

IN THE UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF FLORIDA
MIAMI DIVISION

CASE No. 1:24-cv-21226-RAR

DR. EDWIN A. HERNANDEZ and
EGLA CORP.,

Plaintiffs,

V.

STINGRAY GROUP INC. f/k/a STINGRAY DIGITAL GROUP, INC., STINGRAY MUSIC USA, INC., MOOD MEDIA LLC f/k/a MOOD MEDIA CORPORATION, AT&T ENTERPRISES, LLC f/k/a AT&T CORP., MILLICOM INTERNATIONAL SERVICES, LLC, and BLUE STREAM COMMUNICATIONS, LLC dba BLUE STREAM FIBER,

Defendants.

JURY TRIAL DEMANDED

**DEFENDANTS STINGRAY GROUP, INC., AT&T ENTERPRISES, LLC, MILLICOM
INTERNATIONAL SERVICES, LLC, AND BLUE STREAM COMMUNICATIONS,
LLC'S¹ OPENING CLAIM CONSTRUCTION BRIEF**

¹ Defendants Stingray Music USA, Inc. and Mood Media, LLC are not accused of patent infringement and therefore take no position on the proposed claim constructions in this brief.

TABLE OF CONTENTS

I. Introduction 1

II. Legal Background 2

III. Claim Construction Analysis..... 3

 A. “web page” / “rendering a web page” / “rendered web page” / “browser” 3

 B. “sequence of screen captures of the rendered web page” 6

 C. “illustrate a dynamic change” 8

 D. “output the parameters via MPEG outputs” 11

 E. “multimedia assets” 13

 F. “maintain generation of video and audio files, even in failure cases” 15

IV. Conclusion..... 19

TABLE OF AUTHORITIES

Cases

Cat Tech LLC v. TubeMaster, Inc.
528 F.3d 871 (Fed. Cir. 2008) 10

Geneva Pharm., Inc. v GlaxoSmithKline PLC 349 F.3d 1373 (Fed. Cir. 2003)..... 17

Halliburton Energy Servs., Inc. v. M-I, LLC
514 F.3d 1244 (Fed. Cir. 2008) 18

Kensey Nash Corp. v. Perclose, Inc.
57 Fed. App’x 864 (Fed. Cir. 2003) 10

Media Rights Techs., Inc. v. Capital One Financial Corp.
800 F.3d 1366 (Fed. Cir. 2015) 14

Optikam Tech Inc., v. Am. Bright Signs, Inc.
No. 23-23597, 2024 WL 5263795 (S.D. Fla. Jun. 13, 2024) (Ruiz, J.)..... 2

Panavision Imaging, LLC v. OmniVision Tech., Inc.
No. 2:09-cv-1577, 2012 WL 12952006 (C.D. Cal. May 25, 2012)..... 18

Phillips v. AWH Corp.
415 F.3d 1303 (Fed. Cir. 2005) 2

Synchronoss Techs., Inc. v. Dropbox, Inc.
987 F.3d 1358 (Fed. Cir. 2021) 16

Zeta Global Corp. v. Maropost Mktg. Cloud, Inc.
No. 20-cv-3951, 2021 WL 2823563 (S.D.N.Y. Jul. 7, 2021)..... 16

I. Introduction

Plaintiffs' Second Amended Complaint alleges that Defendants Stingray Group, Inc., AT&T Enterprises, LLC, Millicom International Services, LLC and Blue Stream Communications, LLC (collectively, the "Patent Defendants") infringe over 52 claims from three asserted patents. Pursuant to this Court's schedule and the patent law, the Patent Defendants file this Opening Claim Construction Brief, seeking constructions for six sets of claim terms.

As an initial matter, the Patent Defendants respectfully note that they have moved to stay this case pending the Patent Trial and Appeal Board's resolution of the IPR petitions, which challenge the validity of all claims from the three asserted patents. *See* Dkt. 184. As discussed in that motion, which is fully briefed and currently pending, a stay of this litigation would promote judicial economy by narrowing (if not eliminating) the pending patent claims and potentially avoiding the need for a claim construction hearing in this case. Moreover, to the extent any patent claims survive the IPR challenges, the resulting IPR record will assist this Court in construing various claim terms, including terms raised in this brief.

As discussed below, many of the critical claim terms—including terms that were added during prosecution to overcome the Patent Office's initial rejection—are either not explicitly defined by the patentee or not even mentioned in the patents' shared specification. For example, to distinguish the cited prior art, Plaintiffs amended the claim language to require: (1) "rendering a web page by a browser," (2) "generating a temporal sequence of screen captures of the rendered web page," and (3) that "at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time." *See* Ex. E at 2. In making this amendment, Plaintiffs conceded that these added requirements were "an important distinction between the claimed solution and the cited references." *Id.* at 8-9. Indeed, this amendment formed the basis for the Examiner's subsequent allowance of the claims. *See* Ex. F at 2.

Despite the importance of these additions, Plaintiffs’ current infringement allegations do not identify, for example, any browser rendering any web page, any sequence of screen captures of such a rendered web page, or any dynamic change that would be illustrated by any such screen captures. *See, e.g.*, Dkt. 82, Ex. 24 at 3-4. In their pending Motion to Dismiss, the Patent Defendants highlighted these deficiencies, noting as an example that Plaintiffs’ charts “simply state that ‘Still Pict[ures]’ are generated and fail to identify how UbiquiCAST OSE2 allegedly practices this ‘temporal sequence/dynamic change’ limitation.” *See* Dkt. 148 at 16.

Accordingly, to help streamline disputes, the Patent Defendants seek a construction of the claim terms below in light of the patents’ specification, the prosecution history, and the meaning (if any) that these terms would have to a Person of Ordinary Skill in the Art (“POSITA”).

II. Legal Background

Claim construction is a matter of law for the court. *See Optikam Tech Inc., v. Am. Bright Signs, Inc.*, No. 23-23597, 2024 WL 5263795, at *1 (S.D. Fla. Jun. 13, 2024) (Ruiz, J.). As stated by this Court, “[t]he goal of claim construction is to give disputed terms their ‘ordinary and customary meaning’ as the term would mean to a person of ordinary skill in the art in question as of the effective filing date of the patent application.” *Id.* In construing a claim term, the Federal Circuit has noted that “the claim themselves provide substantial guidance as to the meaning of particular claim terms.” *See Phillips v. AWH Corp.*, 415 F.3d 1303, 1314 (Fed. Cir. 2005). Additionally, a court must also “look at the ordinary meaning in the context of the written description and the prosecution history.” *Id.*

A claim is indefinite under the law if the claim terms, as “viewed in light of the specification and prosecution history, fail to inform those skilled in the art about the scope of the invention with reasonable certainty.” *See Optikam*, 2024 WL 5263795, at *1. Because patents are presumed valid, indefiniteness must be proven by “clear and convincing evidence.” *Id.*

III. Claim Construction Analysis

A. “web page” / “rendering a web page” / “rendered web page” / “browser”

Patents and Claims	Claim Language	Construction
'074: 1-21 '002: 1-13	“web page”	“an HTML document that is accessible on the world wide web”
'441: 10-26	“rendering a web page”	“converting a web page’s HTML code into displayable content”
	“rendered web page”	“displayable content generated from a document’s HTML code”
	“browser”	“an application that converts HTML code into displayable content”

The claims identified in the chart above all require either “render[ing] a web page by a browser” and/or a “rendered web page.” For example, claim 1 of the '074 Patent recites “rendering a web page by a browser” and then “generating a temporal sequence of screen captures of the rendered web page” in order to assemble a media stream. *See* Ex. A at Claim 1.

The intrinsic evidence supports the Patent Defendants’ proposed constructions. With respect to the above claim language, the asserted patents—which share the same specification—disclose “JavaScript, CSS and HTML content files that can be rendered using a browser using WebKit (e.g. PhantomJS or Safari) *or any other browser-rendering engine compatible with HTML4/5 or any future HTML version.*” *See* Ex. A at 7:1-4 (emphasis added). Once the web page’s HTML is rendered, the patents disclose capturing “a sequence of screens that can be used to generate a video file.” *Id.* at 7:22-24. The patents state that “[c]apturing screens can be done using a tool similar to PhantomJS (Webkit-based headless engine for rendering images) or QtCapture, or simply capturing screen frame buffers from a machine running any software tool.” *Id.* at 7:18-21. Based on the patents’ specification, a POSITA would understand that the claims

are directed to the use of a common browser that is already “compatible with HTML” to render web pages into displayable content that can then be screen captured. *See* Ex. D at ¶ 31. As a result, the patents’ reference to “rendering a web page with a browser” comports with a POSITA’s ordinary understanding of this concept. *Id.*

Specifically, a POSITA would understand that the term “web page” is used as is ordinarily understood in the art—*i.e.*, a document on the world wide web that uses HTML, including any embedded CSS or JavaScript code, to define its appearance. *Id.* at ¶ 32; *see also* Ex. L at 4 (stating that “[a] web page consists of an HTML file, with associated files for graphics and scripts”). HTML, or hypertext markup language, is a standardized markup language used to display pages of content on the web. *Id.* A POSITA would understand that a web page includes HTML elements that define its appearance, and browsers render a web page by translating or converting those HTML elements into displayable content for the user. *Id.* at ¶¶ 32-33. As a simple example, the screen shot below shows HTML code on the left side—including code specifying the background color, text color, font style and size—and the rendered graphical representation of that HTML code on the right side, as it would be displayed in a browser:

```
<!DOCTYPE html>
<html>
<body style="background-color:powderblue;">

<h1>Example Heading</h1>

<p style="color:blue;">The corresponding HTML code makes
this text color blue.</p>

<p style="color:green;">The corresponding HTML code makes
this text color green.</p>

<p style="color:red;">The corresponding HTML code makes
this text color red.</p>

<p style="font-family:Ariel; font-size:10vw;">HTML can
also be used to change the font style and size.</p>

</body>
</html>
```

Example Heading

The corresponding HTML code makes this text color blue.

The corresponding HTML code makes this text color green.

The corresponding HTML code makes this text color red.

HTML can also be used
to change the font style
and size.

Id. at ¶ 33.

The proposed constructions are also consistent with relevant extrinsic evidence. The purpose of Plaintiffs’ alleged invention is to assemble screen captures of rendered HTML in order to deliver web-based content to broadcasters, such as satellite or cable television operators. According to Plaintiffs’ own licensing presentation, they specifically characterized their invention as the “[r]endering of HTML-based web pages” for delivery:

Analysis and Inventions on 10,123,074 Patent

- **Linear and VOD Streaming of Music, Music + Video, or Video/TV to Cable TV Operators**
- Rendering of HTML-based webpages from temporal screen captures and creating video streams with audio ready for broadcasting to MPEG Systems (aka Cable TV, DTH) for Linear or VOD Broadcasting
- Creating all screen captures and storing them in MPEG Format for 2-way interaction (VOD)
- Deployment and distribution using virtual machines or instances
- CDN integrated for Web, Mobile, and Cable TV broadcasting
- Fault-tolerance and detection



See Ex. G at 5.

Similarly, on Plaintiffs’ own website, they state that the asserted patents use “software to connect to the web, *retrieve HTML content* and render that content” for broadcast. See Ex. H at 3 (emphasis added). In stark contrast, Plaintiffs unambiguously state on this same website that “[t]he older technology and the solutions available were more hardware-based and *was not HTML-enabled.*” *Id.* at 2 (emphasis added).

Because web pages use HTML to define their appearance, the claims use a common web browser—such as any “browser-rendering engine compatible with HTML4/5 or any future HTML version”—to render content for screen capture. See Ex. A at 7:1-4. A POSITA would understand that a “browser” is an application that converts such HTML code into displayable content. See Ex. D at ¶ 34; see also Ex. L at 3 (defining “web browser” as an “application that enables a user

to view HTML documents on the World Wide Web”). Because web pages use HTML, and because HTML is a standardized language used across the web, applications that are capable of translating HTML code can readily visualize pages on the web. This is why they are called browsers—users can use them to *browse* the billions of pages on the web. Conversely, an application that cannot translate HTML code into displayable content cannot “browse” the web and is not a “browser” as that term is understood in the art. Ex. D at ¶ 34.

In light of the above, a POSITA would understand that a “web page” is an HTML document that is accessible on the world wide web, and that the act of “rendering a web page” means converting a web page’s HTML code into displayable content. *Id.* at ¶ 37. Similarly, a POSITA would understand that a “rendered web page” is the displayable content generated from a document’s HTML code. *Id.* A POSITA would further understand that a “browser” is an application that converts HTML code into displayable content. *Id.*

B. “sequence of screen captures of the rendered web page”

Patents and Claims	Claim Language	Construction
’074: 1-21 ’002: 1-13 ’441: 10-26	“sequence of screen captures of the rendered web page”	“two or more screen captures of the rendered web page taken at different points in time”

The claims above further require a “sequence of screen captures of the rendered web page.” The term “sequence” together with the plural form “screen captures” means there are two or more screen captures taken one after another in series—*i.e.*, at different times. *See* Ex. K (defining “sequence” as “[a] following of one thing after another; succession”). In contrast, a single screen capture would neither be a “sequence” nor a plurality of “screen captures.” Similarly, taking a single screen capture and generating multiple copies of that single screen capture would not be generating a “sequence” of screen captures.

This common understanding is consistent with the patents' specification. With respect to this limitation, the patents state that "the method can *capture a video file* from another rendering function, which can create *a sequence of screens* that can be used to generate a video file." *See* Ex. A at 7:22-24 (emphasis added). The act of capturing a video file involves taking screen captures at different points in time during the playback of the video.

