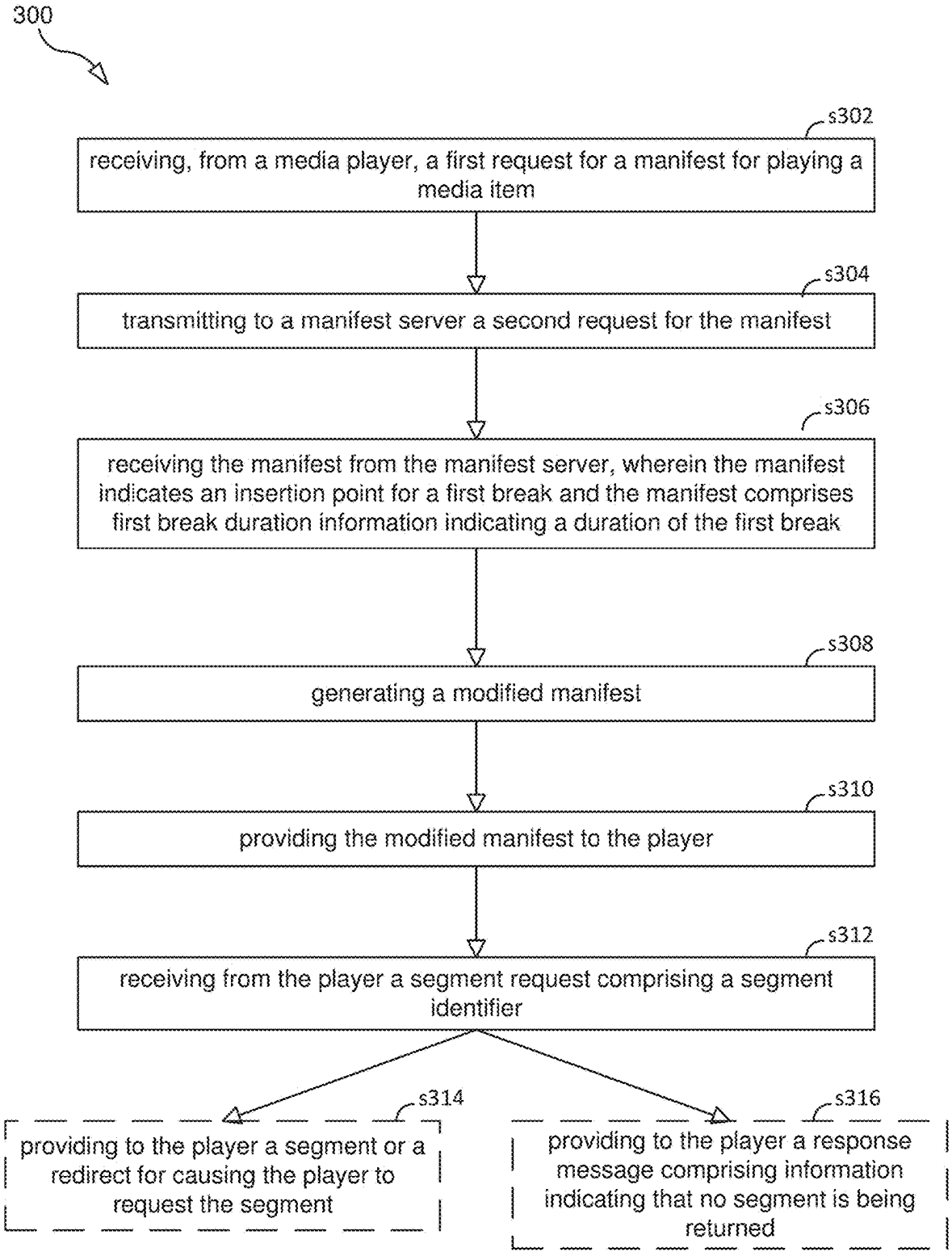

300
s302
receiving, from a media player, a first request for a manifest for playing a media item
s304
transmitting to a manifest server a second request for the manifest
s306
receiving the manifest from the manifest server, wherein the manifest indicates an insertion point for a first break and the manifest comprises first break duration information indicating a duration of the first break
s308
generating a modified manifest
s310
providing the modified manifest to the player
s312
receiving from the player a segment request comprising a segment identifier
s314
providing to the player a segment or a redirect for causing the player to request the segment
s316
providing to the player a response message comprising information indicating that no segment is being returned

FIG. 3

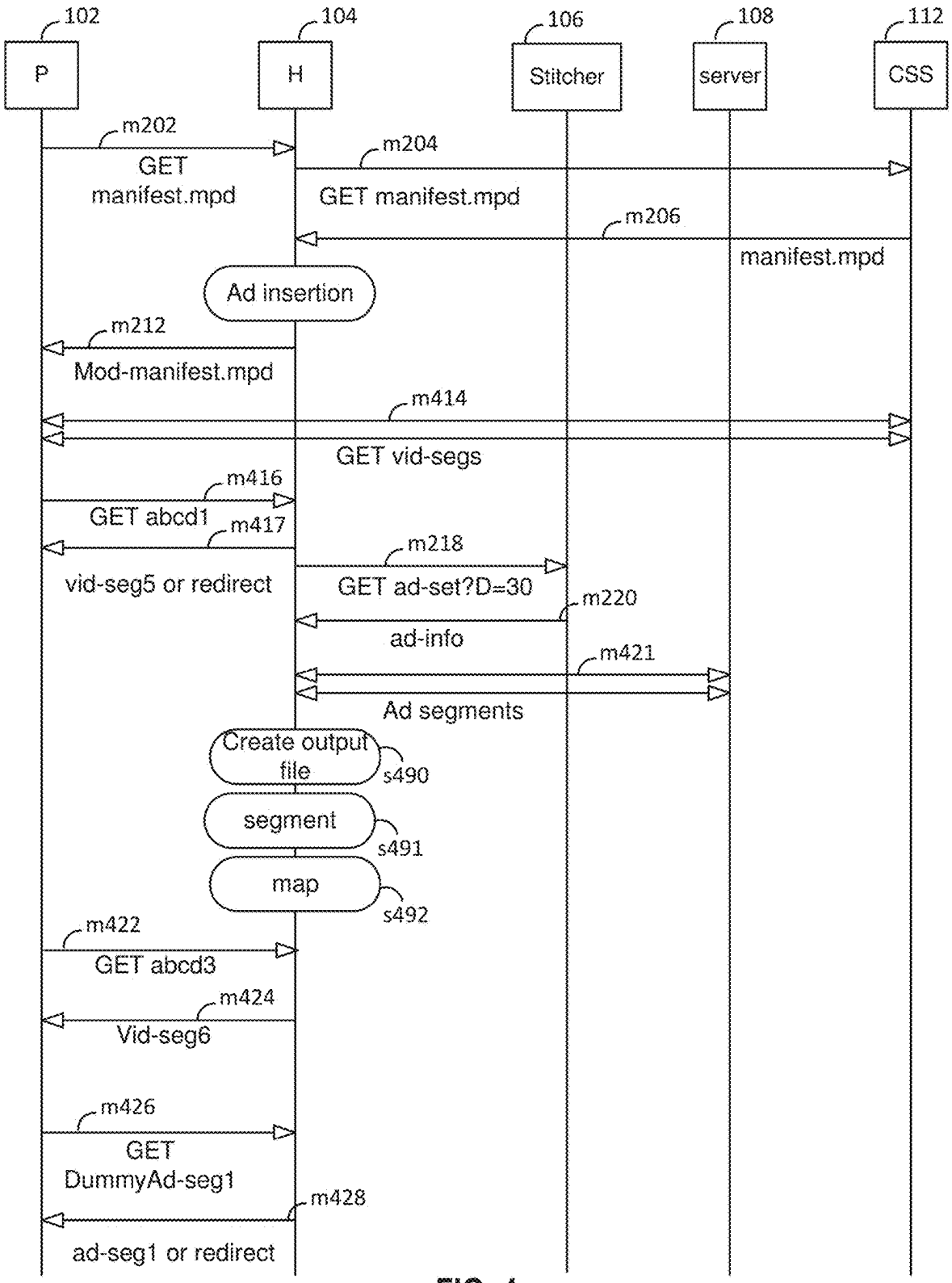

102
104
106
108
112
P
H
Stitcher
server
CSS
m202
GET
manifest.mpd
m204
GET manifest.mpd
m206
manifest.mpd
Ad insertion
m212
Mod-manifest.mpd
m414
GET vid-segs
m416
GET abcd1
m417
vid-seg5 or redirect
m218
GET ad-set?D=30
m220
ad-info
m421
Ad segments
Create output file
s490
segment
s491
map
s492
m422
GET abcd3
m424
Vid-seg6
m426
GET
DummyAd-seg1
m428
ad-seg1 or redirect

FIG. 4

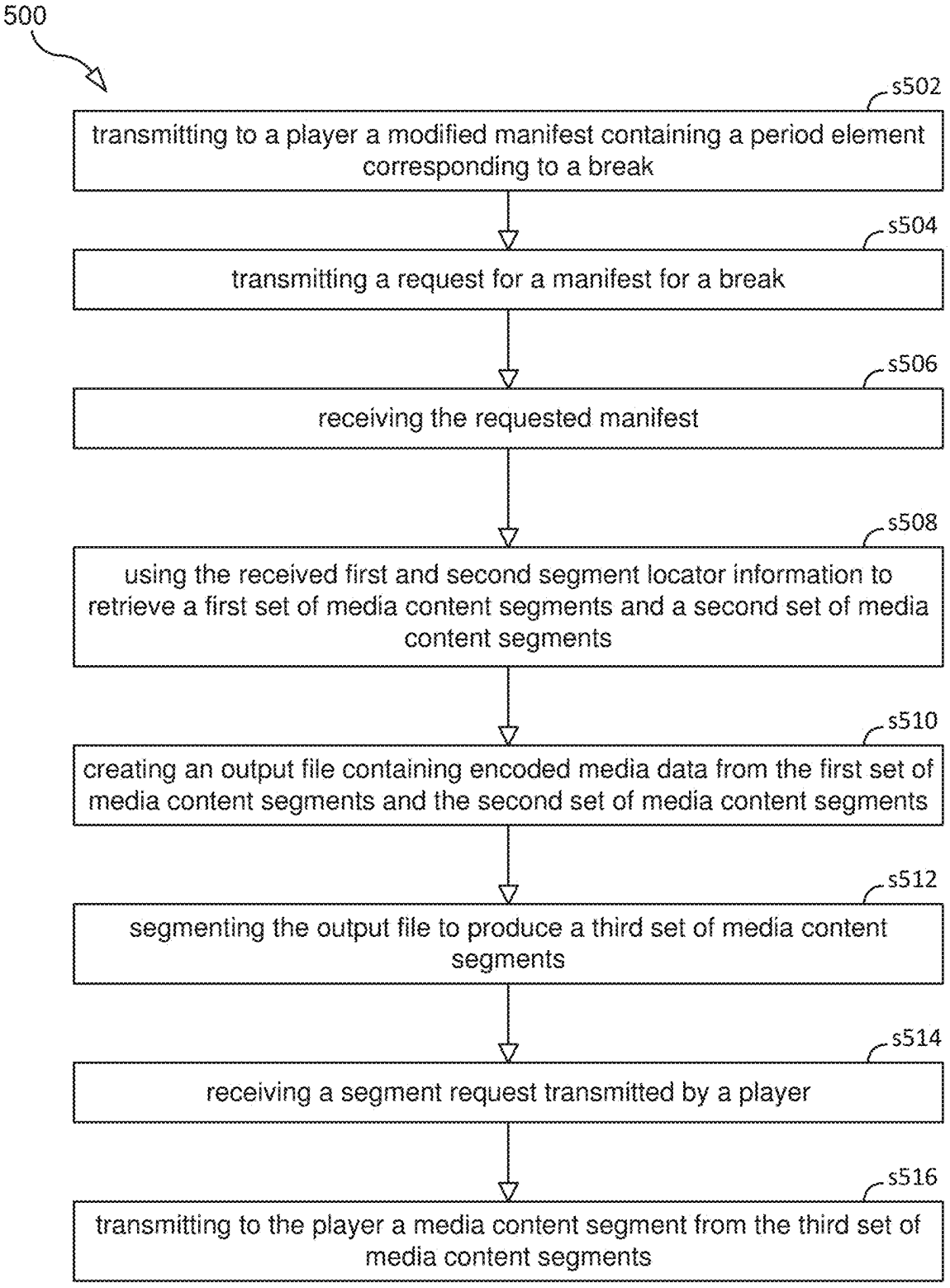

500
s502
transmitting to a player a modified manifest containing a period element corresponding to a break
s504
transmitting a request for a manifest for a break
s506
receiving the requested manifest
s508
using the received first and second segment locator information to retrieve a first set of media content segments and a second set of media content segments
s510
creating an output file containing encoded media data from the first set of media content segments and the second set of media content segments
s512
segmenting the output file to produce a third set of media content segments
s514
receiving a segment request transmitted by a player
s516
transmitting to the player a media content segment from the third set of media content segments

FIG. 5

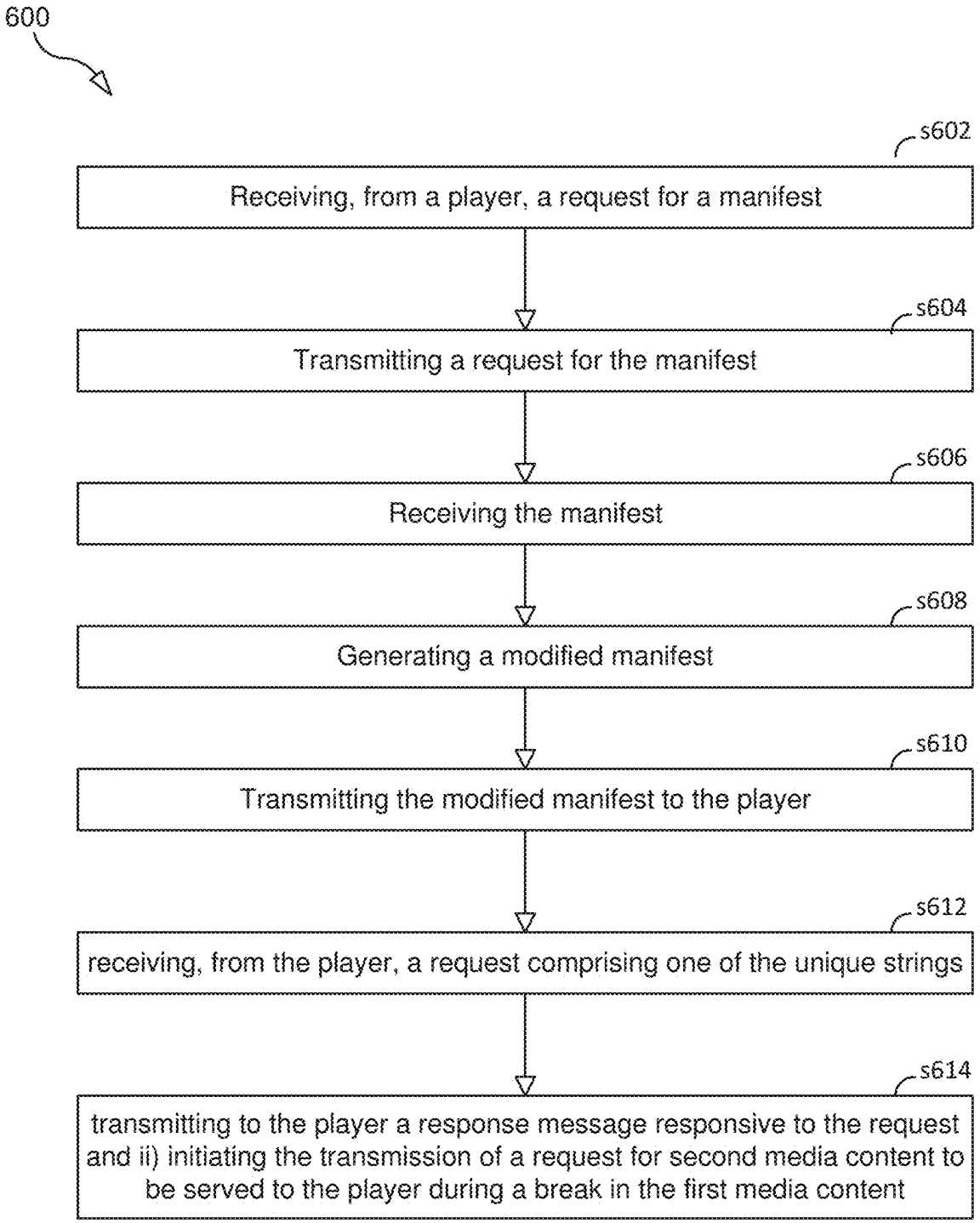

600
s602
Receiving, from a player, a request for a manifest
s604
Transmitting a request for the manifest
s606
Receiving the manifest
s608
Generating a modified manifest
s610
Transmitting the modified manifest to the player
s612
receiving, from the player, a request comprising one of the unique strings
s614
transmitting to the player a response message responsive to the request and ii) initiating the transmission of a request for second media content to be served to the player during a break in the first media content

FIG. 6

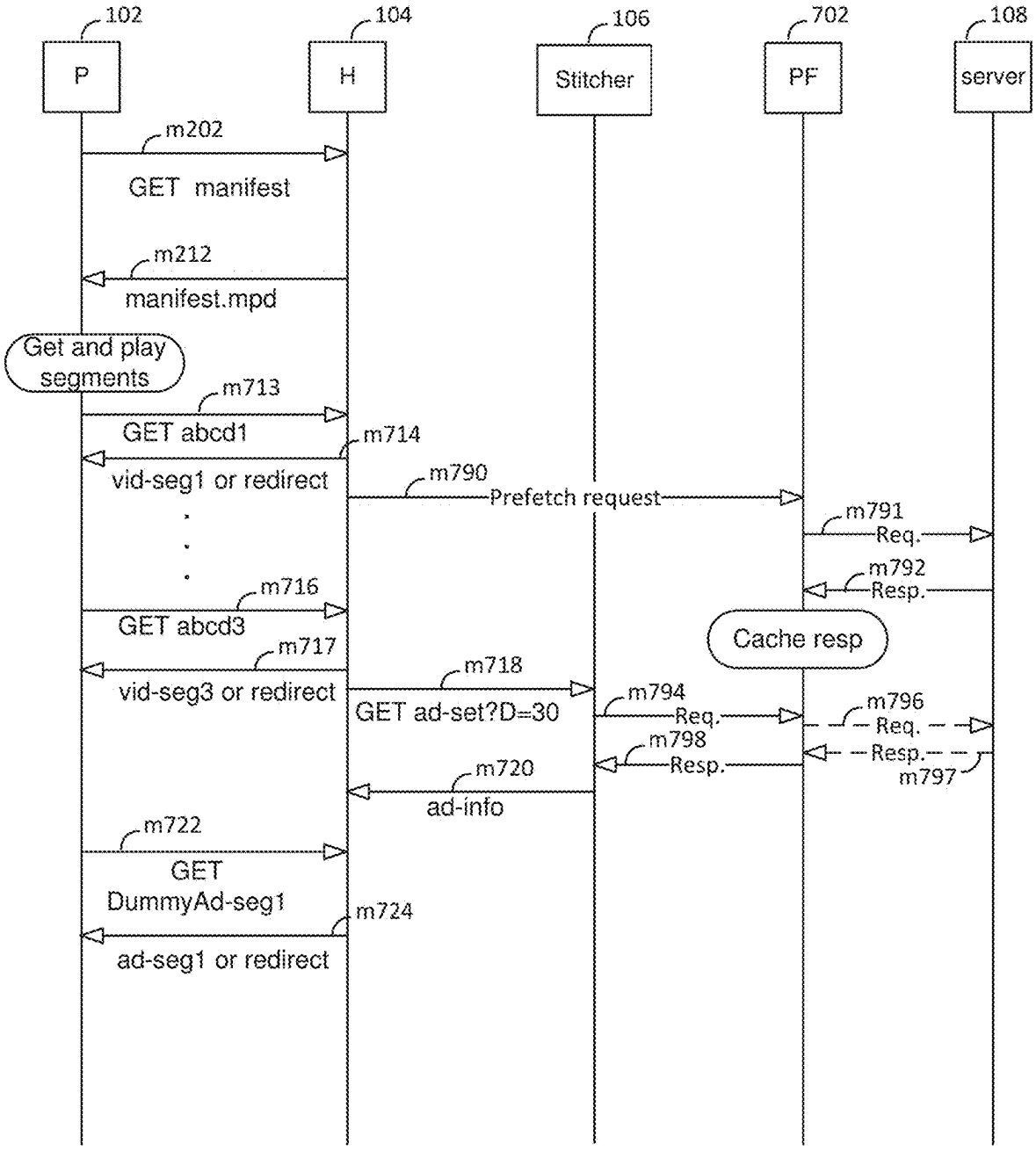

102
104
106
702
108
P
H
Stitcher
PF
server
m202
GET manifest
m212
manifest.mpd
Get and play segments
m713
GET abcd1
m714
vid-seg1 or redirect
m790
Prefetch request
m791
Req.
m792
Resp.
m716
GET abcd3
Cache resp
m717
vid-seg3 or redirect
m718
GET ad-set?D=30
m794
Req.
m796
Req.
m798
Resp.
Resp.
m797
m720
ad-info
m722
GET DummyAd-seg1
m724
ad-seg1 or redirect