Similarly, the proposed construction is also consistent with the prosecution history. During prosecution of the '074 Patent, Plaintiffs amended the claims to add the requirement of a "sequence of screen captures" where "at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time." *See* Ex. E at 2. Plaintiffs even called this addition an "important distinction" over the cited prior art. *Id.* at 8-9. Based on this amendment, the Examiner subsequently allowed the claims, stating in the Notice of Allowability that "[w]hile it is known in the art to have multiple screen shots of a webpage (for example, web.archive.org grabs entire webpages at different points in time) . . . none of these would result applicant's invention." *See* Ex. F at 2. As stated by the Examiner, "[n]one of the art of record teaches the . . . sequence feature which includes *a time* and *two adjacent times* that illustrate a dynamic change of at least a portion of the content over time." *Id.* (emphasis added). As a result, the Examiner's Notice of Allowability recognizes that a "sequence" of screen captures is composed of screen captures from different points in time.

In fact, given that the patents require this "sequence" to include two adjacent screen captures of the rendered web page that "illustrate a dynamic change of its content over time," those two adjacent screen captures must logically be taken at *different* points in time. In other words, the only way two adjacent screen captures could capture *any* change in the rendered web page—let alone the "dynamic change" required by the claims—is if the screen captures are from two different points in time. In contrast, merely duplicating a single screen capture would not illustrate

any change in the rendered web page. Accordingly, a POSITA would understand that a “sequence of screen captures of the rendered web page” means two or more screen captures of the rendered web page taken at different points in time. *See* Ex. D at ¶¶ 38-44.

C. “illustrate a dynamic change”

Patents and Claims	Claim Language	Construction
'074: 1-21 '002: 1-13 '441: 10-26	“illustrate a dynamic change”	“illustrate the playback of video, animation or similar motion or change occurring on the rendered web page”

The identified claims also require that at least two adjacent screen captures “illustrate a dynamic change” of the web page’s content.

As previously discussed, in response to the Patent Office’s rejection, Plaintiffs amended the claims to require a “sequence of screen captures” wherein “each screen capture defines *all the content of the web page* at a given time, and at least two adjacent screen captures *illustrate a dynamic change of at least a portion of the content over time.*” *See* Ex. E at 2; *see also* Ex. A at Claim 1 (emphasis added). Notably, Plaintiffs did not amend the claims to simply require illustrating “a change” of the web page—instead, the amendment specifies that this illustrated change must be a “dynamic change.” However, the patents’ specification does not mention either the terms “dynamic” or “dynamic change,” and this concept is not defined or discussed in the prosecution history.

The Examiner’s subsequent Notice of Allowability recognized that using screen captures to illustrate changes in a rendered web page was already known in the art. Specifically, the Notice of Allowability states that “it is known in the art to have multiple screen shots of a webpage (for example, web.archive.org grabs entire webpages at different points in time).” *See* Ex. F at 2. A POSITA is familiar with the public website Archive.org—popularly known as the Wayback

Machine—which periodically crawls web pages, taking screen captures to preserve their content for future use. *See* Ex. D at ¶ 47. For example, the Wayback Machine may take a screen capture of the front page of ESPN.com today and then take another screen capture a month or even a year later. Indeed, the central purpose of the Wayback Machine is to record **changes** in such web pages over time, forming a vast digital archive of the internet. *Id.* As a result, a POSITA would understand from the Notice of Allowability that a “dynamic” change must mean something different from “any” change in the underlying web page’s content. *Id.* at ¶ 48.

In the context of web design and mobile applications, a POSITA understands that “dynamic content” refers to the playback of video, animation, or similar motion or change (such as content disappearing, reappearing or moving) that is occurring on the rendered web page or in the mobile application. Ex. D at ¶ 49. For example, a technical paper written by Dr. Sam Malek, titled “Automated Accessibility Analysis of Dynamic Content Changes on Mobile Apps,” notes that “[d]ynamically changing visual content of screen (e.g., **through animation**) is a commonly used technique for enhancing the visual aesthetics of an app.” *See* Ex. I at 1 (emphasis added) and 3 (identifying types of dynamic content); Ex. D at ¶ 49. Web designers include dynamic content through the use of Dynamic HTML, which relies on JavaScript to actively modify elements in the HTML document. *See* Ex. D at ¶ 49.

Although the patents’ specification never uses the term “dynamic change,” a POSITA’s understanding of dynamic content—such as the playback of video or animations—is consistent with the disclosed embodiment of the alleged invention. *See* Ex. D at ¶ 50. In particular, the specification describes taking screen captures from the playback of video on a web page, stating “the method can **capture a video file from another rendering function**, which can **create a sequence of screens** that can be used to generate a video file.” *See* Ex. A at 7:22-24 (emphasis

added). Similarly, the patents elsewhere state that the screen captures can include “captures of images *from video* that are available on the website.” *Id.* at 7:65-8:1 (emphasis added).

Given that there is no explicit discussion of the term “dynamic change” in the specification, to the extent it is definite the term should be construed in light of a POSITA’s ordinary understanding of dynamic content. *See* Ex. D at ¶¶ 48, 50. For example, “still images” from a static—*i.e.*, non-dynamic—web page are not dynamic content. *Id.* at ¶ 50. Thus, screen capturing still images from a static web page would not “illustrate a dynamic change” of its content over time. *Id.* Similarly, because the prior art Wayback Machine captures still images from web pages in order to archive any changes over time, a POSITA would understand from the Examiner’s Notice of Allowability that merely capturing “still images” from a web page would not satisfy the “dynamic change” language, even if that web page was modified or similarly rewritten in the intervening time period between screen captures. *Id.*

Given that the claims were narrowed to secure allowance, Plaintiffs are not permitted to define terms in a way that recaptures surrendered claim scope, especially if such an attempt renders the added word “dynamic” a nullity. *See Kensey Nash Corp. v. Perclose, Inc.*, 57 Fed. App’x 864, *2 (Fed. Cir. 2003) (rejecting patent owner’s construction that “attempts to recapture the very scope rejected by the examiner”); *Cat Tech LLC v. TubeMaster, Inc.*, 528 F.3d 871, 885 (Fed. Cir. 2008) (rejecting a construction that rendered an important term “functionally meaningless”). In order to “illustrate a dynamic change” of a web page, that web page must logically be written to display dynamic content, such as video, animation or other active changes that are occurring on the rendered web page. Accordingly, because the patents do not define this term in any other manner, the Court should construe “illustrate a dynamic change” to mean “illustrate the playback of video, animation or similar motion or change occurring on the rendered web page.”

D. “output the parameters via MPEG outputs”

Patents and Claims	Claim Language	Construction
'002: 7-12	“output the parameters via MPEG outputs”	Indefinite.

Claims 7-12 of the '002 Patent require “a multicast server configured to generate parameters relating to at least one of video content, image content and audio content” and also “configured to output the parameters via MPEG outputs.” *See* Ex. B at Claim 7. Those claims then require a “caching server” that is “configured to receive and store the parameters.” *Id.*

The claim language is clear that the required parameters need to be outputted from the multicast server and received by the caching server, where they are stored. *See* Ex. D at ¶ 52. For example, the patents state that “encoding flags or input parameters can be received which may include GOP (Group of Pictures) which is used by certain video encoders, resolution (e.g. 640x480) in pixels, bit rate in Kilobits per second, any encoding type of video (e.g. H.264), encoding type of audio (e.g. AC-3), duration of the video feed, etc.” *See* Ex. B at 6:35-40. “In addition, a ‘format’ parameter can be received which refers to the output format or encapsulation, MPEG TS or MPEG4.” *See* Ex. B at 6:40-42.

Notably, though, the claims do not simply require outputting the parameters to the caching server for storage. Instead, the claim language requires those parameters to be outputted in a specific way—*i.e.*, “via MPEG outputs.” However, the patents do not provide any explanation of what it means to “output the parameters via MPEG outputs,” as required by the claims.

A POSITA would understand that “MPEG” refers to the “Moving Pictures Experts Group,” which is an alliance of working groups that sets standards for media coding. *See* Ex. D at ¶ 54. For example, MPEG has developed various encoding formats, such as MPEG-2 and MPEG-4, which are used to encode video and audio data. Similarly, hardware companies manufacture

dedicated MPEG formatted encoders, such as the H-AV-MPEG2-E hardware encoder sold by Thor Broadcast. *See* Ex. J; *see also* Ex. B at 1:48-49 (stating that “MSOs and DSSs use dedicated hardware encoders” as opposed to software encoders running on general purpose computers). However, “via MPEG outputs” has no known meaning to a POSITA, and nothing in the patents or prosecution history sheds light on its meaning. *See* Ex. D at ¶¶ 55-58.

Because the patents’ specification and prosecution history is silent as to the term “via MPEG outputs,” a POSITA would not know what is required for infringement. *Id.* For example, a POSITA would not know whether outputting parameters “via MPEG outputs” requires that the multicast server first encodes those parameters in accordance with some unidentified MPEG encoding standard or, alternatively, whether this limitation is referring to physical outputs of a dedicated MPEG hardware encoder card on the multicast server. *Id.* Similarly, a POSITA would not know whether “via MPEG outputs” is instead meant to refer to a specific set of “virtual Ethernet ports” or “physical interfaces” on the multicast server and, if so, how those ports/interfaces become “MPEG outputs” for purposes of infringement. *See* Ex. B at 12:15-22 and Fig. 10 (referencing various Ethernet ports and physical interfaces in a described embodiment); *see also* Ex. D at ¶ 56. Neither the patents nor the figures ever refer to any such port or interface—hardware or virtualized—as an “MPEG output.”

Given that there is no explanation in the intrinsic record—and given that outputting parameters “via MPEG outputs” has no understood meaning in the art—the requirement that the multicast server must “output the parameters via MPEG outputs” fails to inform those skilled in the art about the scope of the invention with reasonable certainty. A POSITA would have no way of knowing whether their system is outputting parameters “via MPEG outputs” or via some other type of outputs that would avoid infringement. As a result, this claim term is indefinite.

E. “multimedia assets”

Patents and Claims	Claim Language	Construction
'441: 1-9	“multimedia assets”	Indefinite.

Claims 1-9 of the '441 Patent require “creating a plurality of multimedia assets.” *See* Ex. C at Claim 1. However, the patents do not use the terms “multimedia asset” or “asset” at any point in their specification, and these terms appear nowhere in the prosecution history. On its own, the term “multimedia asset” is not a term of art and has no universal meaning to a POSITA. *See* Ex. D at ¶ 60. Exacerbating this problem, the patents reference a variety of different “multimedia” components, including (1) “multimedia content;” (2) “multimedia files;” (3) “multimedia items;” (4) “multimedia streams;” and (5) “multimedia playlists.”

For example, with respect to the term “**multimedia content**,” the patent discusses broadly “delivering multimedia content from a cloud-based system to cable or satellite content providers.” *See* Ex. C at 1:21-23. In contrast, the term “**multimedia file**” is more narrowly defined and described by the patents as “includ[ing], as non-limiting examples, MPEG-4 Video files with H.264 encoded video and AAC encoded audio or MPEG-layer 3 audio as well (MPEG2 Video could also be used). *Id.* at 4:11-14. Likewise, the patents describe “**multimedia items** such as audio/video multimedia items” that are assembled “using the new plurality of screen captures.” *Id.* at Claim 13. The patents also reference “**multimedia streams**” and state that “[o]nce the audio and video files are created, they can be combined, multiplexed, or encoded to a format that can be utilized by the multimedia stream.” *Id.* at 8:13-16 and 6:8-10 (stating that “Mediaplug can broadcast the multimedia MPEG stream”) and 1:61-64 (equating “multimedia streams” with “TV, radio, [and] Video-on-Demand” programming). Lastly, the patents also reference “**multimedia**

playlists” and state that “the caching server can generate a ‘playlist file’ . . . which will be interpreted as a multimedia playlist file when requested.” *Id.* at 6:27-29.

Given the use of “multimedia” to refer to and modify a variety of different concepts—combined with the fact that there is no mention of “multimedia asset” anywhere in the intrinsic record—a POSITA would not know what the term “multimedia asset” refers to. *See* Ex. D at ¶¶ 61-62. For example, claim 1 of the ’441 Patent requires: (1) that the “multimedia asset” be created at a caching unit; (2) for each “multimedia asset” determining if there is sufficient storage space; (3) generating a channel identifier for the “multimedia asset”; (4) retrieving a plurality of media files to be included in the “multimedia asset”; (5) storing the “multimedia asset” at the caching unit; (6) receiving a request at the caching unit for the “multimedia asset” from the broadcasting unit; and (7) the caching unit providing the “multimedia asset” to the broadcasting unit in the MPEG transport stream format. *See* Ex. C at Claim 1.

However, those requirements could apply equally without regard to whether the “multimedia asset” refers to a “multimedia file,” a “multimedia stream,” a “multimedia item,” a “multimedia playlist,” or to just “multimedia content” in the abstract—or to any of those terms in combination or to something else entirely. Because the claim language fails to indicate which (if any) of those possibilities the term applies to, a POSITA would not understand what the term “multimedia asset” means. *See* Ex. D at ¶ 63; *see also Media Rights Techs., Inc. v. Capital One Financial Corp.*, 800 F.3d 1366, 1371 (Fed. Cir. 2015) (holding that “a claim is indefinite if its language might mean several different things and no informed and confident choice is available among the contending definitions”). As a result, “multimedia asset” fails to inform those skilled in the art about the scope of the invention with reasonable certainty and is indefinite.