FIG. 7

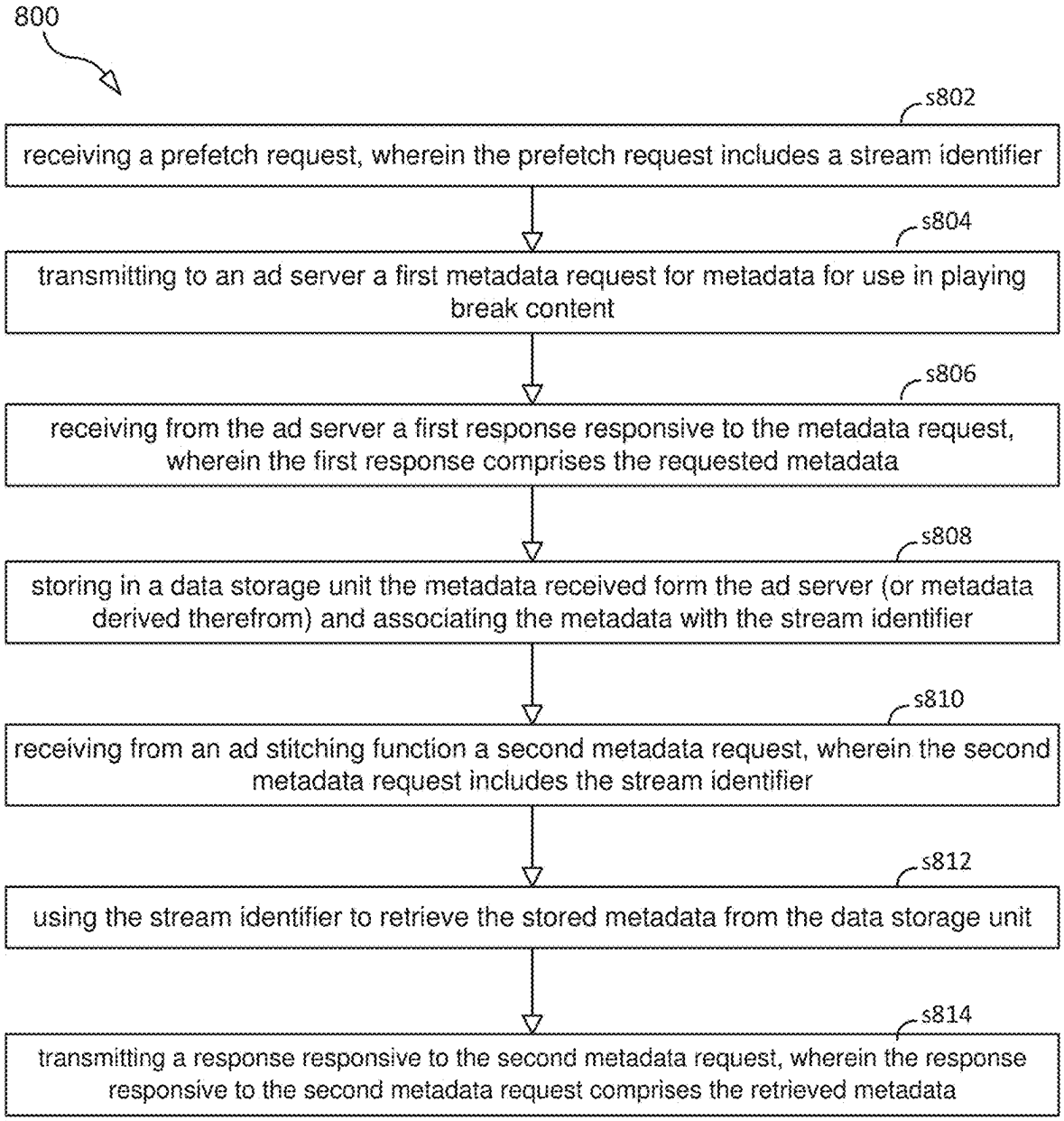

800
s802
receiving a prefetch request, wherein the prefetch request includes a stream identifier
s804
transmitting to an ad server a first metadata request for metadata for use in playing break content
s806
receiving from the ad server a first response responsive to the metadata request, wherein the first response comprises the requested metadata
s808
storing in a data storage unit the metadata received form the ad server (or metadata derived therefrom) and associating the metadata with the stream identifier
s810
receiving from an ad stitching function a second metadata request, wherein the second metadata request includes the stream identifier
s812
using the stream identifier to retrieve the stored metadata from the data storage unit
s814
transmitting a response responsive to the second metadata request, wherein the response responsive to the second metadata request comprises the retrieved metadata

FIG. 8

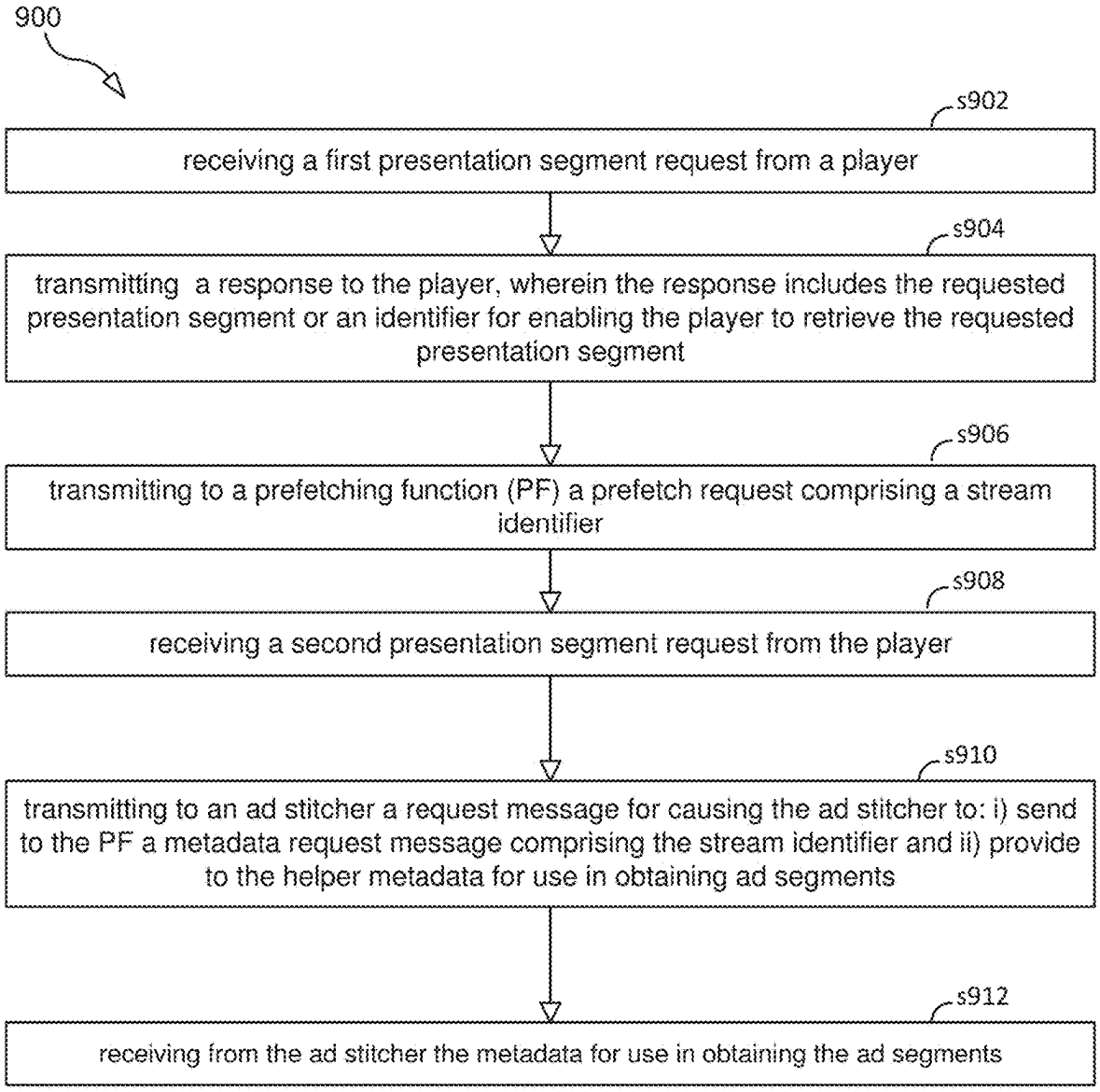

900
s902
receiving a first presentation segment request from a player
s904
transmitting a response to the player, wherein the response includes the requested presentation segment or an identifier for enabling the player to retrieve the requested presentation segment
s906
transmitting to a prefetching function (PF) a prefetch request comprising a stream identifier
s908
receiving a second presentation segment request from the player
s910
transmitting to an ad stitcher a request message for causing the ad stitcher to: i) send to the PF a metadata request message comprising the stream identifier and ii) provide to the helper metadata for use in obtaining ad segments
s912
receiving from the ad stitcher the metadata for use in obtaining the ad segments

FIG. 9

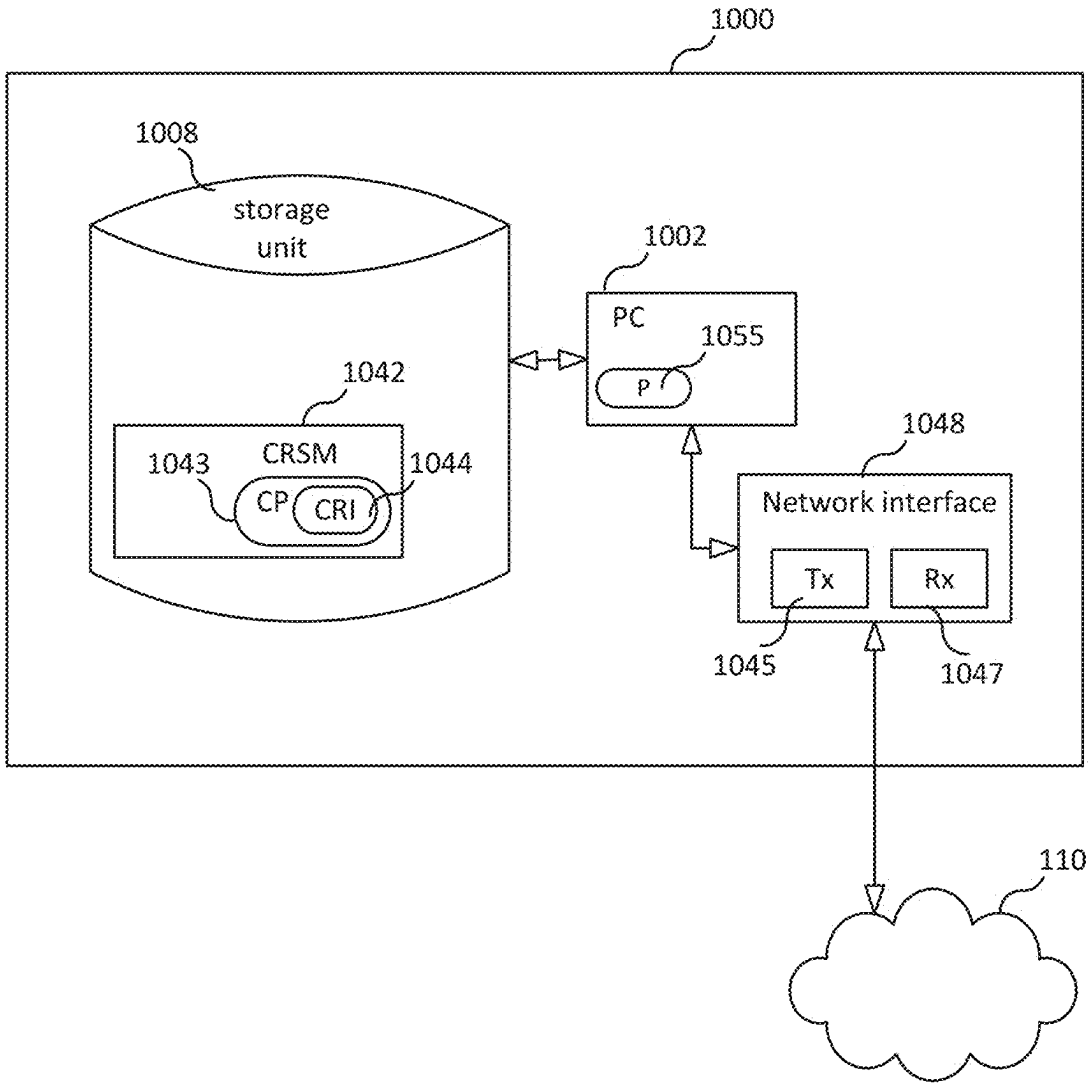

1000
1008
storage unit
1002
PC
1055
P
1042
CRSM
1043
1044
CP
CRI
1048
Network interface
Tx
Rx
1045
1047
110

FIG. 10

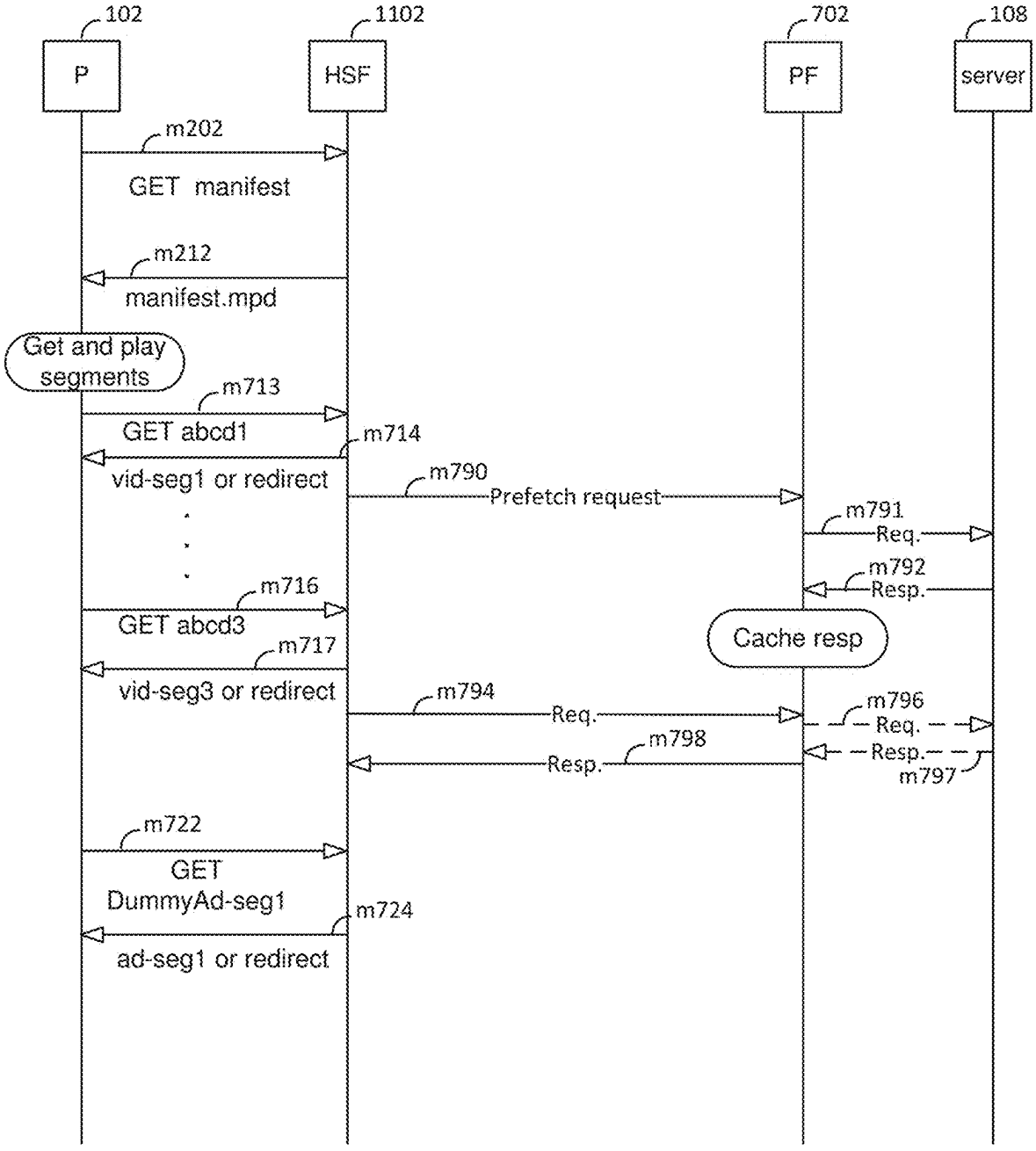

102
P
1102
HSF
702
PF
108
server
m202
GET manifest
m212
manifest.mpd
Get and play segments
m713
GET abcd1
m714
vid-seg1 or redirect
m790
Prefetch request
m791
Req.
m792
Resp.
m716
GET abcd3
Cache resp
m717
vid-seg3 or redirect
m794
Req.
m796
Req.
m798
Resp.
Resp.
m797
m722
GET DummyAd-seg1
m724
ad-seg1 or redirect

FIG. 11

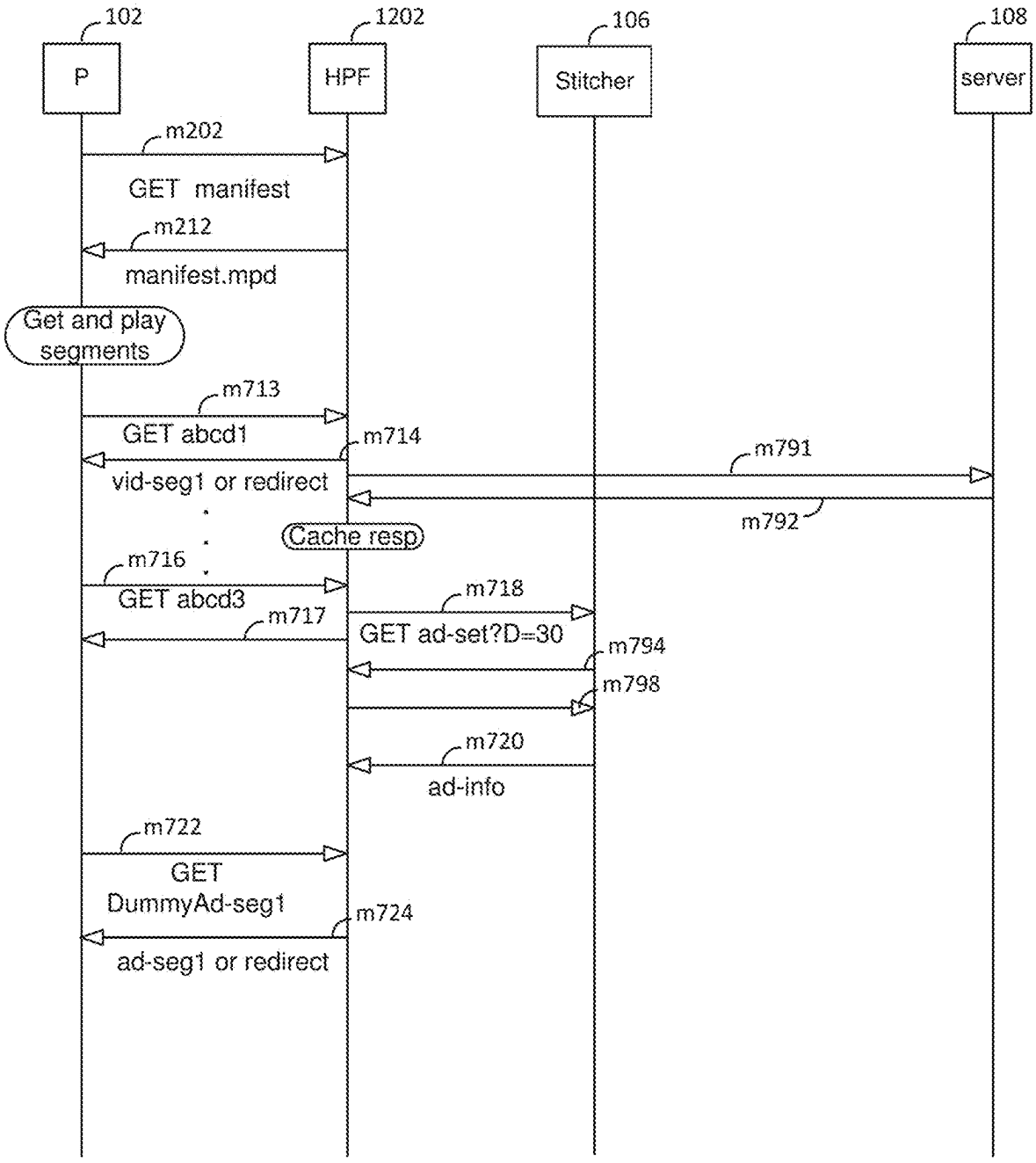

102
P
1202
HPF
106
Stitcher
108
server
m202
GET manifest
m212
manifest.mpd
Get and play segments
m713
GET abcd1
m714
vid-seg1 or redirect
m791
m792
Cache resp
m716
GET abcd3
m718
GET ad-set?D=30
m717
m794
m798
m720
ad-info
m722
GET DummyAd-seg1
m724
ad-seg1 or redirect

FIG. 12

MEDIA STREAMING SYSTEMS AND METHODS WITH A PREFETCHING FUNCTION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Patent Application No. 63/554,573, filed on 2024 Feb. 16, which is incorporated by this reference.

TECHNICAL FIELD

[0002] Disclosed are embodiments related to media (e.g., video and/or audio) streaming systems and methods.

BACKGROUND

[0003] Technology for streaming media content (e.g., video content and/or audio content) to a user is well established. Most streaming technologies, including Hypertext Transfer Protocol (HTTP) Live Streaming (HLS), a technology developed by Apple, Inc., and MPEG-Dynamic Adaptive Streaming over HTTP (DASH), work by dividing a media item (e.g., a 30 minute episode of a TV show) into many segments (e.g., **300** six-second segments) and providing to the player a segment manifest (e.g., an HLS playlist or an MPEG-DASH Media Presentation Description (MPD) or other information element that contains segment locator information) that enables the player to send to a server a request for each segment. Thus, for example, when a user wants to consume (e.g., watch and/or listen to) a particular media item, the user's media player may make a request to a server for a particular segment manifest (or manifest for short) for the media item (e.g., an HLS manifest or MPD), and, then, after obtaining the requested manifest, serially request the segments identified in the manifest (e.g., for each segment of the media item, the manifest may contain a Uniform Resource Locator (URL) that identifies the filename of the file that stores the segment and the location of the file on the Internet and/or segment template information that enables the player to determine the filenames and file locations).

[0004] After a player requests a manifest and before the manifest is provided to player, a process may select media to be inserted (e.g., a set of one or more ads) for an identified break (e.g., using an HLS playlist, the location of the break and its duration can be specified using the #EXT-X-CUE-OUT and #EXT-X-CUE-IN tags) and then insert into the manifest at the break position information for identifying segments of the selected media to be inserted (e.g., a set of ad segment URLs). In this way, selected media can be inserted into a media item at a predefined break.

[0005] For example, Table 1A below illustrates an example HLS playlist as it exists on the playlist server and Table 1B below illustrates a modified version of the HLS playlist shown in Table 1A which is provided to the player. A comparison of the two tables shows that five segment URLs have been inserted into the playlist.

TABLE 1A

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:6.0
http://cdn.com/vid-seg1.ts
#EXTINF:6.0
http://cdn.com/vid-seg2.ts
#EXT-X-CUE-OUT: 30.00
#EXT-X-CUE-IN:
#EXTINF:6.0
http://cdn.com/vid-seg3.ts
```

TABLE 1B

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:6.0
http://cdn.com/vid-seg1.ts
#EXTINF:6.0
http://cdn.com/vid-seg2.ts
#EXT-X-DISCONTINUITY
#EXTINF:6.0
http://adserver.com/Ad-seg1.ts
#EXTINF:6.0
http://adserver.com/Ad-seg2.ts
#EXTINF:6.0
http://adserver.com/Ad-seg3.ts
#EXTINF:6.0
http://adserver.com/Ad-seg4.ts
#EXTINF:6.0
http://adserver.com/Ad-seg5.ts
#EXT-X-DISCONTINUITY
#EXTINF:6.0
http://cdn.com/vid-seg3.ts
```

[0006] As another example, Table 2A below illustrates an example MPEG-DASH MPD as it exists on the manifest server and Table 2B below illustrates a modified version of the manifest shown in Table 2A, which is provided to the player. A comparison of tables **2**A and **2**B shows that a period element corresponding to a break and containing five ad segment URLs has been inserted into the manifest between a first period element containing a first subset of the segment URLs from the period element in the original manifest and a third period containing a second subset of the segment URLs from the period element in the original manifest.

TABLE 2A

```
<MPD ...>
<Period...>
  <AdaptationSet mimeType="video/mp4" ...>
    <Representation id="rep1" bandwidth="10392000" ...>
      <SegmentList ...>
        <SegmentURL media="http://cdn.com/vid-rep1-seg1.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg2.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg3.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg4.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg5.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg6.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg7.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg8.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg9.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg10.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg11.mp4"/>
        <Initialization sourceURL="http://cdn.com/init1.mp4"/>
      </SegmentList>
    </Representation>
  </AdaptationSet>
</Period>
```

TABLE 2B

```
<MPD ...>
<Period id="1" ...>
  <AdaptationSet mimeType="video/mp4" ...>
    <Representation bandwidth="10392000" ...>
      <SegmentList ...>
        <SegmentURL media="http://cdn.com/vid-rep1-seg1.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg2.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg3.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg4.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg5.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg6.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg7.mp4"/>
        <Initialization sourceURL="http://cdn.com/init1.mp4"/>
      </SegmentList>
    </Representation>
  </AdaptationSet>
</Period>
<Period id="ad_break_1"...> /*This period is for the break **/
 <AdaptationSet mimeType="video/mp4" ...>
    <Representation bandwidth="10392000" ...>
      <SegmentList ...>
        <SegmentURL media="http://adserver.com/Ad-rep1-seg1.mp4"/>
        <SegmentURL media="http://adserver.com/Ad-rep1-seg2.mp4"/>
        <SegmentURL media="http://adserver.com/Ad-rep1-seg3.mp4"/>
        <SegmentURL media="http://adserver.com/Ad-rep1-seg4.mp4"/>
        <SegmentURL media="http://adserver.com/Ad-rep1-seg5.mp4"/>
        <Initialization sourceURL="http://adserver.com/init.mp4"/>
      </SegmentList>
    </Representation>
  </AdaptationSet>
</Period>
<Period id="2"...>
 <AdaptationSet mimeType="video/mp4" ...>
    <Representation bandwidth="10392000" ...>
      <SegmentList ...>
        <SegmentURL media="http://cdn.com/vid-rep1-seg8.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg9.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg10.mp4"/>
        <SegmentURL media="http://cdn.com/vid-rep1-seg11.mp4"/>
        <Initialization sourceURL="http://cdn.com/init1.mp4"/>
      </SegmentList>
    </Representation>
  </AdaptationSet>
</Period>
</mpd>
```

SUMMARY

[0007] Certain challenges presently exist. For instance, due to the nature of server side ad-insertion (SSAI), the timeout in the request from an ad stitcher to the ad server is typically set at a maximum of 3 seconds, and, based on observations and market research, this short timeout period causes a significant loss in ads delivered during peak hours.

[0008] Accordingly, in one aspect there is provided a method that overcomes this timing issue. In one embodiment, the method includes providing a manifest to a player, wherein the manifest comprises a segment identifier for identifying a segment of first media content (e.g., a segment of an episode of show or a segment of a move) or template information for generating the segment identifier. The method also includes receiving, from the player, a first segment request comprising the segment identifier for identifying the segment of first media content. The method also includes, in response to receiving the first segment request: i) providing to the player a first response message responsive to the first segment request; and ii) performing a first process for enabling the player to obtain second media content to be played by the player during a break in the first media content. The first process for enabling the player to obtain the second media content comprises transmitting to a prefetching function (PF) a prefetch request for triggering the PF to send to an ad server a metadata request, wherein the ad server responds to the metadata request by providing to the PF metadata for the second media content.

[0009] In another embodiment, the method includes receiving a first segment request from a player. The method also includes, in response to receiving the first segment request: i) transmitting a first response to the player, wherein the first response includes a first segment of first media content or a first segment identifier for enabling the player to retrieve the first segment of the first media content; and ii) transmitting to a prefetching function (PF) a prefetch request comprising a stream identifier. The method also includes receiving a second segment request from the player. The method further includes, in response to receiving the second segment request: i) transmitting a second response to the player, wherein the second response includes a second segment of the first media content or a second segment identifier for enabling the player to retrieve the second segment of the first media content; and ii) performing a process for obtaining segment locator information for use in retrieving segments of second media content.

[0010] In another embodiment, the method includes receiving a first segment request from a player. The method also includes, in response to receiving the first segment request: i) transmitting a first response to the player, wherein the first response includes a first segment of first media content or a first segment identifier for enabling the player to retrieve the first segment of the first media content; and ii) transmitting to an ad server a first metadata request. The method also includes receiving from the ad server a first metadata response responsive to the first metadata request, wherein the first metadata response comprises metadata for second media content. The method also includes caching the metadata. The method also includes receiving a second segment request from the player. The method further includes in response to receiving the second segment request: i) transmitting a second response to the player, wherein the second response includes a second segment of the first media content or a second segment identifier for enabling the player to retrieve the second segment of the first media content; and ii) performing a process for obtaining segment locator information for use in retrieving segments of the second media content.

[0011] In another aspect there is provided a computer program comprising instructions which when executed by processing circuitry of an apparatus causes the apparatus to perform any of the methods disclosed herein. In one embodiment, there is provided a carrier containing the computer program wherein the carrier is one of an electronic signal, an optical signal, a radio signal, and a computer readable storage medium. In another aspect there is provided an apparatus that is configured to perform the methods disclosed herein. The apparatus may include memory and processing circuitry coupled to the memory.

[0012] An advantage of the embodiments disclosed herein is that they enable the selection of an ad set after a manifest has already been provided to the media player.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The accompanying drawings, which are incorporated herein and form part of the specification, illustrate various embodiments.

[0014] FIG. **1** illustrates a system according to an embodiment.

[0015] FIG. **2** is a message flow diagram according to an embodiment.

[0016] FIG. **3** is a flowchart illustrating a process according to an embodiment.

[0017] FIG. **4** is a message flow diagram according to an embodiment.

[0018] FIG. **5** is a flowchart illustrating a process according to an embodiment.

[0019] FIG. **6** is a flowchart illustrating a process according to an embodiment.

[0020] FIG. **7** is a message flow diagram according to an embodiment.

[0021] FIG. **8** is a flowchart illustrating a process according to an embodiment.

[0022] FIG. **9** is a flowchart illustrating a process according to an embodiment.

[0023] FIG. **10** is a block diagram an apparatus according to an embodiment.

[0024] FIG. **11** is a message flow diagram according to an embodiment.

[0025] FIG. **12** is a message flow diagram according to an embodiment.

DETAILED DESCRIPTION

[0026] FIG. **1** illustrates a system **100** according to an embodiment. System **100** includes a player **102**, a proxy **104** (a.k.a., helper **104**), a stitcher **106** (e.g., an ad stitcher or news bulletin stitcher etc.), a media server **108**, and a content serving system (CSS) **112** (e.g., a Content Data Network (CDN). Typically, the player **104** is a computer program that runs on a device **120** (e.g., mobile phone, smart TV, computer, etc.) having a display screen **122** for displaying decoded video frames and a speaker **123** for playing decoded audio frames. While helper **104** is illustrated as being remote from device **102**, in some embodiments, helper is a computer program that also runs on device **120**. Likewise, in some embodiments, helper **104** runs on a computer that is a component of CSS **112**. Similarly, while helper **104** is shown as being separate from stitcher **106**, helper **104** and stitcher **106** may be combined.

[0027] FIG. **2** is a message flow diagram illustrating a process according to an embodiment. The process may begin with player **102** sending a request message m**202** (e.g. HTTP GET message) identifying a manifest (e.g., playlist.m**3***u***8**). Request message m**202** is received at helper **104**.

[0028] In response to receiving request message m**202**, helper **104** sends to CSS **112** a request message m**204** identifying the manifest. In response to receiving message m**204**, CSS **112** sends to helper **104** a response message m**206** comprising the requested manifest.

[0029] In some embodiments, after receiving the requested manifest, helper **104** sends to stitcher **106** a request message m**208** identifying a content break duration (e.g., 30 seconds) and requesting a media set for the content break (for ease of explanation, assume that the media set is an ad set and the break is an ad break). Stitcher **106** responds to the request message by transmitting to helper a response message m**210** that comprises ad information or an indication that no ads are available. The ad information may identify one or more ads and, for each identified ad, may include a set of ad segment URLs for the ad. For example, the ad information may indicate that a single 15 second ad was selected and may contain 3 ad segment URLs for this 15 second ad (e.g., adserver.com/ad-seg**1**.*ts*, adserver.com/ad-seg**2**.*ts*, adserver.com/ad-seg**3**.*ts*).

[0030] For each ad break identified in the manifest, helper **104** performs an ad break placeholder insertion process that, in at least one embodiment, comprises: i) determining the duration of the ad break; ii) determining ad set information based on (e.g., based solely on) the duration of the ad break; and iii) inserting into the manifest at the location of the ad break the determined ad set information (or information derived therefrom), which information includes segment locator information (e.g., a set of URLs). After inserting the information including the segment locator information into the manifest, thereby producing a modified manifest, helper **104** provides to player **102** response message m**212**, which contains the modified manifest.

[0031] In one embodiment, helper **104** has a lookup table that maps (directly or indirectly) each possible ad break duration to corresponding ad set information. Accordingly, in one embodiment the step of determining the ad set information to insert into the manifest based on (e.g., based solely on) the duration of the ad break comprises using the

lookup table to determine the ad set information that is associated with the ad break duration. It is this determined ad set information (or information derived therefrom) that can be inserted into the manifest.

[0032] An example lookup table is shown below Table 3.

TABLE 3

Ad Break Duration	Ad Set Information
30 seconds	http://helper.com/DummyAd-seg1.ts http://helper.com/DummyAd-seg2.ts http://helper.com/DummyAd-seg3.ts #PEB http://helper.com/DummyAd-seg4.ts http://helper.com/DummyAd-seg5.ts #PEB http://helper.com/DummyAd-seg6.ts
60 seconds	http://helper.com/DummyAd-seg1.ts http://helper.com/DummyAd-seg2.ts http://helper.com/DummyAd-seg3.ts #PEB http://helper.com/DummyAd-seg4.ts http://helper.com/DummyAd-seg5.ts #PEB http://helper.com/DummyAd-seg6.ts #PEB http://helper.com/DummyAd-seg7.ts http://helper.com/DummyAd-seg8.ts #PEB http://helper.com/DummyAd-seg9.ts #PEB http://helper.com/DummyAd-seg10.ts #PEB http://helper.com/DummyAd-seg11.ts #PEB http://helper.com/DummyAd-seg12.ts
. . .	

[0033] While each URL included in Table 3 is a unique URL (i.e. is different in some way for each other URL), this is not a requirement. For example, each URL included in table **3** could simply be: "helper.com/DummyAd-seg" or even "DummyAd-seg." The same information shown above can also be stored more efficiently as shown in Table 4 below:

TABLE 4

Ad Break Duration	Data from which Ad Set Information to be inserted can be derived
30 seconds	Max Number of Possible Ad Segments (Max_num) = 6 PEB locations = (3, 5)
60 seconds	Max Number of Possible Ad Segments (Max_num) = 12 PEB locations = (3, 5, 6, 8, 9, 11)
. . .	

[0034] As illustrated above, in one embodiment the ad set information associated with an ad break duration not only includes a value that specifies the maximum number of possible ad segments required for the ad set, but also an array that specifies all potential encoding breaks (PEBs) for the ad break.

Determining the Maximum Number of Possible Ad Segments:

[0035] In one embodiment, to create Table 3 or Table 4, for each possible ad break duration, the maximum number of possible ad segments (Max_num) must be determined. To determine Max_num for a particular ad break duration, one must determine: 1) the set of possible ad sets for that ad break duration and **2**) for each such possible ad set, the number of segments required by that ad set. Max_num is then defined as: Max_num=max (S_1, S_2, . . . , S_N), where N is the number of possible ad sets for the given ad break duration and S_i for i=1 to N is the number of segments required by the ith possible ad set for the given ad break duration.