F. “maintain generation of video and audio files, even in failure cases”

Patents and Claims	Claim Language	Construction
'002: 7-12	“maintain generation of video and audio files, even in failure cases”	Indefinite.

Claims 7-12 of the '002 Patent require “generating a multicast stream with MPEG encoded video and audio” and “a monitoring system configured to maintain generation of video and audio files, even in failure cases.”

With respect to this requirement, the patents state that the claimed system “can also monitor all of the processes to ensure stability and continuous, uninterrupted operation.” *See* Ex. B at 5:65-67 and 8:63-66 (referencing a “monitoring block that can ensure that the process running the ‘Media Parallel encoder’ keeps running at all times and does not stop upon failure”).

As an initial matter, a POSITA would not know what it means to “maintain generation . . . even in failure cases.” *See* Ex. D at ¶ 66. A POSITA would understand the term “failure case” generally means an unintended state that prevents the normal operation of a system. *Id.* For example, intentionally shutting down a system for routine maintenance (*i.e.*, entering an ***intended*** state of non-operation) would not be considered a “failure case” by a POSITA. *Id.* Similarly, a system designed with multiple tiers of redundancy and programmed to load-balance when servers come on-line and go off-line would not be considered to have encountered a “failure case” when the system is simply doing what it is designed to do and maintaining operation. *Id.*

Conversely, a POSITA would understand that a system has encountered a “failure case” when it is instead (unintentionally) no longer able to do what it was designed to do. *Id.* at ¶ 67. For example, a system operating on a single server or computer that experiences an electrical short or a sudden loss in power that shuts the system down would be viewed as encountering a failure case. *Id.* As a result, the concept of “maintaining generation of video and audio files, even in

failure cases” is, in effect, an oxymoron that is internally nonsensical. *See Synchronoss Techs., Inc. v. Dropbox, Inc.*, 987 F.3d 1358, (Fed. Cir. 2021) (finding that the asserted claims “are nonsensical and require an impossibility—that the digital media file contains a directory of digital media files” and holding that the claims are therefore indefinite). The fact that a distribution system is “maintaining generation of video and audio” inherently means that the system is working. As long as it is operating as intended—including through any system of redundancy or monitoring—there is *no failure case*.

For example, in *Zeta Global*, the patent characterized “failure messages” as sent “when the ISP is *unable* to deliver an email message to an email address associated with the ISP.” *See Zeta Global Corp. v. Maropost Mktg. Cloud, Inc.*, No. 20-cv-3951, 2021 WL 2823563, at *5 (S.D.N.Y. Jul. 7, 2021) (emphasis in original). However, in evaluating the claim language, the court emphasized that it required “a failure message that is determined *after delivery* of the email message.” *Id.* (emphasis in original). Because this was “contradictory and nonsensical,” the court held that the term “failure message” was indefinite. *Id.*

Similarly, conjuring impressions of the great Yogi Berra,² the claim language here is akin to saying that “the system works even when it fails.” But this is not a problem of mere semantics. Instead, given this internal contradiction, a POSITA cannot reasonably understand the scope of the claims. For example, a POSITA would not know whether a multimedia distribution platform with perfect redundancy/monitoring is a system that simply *does not fail* or, on the other hand, is a system that consistently “fails” but, nonetheless, still maintains uninterrupted generation of video and audio files. *Id.* at ¶¶ 68-69. As a result, the claim term “configured to maintain generation of

² In addition to being a Hall of Fame baseball player and manager, Yogi Berra was legendary for his sharp wit and head-scratching quips, such as: “when you come to a fork in the road, take it” or “nobody goes there anymore, it’s too crowded” or (Stingray counsel’s personal favorite) “always go to other people’s funerals, otherwise they won’t come to yours.”

video and audio files, even in failure cases” fails to inform those skilled in the art about the scope of the invention with reasonable certainty and is indefinite.

Additionally, this claim term is also indefinite for a second reason. Notably, the limitation does not specify a discrete range or category of “failure cases” in which it must continue operation. Instead, on its face, the language requires that the system maintain generation of video and audio “even in failure cases,” and the claims do not attempt to limit the scope of failures. For example, under the plain language, the system would need to be able to maintain generation even in the event of widespread power outages or malware attacks. Conversely, if the system is unable to maintain generation in such an event, then it would **not** be “configured to maintain generation of video and audio files, even in failure cases,” as required by the claim language.

This problem is critical because a POSITA has no way of knowing whether a given system is so configured. *Id.* at ¶¶ 70-72. For example, even if a POSITA could test a given system under a set of known failure cases, a POSITA would never be able to confirm that the system can maintain generation of video and audio in the limitless number of **unknown** failure cases. As a simple example, a POSITA may be able to design a system that is immune to a variety of known malware such that the system could continue operation even if it becomes infected. However, a POSITA could never say whether that system is immune to every possible malware that could infect its operation. *Id.* As a result, a POSITA has no way of knowing if a system is configured to “maintain generation of video and audio files, even in failure cases” and, consequently, would have no way of knowing whether that system infringes the claims above. *Id.*

In *Geneva Pharmaceuticals*, the Federal Circuit held that the claim term “synergistically effective amount”—*i.e.*, an amount that works to treat an infection—was indefinite because a product “might infringe or not depending on its usage in changing circumstances.” *See Geneva Pharm., Inc. v GlaxoSmithKline PLC*, 349 F.3d 1373, 1384 (Fed. Cir. 2003). As stated by the

court, “a given embodiment would simultaneously infringe and not infringe the claims, depending on the particular bacteria chosen for analysis.” *Id.* “Thus, one of skill in the art would not know from one bacterium to the next whether a particular composition standing alone is within the claim scope or not.” *Id.* (stating that this is “the epitome of indefiniteness”). In the same way, a POSITA would not know whether a given multimedia distribution system that “maintain[s] generation of video and audio files” in one failure case would similarly work in all failure cases, which would depend on each unique circumstance.

In *Halliburton*, the Federal Circuit similarly held that the term “fragile gel” was indefinite. *See Halliburton Energy Servs., Inc. v. M-I, LLC*, 514 F.3d 1244, 1250 (Fed. Cir. 2008). Noting that “fragile gel” was defined in functional language as a fluid “capable of suspending drill cuttings,” the court observed that “a given fluid might be adequate to suspend drill cuttings in some formations and/or well configurations, whereas in others it would not be.” *Id.* at 1254–55. The court cautioned that “[w]hen a proposed construction requires an artisan to make ***a separate infringement determination for every set of circumstances*** in which the composition might be used, and when such determinations are likely to result in differing outcomes (sometimes infringing and sometimes not), that construction is likely to be indefinite.” *Id.* at 1255 (warning that this is the “danger” of functional claim language, which defines an element “by what it does rather than what it is”) (emphasis added); *see also Panavision Imaging, LLC v. OmniVision Tech., Inc.*, No. 2:09-cv-1577, 2012 WL 12952006, at *4 (C.D. Cal. May 25, 2012) (holding that “high-impedance” was indefinite because the patent’s usage “depends not on some measurable characteristic of the amplifier but on whether the amplifier produces the desired result”).

Here, because “configured to maintain generation of video and audio files” is expressed in similar functional language—*i.e.*, expressed as what it does, not what it is—an artisan would have to separately determine whether a system maintains the required generation for every possible set

of failure cases, and the outcome for any particular failure case would depend entirely on that failure case's unique circumstances. Simply put, whether a given media distribution system affirmatively infringes these claims cannot be determined at any time. As a result, under the law, this claim term is indefinite.

IV. Conclusion

Defendants' proposed constructions reflect the ordinary meaning (if any) that POSITAs would assign to each term based on a reading of the claims, the specification, and the prosecution history. Adopting such constructions would not only assist the factfinder in evaluating Plaintiffs' claims of infringement, but it would streamline the issues in dispute and help conserve the parties' resources going forward. Additionally, for some terms addressed above, the Patent Defendants have explained why the terms are indefinite under the law. Importantly, for these indefinite terms, the intrinsic record is silent as to what the terms mean, and a POSITA would be unable to determine the scope of the invention with reasonable certainty, including determining whether any product is infringing. Based on the above, the Patent Defendants respectfully request that the Court enter an order adopting their proposed constructions.

Dated: February 6, 2025

Respectfully submitted,

RIVERO MESTRE LLP

2525 Ponce de Leon Boulevard, Suite 1000

Miami, Florida 33134

Telephone: (305) 445-2500

Fax: (305) 445-2505

Email: jmestre@riveromestre.com

Email: ahenry@riveromestre.com

/s/ Allison Henry

JORGE A. MESTRE

Florida Bar No. 88145

ALLISON HENRY

Florida Bar No. 1003008

*Counsel for Defendants Stingray Group Inc.,
Stingray Music USA, Inc., Millicom
International Services LLC, and Blue Stream
Communications LLC*

/s/ Demetrios Anaipakos

Demetrios Anaipakos (admitted pro hac vice)

Texas Bar No. 00793258

danaipakos@aatriallaw.com

Michael McBride (admitted pro hac vice)

Texas Bar No. 24065700

mmcbride@aatriallaw.com

Steven Jugle (admitted pro hac vice)

Texas Bar No. 24083280

sjuggle@aatriallaw.com

ALAVI & ANAIPAKOS PLLC

609 Main Street, Suite 3200

Houston, Texas 77002

Telephone: (713) 751-2362

Facsimile: (713) 751-2341

*Counsel for Defendants Stingray Group Inc.,
Stingray Music USA, Inc., Millicom
International Services LLC, and Blue Stream
Communications LLC*

/s/ Harvey W. Gurland, Jr.

Harvey W. Gurland, Jr. (FBN 284033)

hwgurland@duanemorris.com

DUANE MORRIS LLP

201 S. Biscayne Boulevard, Ste. 3400

Miami, Florida 33131
Telephone: 305.960.2214
Facsimile: 305.397.1874

Matthew S. Yungwirth
msyungwirth@duanemorris.com
(admitted pro hac vice)
Alice E. Snedeker
aesnedeker@duanemorris.com
(admitted pro hac vice)

DUANE MORRIS LLP
1075 Peachtree Street NE, Ste. 1700
Atlanta, Georgia 30309
Telephone: 404.253.6900
Facsimile: 404.253.6901

*Counsel for Defendant AT&T Enterprises,
LLC*

CERTIFICATE OF SERVICE

The undersigned hereby certifies that a true and correct copy of the above and foregoing document was filed electronically in compliance with Local Rules 5.2(a) on February 6, 2025. As such, this document was served on all counsel of record pursuant to Local Rules 5.2(a) and the Federal Rules of Civil Procedure.

/s/ Allison Henry
Allison Henry

IN THE UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF FLORIDA
MIAMI DIVISION

CASE No. 1:24-cv-21226-RAR

DR. EDWIN A. HERNANDEZ and
EGLA CORP.,

Plaintiffs,

V.

STINGRAY GROUP INC. f/k/a
STINGRAY DIGITAL GROUP, INC.,
STINGRAY MUSIC USA, INC.,
MOOD MEDIA LLC f/k/a MOOD
MEDIA CORPORATION, AT&T
ENTERPRISES, LLC f/k/a AT&T
CORP., MILLICOM
INTERNATIONAL SERVICES, LLC,
and BLUE STREAM
COMMUNICATIONS, LLC dba
BLUE STREAM FIBER,

Defendants.

JURY TRIAL DEMANDED

DECLARATION OF MICHAEL D. MCBRIDE

I, Michael D. McBride, declare as follows:

1. My name is Michael D. McBride. I am over the age of eighteen, I am of sound mind, and I am fully competent to testify herein. I have personal knowledge of the facts set forth herein, or if so stated, I am informed and believe in their truth and accuracy and, if called to testify, I could and would do so competently and under oath.
2. I am an attorney at the law firm of Alavi & Anaipakos, PLLC and I am one of the attorneys representing Defendants Stingray Group Inc., Stingray Music USA, Inc., Millicom International Services, LLC., and Blue Stream Communications, LLC in the above

captioned case. I am submitting this declaration in support of Patent Defendants' Opening Claim Construction Brief.

3. Attached as **Exhibit A** is a true and correct copy of U.S. Patent No. 10,123,074, which is asserted by Plaintiffs in this case.
4. Attached as **Exhibit B** is a true and correct copy of U.S. Patent No. 10,524,002, which is asserted by Plaintiffs in this case.
5. Attached as **Exhibit C** is a true and correct copy of U.S. Patent No. 11,140,441, which is asserted by Plaintiffs in this case.
6. Attached as **Exhibit D** is a true and correct copy of the declaration of Dr. Sam Malek, which is offered in support of Patent Defendants' Opening Claim Construction Brief.
7. Attached as **Exhibit E** is a true and correct copy of an excerpt from the Prosecution History of U.S. Patent No. 10,123,074, showing Applicant's Amendment to the Claims and accompanying Remarks.
8. Attached as **Exhibit F** is a true and correct copy of an excerpt from the Prosecution History of U.S. Patent No. 10,123,074, showing Examiner's Notice of Allowability.
9. Attached as **Exhibit G** is a true and correct copy of a licensing presentation authored by Plaintiffs, titled "Licensing Opportunity," which is publicly available at <https://eglacorp.com/wp-content/uploads/2019/09/Licensing-Technologies-Presentation-small.pdf>. The presentation was downloaded at my direction on or about February 5, 2025.
10. Attached as **Exhibit H** is a true and correct copy of a page from Plaintiffs' website, titled "Mevia Patent Portfolio," which is publicly available at <https://eglacorp.com/mevia-patent-portfolio>. The page was downloaded at my direction on or about February 5, 2025.
11. Attached as **Exhibit I** is a true and correct copy of an article titled "Automated Accessibility Analysis of Dynamic Content Changes on Mobile Apps" co-authored by Dr. Sam Malek.
12. Attached as **Exhibit J** is a true and correct copy of a technical data sheet for the H-AV-MPEG2-E hardware encoder from Thor Broadcast, which is available at Thor Broadcast's website: www.thorbroadcast.com.
13. Attached as **Exhibit K** is a true and correct copy of the American Heritage Dictionary's entries for the terms "sequence" and "series," which are available at American Heritage Dictionary's website: <https://ahdictionary.com>.