[0036] As a specific example, given a 30 second ad break duration and given that ads come in only two flavors (15 second ad or 30 second ad), one can deduce the set of possible ad sets for the 30 second ad break. Specifically, for this example, there are two possible ad sets that can fit within the 30 second ad break (i.e., N=2): i) a 30 second ad and ii) a two consecutive 15 second ads.

[0037] As another specific example, given a 60 second ad break duration and given that ads come in only two flavors (15 second ad or 30 second ad), one can deduce the set of possible ad sets for the 60 second ad break. Specifically, for this example, there are six possible ad sets that can fit within the 60 second ad break (i.e., N=6): i) a 60 second ad, ii) two consecutive 30 second ads, iii) a 30 second ad followed by two 15 second ads, iv) two 15 seconds followed by a 30 second ad, and v) a 15 second ad followed by a 30 second ad followed by a 15 second ad, and vi) four consecutive 15 second ads.

[0038] The value S_i is equal to sum (S_{i1}, S_{i2}, . . . , S_{iM}), wherein M is the number of ads within the ith possible ad set and s_{ij} is number of segments required by the jth ad within the ith ad set. The value s_{ij} is dependent on two values: 1) the duration of the ad and **2**) a segment duration. More specifically, S_{ij}=Ceiling ((Dur of adij)/(Seg_Dur)). In one embodiment, Seg_Dur is equal to 6 seconds. According, if an ad is 15 seconds, the number of segments required for that ad is: Ceiling (15/6)=Ceiling (2.5)=3. Hence, given Seg_Dur=6, an ad set that consists of two 15 second ads requires 3+3=6 segments; likewise, an ad set that consists of four 15 second ads requires 3× 4=12 segments.

[0039] Determining the Locations of Potential Encoding Breaks (PEBs)

[0040] Every possible ad set associated with a given ad break is associated with a set of encoding breaks. Accordingly, to determine all of the potential encoding breaks (PEBs) for the given ad break, one forms the union of these sets. For example, two ad sets are associated with the 30 second ad break: a first ad set consisting of a 30 second ad and a second ad set consisting of two consecutive 15 second ads. Given Seg_Dur=6, the first ad set requires only 5 segments, but Max_num for this ad break is 6 segments; hence, for the first ad set there is an encoding break located at segment **5** (or alternatively an encoding break located at segment **1**). The second ad set requires 3 segments for the first 15 second ad and 3 segments for the second 15 second ad; hence for the second ad set there is an encoding break located at segment **3**. Accordingly, the PEB locations for the 30 second ad break are: **3** and **5** (or **1** and **3**).

[0041] As another example, given an ad break of 60 seconds and Seg_Dur=6 seconds, then there is the potential to have 10-12 ad segments in that break. In the case of HLS, that means that discontinuity tags need to be placed just before the first ad segment, just after the last ad segment, and anywhere within the **10-12** ad segments that an ad switches

to a different ad or to a black segment. For example, give two 30 second ads, in one embodiment, the segments are arranged as follows: (Aij=Ad segment i for Ad j, P=Placeholder Segment):

[0042] A**11**-A**12**-A**13**-A**14**-A**15**-B-A**21**-A**22**-A**23**-A**24**-A**25**-B,

[0043] so that there are encoding breaks at location **5**, **6**, and **11**.

[0044] As another example, given two 15 second ads and a 30 second ad, in one embodiment the segments are arranged as:

[0045] A**11**-A**12**-A**13**-A**21**-A**22**-A**23**-A**31**-A**32**-A**33**-A**34**-A**35**-P,

[0046] so that there are encoding breaks at location **3**, **6**, and **11**.

[0047] As another example, given four 15 second ads, in one embodiment the segments are arranged as:

[0048] A**11**-A**12**-A**13**-A**21**-A**22**-A**23**-A**31**-A**32**-A**33**-A**41**-A**42**-A**43**,

[0049] so that there are encoding breaks at location **3**, **6**, and **9**.

[0050] Thus, a final structure that would support all 3 of these segment breakouts would be (where S=generic segment placeholder, and PEB indicates an potential encoding break): S-S-S-PEB-S-S-PEB-S-PEB-S-S-S-PEB-S-S-PEB-S.

[0051] Hence, for this example the PEB array corresponding to the 60 second ad break could be: PEB=(3, 5, 6, 9, 11). As is known in the art, there is an encoding break just prior to the beginning of the ad break and another encoding break at the end of the ad break.

[0052] In some embodiments, rather than determining all of the potential encoding breaks, helper **104** simply assumes there is an encoding break after each segment, leading to the following structure:

[0053] S-PEB-S-PEB-S-PEB-S-PEB-S-PEB-S-PEB-S-PEB-S-PEB-S-PEB-S-PEB-S-PEB-S.

[0054] Hence, for this example the PEB array corresponding to the 60 second ad break could be: PEB=(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11).

[0055] Inserting Ad Set Information into the Manifest

[0056] As noted above, in one embodiment helper **104** inserts into the manifest at the location of the ad break the determined ad set information (or, for example, segment locator information derived therefrom). Here will provide a specific example using an HLS playlist, but this disclosure also applies to other protocols, such as, for example, MPEG-DASH. For this example, the playlist shown in Table 1A is the playlist helper **104** received in response to its manifest request. For this example, we will also assume that the ad set information corresponding to the 30 second ad break identified in the playlist shown in Table 1A is the same ad set information corresponding to the 30 second ad break shown in Table 4. That is, in this example, the ad set information has Max_num=6 and PEB locations=(3, 5). Given this ad set information, helper may produce a modified playlist as shown in Table 5.

TABLE 5

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
```

TABLE 5-continued

```
#EXTINF:6.0
http://helper.com/vid-seg1.ts
#EXTINF:6.0
http://helper.com/vid-seg2.ts
#EXT-X-DISCONTINUITY
#EXTINF:6.0
http://helper.com/DummyAd-seg1.ts
#EXTINF:6.0
http://helper.com/DummyAd-seg2.ts
#EXTINF:6.0
http://helper.com/DummyAd-seg3.ts
#EXT-X-DISCONTINUITY
#EXTINF:6.0
http://helper.com/DummyAd-seg4.ts
#EXTINF:6.0
http://helper.com/DummyAd-seg5.ts
#EXT-X-DISCONTINUITY
#EXTINF:3.0
http://helper.com/DummyAd-seg6.ts
#EXT-X-DISCONTINUITY
#EXTINF:6.0
http://helper.com/vid-seg3.ts
```

[0057] More specifically, because Max_num=6, helper will insert at the ad break (i.e., between vid-seg**2** and vid-seg**3**) six placeholder (or 'dummy') URLs into the playlist. Also, because this is an HLS example, for each dummy URL there must be a preceding #EXTINF tag. Additionally, because PED=(3, 5), in addition to inserting an #EXT-X-DISCONTINUITY tag between vid-seg**2** and DummyAd-seg**1** and inserting an #EXT-X-DISCONTINUITY tag between DummyAd-seg**5** and vid-seg**3** (due to the known encoding breaks there), helper also inserts an #EXT-X-DISCONTINUITY tag between DummyAd-seg**3** and DummyAd-seg**4** and insert an #EXT-X-DISCONTINUITY tag between DummyAd-seg**5** and DummyAd-seg**6** because these are locations of the potential encoding breaks.

[0058] Additionally, as shown in the table above, in some embodiments, helper **104** may replace every string "cdn.com" with "helper.com" so that all of the content segment requests in addition to the ad segment requests will be sent by player **102** to helper **104**. In other embodiments (see e.g., Table 7), helper selectively replaces the string "cdn.com" with "helper.com" so that not all occurrences are changed. More generically, helper may modify the manifest so that at least some (i.e., all or less than all) segment requests are sent to the helper.

[0059] Additionally, in some embodiments, helper **104** will also append the following text to the playlist:

TABLE 6

```
#EXT-X-DISCONTINUITY
#EXTINF:1.0,
http://helper.com/AdditionalDummyAd-seg1.ts
#EXTINF:1.0,
http://helper.com/AdditionalDummyAd-seg2.ts
#EXTINF:1.0,
http://helper.com/AdditionalDummyAd-seg3.ts
#EXTINF:1.0,
http://helper.com/AdditionalDummyAd-seg4.ts
#EXTINF:1.0,
http://helper.com/AdditionalDummyAd-seg5.ts
#EXTINF:1.0,
http://helper.com/AdditionalDummyAd-seg6.ts
```

[0060] That is, in one embodiment, helper appends to the manifest a number of additional dummy URLs equal in number to the number dummy URLs added to the manifest.

Additionally, as shown in Table 6 above, helper **104** may specify that the duration of each additional dummy segment is 1 second.

[0061] As noted above, in some embodiments helper **104** may send request m**208** to stitcher **106** prior to performing the ad break placeholder insertion process. In this embodiment, helper **104** may be configured to modify the ad break placeholder insertion process so that instead of inserting all of the dummy URLs into the manifest it inserts less than all, depending on the response m**210** from stitcher **106**. For example, if the ad information included in response m**210** indicates that a single 15 second ad was selected and contains 3 ad segment URLs for this 15 second ad (e.g., adserver.com/ad-seg**1**.*ts*, adserver.com/ad-seg**2**.*ts*, adserver.com/ad-seg**3**.*ts*), then the modified manifest produced by helper **104** may appear as shown in Table 7 below. Alternatively, all of the dummy URLs are inserted into the manifest as shown in Table 5 and helper maps the first three dummy URLs to the ad segment URLs received in response m**210** (an example of such a mapping is shown in Tables 9 and 10, below).

TABLE 7

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:6.0
http://cdn.com/vid-seg1.ts
#EXTINF:6.0
http://helper.com/vid-seg2.ts
#EXT-X-DISCONTINUITY
#EXTINF:6.0
http://helper.com/ad-seg1.ts
#EXTINF:6.0
http://helper.com/ad-seg2.ts
#EXTINF:3.0
http://helper.com/ad-seg3.ts
#EXT-X-DISCONTINUITY
#EXTINF:6.0
http://helper.com/DummyAd-seg1.ts
#EXTINF:6.0
http://helper.com/DummyAd-seg2.ts
#EXT-X-DISCONTINUITY
#EXTINF:3.0
http://helper.com/DummyAd-seg3.ts
#EXT-X-DISCONTINUITY
#EXTINF:6.0
http://helper.com/vid-seg3.ts
```

[0062] As yet another example, if the ad information included in response m**210** indicates that a single 30 second ad was selected and contains 5 ad segment URLs for this 30 second ad (e.g., adserver.com/ad-seg**1**.*ts*, adserver.com/ad-seg**2**.*ts*, adserver.com/ad-seg**3**.*ts*, adserver.com/ad-seg**4**.*ts*, adserver.com/ad-seg**5**.*ts*), then the modified manifest produced by helper **104** may appear as shown in Table 8 below. Alternatively, all of the dummy URLs are inserted into the manifest as shown in Table 5 and helper maps five consecutive dummy URLs to the ad segment URLs received in response m**210** (an example of such a mapping is shown in Tables 11 and 12, below).

TABLE 8

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
```

TABLE 8-continued

```
#EXTINF:6.0
http://helper.com/vid-seg1.ts
#EXTINF:6.0
http://helper.com/vid-seg2.ts?trigger=1
#EXT-X-DISCONTINUITY
#EXTINF:6.0
http://helper.com/ad-seg1.ts
#EXTINF:6.0
http://helper.com/ad-seg2.ts
#EXTINF:6.0
http://helper.com/ad-seg3.ts
#EXTINF:6.0
http://helper.com/ad-seg4.ts
#EXTINF:6.0
http://helper.com/ad-seg5.ts
#EXT-X-DISCONTINUITY
#EXTINF:3.0
http://helper.com/DummyAd-seg1.ts
#EXT-X-DISCONTINUITY
#EXTINF:6.0
http://helper.com/vid-seg3.ts
```

[0063] Referring back to FIG. **2**, after player **102** receives the modified manifest (e.g., the playlist shown in Table 5, Table 7 or Table 8), player **102** begins sending segment requests according to the manifest (e.g., requests for segments of the presentation, which is a video in this example). For example, as shown in FIG. **2**, player **102** sends to CSS **112** a request message m**213** for vid-seg**1**.*ts* and sends to helper **104** a request message m**214** for vid-seg**2**.*ts*. CSS responds to request m**213** in a conventional manner and helper **104** responds to request m**214** by, at a minimum, providing to player **102** a response message m**216**, which either contains the requested segment (vid-seg**2**.*ts*) or a redirect message that causes player **102** to send a request message to a server (e.g. a server within CSS) for the segment.

[0064] At some point before player **102** requests the segment DummyAd-seg**1**.*ts*, or in response to receiving the request for segment DummyAd-seg**1**.*ts* (e.g., an HTTP GET request that includes a request target containing or consisting of the string "/DummyAd-set**1**.*ts*"), helper **104** transmits to stitcher **106** a request message m**218** indicating that helper is requesting one or more ads (i.e. a set of ads) to fill a 30 second ad break.

[0065] For example, in one embodiment, in response to receiving request m**214**, the helper may: 1) send to the player response message m**216** and **2**) initiate the transmission of a request to stitcher **106** (e.g., determine whether a request to stitcher **106** is needed, and, if so, send to stitcher a request message m**218** indicating that helper is requesting one or more ads (i.e. a set of ads) to fill an ad break). For instance, in one embodiment, the helper determines whether a request to stitcher **106** is needed by checking whether a flag is set to TRUE (e.g., equal to 1) and if the flag is set to TRUE, then it is determined that a request to stitcher **106** is needed. In another embodiment, the helper determines whether a request to stitcher **106** is needed by checking both whether the flag is set to TRUE and whether the request includes a trigger string (e.g., "trigger=1"), and if the flag is set to TRUE and request includes the trigger string, then it is determined that a request to stitcher **106** is needed. After sending the request, the flag is set to FALSE. The flag may be set back to TRUE after the ad break has finished.

[0066] Stitcher **106** responds to the request message m**218** by transmitting to helper **104** a response message m**220** that contains either segment location information for use in retrieving X ad segments (e.g,. X ad segment URLs or template information specifying the X ad segment URLs) or X ad segments, where X is greater than or equal to zero. More specifically, assuming only two flavors of ads (15 second ads or 30 second ads) and a segment duration of 6 seconds, then X is one of: **0**, **3**, **5**, or **6**. If X is 3, this means stitcher has selected a single 15 second ad; if X is 5, this means stitcher has selected a single 30 second ad; if X is 6, this means stitcher selected two fifteen second ads.

[0067] If X is greater than zero, then helper **104** will perform a mapping step s**202** that results in mapping X of the dummy URLs to a corresponding one of the X ad segment URLs. For example, assume that stitcher **106** returned X=3 ad segment URLs as follows: adserver.com/ad**1**-seg**1**.*ts*; adserver.com/ad**1**-seg**2**.*ts*; and adserver.com/ad**1**-seg**3**.*ts*. In this case, in performing mapping step s**02**, helper maps three consecutive dummy URLs to these ad segment URLs as shown in the table **9** or table **10** below as examples, whereas the other three dummy URLs that make up the ad break are either mapped to a short (e.g. 500 millisecond) placeholder segment (e.g., a segment that contains nothing but blank video frames or a segment that contains for example still video of a logo or a segment containing a house ad or branded slate any other video content to fill the time) or mapped to i) ad segment URLs received in response message m**210** and/or ii) ad segment URLs corresponding to ad segments received in response message m**210**.

TABLE 9

http://helper.com/DummyAd-seg1.ts	adserver.com/ad1-seg1.ts
http://helper.com/DummyAd-seg2.ts	adserver.com/ad1-seg2.ts
http://helper.com/DummyAd-seg3.ts	adserver.com/ad1-seg3.ts
http://helper.com/DummyAd-seg4.ts	placeholder_seg.ts
http://helper.com/DummyAd-seg5.ts	placeholder_seg.ts
http://helper.com/DummyAd-seg6.ts	placeholder_seg.ts

TABLE 10

http://helper.com/DummyAd-seg1.ts	placeholder_seg.ts
http://helper.com/DummyAd-seg2.ts	placeholder_seg.ts
http://helper.com/DummyAd-seg3.ts	placeholder_seg.ts
http://helper.com/DummyAd-seg4.ts	adserver.com/ad1-seg1.ts
http://helper.com/DummyAd-seg5.ts	adserver.com/ad1-seg2.ts
http://helper.com/DummyAd-seg6.ts	adserver.com/ad1-seg3.ts

[0068] As another example, assume that stitcher **106** returned X=5 ad segment URLs as follows: adserver.com/ad**1**-seg**1**.*ts*; adserver.com/ad**1**-seg**2**.*ts*; adserver.com/ad**1**-seg**3**.*ts*; adserver.com/ad**1**-seg**4**.*ts*; and; adserver.com/ad**1**-seg**5**.*ts*. In this case, helper will map five consecutive dummy URLs to these ad segment URLs as shown in Table 11 or Table 12 below.

TABLE 11

http://helper.com/DummyAd-seg1.ts	adserver.com/ad1-seg1.ts
http://helper.com/DummyAd-seg2.ts	adserver.com/ad1-seg2.ts
http://helper.com/DummyAd-seg3.ts	adserver.com/ad1-seg3.ts
http://helper.com/DummyAd-seg4.ts	adserver.com/ad1-seg4.ts
http://helper.com/DummyAd-seg5.ts	adserver.com/ad1-seg5.ts
http://helper.com/DummyAd-seg6.ts	placeholder_seg.ts

TABLE 12

http://helper.com/DummyAd-seg1.ts	placeholder_seg.ts
http://helper.com/DummyAd-seg2.ts	adserver.com/ad1-seg1.ts
http://helper.com/DummyAd-seg3.ts	adserver.com/ad1-seg2.ts
http://helper.com/DummyAd-seg4.ts	adserver.com/ad1-seg3.ts
http://helper.com/DummyAd-seg5.ts	adserver.com/ad1-seg4.ts
http://helper.com/DummyAd-seg6.ts	adserver.com/ad1-seg5.ts

[0069] At some point after receiving the response from stitcher **106** and mapping dummy URLs to ad segment URLs, helper **104** will respond to request messages corresponding to the dummy URLs, and helper may respond according to the mappings. For example, assuming the mapping shown in Table 9, when helper **104** processes a request message m**222** for DummyAd-seg**1**.*ts* (e.g., an HTTP Get message wherein the request target of the GET message contains or consists of the string "DummyAd-seg**1**.*ts*"), helper **104** responds to such request by providing to player **102** a response message m**224** that contains ad**1**-seg**1**.*ts* or an HTTP redirect containing the corresponding ad segment URL. If response message m**224** includes the redirect, this causes player **102** to transmit to media server **108** a request message m**226** identifying the requested ad segment and then media server **108** responds by transmitting a response message m**228** that contains the requested ad segment (or it could possibly contain another redirect).

[0070] Likewise, assuming the mapping shown in Table 9, when helper **104** receives request messages for DummyAd-seg**2**.*ts* and DummyAd-seg**3**.*ts*, helper will respond to each such request, respectively, by either providing to the player ad**1**-seg**2**.*ts* and ad**1**-seg**3**.*ts*, respectively, or providing to the player an HTTP redirect containing the corresponding ad segment URL. That is, when helper receives a request for DummyAd-segY.ts, helper either returns ad**1**-segY.ts or an HTTP redirect message containing the ad segment URL for ad**1**-segY.ts (i.e., adserver.com/ad**1**-segY.ts), where Y=1, 2, 3. In contrast, (still assuming the mapping shown in Table 9) when helper **104** receives a request for DummyAd-segX.ts (X=4, 5, or 6), then helper **104** provides to the player a short placeholder segment (placeholder_seg.ts) (e.g., a 500 ms or 1 second segment) (this can happen even if DummyAd-segX.ts is not mapped to a placeholder segment because, in such a case, helper may have logic that instructs it to respond to a request for a segment with a placeholder (or other selected segment) when the requested segment is not mapped to an ad segment or placeholder segment).

[0071] As noted above, in embodiments in which helper **104** receives from stitcher **106** ad segment URLs identifying ad segments selected by stitcher **106** (or the ad segments themselves) before helper **104** performs the ad break placeholder insertion process (see above description of message m**210**), helper **104** may insert into the manifest the ad segment URLs corresponding to the ad segments selected by stitcher **106**. In such an embodiment, there is a possibility that, in performing the mapping process described above, helper **104** decides to cause the player to request ad segments identified in message m**220** rather than the ad segments identified in message m**210**. For example, assume that message m**210** identified the following three add segments (ad**1**-seg**1**.*ts*, ad**1**-seg**2**.*ts*, and ad**1**-seg**3**.*ts*) and assume that message m**220** identified at the following five add segments (ad**2**-seg**1**.*ts*, ad**2**-seg**2**.*ts*, ad**2**-seg**3**.*ts*, ad**2**-seg**4**.*ts*, ad**2**-seg**5**.*ts*) and that helper **104** has decided to replace ad**1** with

ad**2** (e.g., ad**2** may be more valuable than ad**1**), then in this scenario the mapping process may produce the following mapping:

TABLE 13

Ad segment URLs inserted into Manifest	Corresponding URLs
http://helper.com/ad1-seg1.ts	adserver.com/ad2-seg1.ts
http://helper.com/ad1-seg2.ts	adserver.com/ad2-seg2.ts
http://helper.com/ad1-seg3.ts	adserver.com/ad2-seg3.ts
http://helper.com/DummyAd-seg1.ts	adserver.com/ad2-seg4.ts
http://helper.com/DummyAd-seg2.ts	adserver.com/ad2-seg5.ts
http://helper.com/DummyAd-seg3.ts	placeholder_seg.ts

[0072] As noted above, there are embodiments in which helper **104** inserts additional dummy URLs to the manifest (see Table 6). In these embodiments, when a URL for a dummy ad segment is not mapped to an ad segment URL or actual ad segment, but rather is mapped to the short placeholder segment (see, e.g., DummyAd-seg**6**.*ts* in Table 9), then helper will map the corresponding additional dummy URL (e.g., the URL for AdditionalDummyAd-seg**6**.*ts*) to a long placeholder segment (e.g., a 3 second or 6 second placeholder segment) such that when helper receives a request for this corresponding additional dummy segment helper returns the long placeholder segment rather than the short placeholder segment it would have otherwise returned. For specific examples with reference to Table 9, when helper **104** receives a request for AdditionalDummyAd-segY.ts (Y=1, 2, 3), helper return the short (e.g., 500 ms) placeholder segment because DummyAd-segY.ts is mapped to ad**1**-segY.ts, whereas when helper receives a request for AdditionalDummyAd-segX.ts (X=4,5), helper return a first long (e.g., a 6 second) placeholder segment because DummyAd-segX.ts is mapped to a placeholder segment and the #EXTINF value for these dummy segments is set to 6 seconds, and when helper receives a request for AdditionalDummyAd-seg**6**.*ts*, helper return a second long (e.g., a 3 second) placeholder segment because DummyAd-seg**6**.*ts* is mapped to a placeholder segment and the #EXTINF value for this dummy segments is set to 3 seconds. In this manner, the total duration of the media, as defined in the manifest to the player, is maintained.

[0073] In other words, when player **102** requests, for example, DummyAd-seg**4**.*ts*, which we will assume is mapped to a short placeholder segment, instead of returning a 6 second segment as specified in the playlist by the #EXTINF tag for that segment, the helper instead returns a short (e.g. 500 ms) placeholder segment. Accordingly, when player requests AdditionalDummyAd-seg**4**.*ts*, which corresponds to DummyAd-seg**4**.*ts*, instead of returning a 1 second segment as specified by the #EXTINF tag for that segment, helper **104** returns a 6 second black segment. In this manner, the duration of the media remains constant and helper **104** is simply rearranging (conceptually) where the "black video" is being played within the timeline. In an extreme example, helper **104** could do this for the entire ad break, reducing what would be a 60 second break to 6 seconds of black (or a chosen placeholder video).

[0074] In another embodiment, rather than mapping dummy URLs that were inserted into the manifest to a placeholder segment when there is no available ad segment, the dummy URLs are mapped to information (e.g., a pre-defined "not available" status code, such as an HTTP 404 status code or other status code, or other information) that indicates to the player that the requested segment is being skipped. That is, in this embodiment, when helper receives a request for, for example, DummyAd-seg**1**.*ts* and this dummy URL has not been mapped to any ad segment, helper responds to the request by sending to the player a response message with information (e.g., a pre-defined status code, such as an HTTP 404 or other information) that indicates to the player that the requested segment is being skipped. Such a response message may contain a status code with no response body where the status code indicates that no segment is being returned in response to the request (i.e., the segment is being skipped). As another example, the response message includes a status code (e.g., 200 OK) and a content-length parameter set to 0 to indicate that no segment is being returned in response to the request. As yet another example, the response message comprises a header and the information that indicates to the player that the requested segment is being skipped is contained in the header. As yet another example, the response message comprises a non-zero length body and the information that indicates to the player that the requested segment is being skipped is contained in the body of the message.

[0075] FIG. **3** is a flowchart illustrating a process **300** that includes: receiving (step s**302**), from a media player, a first request for a manifest for playing a media; transmitting (step s**304**) to a manifest server a second request for the manifest; receiving (step s**306**) the manifest from the manifest server, wherein the manifest indicates an insertion point for a first ad break and the manifest comprises first ad break duration information indicating a duration of the first ad break; generating (step s**308**) a modified manifest, wherein generating the modified manifest comprises inserting into the manifest at the insertion point first segment locator information for causing the player to request a number of segments, wherein the first segment locator information comprises a first uniform resource locator (URL) (absolute or relative) or URL template information for generating the first URL, and wherein the first URL includes a segment identifier (e.g., the first URL may consist of or comprise the segment identifier), and further wherein the modified manifest comprises segment duration information specifying a segment duration associated with the segment identifier; providing (step s**310**) the modified manifest to the player; receiving (step s**312**) from the player a segment request comprising the segment identifier; and in response to receiving the segment request comprising the segment identifier, i) providing (step s**314**) to the player a segment or a redirect message for causing the player to request the segment, wherein the segment is either a segment identified by the segment identifier or another segment (e.g., a segment to which the segment identifier is mapped), and the duration of the segment is less than the segment duration associated with the segment identifier or ii) providing (step s**316**) to the player a response message comprising information indicating that no segment is being returned.

[0076] In some embodiments, the first segment locator information comprises a list of N**1** URLs comprising the first URL (but there is no requirement that the first URL be positioned first in the list; the first URL may be positioned anywhere in the list), where N1 >1, or the first segment locator information comprises URL template information that can be used by the player to generate the list of N**1** URLs.

[0077] In some embodiments, the duration of the first ad break is associated with a set of two or more potential ad sets, wherein each potential ad set requires a specific number of segments, and N1=max (s**1**, s2, . . . , sM), where si for i=1 to M is the specific number of segments required by the ith potential ad set.

[0078] In some embodiments, the first segment locator information futher comprises, for each URL in the list of N**1** URLs, metadata for the URL (e.g., #EXT-X-INF tag, byte range information).

[0079] In some embodiments, generating the modified manifest comprises inserting into the manifest first segment grouping information for forming two or more groups of the N1 number of URLs.

[0080] In some embodiments, the N**1** URLs further comprises a second URL, and inserting into the manifest the first segment grouping information comprises inserting into the manifest a discontinuity tag (e.g., an HLS EXT-X-DISCONTINUITY tag) such that in the modified manifest the discontinuity tag comes after the first URL but comes before the second URL.

[0081] In some embodiments, the first segment grouping information comprises: a first period element comprising: i) a first subset of the N**1** URLs or ii) URL template information for use in generating the first subset of URLs; and a second period element comprising: i) a second subset of the N**1** URLs or ii) URL template information for use in generating the second subset of URLs.

[0082] In some embodiments, process **300** further comprises: after providing the modified manifest to the player, submitting to an stitcher an ad request for triggering the stitcher to select a set of ads to fill the first ad break; receiving from the stitcher a response responsive to the ad request the response comprising a set of M segment URLs or a set of M ad segments, where M≤N**1**, wherein the set of M segment URLs comprises a first ad segment URL or the set of M ad segments comprises a first ad segment; and mapping the first URL included in the list of N**1** URLs with the first ad segment URL or the first ad segment.

[0083] In some embodiments, at the time the first segment locator information is inserted into the manifest, the first URL is a dummy URL (e.g. DummyAd-seg**1**.*ts*) that is associated with no segment or a placeholder segment (e.g., a segment which consists essentially of black frames).

[0084] In some embodiments, process **300** further comprises: prior to generating the modified manifest, using information indicating the duration of the first ad break to obtain first ad set information, wherein the first ad set information comprises the first segment locator information or the first segment locator information is derived using the first ad set information.

[0085] In some embodiments, the duration of the segment is less than D-**1**, wherein D is the specified segment duration in units of seconds.

[0086] In some embodiments, the duration of the segment is less than 1 second.

[0087] In some embodiments, the segment is a placeholder segment.

[0088] In some embodiments, the manifest indicates an insertion point for a second ad break and the manifest comprises second ad break duration information indicating a duration of the second ad break, and generating the modified manifest further comprises inserting into the manifest at the insertion point for the second ad break second segment locator information for causing the player to request a number of segments, wherein the second segment locator information comprises a list of N**2** URLs, where N2 >1, or the second segment locator information comprises second URL template information that can be used by the player to generate the list of N**2** URLs.

[0089] Additional Ad Break Use Case

[0090] FIG. **4** is a message flow diagram illustrating a process according to another use case. The process may begin with player **102** sending request message m**202** (e.g. HTTP GET message) identifying a manifest (e.g., manifest.mpd). Request message m**202** is received at helper **104**.

[0091] In response to receiving request message m**202**, helper **104** sends to CSS **112** request message m**204** identifying the manifest. In response to receiving message m**204**, CSS **112** sends to helper **104** a response message m**206** comprising the requested manifest. Helper **104** may obtain information about ad breaks to be inserted into the manifest. For instance, the information about the ad breaks may come from an EventStream element included in the received manifest or helper **104** may make a Video Multiple Ad Playlist (VMAP) request to get the information as is known in the art (this is also true for the HLS).

[0092] For each ad break to be inserted into the manifest, helper **104** performs an ad break placeholder insertion process that, in at least one embodiment, as explained above, comprises: i) determining the duration of the ad break; ii) determining ad set information based on (e.g., based solely on) the duration of the ad break; and iii) inserting into the manifest at the "location" of the ad break the determined ad set information (or information derived therefrom), which information includes segment locator information (e.g., a set of URLs or a template for generating URLs). After inserting the information including the segment locator information into the manifest, thereby producing a modified manifest, helper **104** provides to player **102** a response message m**212**, which contains the modified manifest (e.g., Mod-manifest.mpd).

[0093] Inserting Ad Set Information into the Manifest

[0094] As noted above, in one embodiment helper **104** inserts into the manifest at the location of the ad break the determined ad set information (or, for example, segment locator information derived therefrom). A specific example using an MPEG-DASH manifest is illustrated below in Table 14. For this example, the manifest shown in Table 2A is the manifest helper **104** received in response to its manifest request m**204**. For this example, we will also assume that the ad set information is the same ad set information corresponding to the 30 second ad break shown in Table 4. Given this ad set information, helper may produce a modified manifest as shown in Table 14.

TABLE 14

```
<MPD ...>
<Period id="1"...>
  <AdaptationSet mimeType="video/mp4" ...>
```

TABLE 14-continued

```
        <Representation id="rep1" bandwidth="10392000" ...>
          <SegmentList ...>
            <SegmentURL media="http://cdn.com/vid-rep1-seg1.mp4"/>
            <SegmentURL media="http://cdn.com/vid-rep1-seg2.mp4"/>
            <SegmentURL media="http://cdn.com/vid-rep1-seg3.mp4"/>
            <SegmentURL media="http://cdn.com/vid-rep1-seg4.mp4"/>
            <SegmentURL media="http://helper.com/abcd1"/>
            <SegmentURL media="http://helper.com/abcd2"/>
            <SegmentURL media="http://helper.com/abcd3"/>
            <Initialization sourceURL="http://cdn.com/init1.mp4"/>
          </SegmentList>
        </Representation>
      </AdaptationSet>
    </Period>
    <Period id="ad_break_1" ...> /** This period corresponds to an ad
    break **/
      <AdaptationSet mimeType="video/mp4" ...>
        <Representation id="rep1" bandwidth="10392000" ...>
          <SegmentList ...>
            <SegmentURL media="http://helper.com/DummyAd-rep1-seg1"/>
            <SegmentURL media="http://helper.com/DummyAd-rep1-seg2"/>
            <SegmentURL media="http://helper.com/DummyAd-rep1-seg3"/>
            <SegmentURL media="http://helper.com/DummyAd-rep1-seg4"/>
            <SegmentURL media="http://helper.com/DummyAd-rep1-seg5"/>
            <SegmentURL media="http://helper.com/DummyAd-rep1-seg6"/>
            <Initialization sourceURL="http://helper.com/init.mp4"/>
          </SegmentList>
        </Representation>
      </AdaptationSet>
    </Period>
    <Period id="2"...>
      <AdaptationSet mimeType="video/mp4" ...>
        <Representationid="rep1" bandwidth="10392000" ...>
          <SegmentList ...>
            <SegmentURL media="http://cdn.com/vid-rep1-seg8.mp4"/>
            <SegmentURL media="http://cdn.com/vid-rep1-seg9.mp4"/>
            <SegmentURL media="http://cdn.com/vid-rep1-seg10.mp4"/>
            <SegmentURL media="http://cdn.com/vid-rep1-seg11.mp4"/>
            <Initialization sourceURL="http://cdn.com/init1.mp4"/>
          </SegmentList>
        </Representation>
      </AdaptationSet>
    </Period>
    </mpd>
```

[0095] In the example shown, the dummy URLs are of the form: http://FQDN/stringN, where "FQDN" is the helper's fully qualified domain name (e.g., helper.com), "string" is a constant (e.g., "DummyAd-rep**1**-seg") and "N" is a variable where N=1, 2, . . . , or **6**, so that each dummy URL is unique. However, in another embodiment, each dummy URL may be of the form: http://FQDN/string so that each dummy URL is the same as each other dummy URL.

[0096] Additionally, as shown in the table above, in some embodiments, helper **104** may modify one or more of the segment URLs included in the original manifest to cause the player to transmit to the helper one or more of the presentation segments requests (i.e., requests of obtaining segments of the presentation). For instance, in the example shown, the helper replaced the 5th, 6th, and 7th segment URL in the first period element with the following replacement URLs "http://helper.com/abcd1," "http://helper.com/abcd2," and "http://helper.com/abcd3," respectively. As should be noted, each replacement URL within the same segment list element includes a unique string (e.g., "abcd**1**") within the segment list and each points to helper **104** (e.g., each replacement URL includes the helper's fully qualified domain name (FQDN)). In this way, the 5th, 6th, and 7th presentation segment requests in addition to the ad segment requests will be sent by player **102** to helper **104**. More generically, helper may modify the manifest so that at least some (i.e., all or less than all) presentation segments requests are sent to the helper.

[0097] If any presentation segment requests are sent to the helper, some of the replacement URLs may include a "trigger" indicator. For instance (using an MPD as an example), the helper may replace the following URLs in the original manifest:

[0098] <SegmentURL media= "http://cdn.com/vid-rep1-seg1.mp4"/>

[0099] <SegmentURL media= "http://cdn. com/vid-rep**1**-seg**2**. mp4"/>

[0100] <SegmentURL media= "http://cdn. com/vid-rep**1**-seg**3**.*mp*4"/>

[0101] <SegmentURL media= "http://cdn.com/vid-rep1-seg4.mp4"/>

[0102] <SegmentURL media= "http://cdn.com/vid-rep1-seg5.mp4"/>

[0103] <SegmentURL media= "http://cdn.com/vid-rep1-seg6.mp4"/>

[0104] <SegmentURL media= "http://cdn.com/vid-rep1-seg7.mp4"/>

with the following replacement URLs:

[0105] <SegmentURL media= "http://helper.com/abcd1"/>

[0106] <SegmentURL media= "http://helper.com/abcd2"/>

[0107] <SegmentURL media= "http://helper. com/abcd3"/>

[0108] <SegmentURL media= "http://helper.com/abcd4"/>

[0109] <SegmentURL media= "http://helper.com/abcd5?trigger=1"/>

[0110] <SegmentURL media= "http://helper.com/abcd6?trigger=1"/>

[0111] <SegmentURL media= "http://helper.com/abcd7?trigger=1"/>

[0112] If the helper replaces a URL (e.g., http://cdn.com/vid-rep1-seg4.mp4) in the original manifest with a replacement URL that points to the helper (e.g., http://helper.com/abcd1), then the helper will associate the replacement URL with the segment identified by the replaced segment URL. For example, the helper may create a data structure to map each replacement URL with the URL that the replacement URL replaced (or other information that identifies the segment that was identified by the URL that was replaced by the replacement URL). Accordingly, the information to which the replacement URL is mapped may comprise all or a portion of the replaced URL.

[0113] Another example of a manifest that may be received in message m**206** is illustrated below in Table 15.

TABLE 15

```
<MPD mediaPresentationDuration="3740.08" ...>
 <Period id="1" duration="3740.08" ...>
```

TABLE 15-continued

```
  <EventStream>
   <Event
    presentationTime="400"
    duration="0"
    id="10">
   </Event>
  </EventStream>
  <BaseURL>http://cdn.com/dash/</BaseURL>
   <SegmentTemplate timescale="1"
    initialization="init-$RepresentationID$.mp4"
    media="media-$RepresentationID$-$Time$.mp4">
    <SegmentTimeline>
     <S t="0" d="4" r="934" />
     <S d="0.08" />
    </SegmentTimeline>
   </SegmentTemplate>
   <Representation id="rep1"> </Representation>
  </Period>
</MPD>
```

[0114] The manifest shown in Table 15 is a single period manifest for a presentation that has a presentation time of 3740.08 seconds. The manifest includes information that indicates that an ad break should be inserted after 400 seconds of presentation time (in this example, the information is in the form of an Event element within an EventStream element. The manifest also includes a SegmentTemplate element for generating a list of segmenet URLs, as opposed to including the segment URLs in the manifest itself as illustrated in Table 14 (this feature can be advantageous because it makes the manifest smaller).

[0115] Given the manifest shown in Table 15, helper may produce a modified manifest as shown in Table 16.

TABLE 16