14. Attached as **Exhibit L** is a true and correct copy of the Microsoft Press Computer Dictionary's entries for the terms "web browser" and "web page."

I declare under penalty of perjury that the foregoing is true and correct.

Executed in Houston, Texas on February 6, 2025.

/s/ Michael D. McBride

Michael D. McBride

Exhibit A



US010123074B2

(12) **United States Patent**
Hernandez-Mondragon

(10) **Patent No.:** **US 10,123,074 B2**

(45) **Date of Patent:** **Nov. 6, 2018**

(54) **METHOD, SYSTEM, AND APPARATUS FOR MULTIMEDIA CONTENT DELIVERY TO CABLE TV AND SATELLITE OPERATORS**

(58) **Field of Classification Search**

CPC H04N 21/440218; H04N 21/233; H04N 21/234309; H04N 21/2387;

(Continued)

(71) Applicant: **Edwin A. Hernandez-Mondragon**,
Coral Springs, FL (US)

(56)

References Cited

(72) Inventor: **Edwin A. Hernandez-Mondragon**,
Coral Springs, FL (US)

U.S. PATENT DOCUMENTS

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

6,886,178 B1 4/2005 Mao et al.
7,020,888 B2 3/2006 Reynolds et al.

(Continued)

(21) Appl. No.: **15/538,911**

OTHER PUBLICATIONS

(22) PCT Filed: **Dec. 22, 2015**

Justin Pot: "Save Videos From Any Site—Even Netflix—With Applian's Replay Capture Suite", Jun. 3, 2014 (Jun. 3, 2014), XP055472253, Retrieved from the Internet: URL:https://www.makeuseof.com/tag/savevideos- from-any-site-even-netflix-with-applians-replay-capture-suite/ [retrieved on May 3, 2018].

(86) PCT No.: **PCT/US2015/067464**

§ 371 (c)(1),

(2) Date: **Jun. 22, 2017**

(Continued)

(87) PCT Pub. No.: **WO2016/106360**

PCT Pub. Date: **Jun. 30, 2016**

Primary Examiner — Hunter B Lonsberry

(74) *Attorney, Agent, or Firm* — Fox Rothschild LLP

(65) **Prior Publication Data**

US 2017/0374417 A1 Dec. 28, 2017

Related U.S. Application Data

(60) Provisional application No. 62/095,504, filed on Dec. 22, 2014.

(51) **Int. Cl.**

H04N 21/4402 (2011.01)

H04N 21/2343 (2011.01)

(Continued)

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

CPC ... **H04N 21/440218** (2013.01); **H04N 21/233** (2013.01); **H04N 21/2387** (2013.01);

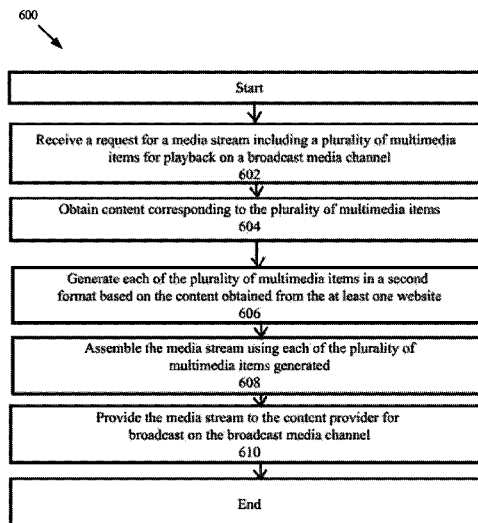
(Continued)

(57)

ABSTRACT

Systems, methods, and computer-readable media for delivering multimedia content from the cloud to cable operators are disclosed. A device located at the cable headend or implemented in the cloud can receive a request for at least one media stream for playback on a broadcast media channel. Content corresponding to a plurality of multimedia files in the media stream can be obtained from the internet or a cloud based service. The content can be used to generate the multimedia files in a format that is compatible with the cable operator. The multimedia files can be used to assemble the at least one media stream which can be provided to the cable operator for broadcast on the broadcast media channel.

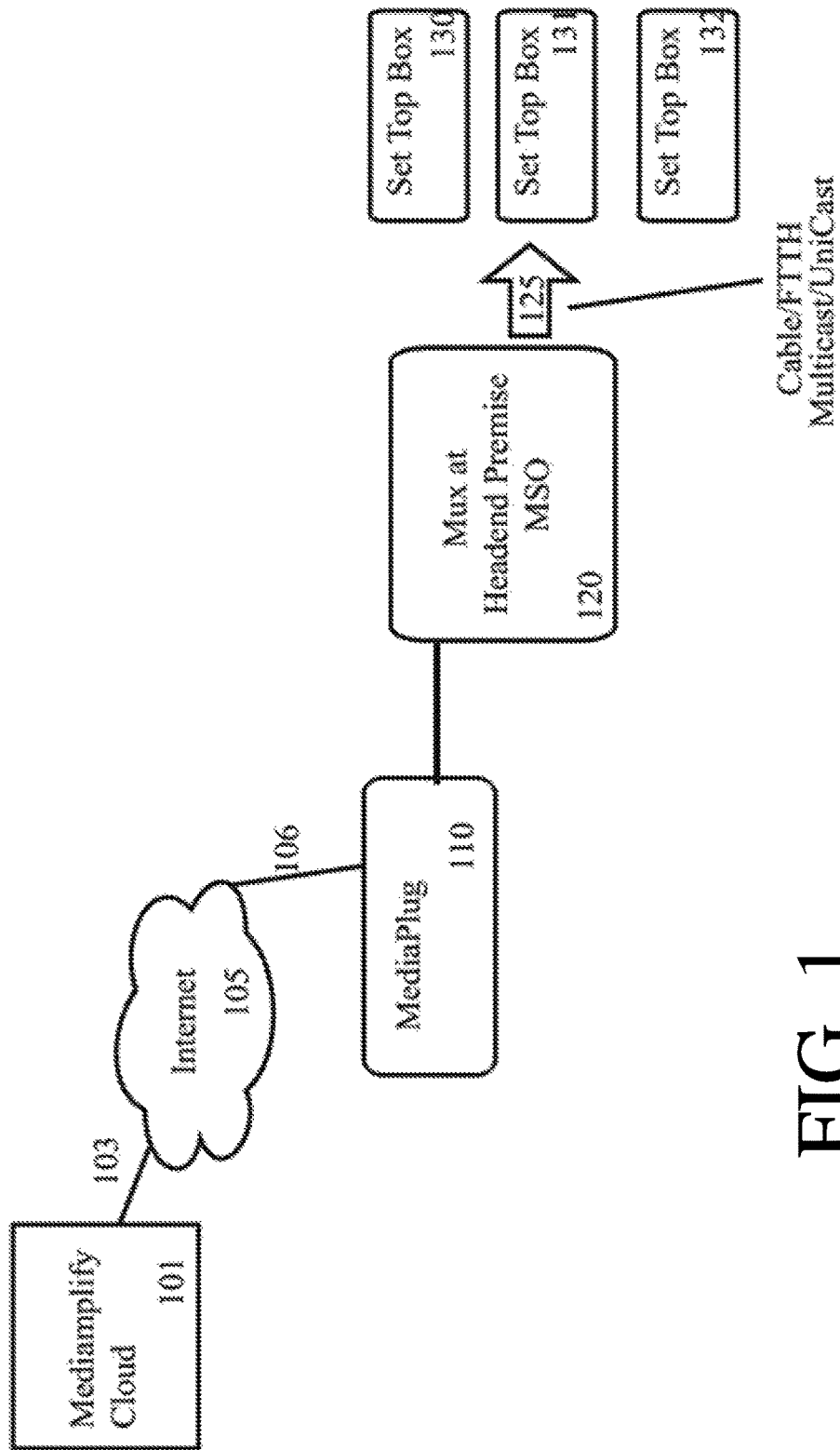
21 Claims, 12 Drawing Sheets



US 10,123,074 B2

Page 2

-
- (51) **Int. Cl.**
- | | | | | | |
|---------------------|-----------|------------------|--------|-----------------|------------------------|
| <i>H04N 21/84</i> | (2011.01) | 2002/0138852 A1 | 9/2002 | Reynolds et al. | |
| <i>H04N 21/81</i> | (2011.01) | 2003/0115612 A1 | 6/2003 | Mao et al. | |
| <i>H04N 21/2387</i> | (2011.01) | 2008/0040354 A1 | 2/2008 | Ray et al. | |
| <i>H04N 21/472</i> | (2011.01) | 2008/0098426 A1* | 4/2008 | Candelore | H04N 5/4403
725/38 |
| <i>H04N 21/233</i> | (2011.01) | 2009/0178003 A1 | 7/2009 | Fiedler | |
| <i>H04N 21/61</i> | (2011.01) | 2010/0192178 A1* | 7/2010 | Candelore | G06K 9/325
725/39 |
| <i>H04N 21/431</i> | (2011.01) | 2012/0204217 A1 | 8/2012 | Regis et al. | |
| <i>H04N 21/6405</i> | (2011.01) | 2014/0068690 A1 | 3/2014 | Luthra et al. | |
| <i>H04N 21/643</i> | (2011.01) | 2014/0098815 A1* | 4/2014 | Mishra | H04L 12/185
370/390 |
- (52) **U.S. Cl.**
- CPC . *H04N 21/234309* (2013.01); *H04N 21/4312* (2013.01); *H04N 21/47217* (2013.01); *H04N 21/6118* (2013.01); *H04N 21/6143* (2013.01); *H04N 21/8106* (2013.01); *H04N 21/8133* (2013.01); *H04N 21/8153* (2013.01); *H04N 21/84* (2013.01); *H04N 21/6405* (2013.01); *H04N 21/64322* (2013.01)
- (58) **Field of Classification Search**
- CPC H04N 21/4312; H04N 21/47217; H04N 21/6118; H04N 21/6143; H04N 21/8106; H04N 21/8133; H04N 21/8153; H04N 21/08
- See application file for complete search history.
- (56) **References Cited**
- U.S. PATENT DOCUMENTS
- | | | | |
|--------------|--------|--------------------|--|
| 7,320,025 B1 | 1/2008 | Steinberg et al. | |
| 8,769,602 B1 | 7/2014 | Del Beccaro et al. | |
- OTHER PUBLICATIONS
- Fawad Mir: "Schedule Auto Song Recording From Multiple Radio Streams With streamWriter", Jan. 9, 2013 (Jan. 9, 2013), XP055472533, Retrieved from the Internet: URL:https://www.addictivetips.com/windows-tips/schedule-auto-songrecording-from-multiple-radio-streams-streamwriter/ [retrieved on May 4, 2018].
- Chris Robley, How to make album art videos for your songs on YouTube, Jul. 16, 2014.
- International Search Report dated Apr. 20, 2016 for PCT/US15/67464.
- * cited by examiner



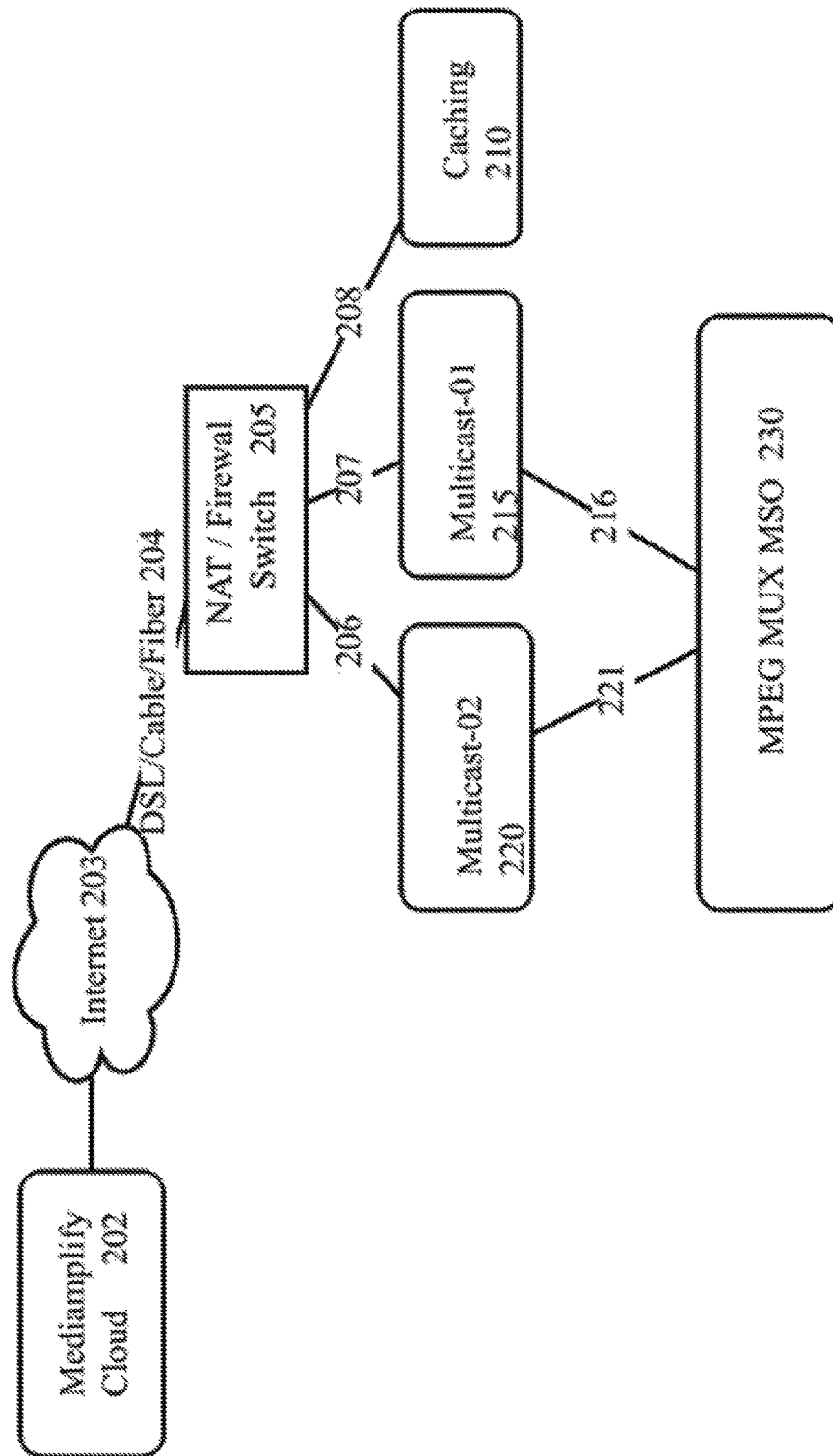


FIG. 2

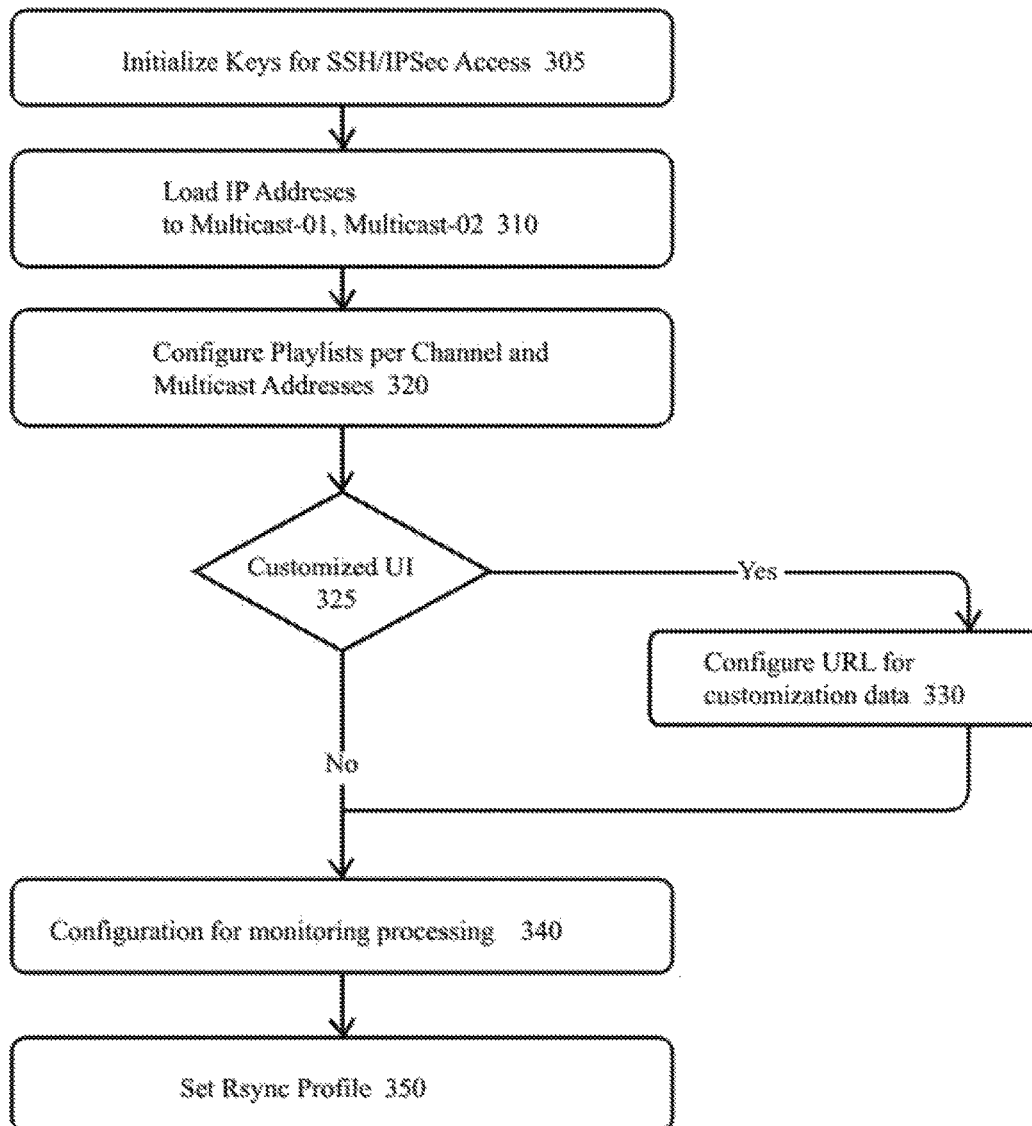


FIG. 3

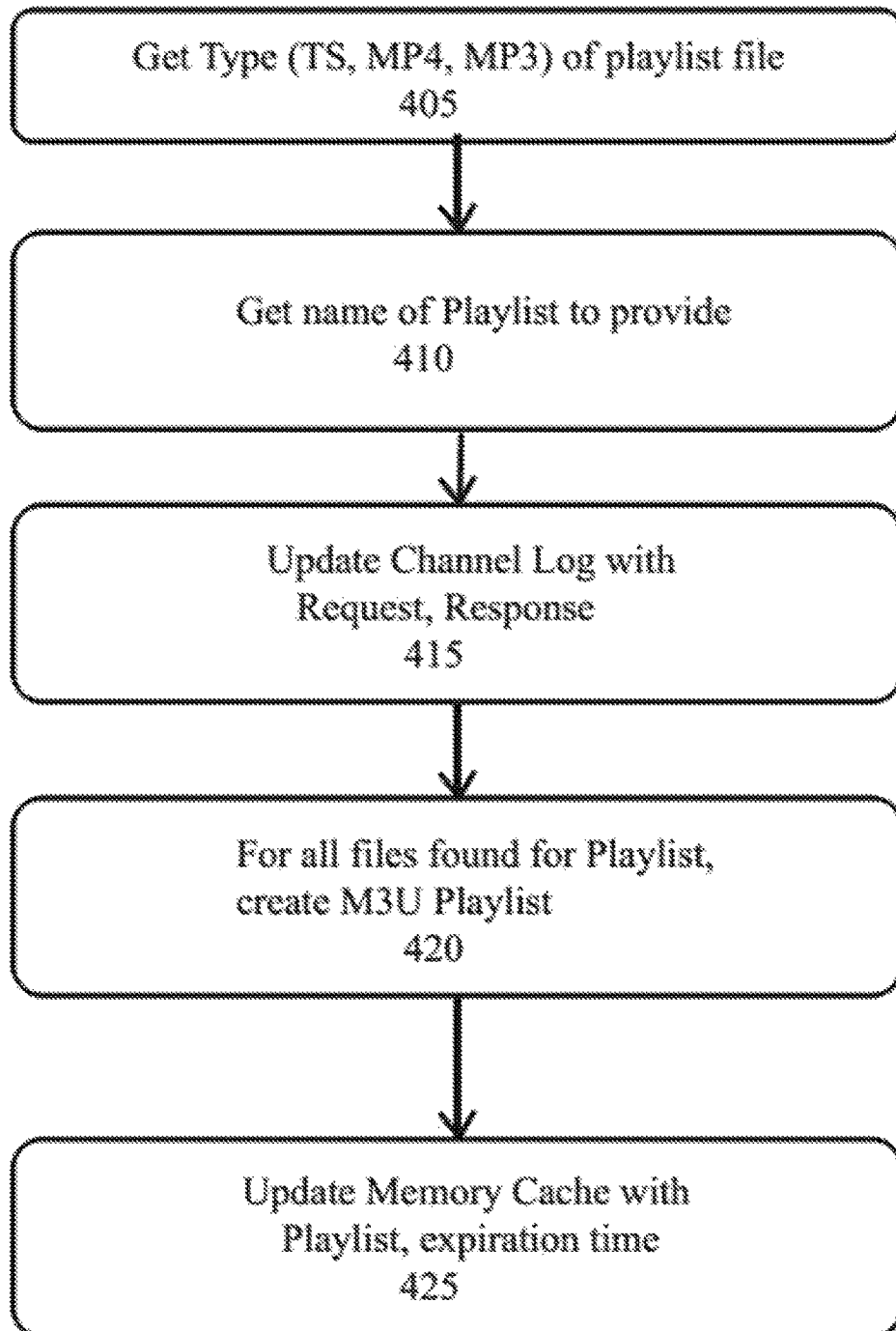


FIG. 4