```
<MPD mediaPresentationDuration="3770.08" ...>
 <Period id="1" duration="400">
  <BaseURL>http://cdn.com/dash/</BaseURL>
   <SegmentTemplate timescale="1"
    initialization="init-$RepresentationID$.mp4"
    media="media-$RepresentationID$-$Time$.mp4">
    <SegmentTimeline>
     <S t="0" d="4" r="99" />
    </SegmentTimeline>
   </SegmentTemplate>
   <Representation id="rep1" ...> </Representation>
 </Period>
<Period id="2" duration="30">
  <BaseURL>http://helper.com/</BaseURL>
   <SegmentTemplate timescale="1"
    startnumber=1
    initialization="init.mp4"
    media="DummyAd-$RepresentationID$-seg$Number$.mp4">
    <SegmentTimeline>
     <S t="0" d="6" r="4" />
    </SegmentTimeline>
   </SegmentTemplate>
   <Representation id="rep1" ...> </Representation>
 </Period>
 <Period id="3" duration="3340.08">
 <BaseURL>http://cdn.com/dash/</BaseURL>
   <SegmentTemplate timescale="1"
    initialization="init-$RepresentationID$.mp4"
    media="media-$RepresentationID$-$Time$.mp4">
    <SegmentTimeline>
     <S t="400" d="4" r="834" />
     <S d="0.08" />
    </SegmentTimeline>
   </SegmentTemplate>
   <Representation id="rep1" presentationTimeOffset="400" ...> </Representation>
 </Period>
</MPD>
```

[0116] As shown in Table 16, helper converted the single-period manifest into a multi-period manifest. More specifically, the helper "split" the single Period of the original manifest into two periods and inserted a period between the two. That is, the modified manifest includes a first Period element for the first 400 seconds of the presentation, a second Period immediately following the first period for the ad break (in this example, a 30 second ad break was chosen), and a third period immediately following the second period for the remaining 3340.08 seconds of the presentation. Because a 30 second ad break was inserted the mediaPresentationDuration increased from 3740.08 to 3770.08 seconds as illustrated in the table above. As noted in Table 16, the third Period element includes presentationTimeOffset attribute set to a value of 400 seconds to indicate to the player the timecodes in the returned segments need to be shifted back by 400 seconds to match the period's zero presentation timestamp. More generically, for example, if the original manifest indicated that N ad breaks should be inserted, then the helper may split the single period into N+1 periods and insert an ad break period between each period. As another example, helper may split the single period into N periods and insert an ad break period either after or before each one of the N periods.

[0117] As another example, given the manifest shown in Table 15, helper may produce a modified manifest as shown in Table 17.

TABLE 17

```
<MPD mediaPresentationDuration="3770.08"...>
 <Period id="1" duration="368">
  <BaseURL>http://cdn.com/dash/</BaseURL>
   <SegmentTemplate timescale="1"
    initialization="init-$RepresentationID$.mp4"
    media="media-$RepresentationID$-$Time$.mp4">
    <SegmentTimeline>
     <S t="0" d="4" r="91" />
    </SegmentTimeline>
   </SegmentTemplate>
   <Representation id="rep1" ...> </Representation>
 </Period>
 <Period id="2" duration="32">
  <BaseURL>http://helper.com/</BaseURL>
   <SegmentTemplate timescale="1"
    startNumber=1
    initialization="init-$RepresentationID$.mp4"
    media="abcd$Number$">
    <SegmentTimeline>
     <S t="368" d="4" r="7" />
    </SegmentTimeline>
   </SegmentTemplate>
   <Representation id="rep1" presentationTimeOffset="368" ...> </Representation>
 </Period>
<Period id="3" duration="30">
  <BaseURL>http://helper.com/dash/</BaseURL>
   <SegmentTemplate timescale="1"
    startnumber=1
    initialization="init.mp4"
    media="DummyAd-$RepresentationID$-seg$Number$.mp4">
    <SegmentTimeline>
     <S t="0" d="6" r="4" />
    </SegmentTimeline>
   </SegmentTemplate>
   <Representation id="rep1" ...> </Representation>
 </Period>
 <Period id="4" duration="3340.08">
  <BaseURL>http://cdn.com/dash/</BaseURL>
   <SegmentTemplate timescale="1"
    initialization="init-$RepresentationID$.mp4"
    media="media-$RepresentationID$-$Time$.mp4">
    <SegmentTimeline>
     <S t="400" d="4" r="834" />
     <S d="0.08" />
    </SegmentTimeline>
   </SegmentTemplate>
   <Representation id="rep1" presentationTimeOffset="400" ...> </Representation>
 </Period>
</MPD>
```

[0118] The modified manifest shown in Table 17 is nearly identical to the one shown in Table 16 with the most significant difference being that the manifest shown in Table 17 is configured to cause the player transmit to the helper eight (8) requests for presentation segments prior to transmitting a request for an inserted segment (i.e., an ad segment in this example). More specifically, the first Period element shown in Table 16, which has a duration of 400 seconds, is split into two Period elements: a first Period element with a duration 368 seconds and a second Period element with a duration of 32 seconds. The **92** URLs that are generated based on the segment template included in the first period shown in Table 17 are identical to the first **92** URLs that are generated based on the segment template included in the first period shown in Table 16. But the **8** URLs that are generated based on the segment template included in the second period shown in Table 17 are different than the remaining 8 URLs that are generated based on the segment template included in the first period shown in Table 16. More specifically, each these 8 URLs generated based on the segment template included in the second period shown in Table 17 cause the player to transmit to the helper a request (e.g., an HTTP Get message wherein the request target of the GET message contains or consists of the string "abcdN", where N=1, 2, . . . , or **8**)

[0119] Referring back to FIG. **4**, after player **102** receives the modified manifest (e.g., the manifest shown in Table 14), player **102** begins sending segment requests according to the manifest (e.g., requests for segments of the presentation, which is a video in this example). For example, as shown in FIG. **4**, player **102** may initially send to CSS **112** one or more segment requests m**414** (and CSS **112** responds by sending the requested segments to player **102**) followed by player **102** sending to helper **104** one or more segment requests (e.g., request m**416**). For instance, using the manifest shown in Table 14 as an example, player **102** will i) send to CSS **112** (i.e., "cdn.com" in this example) four segment requests (i.e., a request for vid-rep**1**-seg**1**.*mp*4, a request for vid-rep**1**-seg**2**.*mp*4, a request for vid-rep**1**-seg**3**.*mp*4, and a request for vid-rep**1**-seg**4**.*mp*4) and ii) after sending the segment requests to CSS **112**, send a request m**416** to helper **104** (i.e., a request containing the string "abcd**1**"). Likewise, using the manifest shown in Table 17 as an example, player **102** will send multiple segment requests to CSS **112** before sending to helper request m**416**.

[0120] In one embodiment, in response to request m**416**, the helper will: 1) send to the player a response m**417** responsive to the request and **2**) initiate the transmission of a request to stitcher **106** (e.g., determine whether a request to stitcher **106** is needed, and, if so, send to stitcher request message m**218** indicating that helper is requesting one or more ads (i.e. a set of ads) to fill an ad break). For instance, in one embodiment, the helper determines whether a request to stitcher **106** is needed by checking whether a flag is set to TRUE (e.g., equal to 1) and if the flag is set to TRUE, then it is determined that a request to stitcher **106** is needed. In another embodiment, the helper determines whether a request to stitcher **106** is needed by checking both whether the flag is set to TRUE and whether the request includes a trigger string (e.g., "trigger=1"), and if the flag is set to TRUE and request includes the trigger string, then it is determined that a request to stitcher **106** is needed. After sending the request, the flag is set to FALSE. The flag may be set back to TRUE after the ad break has finished. Response message m**417** either contains the requested segment or a redirect message that causes player **102** to send a request message to a server (e.g. a server within CSS) for the segment.

[0121] In this example, stitcher **106** responds to request m**218** by selecting one or more ads (or other pieces of media content) and transmitting to helper **104** a response message m**220** that contains a manifest containing segment locator information (e.g., segment URLs and/or template information enabling helper to produce segment URLs) for retrieving the segments of the one or more ads selected by stitcher **106**. For instance, the manifest returned by stitcher **106** may include a first period element containing segment URLs (or corresponding template information) for the first ad and a second period element containing segment URLs (or corresponding template information) for the second ad.

[0122] After receiving response message m**220**, helper **104** uses the segment locator information in the response message to retrieve all of the identified segments (e.g., by sending segment requests m**421**). Next, the helper creates an output file (step s**490**) comprising encoded media data from the retrieved segments. Next, the helper segments the output file to produce a set of media content segments (step s491), wherein each media content segment contains a portion of the encoded media data contained in the output file (an initialization segment may also be produced in addition to the media content segments).

[0123] In one embodiment, step s**490** (creating the output file) comprises the helper: i) creating a first temporary file containing a first set of segments (e.g., all of the segments of ad**1**), ii) creating a second temporary file containing a second set of segments (e.g., all of the segments for ad**2**), and iii) concatenating the first and second temporary files, thereby producing the output file. In one embodiment, each media content segment in the second temporary file contains one or more timestamps, and the step of concatenating the first and second temporary files comprises modifying the timestamp(s).

[0124] Assuming the segments for the first ad consist of the following segments: s**11**, s12, . . . , s**1**N and the segments for the second ad consist of the following segments: s**21**, s22, . . . , s**2**M (s**11** and s**21** may be initialization segments), then, in one embodiment, the first temporary file is created running the cat commands shown in Table 18 and the second temporary file is created by running the cat commands shown in Table 19.

TABLE 18

```
cat s11.mp4 > tempFile1.mp4
cat s12.mp4 >> tempFile1.mp4
cat s13.mp4 >> tempFile1.mp4
...
cat s1N.mp4 >> tempFile1.mp4
```

TABLE 19