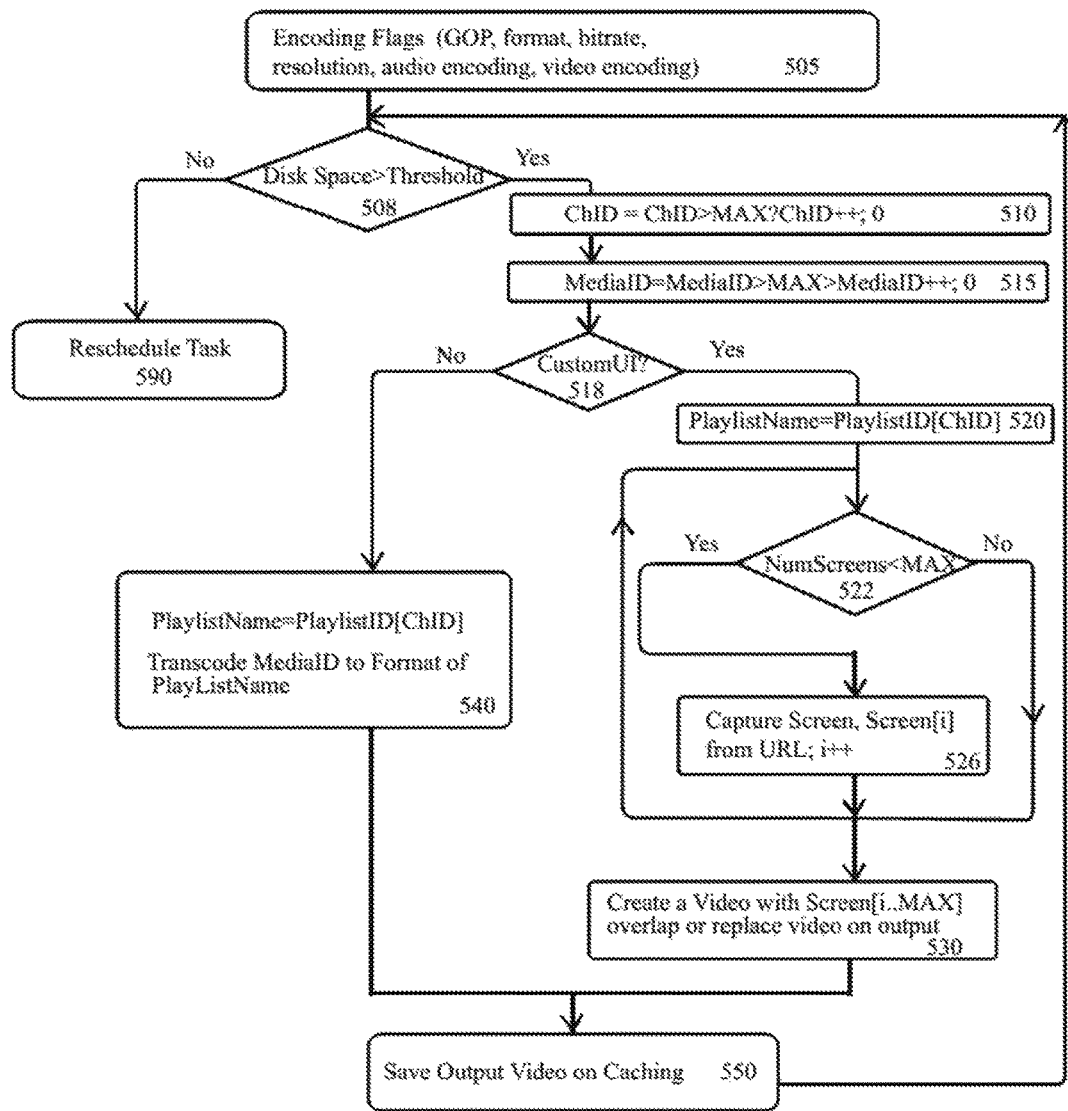


FIG. 5

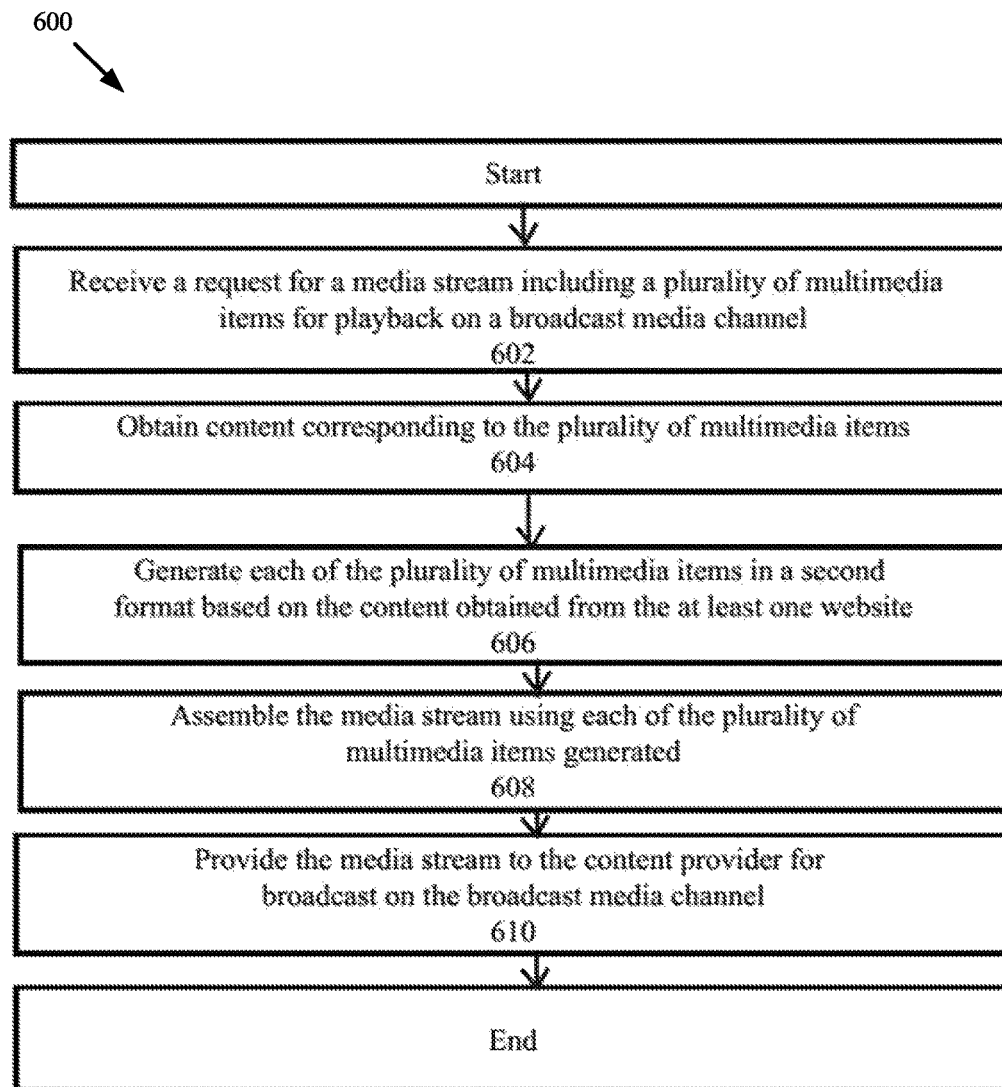


FIG. 6

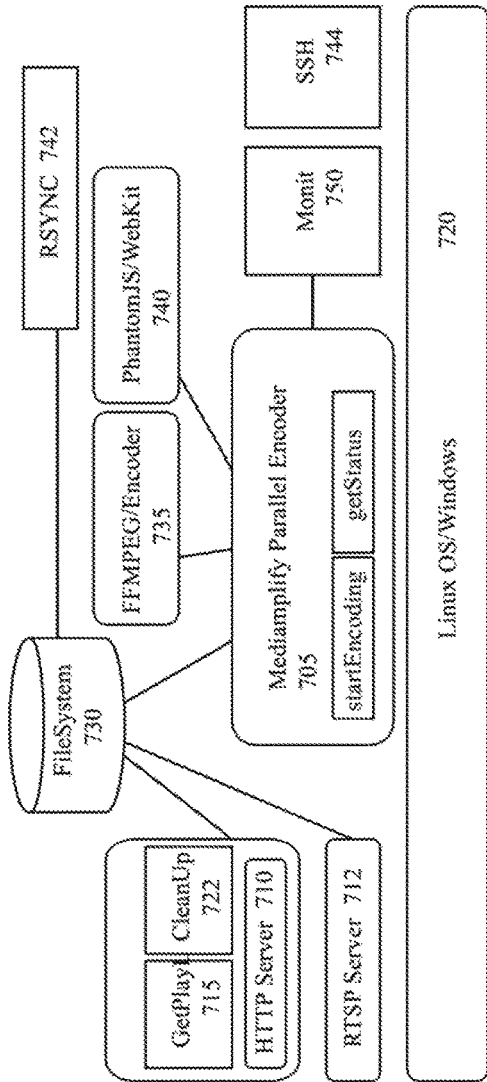
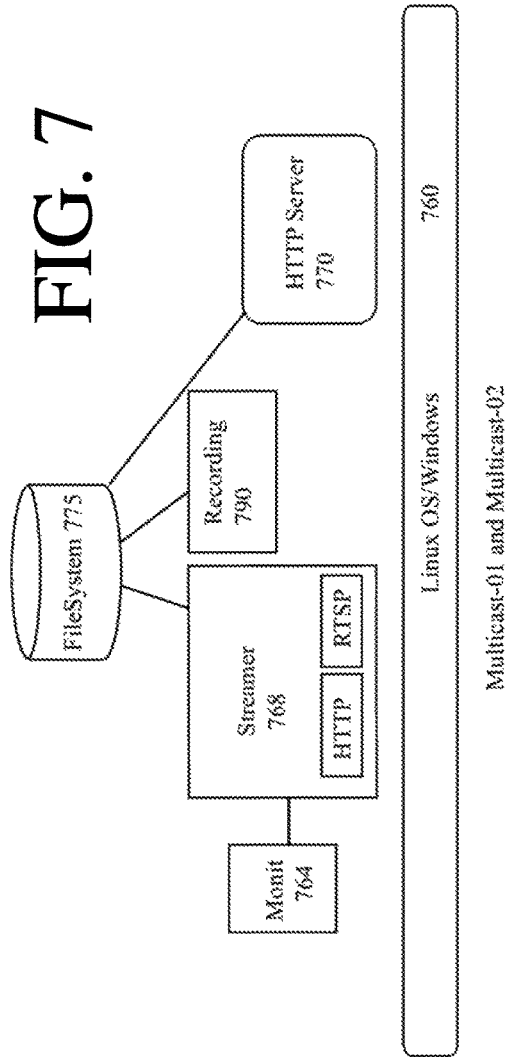


FIG. 7



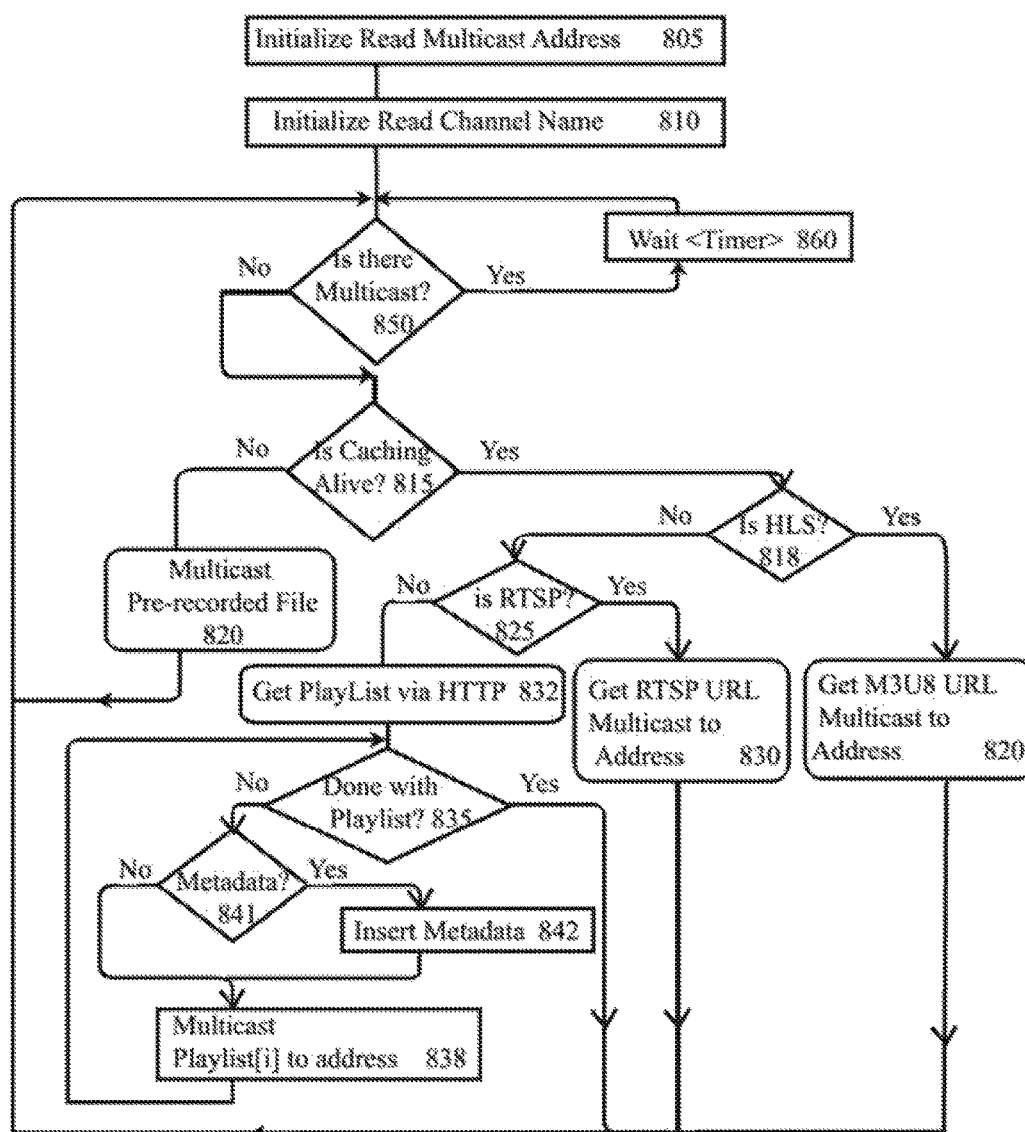


FIG. 8

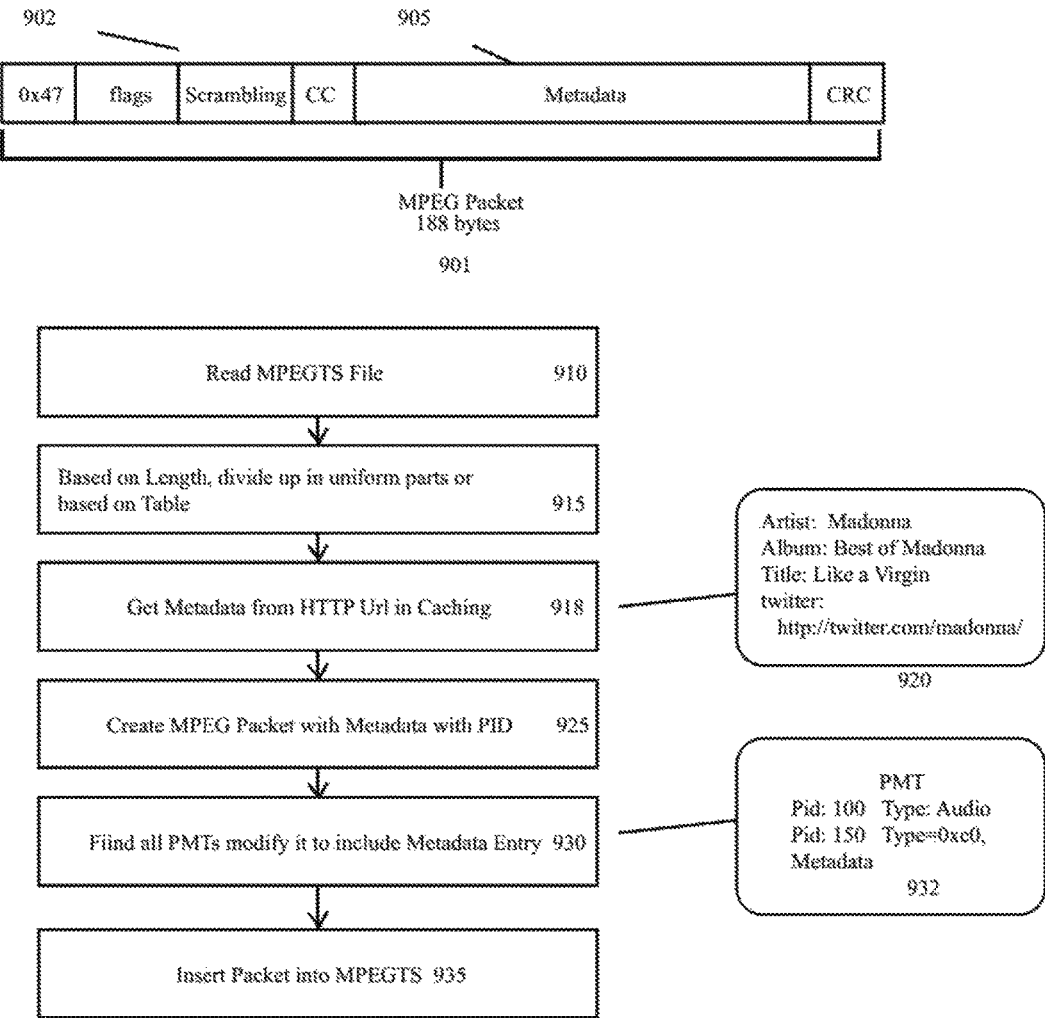


FIG. 9

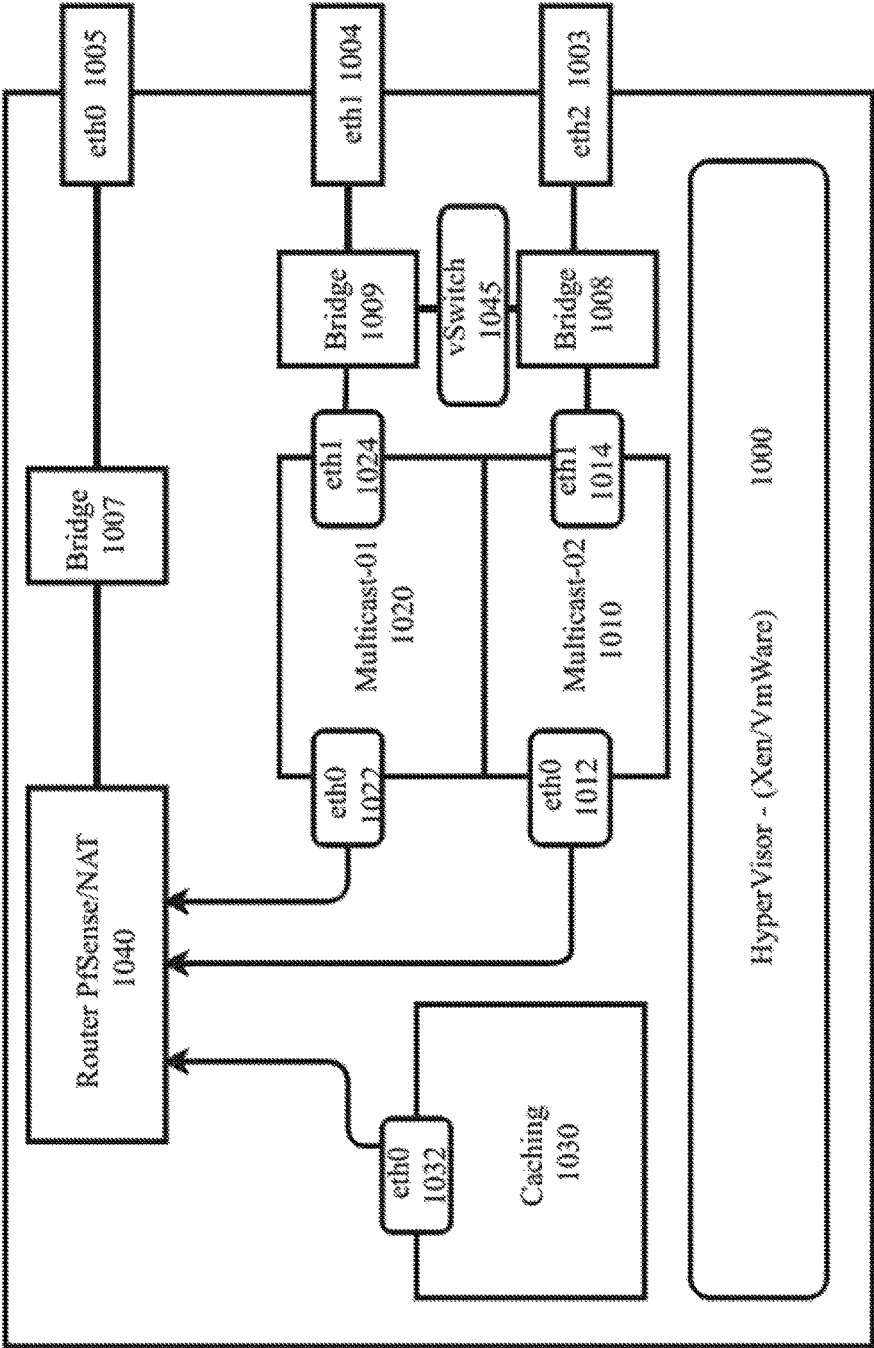


FIG. 10

U.S. Patent

Nov. 6, 2018

Sheet 11 of 12

US 10,123,074 B2

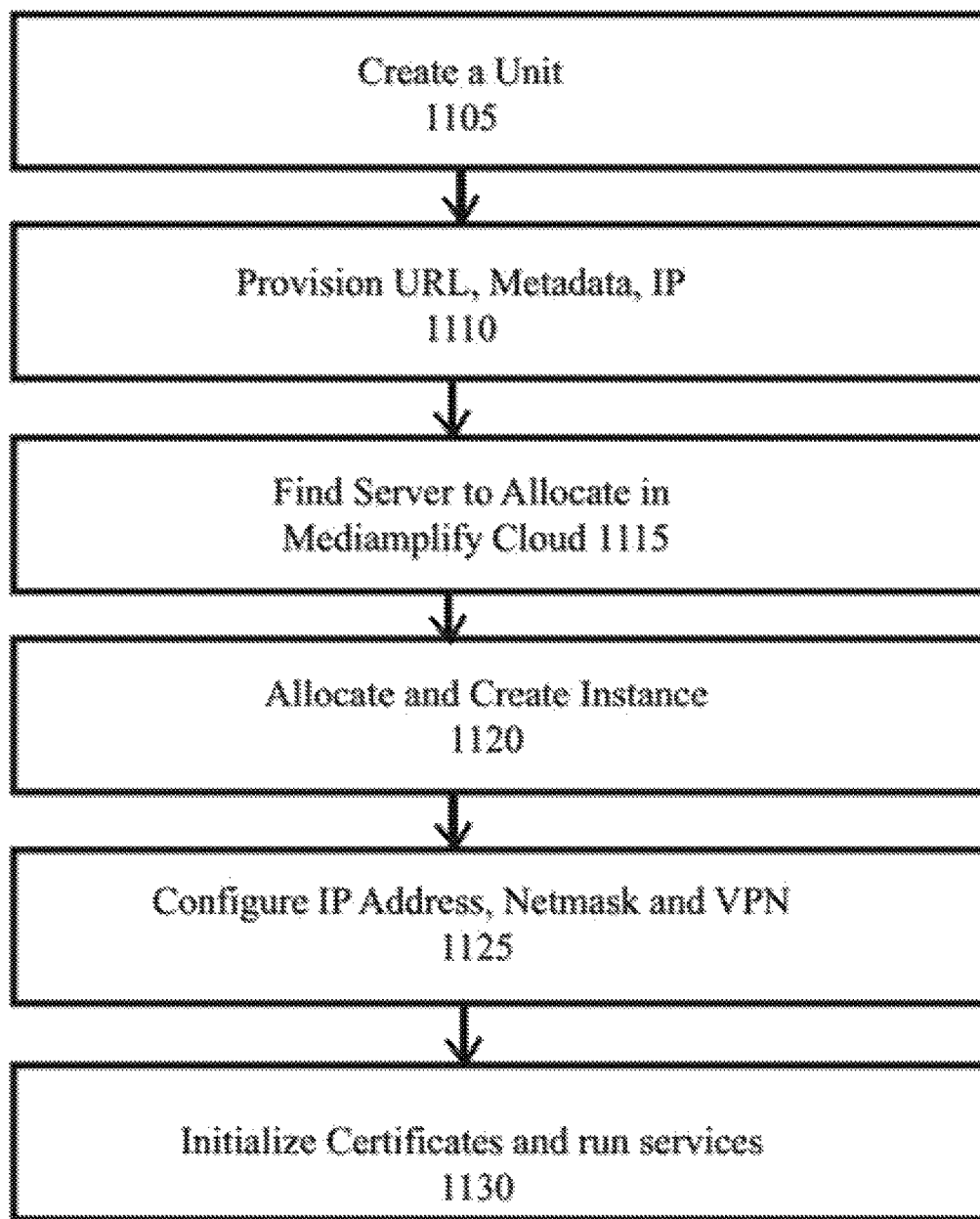
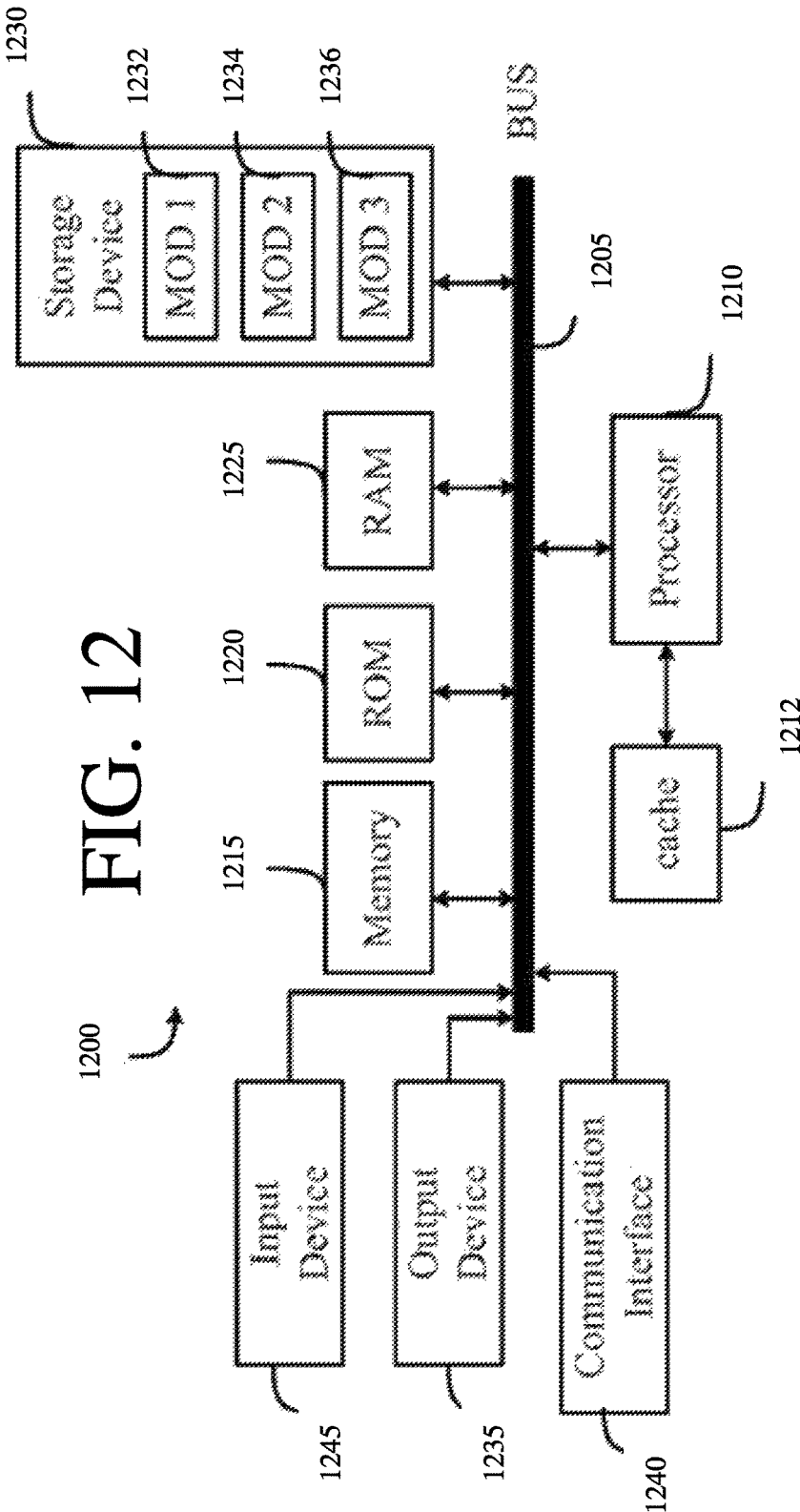


FIG. 11



US 10,123,074 B2

1

**METHOD, SYSTEM, AND APPARATUS FOR
MULTIMEDIA CONTENT DELIVERY TO
CABLE TV AND SATELLITE OPERATORS****CROSS-REFERENCE TO RELATED
APPLICATION**

This application claims priority to International Patent Application No. PCT/US2015/067464, filed on Dec. 22, 2015, and U.S. Provisional Application 62/095,504, filed Dec. 22, 2014, the complete disclosures of which are incorporated herein by reference in their entirety.

FIELD OF THE DISCLOSURE

The present disclosure provides a mechanism for delivering multimedia content from a cloud-based system to cable or satellite content providers.

The disclosure describes a novel method and a system for continuous media delivery and secured synchronization of multimedia files that are broadcasted to set top boxes or subscriber units in the field that are connected to IPTV or Cable Systems. The present technology can securely connect to a cloud-based platform (e.g. OpenStack, Amazon EC2, Mediampify Cloud, etc.), creates a set of multimedia files (e.g. MPEG4 movies) into a secured repository (e.g. Secured File System, Linux/Windows, etc.), modifies those files according to certain rules as defined by the method herein, and reproduces those signals via a multicast video stream (e.g. User Datagram Protocol (UDP) multicast, etc.) encapsulated in a Transport Stream (TS) frame that can be a multiple transport stream (several transports in one TS frame) or a single transport stream (one transport stream according to the MPEG-2 Part 1 Specification or similar multimedia streaming that is compatible with MPEG-TS at a predetermined rate or at a variable bit rate).