```
cat s21.mp4 > tempFile2.mp4
cat s22.mp4 >> tempFile2.mp4
cat s23.mp4 >> tempFile2.mp4
...
cat s2M.mp4 >> tempFile2.mp4
```

[0125] In one embodiment, helper **104** uses the FFmpeg tool to create the output file using the temporary files as

input. For example, to concatenate tempFile**1**.*mp*4 with tempFile**2**.*mp*4 to produce the output file (e.g., out.mp4), helper may run the following command: ffmpeg-f concat-safe **0**-*i* list.txt-c copy out.mp4, where list.txt is a two line text file wherein the first line is "tempFile**1**.*mp*4" and the second line is "tempFile**2**.*mp*4." Executing this ffmpeg command demuxes the files identified in list.txt one after the other, as if all their packets had been muxed together. The timestamps in the files are adjusted so that the first file starts at **0** and each next file starts where the previous one finishes. The duration of each file is used to adjust the timestamps of the next file (i.e., the duration of tempFile**1** is used to adjust the timestamps of tempFile**2**).

[0126] After creating the output file, the helper may run the following ffmpeg command to create the segments (i.e., step s**491**):

[0127] ffmpeg-i out.mp4-codec copy-use_timeline **0**-use_template **0**-seg_duration 6-f dash out.mpd

[0128] This command generates a single-period MPD that contains a period element matching the period element corresponding to the ad break (see e.g., the period element with id= "ad_break_**1**" in Table 14) as well as the individual segments to be returned for the ads.

[0129] Each one of the media content segments generated by helper **104** in step s**491** is mapped to one of the dummy URLs (step s**492**). Tables 20 and 21 illustrate possible mappings of dummy URLs to the media content segments (e.g., Table 20 illustrate possible mappings when the number of media content segments is three and Table 21 illustrate possible mappings when the number of media content segments is five).

TABLE 20

helper.com/DummyAd-seg1.mp4	seg1.mp4
helper.com/DummyAd-seg2.mp4	seg2.mp4
helper.com/DummyAd-seg3.mp4	seg3.mp4
helper.com/DummyAd-seg4.mp4	placeholder_seg.mp4 (or a "not available" status code)
helper.com/DummyAd-seg5.mp4	placeholder_seg.mp4 (or a "not available" status code)
helper.com/DummyAd-seg6.mp4	placeholder_seg.mp4 (or a "not available" status code)

TABLE 21

helper.com/DummyAd-seg1.mp4	seg1.mp4
helper.com/DummyAd-seg2.mp4	seg2.mp4
helper.com/DummyAd-seg3.mp4	seg3.mp4
helper.com/DummyAd-seg4.mp4	seg4.mp4
helper.com/DummyAd-seg5.mp4	seg5.mp4
helper.com/DummyAd-seg6.mp4	placeholder_seg.mp4 (or a "not available" status code)

[0130] Referring back to FIG. **4**, player **102** continues sending segment requests according to the manifest. For example, as shown in FIG. **4**, player **102** sends to helper **104** one or more segment requests, such as, for example, request m**422**, which is a request containing the string "abcd**3**". In one embodiment, in response to request m**422**, the helper sends to the player a response responsive to the request, such as response message m**424**, which either contains a segment (e.g., vid-rep**1**-seg**3**.*mp*4 or vid-rep**1**-seg**7**.*mp*4) or a redirect message that causes player **102** to send a request message to a server (e.g. a server within CSS) for the segment . . .

[0131] At some point in the process, player **102** sends requests corresponding to the dummy URLs, and helper may respond according to the mappings.

[0132] For example, assuming the mapping shown in Table 20, when helper **104** processes a request message m**426** for DummyAd-seg**1**.*mp*4 (e.g., an HTTP Get message wherein the request target of the GET message contains or consists of the string "DummyAd-seg**1**.*mp*4"), helper **104** responds to such request by providing to player **102** a response message m**428** that contains seg**1**.*mp*4 or an HTTP redirect containing a URL pointing to seg**1**.*mp*4.

[0133] Likewise, assuming the mapping shown in Table 20, when helper **104** receives request messages for DummyAd-seg**2**.*mp*4 and DummyAd-seg**3**.*mp*4, helper will respond to each such request, respectively, by either providing to the player seg**2**.*mp*4 and seg**3**.*mp*4, respectively, or providing to the player an HTTP redirect containing the corresponding URL. That is, when helper receives a request for DummyAd-segY.mp4, helper either returns segY.mp4 (or an HTTP redirect), where Y=1, 2, 3. In contrast, (still assuming the mapping shown in Table 20) when helper **104** receives a request for DummyAd-segX.mp4 (X=4, 5, or 6), then helper **104** provides to the player a short placeholder segment (e.g., a 500 ms or 1 second segment) or a "not available" (e.g., a "Not Found") response status code or some other default response.

[0134] While the embodiments have been explained for media segments, the embodiments apply equally to audio segments. As is known in the art, each period element of an MPD may include a video adaptation set (e.g., adaptation set with mimeType= "video/mp4") and an audio adaptation set (e.g., adaptation set with mimeType= "audio/mp4"). Hence, the embodiments may function with content that includes only video, only audio, or both audio and video.

[0135] In an embodiment, the manifest for the ad break may include: i) a first period element comprising first media segment locator information for retrieving media content segments for the period and first audio segment locator information for retrieving corresponding audio content segments for the period and ii) a second period element comprising second media segment locator information for retrieving media content segments for the period and second audio segment locator information for retrieving corresponding audio content segments for the period.

[0136] Accordingly, in such an embodiment, the helper performs a process that includes: 1) using the received first and second audio segment locator information to retrieve a first set of audio content segments and a second set of audio content segments, respectively; 2) creating an audio output file containing encoded audio data from the first set of audio content segments and encoded audio data from the second set of audio content segments; 3) segmenting the audio output file to produce a third set of audio content segments, wherein each audio content segment in the third set of segments contains a portion of the encoded audio data contained in the output file; and 4) for each audio content segment included in the third set of audio content segments, associating one of the dummy URLs in the audio adaptation set element with the audio content segment. Thus, when the helper receives a request corresponding to one of the dummy URLs in the audio adaptation set element, the helper may respond by transmitting to the player the audio content segment associated with the dummy URL.

[0137] FIG. **5** is a flowchart illustrating a process **500** that includes: transmitting a modified manifest to a player (step s**502**), the modified manifest containing a period element corresponding to a content break (e.g., an ad break); after transmitting the modified manifest, transmitting a request for a manifest for the content break (step s**504**); receiving the manifest for the content break (step s506), wherein the manifest for the content break comprises i) a first period element containing first segment locator information for use in retrieving a first set of media content segments and ii) a second period element containing second segment locator information for use in retrieving a second set of media content segments; using the received first and second segment locator information to retrieve the first set of media content segments and the second set of media content segments, respectively (step s**508**); creating an output file containing encoded media data from the first set of media content segments and encoded media data from the second set of media content segments (step s**510**); segmenting the output file to produce a third set of media content segments (step s512), wherein each media content segment in the third set of segments contains a portion of the encoded media data contained in the output file; receiving a segment request transmitted by a player (step s**514**); and after receiving the segment request, transmitting to the player a media content segment from the third set of media content segments (step s**516**).

[0138] In some embodiments, creating the output file comprises: creating a first temporary file containing the first set of segments; creating a second temporary file containing the second set of segments; and concatenating the first and second temporary files, thereby producing the output file.

[0139] In some embodiments, each segment in the second temporary file contains one or more timestamps, and concatenating the first and second temporary files comprises modifying the timestamp(s) of each segment in the second temporary file.

[0140] In some embodiments, process **500** further includes: receiving, from the player, a first request for a manifest (e.g., Dash MPD, HLS playlist) for retrieving first media content (e.g., segments of a movie, an episode, etc.); after receiving the first request for the manifest, transmitting a second request for the manifest; after transmitting the second request for the manifest, receiving the requested manifest, wherein the requested manifest includes at least a first ordered list of N segment locators, and each segment locator in the first ordered list includes a first server identifier that identifies a first server; generating the modified manifest based on the received manifest, wherein the modified manifest includes at least a second ordered list of N segment locators, each one of the first M segment locators included in the second ordered list includes the first server identifier, where M>1 and M<N, each one of the N-M segment locators in the second ordered list that follow the first M segment locators in the second ordered list includes: i) a second server identifier that identifies a second server different than the first server and ii) a unique string.

[0141] In some embodiments, process **500** further includes: receiving, from the player, a request comprising one of the unique strings; and in response to receiving the request comprising one of the unique strings, i) transmitting to the player a response message responsive to the request comprising one of the unique strings and ii) transmitting a request for second media content to be served to the player during a break in the first media content (e.g., an ad break or some other break).

[0142] In some embodiments, the request for the second media content is a request for a manifest for the second media content, wherein the manifest comprises segment locator information for retrieving segments of the second media content.

[0143] In some embodiments, initiating the transmission of the request for the second media content comprises determining whether a request for the second media content is needed.

[0144] In some embodiments, the first set of media content segments consists of a first set of video content segments, wherein each video content segment in the first set of video content segments does not contain any encoded audio data, and the second set of media content segments consists of a second set of video content segments, wherein each video content segment in the second set of video content segments does not contain any encoded audio data.

[0145] In some embodiments, the first period element further contains first audio segment locator information for use in retrieving a first set of audio content segments, the second period element further contains second audio segment locator information for use in retrieving a second set of audio content segments, and the method further comprises: using the first and second audio segment locator information to retrieve the first set of audio content segments and the second set of audio content segments, respectively; creating an audio output file containing encoded audio data from the first set of audio content segments and encoded audio data from the second set of audio content segments; segmenting the audio output file to produce a third set of audio content segments, wherein each audio content segment in the third set of audio content segments contains a portion of the encoded audio data contained in the audio output file; receiving an audio segment request transmitted by the player; and after receiving the audio segment request, transmitting to the player an audio content segment from the third set of audio content segments.

[0146] In some embodiments, the first set of media content segments consists of a first set of audio content segments, wherein each audio content segment in the first set of audio content segments does not contain any encoded video data, and the second set of media content segments consists of a second set of audio content segments, wherein each audio content segment in the second set of audio content segments does not contain any encoded video data.

[0147] FIG. **6** is a flowchart illustrating a process **600** that includes: receiving, from a player, a first request for a manifest (e.g., Dash MPD, HLS playlist) for retrieving first media content (e.g., a movie, an episode, etc.) (step s**602**); after receiving the first request for the manifest, transmitting a second request for the manifest (step s**604**); after transmitting the second request for the manifest, receiving the requested manifest, wherein the requested manifest includes at least a first ordered list of N segment locators, and each segment locator in the first ordered list includes a first server identifier that identifies a first server (step s**606**); generating a modified manifest based on the received manifest (step s608), wherein the modified manifest includes at least a second ordered list of N segment locators, each one of the first M segment locators included in the second ordered list includes the first sever identifier, where M ≥0 and M <N,

each one of the N-M segment locators in the second ordered list that follow the first M segment locators in the second ordered list includes: i) a second server identifier that identifies a second server different than the first server and ii) a unique string; transmitting the modified manifest to the player (step s**610**); receiving, from the player, a request comprising one of the unique strings (step s**612**); and in response to receiving the request comprising one of the unique strings, i) transmitting to the player a response message responsive to the request and ii) initiating the transmission of a request for second media content to be served to the player during a break in the first media content (step s**614**).

[0148] In some embodiments, the request for the second media content is a request for a manifest for the second media content, wherein the manifest comprises segment locator information for retrieving segments of the second media content.

[0149] In some embodiments, initiating the transmission of the request for the second media content comprises determining whether a request for the second media content is needed.

[0150] Prefetching Use Cases

[0151] FIG. **7** is a message flow diagram illustrating a process according to a prefetching use case. The process may begin with player **102** sending request message m**202**, which may be a standard HTTP request and which may contain request data. In some embodiments, the request data is provided via HTTP standard mechanisms, such as HTTP headers, query parameters, or HTTP POST body data. In one embodiment, request message m**202** identifies a manifest (e.g., manifest.mpd). In addition to including request data identifying the manifest, request message **202** may include other information such as, for example, "ad_params=<ad parameter values>" which specifies ad targeting parameters, "Pod_max_dur-**60**", which specifies 60 second ad pods, and "Cuepoints=60,120", which identifies two cue points, one 60 seconds after the start of the video and another 120 seconds after the start of the video. An example of attribute-value-pairs (AVPs) that may be included in message m**202** is shown in table 22 below. Request message m**202** is received at helper **104**. As described above, after receiving request message m**202**, helper **104** provides to player **102** message m**212**, which contains a manifest.

TABLE 22

manifest=server.com/episode1.mpd&
lookahead=10000&
assetId=tos&
pod_max_dur=60&
cuepoints=60,120&
cue_preference=query&
ad_params=<data>&
otherAdParams=<data>

[0152] After player **102** receives the manifest from helper **104** (e.g., the manifest shown in Table 14), player **102**, as described above, begins sending segment requests according to the manifest. For example, as shown in FIG. **4**, player **102** may initially send to CSS **112** one or more segment requests m**414** (and CSS **112** responds by sending the requested segments to player **102**) followed by player **102** sending to helper **104** one or more segment requests (e.g., request m**416**). For instance, using the manifest shown in Table 14 as an example, player **102** i) sends to CSS **112** (i.e., "cdn.com" in this example) four segment requests (i.e., a request for vid-rep**1**-seg**1**.*mp*4, a request for vid-rep**1**-seg**2**.*mp*4, a request for vid-rep**1**-seg**3**.*mp*4, and a request for vid-rep**1**-seg**4**.*mp*4) and ii) after sending the segment requests to CSS **112**, sends a segment request m**713** to helper **104** (i.e., a request containing the string "abcd**1**" in the example shown).

[0153] In one embodiment, in response to request m**713**, helper **104**:**1**) sends to the player a response m**714** responsive to the request, where the response message m**714** either contains the requested segment or a redirect message that causes player **102** to send a request message to a server for the segment, and 2) initiates the transmission of a prefetch request to a prefetch function (PF) **702**. That is, for example, helper **104** determines whether a prefetch request to PF **702** is needed, and, if so, sends to PF **702** a prefetch request message m**790** for causing PF **702** to obtain from server **108** metadata for use in playing one or more ads (or other break content) and cache the obtained metadata together with a stream identifier, which can be a single value that identifies a stream or multiple values that together identify the stream. As an example, a stream identifier can be a universally unique identifier (UUID) as defined by Request for Comment (RFC) 4122. In one embodiment, the prefetch request message m**790** includes the stream identifier and i) a conventional Video Ad Serving Template (VAST) request for requesting a single ad or an ad pod (a set of one or more ads) or ii) information that enables PF **702** to generate the conventional VAST request. If the VAST request is for an ad pod, the VAST request may include a pod minimum duration attribute-value-pair (e.g., pmnd=20) and a pod maximum duration attribute attribute-value-pair (e.g., pmxd=35). In some embodiments, in addition to including the stream identifier, request message m**790** includes one or more the AVPs that were included in message m**202**. In the embodiment shown in FIG. **7**, PF **702** is separate from stitcher **106**, but in other embodiments, PF **702** may be a component of stitcher **106** or stitcher **106** may be a component of PF **720**. That is, in some embodiments, stitcher may perform the functionality of PF **702** or PF **702** may perform the functionality of stitcher **106**.

[0154] In one embodiment, the helper determines whether a prefetch request to PF **702** is needed by checking whether a prefetch flag is set to TRUE (e.g., equal to 1) and if the prefetch flag is set to TRUE, then it is determined that a prefetch request to PF **702** is needed. In another embodiment, the helper determines whether a prefetch request to PF **702** is needed by checking both whether the prefetch flag is set to TRUE and whether the request m**713** includes a trigger string (e.g., "trigger=1"), and if the prefetch flag is set to TRUE and request m**713** includes the trigger string, then it is determined that a prefetch request to PF **702** is needed. After sending the prefetch request to PF **702**, the flag is set to FALSE. The flag may be set back to TRUE after an ad break has finished. In one embodiment, after sending the prefetch request to PF **702**, helper **104** sets an ad-is-needed flag is set to TRUE.

[0155] In the illustrated embodiment, request message m**790** may include ad targeting properties (e.g., the "ad_params" AVP from message m**202**), which may include VAST request properties and/or other proprietary data, and a timeout duration, which may be set to, for example, 10 seconds or longer. In response to receiving message m**790**, PF **702** transmits to an ad server (e.g., ad server **108** in the

example shown) a request m**791** (e.g., a VAST request) for ad metadata. For example, request m**791** may include the ad targeting properties that were included in prefetch request message m**790** or it may include ad targeting properties that are generated by PF **702** using information from message m**790**.

[0156] In response to receiving request message m**791**, server **108** selects a single ad if the request message m**791** is for a single ad or selects an ad pod if the request message m**791** is for an ad pod. Server **108** may select the ad or ads using the ad targeting properties. After performing the selection, sever **108** transmits to PF **702** a response message m**792** containing metadata for the selected ad or selected ad pod. In one embodiment, message m**792** contains a conventional VAST response (e.g., an XML file containing, among other things, one or more media file identifiers (e.g., one or more Uniform Resource Locators each pointing to a different media file for an ad)).

[0157] In one embodiment, after receiving the response message m**792** from server **108**, PF **702** caches (i.e., stores in a data storage unit) the metadata (e.g., XML file) included in the message m**792** such that the stored metadata is associated with the stream identifier so that, at a later point in time, given the stream identifier, PF **702** can retrieve the metadata from storage.

[0158] In another embodiment, after receiving the response message m**792** from server **108**, PF **702** determines, based on the metadata, whether the ad(s) selected by ad server **108** are satisfactory (i.e., whether the ads satisfy one or more criterions). If the ads are satisfactory, then PF **702** caches the metadata such that the stored metadata is associated with the stream identifier so that, at a later point in time, given the stream identifier, PF **702** can retrieve the metadata from storage. If, however, the metadata is not satisfactory and if sufficient time remains, PF **702** re-sends message m**791** to ad server **108** to trigger ad server **108** to select a new, second set of ads and transmit to PF **702** a response message comprising new metadata for the new set of ads. In some embodiments, rather than resending only message m**791**, PF**702** resends message m**791** with additional properties (e.g., additional AVPs), which may identify the set of previously chosen ads and/or new targeting criterion which the ad server **108** may use to select more suitable ads. Assuming the new set of ads is satisfactory, then PF **702** caches the new metadata. Alternatively, if some combination of the new set of ads and the original set of ads is better than the new set of ads and the original set of ads, then PF **702** may produce metadata using the first metadata and the second metadata. For example, if the first received metadata identifies the following ads: Ad**1**, Ad**2**, and Ad**3**, and the second set (new set) of metadata identifies the following ads: Ad**4**, Ad**5**, and Ad**6**, and the following set of ads is the most optimal set: Ad**2**, Ad**5**, and Ad**6**, then PF **702** may produce a new metadata file that includes the metadata for Ad**2** from the first metadata and the metadata for ads Ad**5** and Ad**5** from the new metadata. In yet another embodiment, PF **702** may send request message m**791** multiple times, with potential variations to the supplied AVP(s), to ad server **108** so that PF **702** can select the most optimal set of ads and cache metadata for this optimal ad set.

[0159] At a later point in time, helper **104** receives from player **102** a segment request m**716** (i.e., a request containing the string "abcd**3**" in the example shown).

[0160] In one embodiment, in response to segment request m**716**, the helper: 1) sends to the player a response m**717** responsive to the request, where the response message m**717** either contains the requested segment or a redirect message that causes player **102** to send a request message to a server for the segment and **2**) initiates the transmission of a request m**718** (e.g., an HTTP GET request) to stitcher **106** (e.g., the helper determines whether a request m**718** to stitcher **106** is needed, and, if so, sends to stitcher request message m**718** indicating that helper is requesting one or more ads (e.g., an ad pod) to fill an ad break of a specified duration). In one embodiment, the request message m**718** includes the same stream identifier that was included in the prefetch request message m**790**. In some embodiments, in addition to including the stream identifier, request message m**718** includes AVPs that were included in message m**202** (e.g., the "ad_params" AVP), which may, as an example, be sent as a combination of HTTP headers, URL query parameters, or HTTP POST body data. That is, for example, stitcher **106** may include an HTTP server.

[0161] In one embodiment, the helper determines whether a request m**718** to stitcher **106** is needed by checking whether an 'ad-is-needed flag' is set to TRUE (e.g., equal to 1) and if the ad-is-needed flag is set to TRUE, then it is determined that a request to stitcher **106** is needed. In another embodiment, the helper determines whether a request to stitcher **106** is needed by checking both whether the ad-is-needed flag is set to TRUE and whether the request m**716** includes a trigger string (e.g., "trigger=1"), and if the ad-is-needed flag is set to TRUE and request includes the trigger string, then it is determined that a request to stitcher **106** is needed. After sending the request, the ad-is-needed flag is set to FALSE. The ad-is-needed flag may be set back to TRUE after the ad break has finished.

[0162] In this example, stitcher **106** responds to request m**718** by transmitting to PF **702** a request m**794** for metadata (such as a VAST response) for use in playing one or more ads (or other break content). Request m**794** includes the stream identifier.

[0163] In response to receiving request message m**794**, PF **702** uses the stream identifier to determine whether the metadata cache includes metadata (e.g., a VAST XML file) associated with the stream identifier and, if so, retrieves the metadata from the cache and sends to stitcher a response message m**798** containing the retrieved metadata. If, however, the metadata cache does not include metadata associated with the stream identifier, then PF 702 May 1) transmit a VAST request m**796** to server **108** and receive from server **108** a VAST response m**797** and include in message m**798** the content of the VAST response m**797** or **2**) send to stitcher **106** a redirect message for causing stitcher **106** to send message m**794** to ad server **108**.

[0164] After receiving response message m**798** from PF **702**, stitcher **106** transmits to helper **104** a response message m**720** that contains a manifest containing segment locator information (e.g., segment URLs and/or template information enabling helper to produce segment URLs) for retrieving the segments of the one or more ads selected by server **108** in response to the request m**791** or m**796** from PF **702**. For instance, the manifest returned by stitcher **106** may include a first period element containing segment URLs (or corresponding template information) for the first ad and a second period element containing segment URLs (or corresponding template information) for the second ad.

[0165] After receiving response message m**720**, helper **104** may perform the steps described above with respect to either FIG. **2** or FIG. **4**.

[0166] The description above illustrates an embodiment in which there is a single ad server (i.e., ad server **108**). In other embodiments, there may be any number of available ad servers. In such an embodiment, PF **702** may be configured such that, in response to receiving the prefetch request message m**790**, PF **702** selects one or more of the available ad servers and then sends message m**791** (or similar message) to each one of the selected ad servers.

[0167] The selection of the ad server(s) may be based on information included in message m**790**. For instance, message m**790** may include an AVP that identifies a specific one or more of the available ad servers and PF **702** may be configured to select the identified ad server(s).

[0168] In another embodiment, when there are multiple ad servers to choose from, PF **702** may be configured to select an ad server in response to a prefetch request based on pre-defined configuration information. For example, considering a scenario where there are two available ad servers (ADS**1** and ADS**2**), the pre-defined configuration information may specify that ADS**1** should be selected 30% of the time and ADS**2** be selected 70% of the time. For instance, given this 30/70 split between ADS**1** and ADS**2**, if PF **702** is expected to receive **100** prefetch requests per window of time (e.g., **100** per day), then, during such a window of time, PF **702** may select ADS1 30 times in a row and then cease selecting ADS**1** until PF **702** has selected ADS**2** at least 70 times. That is, for example, PF **702** may be configured to front-load requests to ADS**1** before sending requests to ADS**2**. In one embodiment, PF**702** may select ADS**1** and ADS**2** in round-robin or other similar rotation, ensuring a constant configured distribution to each ad server.

[0169] In one embodiment, if PF **702** selects more than one ad server and sends to each selected ad server the message m**791**, then PF **702** will select one of the response message m**792** to cache. That is, for example, if two or more ad servers are selected and two or more of the ad servers select an ad pod and provide to PF **702** metadata for the respective selected ad pod, PF **702** may compare the ad pods to determine which is the most valuable and then cache only the metadata for the most valuable ad pod. As another example, if two or more ad servers are selected and two or more of the ad servers select an ad pod and provide to PF **702** metadata for the respective selected ad pod, PF **702** may generate an ad pod by selecting one more ads from two or more of the selected ad pods and then cache metadata corresponding to the generated ad pod. That is, if some combination of the selected ad pods is better than any one of the selected ad pods, then PF **702** may produce metadata using the metadata received from each of the two or more ad servers.

[0170] FIG. **11** is a message flow diagram illustrating a process according to another prefetching use case. In this use case a single function **1102**, called the helper-stitcher function (HSF) **1102**, replaces helper **104** and stitcher **106**. The process may begin with player **102** sending request message m**202**, as described above. Request message m**202** is received at HSF **1102**. After receiving request message m**202**, HSF **1102** provides to player **102** message m**212**.

[0171] After player **102** receives message m**221**, player **102**, as described above, begins sending segment requests according to the manifest, such as segment request m**713**, which is received at HSF **1102**.

[0172] In one embodiment, in response to request m**713**, HSF **1102**:**1**) sends to the player response m**714** and **2**) initiates the transmission of a prefetch request to a prefetch function (PF) **702**. That is, for example, HSF **1102**, like helper **104**, determines whether a prefetch request to PF **702** is needed, and, if so, sends to PF **702** the prefetch request message m**790** for causing PF **702** to obtain from server **108** metadata for use in playing one or more ads (or other break content). At a later point in time, HSF **1102** receives from player **102** a segment request m**716** (i.e., a request containing the string "abcd**3**" in the example shown).

[0173] In one embodiment, in response to segment request m**716**, the HSF: 1) sends to the player the response m**717** and **2**) initiates the transmission of the request m**794** to PF **702** (e.g., the HSF determines whether a request m**794** to PF **702** is needed, and, if so, sends to PF **702** request message m**794**. As described above, in response to receiving request message m**794**, PF **702** uses the stream identifier to determine whether the metadata cache includes metadata (e.g., a VAST XML file) associated with the stream identifier and, if so, retrieves the metadata from the cache and sends to HSF **1102** the response message m**798** containing the retrieved metadata. If, however, the metadata cache does not include metadata associated with the stream identifier, then PF 702 May 1) transmit the VAST request m**796** to server **108** and receive from server **108** the VAST response m**797** and include in message m**798** the content of the VAST response m**797** or **2**) send to HSF **1102** a redirect message for causing HSF **1102** to send message m**794** to ad server **108**.

[0174] After receiving response message m**798** from PF **702**, HSF **1102** obtains (e.g., generates) segment locator information (e.g., segment URLs and/or template information enabling HSF to produce segment URLs) for retrieving the segments of the one or more ads selected by server **108** in response to the request m**791** or m**796** from PF **702**. After receiving response message m**798**, HSF **1102** may perform the steps described above with respect to either FIG. **2** or FIG. **4**.

[0175] FIG. **12** is a message flow diagram illustrating a process according to another prefetching use case. In this use case a single function **1202**, called the helper-prefetch function (HPF) **1202**, replaces helper **104** and PF **702**. The process may begin with player **102** sending request message m**202**, as described above. Request message m**202** is received at HPF **1202**. After receiving request message m**202**, HPF **1202** provides to player **102** message m**212**, described above.

[0176] After player **102** receives message m**212**, player **102**, as described above, begins sending segment requests, such as segment request m**713**, which is received at HPF **1202**.

[0177] In one embodiment, in response to request m**713**, HPF **1202**:**1**) sends to the player response m**714** responsive to the request and **2**) initiates the transmission of request m**791** to server **108**. That is, for example, HPF **1202**, like helper **104**, determines whether a request is needed, and, if so, sends to server **108** message m**791** for obtaining from server **108** metadata for use in playing one or more ads (or other break content).

[0178] In response to receiving request message m**791**, server **108** functions as described above with respect to FIG. **7** and transmits a response message m**792**, which is received

at HPF **1202**. Message m**792**, as noted above, contains metadata for the selected ad or selected ad pod.

[0179] After received message m**792**, HPF **1202** performs the same steps as described above with respect to PF **702**. For example, in one embodiment, after receiving the response message m**792** from server **108**, HPF **1202** caches the metadata included in the message m**792** such that the stored metadata is associated with the stream identifier so that, at a later point in time, given the stream identifier, HPF **1202** can retrieve the metadata from storage. And in another embodiment, HPF **1202**, among the other steps described above, determines, based on the metadata, whether the ad(s) selected by ad server **108** are satisfactory.

[0180] At a later point in time, HPF **1202** receives from player **102** a segment request m**716** (i.e., a request containing the string "abcd**3**" in the example shown). In response to segment request m**716**, HPF **1202** functions in the same manner as helper **104**, i.e., HPF **1202** sends to the player response m**717** and initiates the transmission of request m**718** to stitcher **106**.

[0181] In this example, stitcher **106** responds to request m**718** by transmitting to HPF **1202** a request m**794** for metadata (such as a VAST response) for use in playing one or more ads (or other break content). Request m**794** includes the stream identifier.

[0182] In response to receiving request message m**794**, HPF **1202** uses the stream identifier to determine whether the metadata cache includes metadata (e.g., a VAST XML file) associated with the stream identifier and, if so, retrieves the metadata from the cache and sends to stitcher a response message m**798** containing the retrieved metadata. If, however, the metadata cache does not include metadata associated with the stream identifier, then HPF **1202** May 1) transmit a VAST request to server **108** and receive from server **108** a VAST response and include in message m**798** the content of the VAST response or **2**) send to stitcher **106** a redirect message for causing stitcher **106** to send message m**794** to ad server **108**.

[0183] After receiving response message m**798** from HPF **1202**, stitcher **106** transmits to HPF **1202** a response message m**720** that contains a manifest containing segment locator information (e.g., segment URLs and/or template information enabling helper to produce segment URLs) for retrieving the segments of the one or more ads selected by server **108** in response to the request m**791** or m**796**. For instance, the manifest returned by stitcher **106** may include a first period element containing segment URLs (or corresponding template information) for the first ad and a second period element containing segment URLs (or corresponding template information) for the second ad. After receiving response message m**720**, HPF **1202** may perform the steps described above with respect to either FIG. **2** or FIG. **4**.

[0184] FIG. **8** is a flowchart illustrating a process **800** that includes: a prefetching function (PF) receiving (step s**802**) a prefetch request transmitted by a helper, wherein the prefetch request includes a stream identifier; after receiving the prefetch request, the PF transmitting (step s**804**) to an ad server a first metadata request for metadata for use in playing break content (e.g., one or more ads); the PF receiving (step s**806**) from the ad server a first response responsive to the metadata request, wherein the first response comprises the requested metadata; the PF storing (step s**808**) in a data storage unit the metadata received form the ad server (or metadata derived therefrom) and associating the metadata with the stream identifier; after storing the metadata, the PF receiving (step s**810**) from an ad stitching function a second metadata request, wherein the second metadata request includes the stream identifier; in response to receiving the second metadata request, the PF using (step s**812**) the stream identifier to retrieve the stored metadata from the data storage unit; and the PF transmitting (step s**814**) a response responsive to the second metadata request, wherein the response responsive to the second metadata request comprises the retrieved metadata.

[0185] FIG. **9** is a flowchart illustrating a process **900** that includes: a helper receiving (step s**902**) a first presentation segment request from a player; in response to receiving the first presentation segment request, the helper: i) transmitting (step s**904**) a response to the player, wherein the response includes the requested presentation segment or an identifier for enabling the player to retrieve the requested presentation segment and ii) transmitting (step s**906**) to a prefetching function (PF) a prefetch request comprising a stream identifier; the helper receiving (step s**908**) a second presentation segment request from the player; in response to receiving the second presentation segment request, the helper transmitting (step s**910**) to an ad stitcher a request message for causing the ad stitcher to: i) send to the PF a metadata request message comprising the stream identifier and ii) provide to the helper metadata for use in obtaining ad segments; and the helper receiving (step s**912**) from the ad stitcher the metadata for use in obtaining the ad segments.

[0186] FIG. **10** is a block diagram of apparatus **1000**, according to some embodiments, that may implement helper **104**. In some embodiments, apparatus **1000** (or a component thereof) may be a component of device **120**. As shown in FIG. **10**, apparatus **1000** may comprise: processing circuitry (PC) **1002**, which may include one or more processors (P) **1055** (e.g., one or more general purpose microprocessors and/or one or more other processors, such as an application specific integrated circuit (ASIC), field-programmable gate arrays (FPGAs), and the like), which processors may be co-located in a single housing or in a single data center or may be geographically distributed (e.g., apparatus **1000** may be a distributed computing apparatus comprising two or more computers or a single computer); at least one network interface **1048** (e.g., a physical interface or air interface) comprising a transmitter (Tx) **1045** and a receiver (Rx) **1047** for enabling apparatus **1000** to transmit data to and receive data from other network nodes connected to network **110** (e.g., an Internet Protocol (IP) network) to which network interface **1048** is connected (physically or wirelessly) (e.g., network interface **1048** may be coupled to an antenna arrangement comprising one or more antennas for enabling apparatus **1000** to wirelessly transmit/receive data); and a storage unit (a.k.a., "data storage system") **1008**, which may include one or more non-volatile storage devices and/or one or more volatile storage devices. In embodiments where PC **1002** includes a programmable processor, a computer readable storage medium (CRSM) **1042** may be provided. CRSM **1042** may store a computer program (CP) **1043** comprising computer readable instructions (CRI) **1044**. CRSM **1042** may be a non-transitory computer readable medium, such as, magnetic media (e.g., a hard disk), optical media, memory devices (e.g., random access memory, flash memory), and the like. In some embodiments, the CRI **1044** of computer program **1043** is configured such that when executed by PC **1002**, the CRI causes apparatus **1000** to

perform steps described herein (e.g., steps described herein with reference to the flowcharts). In other embodiments, apparatus **1000** may be configured to perform steps described herein without the need for code. That is, for example, PC **1002** may consist merely of one or more ASICs. Hence, the features of the embodiments described herein may be implemented in hardware and/or software.

Summary of Various Embodiments

[0187] A**1**. A method comprising: providing a manifest to a player, wherein the manifest comprises a segment identifier for identifying a segment of first media content or template information for generating the segment identifier; receiving, from the player, a first segment request m**713** comprising the segment identifier for identifying the segment of first media content; and in response to receiving the first segment request: i) providing to the player a first response message m**714** responsive to the first segment request m**713**; and ii) performing a first process for enabling the player to obtain second media content to be played by the player during a break in the first media content, wherein the first process for enabling the player to obtain the second media content comprises transmitting to a prefetching function (PF) a prefetch request m**790** for triggering the PF to send to an ad server a metadata request m**791**, wherein the ad server responds to the metadata request by providing to the PF metadata for the second media content.

[0188] A**2**. The method of embodiment A**1**, wherein the method further comprises: receiving, from the player, a second segment request m**716** comprising a second segment identifier; and in response to receiving the second segment request m**716**: i) providing to the player a second response message m**717** responsive to the second segment request m**716**; and ii) initiating a transmission of a request m**794** to the PF, wherein the PF responds to the request (m**794** by providing the metadata for the second media content.

[0189] A**3**. The method of embodiment A**2**, wherein initiating the transmission of the request m**794** to the PF comprises transmitting to an ad stitcher a message m**718** triggering the ad stitcher to send the request to the PF.

[0190] A**4**. The method of embodiment A**3**, wherein the method further comprises: after transmitting the message m**718** to the ad stitcher, receiving from the ad stitcher a response m**720** responsive to the message m**718**, wherein the response m**720** comprises segment locator information for retrieving segments of the second media content.

[0191] A**5**. The method of embodiment A**4**, wherein the segment locator information for retrieving segments of the second media content comprises a segment identifier identifying a segment of the second media content, and the method further comprises: after providing to the player the second response message m**717** responsive to the second segment request m**716**, receiving from the player a third segment request m**722**; and in response to receiving the third segment request m**722**, performing a second process that includes providing to the player i) the segment identifier identifying the segment of the second media content or ii) the segment of the second media content.

[0192] A**6**. The method of embodiment A**2**, wherein the method further comprises: after providing to the player the second response message m**717** responsive to the second segment request m**716**, receiving from the player a third segment request m**722**; and in response to receiving the third segment request m**722**, performing a second process that includes providing to the player i) a segment identifier identifying a segment of the second media content or ii) the segment of the second media content.

[0193] A**7**. The method of embodiment A**6**, wherein the method further comprises, prior to receiving from the player the third segment request m**722**, mapping an identifier to the segment identifier identifying the segment of the second media content, the third segment request m**722** comprises the identifier mapped to the segment identifier identifying the segment of the second media content, and the second process further comprises, using the identifier included in the third segment request m**722** to retrieve the segment identifier identifying the segment of the second media content.

[0194] A**8**. The method of embodiment A**1**, wherein the first process further comprises, prior to transmitting the prefetch request m**790** to the PF, determining whether the first segment request m**713** comprises a trigger string, and the step of transmitting the prefetch request m**790** to the PF is performed as a result of determining that the first segment request m**713** comprises a trigger string.

[0195] A**9**. The method of embodiment A**1**, wherein the manifest provided to the player includes an ordered list of N segment locators, each one of the first M segment locators included in the ordered list includes a first sever identifier, where M>0 and M<N, each one of the N-M segment locators in the ordered list that follow the first M segment locators in the ordered list includes: i) a second server identifier that is different than the first server identifier and ii) a unique string.

[0196] A**10**. The method of embodiment A**9**, wherein the first process further comprises, prior to transmitting the prefetch request m**790** to the PF, determining, based on the value of a variable, whether a prefetch is needed, and the step of transmitting the prefetch request m**790** to the PF is performed as a result of determining that a prefetch is needed.

[0197] B**1**. A method, the method comprising: receiving a first segment request m**713** from a player; in response to receiving the first segment request: i) transmitting a first response m**714** to the player, wherein the first response includes a first segment of first media content or a first segment identifier for enabling the player to retrieve the first segment of the first media content; and ii) transmitting to a prefetching function (PF) a prefetch request comprising a stream identifier; receiving a second segment request m**716** from the player; and in response to receiving the second segment request m**716**: i) transmitting a second response m**717** to the player, wherein the second response includes a second segment of the first media content or a second segment identifier for enabling the player to retrieve the second segment of the first media content; and ii) performing a process for obtaining segment locator information for use in retrieving segments of second media content.

[0198] B**2**. The method of embodiment B**1**, wherein performing the process for obtaining the segment locator information comprises: transmitting to a server (e.g., stitcher **106**) a request message m**718** for causing the server to: i) send to the PF a metadata request message m**794** comprising the stream identifier and ii) provide the segment locator information; and receiving from the server a response message m**720** comprising the segment locator information.

[0199] B**3**. The method of embodiment B**1**, wherein performing the process for obtaining the segment locator infor-

mation comprises: transmitting to the PF a metadata request message m**794** comprising the stream identifier; receiving from the PF a response message m**798** responsive to the metadata request message; and obtaining the segment locator information using metadata included in the response message from the PF.

[0200] B**4**. The method of embodiment B**3**, wherein obtaining the segment locator information using metadata included in the response message from the PF comprises: retrieving the second media content using information included in the metadata included in the response message from the PF; segmenting the second media content into a set of N segments, N>1; and generating the segment locator information.

[0201] B**5**. The method of embodiment B**4**, wherein the segment locator information comprises, for each one of the N segments, a unique identifier associated with the segment.

[0202] C**1**. A method, the method comprising: receiving a first segment request m**713** from a player; in response to receiving the first segment request: i) transmitting a first response m**714** to the player, wherein the first response includes a first segment of first media content or a first segment identifier for enabling the player to retrieve the first segment of the first media content; and ii) transmitting to an ad server a first metadata request m**791**; receiving from the ad server a first metadata response m**792** responsive to the first metadata request, wherein the first metadata response m**792** comprises metadata for second media content; caching the metadata; receiving a second segment request m**716** from the player; and in response to receiving the second segment request m**716**: i) transmitting a second response m**717** to the player, wherein the second response includes a second segment of the first media content or a second segment identifier for enabling the player to retrieve the second segment of the first media content; and ii) performing a process for obtaining segment locator information for use in retrieving segments of the second media content.

[0203] C**2**. The method of claim C**1**, wherein performing the process for obtaining the segment locator information comprises: transmitting to a server (e.g., stitcher **106**) a second metadata request m**718** for causing the server to send a third metadata request m**794**; and receiving the third metadata request m**794**; in response to receiving the third metadata request m**794**, transmitting to the server a third metadata response m**798** comprising the cached metadata; after transmitting the third metadata response m**798**, receiving from the server a second metadata response m**720** responsive to the second metadata request, wherein second metadata response m**720** comprises the segment locator information.

[0204] D**1**. A method, the method comprising: the PF receiving a prefetch request (e.g., m**790**, wherein the prefetch request includes a stream identifier; after receiving the prefetch request, the PF transmitting to a server a first metadata request m**791** for metadata for use in playing break content; the PF receiving from the server a first response m**792** responsive to the metadata request, wherein the first response comprises the requested metadata; the PF storing in a data storage unit the received metadata or derived metadata derived from the received metadata, and associating the received or derived metadata with the stream identifier; after storing the metadata, the PF receiving a second metadata request m**794**, wherein the second metadata request includes the stream identifier; in response to receiving the second metadata request, the PF using the stream identifier to retrieve the stored metadata from the data storage unit; and the PF transmitting a response m**798** responsive to the second metadata request, wherein the response comprises the metadata retrieved from the data storage unit.

[0205] While various embodiments are described herein, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of this disclosure should not be limited by any of the above-described exemplary embodiments. Moreover, any combination of the above-described elements in all possible variations thereof is encompassed by the disclosure unless otherwise indicated herein or otherwise clearly contradicted by context.

[0206] As used herein transmitting a message "to" or "toward" an intended recipient encompasses transmitting the message directly to the intended recipient or transmitting the message indirectly to the intended recipient (i.e., one or more other nodes are used to relay the message from the source node to the intended recipient). Likewise, as used herein receiving a message "from" a sender encompasses receiving the message directly from the sender or indirectly from the sender (i.e., one or more nodes are used to relay the message from the sender to the receiving node). Further, as used herein "a" means "at least one" or "one or more."

[0207] Additionally, while the processes described above and illustrated in the drawings are shown as a sequence of steps, this was done solely for the sake of illustration. Accordingly, it is contemplated that some steps may be added, some steps may be omitted, the order of the steps may be re-arranged, and some steps may be performed in parallel.

[0208] Further, while all of the examples used herein show each segment being contained in its own file, this was done solely for the sake of illustration. For example, it is known in the art that each segment for a given representation can be stored in the same file and the player requests a particular segment by transmitting to the server not only the identifier for the file but also the identifier (e.g., byte offset) for the segment within the file.

1. A method comprising:

providing a manifest to a player, wherein the manifest comprises a segment identifier for identifying a segment of first media content or template information for generating the segment identifier;

receiving, from the player, a first segment request comprising the segment identifier for identifying the segment of first media content; and

in response to receiving the first segment request:

i) providing to the player a first response message responsive to the first segment request; and

ii) performing a first process for enabling the player to obtain second media content to be played by the player during a break in the first media content, wherein

the first process for enabling the player to obtain the second media content comprises transmitting to a prefetching function (PF) a prefetch request for triggering the PF to send to an ad server a metadata request, wherein the ad server responds to the metadata request by providing to the PF metadata for the second media content.

2. The method of claim 1, wherein the method further comprises:

receiving, from the player, a second segment request comprising a second segment identifier; and

in response to receiving the second segment request:

i) providing to the player a second response message responsive to the second segment request; and

ii) initiating a transmission of a request to the PF, wherein the PF responds to the request (m794) by providing the metadata for the second media content.

3. The method of claim 2, wherein initiating the transmission of the request to the PF comprises transmitting to an ad stitcher a message triggering the ad stitcher to send the request to the PF.

4. The method of claim 3, wherein the method further comprises:

after transmitting the message to the ad stitcher, receiving from the ad stitcher a response responsive to the message, wherein the response comprises segment locator information for retrieving segments of the second media content.

5. The method of claim 4, wherein

the segment locator information for retrieving segments of the second media content comprises a segment identifier identifying a segment of the second media content, and

the method further comprises:

after providing to the player the second response message responsive to the second segment request, receiving from the player a third segment request; and

in response to receiving the third segment request, performing a second process that includes providing to the player i) the segment identifier identifying the segment of the second media content or ii) the segment of the second media content.

6. The method of claim 2, wherein

the method further comprises:

after providing to the player the second response message responsive to the second segment request, receiving from the player a third segment request; and

in response to receiving the third segment request, performing a second process that includes providing to the player i) a segment identifier identifying a segment of the second media content or ii) the segment of the second media content.

7. The method of claim 6, wherein

the method further comprises, prior to receiving from the player the third segment request, mapping an identifier to the segment identifier identifying the segment of the second media content,

the third segment request comprises the identifier mapped to the segment identifier identifying the segment of the second media content, and

the second process further comprises, using the identifier included in the third segment request to retrieve the segment identifier identifying the segment of the second media content.

8. The method of claim 1, wherein

the first process further comprises, prior to transmitting the prefetch request to the PF, determining whether the first segment request comprises a trigger string, and

the step of transmitting the prefetch request to the PF is performed as a result of determining that the first segment request comprises a trigger string.

9. The method of claim 1, wherein

the manifest provided to the player includes an ordered list of N segment locators,

each one of the first M segment locators included in the ordered list includes a first sever identifier, where $M \geq 0$ and $M<N$,

each one of the N-M segment locators in the ordered list that follow the first M segment locators in the ordered list includes: i) a second server identifier that is different than the first server identifier and ii) a unique string.

10. The method of claim 9, wherein

the first process further comprises, prior to transmitting the prefetch request to the PF, determining, based on the value of a variable, whether a prefetch is needed, and

the step of transmitting the prefetch request to the PF is performed as a result of determining that a prefetch is needed.

11. A method, the method comprising:

receiving a first segment request from a player;

in response to receiving the first segment request:

i) transmitting a first response to the player, wherein the first response includes a first segment of first media content or a first segment identifier for enabling the player to retrieve the first segment of the first media content; and

ii) transmitting to a prefetching function (PF) a prefetch request comprising a stream identifier;

receiving a second segment request from the player; and

in response to receiving the second segment request:

i) transmitting a second response to the player, wherein the second response includes a second segment of the first media content or a second segment identifier for enabling the player to retrieve the second segment of the first media content; and

ii) performing a process for obtaining segment locator information for use in retrieving segments of second media content.

12. The method of claim 11, wherein performing the process for obtaining the segment locator information comprises:

transmitting to a server a request message for causing the server to: i) send to the PF a metadata request message comprising the stream identifier and ii) provide the segment locator information; and

receiving from the server a response message comprising the segment locator information.

13. The method of claim 11, wherein performing the process for obtaining the segment locator information comprises:

transmitting to the PF a metadata request message comprising the stream identifier;

receiving from the PF a response message responsive to the metadata request message; and

obtaining the segment locator information using metadata included in the response message from the PF.

14. The method of claim 13, wherein obtaining the segment locator information using metadata included in the response message from the PF comprises:

retrieving the second media content using information included in the metadata included in the response message from the PF;

segmenting the second media content into a set of N segments, N>1; and

generating the segment locator information.

15. The method of claim **14**, wherein the segment locator information comprises, for each one of the N segments, a unique identifier associated with the segment.

16. A method, the method comprising:

receiving a first segment request from a player;

in response to receiving the first segment request:

i) transmitting a first response to the player, wherein the first response includes a first segment of first media content or a first segment identifier for enabling the player to retrieve the first segment of the first media content; and

ii) transmitting to an ad server a first metadata request;

receiving from the ad server a first metadata response responsive to the first metadata request, wherein the first metadata response comprises metadata for second media content;

caching the metadata;

receiving a second segment request from the player; and

in response to receiving the second segment request:

i) transmitting a second response to the player, wherein the second response includes a second segment of the first media content or a second segment identifier for enabling the player to retrieve the second segment of the first media content; and

ii) performing a process for obtaining segment locator information for use in retrieving segments of the second media content.

17. The method of claim **16**, wherein performing the process for obtaining the segment locator information comprises:

transmitting to a server a second metadata request for causing the server to send a third metadata request; and

receiving the third metadata request;

in response to receiving the third metadata request, transmitting to the server a third metadata response comprising the cached metadata;

after transmitting the third metadata response, receiving from the server a second metadata response responsive to the second metadata request, wherein second metadata response comprises the segment locator information.

* * * * *