BACKGROUND

Cable operators, Multiple System Operators (MSO), and Digital Satellite Systems (DSS) deliver media content by utilizing satellite, Radio Frequency (RF) and Integrated Receiver and Decoder (IRD) technologies, which provide outputs that can include Asynchronous Serial Interface (ASI) signal formats. Additionally, MSOs and DSSs use dedicated hardware encoders and transcoding systems if the source video/audio is incompatible with the set top boxes of the cable or satellite operator. In general, the video/audio content delivered via the satellite to the cable or satellite operator arrives via wireless link or receiver, which is processed by an IRD that receives the stream from the satellite and can ultimately broadcast that signal to the cable operator head end in several formats. In addition, some systems may include multicast units connected with IRDs that tune to the right transponder system.

Satellite systems are used to deliver data (e.g. TV, radio, Video-on-Demand, multimedia streams) from a particular source (e.g. third-party content provider) to cable head end systems or other satellite systems. Satellite delivery is performed via a broadcast signal containing digital pictures and audio inside a transponder frequency. Transponders are radio frequency (RF) space allocated in a particular satellite.

For example, a third-party content provider can maintain multimedia files in CD/DVDs, pre-recorded storage, or as pre-recorded files that are reproduced in a particular audio or video encoding format using a video or audio encoder (e.g. MP3, H.264, AC3, and others). The audio and video is then

2

made part of an RF signal that is broadcasted to the satellite system. This signal is then received by the cable or satellite operator using an antenna receptor and is decoded to generate either an analog signal (e.g. RCA Cables or any other analog interface) or digital signal (e.g. ASI, Asynchronous Serial Interface). If an analog signal is received, modern DVB (Digital Video Broadcasting) Cable and Satellite systems will encode this signal to their particular video/audio encoding requirements such that it can be broadcasted in digital packets via a multicast MPEG Transport Stream. The same occurs if a receiver decodes the signal in ASI. As such, this signal is converted from ASI to Ethernet, or simply redirected to a multicast MPEG Transport stream if arriving in a compatible encoding format for the receiving head end cable system.

When no satellite is being used, video, pictures and music, are read out from the DVD or a hard-drive containing MPEG-layer 3 or MPEG-layer 2 files that were encoded at a pre-determined bit rate according. Those video feeds can be reproduced at a synchronized timing using a multicast MPEG Transport Stream using RTP (Real-time Transport Protocol)-based multicast communication and RTCP (RTP Control Protocol)-based mechanisms, as well as standard UDP (User Datagram Protocol) multicast streams with proprietary timing.

Additionally, IPTV systems can provide mechanisms to broadcast a signal using the multicast protocol in MPEG transport streams to a customer premises with compatible set top boxes (same encoder, same streams, encryption keys, etc). A compatible set top box is such that when connected to the cable TV system it can properly decode video/audio from the cable operator's head end system. Some of these IPTV systems rely on CDN (Content Delivery Networks) that enable distribution of multimedia files to IP-enabled devices. For example, a mechanism can be provided whereby a head end content (MSO) that is being broadcasted to a cable subscriber is also available to an IP-enabled device such as a smartphone, PC, or tablet computer.

SUMMARY

Disclosed are systems, devices, methods, and non-transitory computer-readable media for media delivery from a cloud or IP-based distribution network, or CDN (Content Delivery Network) to MSO (Multiple System Operators) or head-ends that include cable and satellite delivery mechanisms. The present technology unifies cloud-based delivery (Mediampify cloud) with the cable-based mechanism (e.g. Comcast, Verizon FiOS).

An example computer-implemented method can receive, from a content provider such as a cable or satellite operator, a request for at least one media stream for playback on a broadcast media channel. The request for the at least one media stream can include a request for a plurality of multimedia items. Content corresponding to the plurality of multimedia items can be obtained from at least one website or cloud service that offers the content in at least one first format. Based on the obtained content, the plurality of multimedia files can be generated in a second format that is compatible with the content provider. The media stream can be assembled using each of the multimedia items generated in the second format. The media stream can be provided to the content provider for broadcast on the broadcast media channel.

In some aspects of the present technology, obtaining the content can include retrieving at least one audio file corresponding to an audio component of the multimedia item and

US 10,123,074 B2

3

retrieving a plurality of images as screen captures that can correspond to a video component of the multimedia item. Generating the multimedia files in the second format can include combining the plurality of images and the at least one audio file to create each of the multimedia items.

BRIEF DESCRIPTION OF THE DRAWING

There are shown in the drawings, embodiments which are presently preferred, it being understood, however, that the disclosure is not limited to the precise arrangements and instrumentalities shown

FIG. 1 shows an example block diagram illustrating an exemplary head end (MSO);

FIG. 2 shows an example block diagram of the apparatus located at the MSO facility (mediaplug);

FIG. 3 shows an example method for initialization of Mediaplug components;

FIG. 4 shows an example method to obtain list of files to multicast to a cable operator MSO;

FIG. 5 shows an example method for encoding video and audio files;

FIG. 6 shows an example method for providing multimedia content to a cable operator;

FIG. 7 shows an example block diagram of example software components of the Mediaplug;

FIG. 8 shows an example method to multicast from a component in the Mediaplug;

FIG. 9 shows an example method to insert metadata into an MPEG Transport Stream;

FIG. 10 shows an example block diagram for a Virtualized Mediaplug

FIG. 11 shows an example method to initialize virtual mediaplug components; and

FIG. 12 shows an example possible system embodiment for implementing various embodiments of the present technology

While the present technology is amenable to various modifications and alternative forms, specifics thereof have been shown by way of example in the drawings and will be described in detail. It should be understood, however, that the intention is not to limit the present technology to the particular embodiments described. On the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present technology.

DETAILED DESCRIPTION

The following description should be read with reference of the drawings, in which like elements are numbered in similar fashion. The drawings are not necessarily at scale and are not intended to limit the scope of the disclosure. Those skilled in the art understand that there are alternatives that may be used as well which are also considered to be within the scope of the disclosure.

Generally, the disclosed technology pertains to a delivery mechanism for providing multimedia content to cable systems without the use of a satellite system in a fault-tolerant and reliable fashion. FIG. 1 illustrates a block diagram that includes a Mediaplugin Cloud 101. The cloud 101 can refer to a set of virtualized instances or non-virtualized instances of one or more servers that provide access to archives and digital files with media content stored in them. For example, the cloud 101 can be implemented using VMWave or Xen and OpenStack, or simply reside in Amazon Web Services or

4

Rackspace. In addition, special provisioning can be provided in cloud 101 for secured access.

The cloud 101 service files can be accessible via the internet 105 and can be stored in a cloud-based file system or cluster file system (e.g. GlusterFS or Hadoop, Amazon S3). These files can be multimedia files that include, as non-limiting examples, MPEG-4 Video files with H.264 encoded video and AAC encoded audio or MPEG-layer 3 audio as well (MPEG2 Video could also be used). These files can be accessed using a secured transport protocol such as, but not limited to, SSL or a synchronization tool as “rsync,” commonly found in Linux/MacOS environments. The communication link 106 can be a DSL, Cable Modem, Wireless WAN, Fiber, Metro Ethernet, etc. and can be linked with the Mediaplugin 110 unit. The Mediaplugin 110 can reside at the “MSO” facility and can be collocated with all the cable or satellite operator equipment. The MSO is a facility used by the Cable Operator to receive all TV and Radio Programming that the cable operator will then multiplex, re-encode, and distribute from the MSO to a set of subscriber units with set top boxes, or modulate the signal in TV analog channels (2-60+).

One function of the Mediaplugin 110 is to provide multimedia content, e.g., video and audio, to an MPEG multiplexer at the headend 120. The content can be pre-formatted in MPEG Transport Streams that the multiplexer 120 (e.g. Arris TMX-2010) will then be able to capture, reformat in any way the operator requires, and deliver to the cable operator’s subscribers via Coaxial Cable 125 or any other means suitable for a multicast or unicast delivery to its customers with “Set Top boxes” 130, 131, 132. A “Set Top Box” is customer premise equipment that will be able to decode and display in a TV set or other screen display, via a signal received by its HDMI or analog input, the contents of the TV programming the cable operator offers.

The Mediaplugin 110 can be a computer or server or set of computers or servers that can communicate with the Cloud 101 using a standard Local Area Network (e.g. Gigabit Ethernet). The Mediaplugin 110 can store, encode, and transmit multimedia files in the format that the cable operator at its MSO requires. This cable format may include audio, video, and metadata (subtitles, control data, and any additional component that can be packaged in an MPEG Packet that is part of the MPEG Transport Stream). In one embodiment, Mediaplugin 110 can include components such as a caching unit, and one or more multicasting servers.

FIG. 2 illustrates an example configuration of the hardware components of Mediaplugin 110 which includes three servers. A server is a computer device, usually rack mountable unit with volatile memory such as RAM, one or more micro-processors, and non-volatile memory such as a hard disk. However, one skilled in the art will recognize that the present technology is not limited to a particular configuration and can be varied according to different specifications. FIG. 2 illustrates Mediaplugin Cloud 202 which is connected to the Internet backbone 203. Link 204 (e.g. DSL, Cable Modem, Fiber) provides connectivity between cloud 202 and a Network Address Translator unit 205 (e.g. Cisco RV54000, or any other NAT device) which can provide Gigabit Ethernet switching capabilities. The lines 206, 207, and 208 show Gigabit Ethernet links, however one skilled in the art will understand that Gigabit Ethernet or a Wireless LAN using 802.11g/n would be also a suitable interface to be used in a Mediaplugin configuration. The Caching unit 210 can be the main storage component in the system and it can contain all of the multimedia files in the format required by the cable operators, such as the Transport Stream format.

US 10,123,074 B2

5

The caching unit can communicate with Multicast-01 215 and Multicast-02 220. The two Multicast units 215, 220 can pull or obtain information from the caching unit via TCP/IP or UDP, using protocols such as HTTP (Hypertext Transport Protocol) or RTSP (Real-time Streaming Transport). The caching unit can also encode and transcode the multimedia files it downloads from the Cloud 202. The caching unit can also keep updated information of all the metadata and create customizable screens that can be converted into video formats compatible with the Transport Stream content that will be provided to the cable operator at its MSO. MPEG MUX MSO 230 can correspond to a program residing in a computer system that processes and handles the MPEG streams from multiple sources. The MUX 230 can also function as the bridge between the source of the signal and the customer's set top box units. A non-limiting example program would be the CherryPicker DM6400 by MOTOROLA. Each of the components or modules of the Mediaplugin can include software for performing functions and/or methods according to the present technology.

FIG. 3 illustrates an example method that can be used to configure and provision the units at boot-time. Each unit can initialize keys for SSH/IP Sec access at 305. For example, the units can use IPsec and SSH Tunnel keys that can be stored in the administrator user account. The keys can be generated by the Mediaplugin Cloud that are private keys or certificates, shared secrets or similar components by which the Mediaplugin can connect securely with the Mediaplugin cloud. Each cable operator can provide custom-based IP addressing at 310 for the Gigabit Ethernet ports that connect to the operators' network. Changing the IP addresses of the Multicast-01 and/or Multicast-02 refers to the IP address by which the links 221 and 216 in FIG. 2 are linked to the MPEG MUX MSO unit. A change of the IP address can refer to the 'ifconfig' command which is used in Linux-based systems and that can be changed in an Ubuntu box at/etc/network/interfaces file. Similarly, the network routing can be changed if the multicast address cannot be routed to the default gateway which points to the 205 NAT, but to 221 and 216 respectively for Multicast-02 and Multicast-01. In general, the 239.*.* network address may be routed to those interfaces.

At step 320, the Mediaplugin can configure playlists per channel and multicast address. The playlist and channel configuration can be obtained from the cable/satellite operator and can include any number of channels for broadcasting multimedia content to end customers. As an example, the channels may include music channels corresponding to different music genres, e.g. rock, alternative, classical, jazz, hip hop, etc. Based on the particular genre, a corresponding playlist can be selected and/or configured. The Mediaplugin can also determine the multicast address for each of the channels.

In some configurations a customizable user interface (UI) can be added to the output video MPEG Transport stream. As such, if the "Custom UI" variable is then true and valid, 325, a link or URL 330 is loaded into the Mediaplugin configuration that can be used when the video interface is created. The custom video interface can be created and fetched as needed or it can be created in real-time for a particular channel or for all the channels. The custom user interface variable can be uniquely configured per channel in the Mediaplugin. The mediaplugin can also monitor 340 all of the processes to ensure stability and continuous, uninterrupted operation. The initialization of the Mediaplugin can also include setting the RSYNC profile 350. The RSYNC profile can be used to update and synchronize media content

6

with the cloud when the Mediaplugin determines that the content has changed or been modified.

As described above, the Mediaplugin can broadcast the multimedia MPEG stream using the programmed IP Address in multicast-01 and multicast-02 according to the method illustrated in FIG. 3. In FIG. 4, an example method is shown for obtaining the list of files to playback or broadcast to the MSO system. The file type or format requested by the multicasting embodiment, e.g. multicast-01 or multicast-02, can include Transport Stream (e.g. TS) and/or MPEG-4 Part 14 (e.g. MP4) values 405. This request is executed from any of the multicasting embodiments, usually making use of the HTTP GET Request. The multicasting unit can connect to an HTTP Server and retrieve the list of files based on the type identified by the request. As the caching unit may handle multiple channels of different types and name, each channel can have a unique identifier corresponding to the name of the playlist. For example channel name "Cool" or "Rock," 410 are names that can identify the channel to be retrieved from the caching server. The activity log in the request can also be updated keeping track what step of the process is being completed 415. At 420, the caching server can generate a "playlist file" of mime type "audio/x-mpegurl" which will be interpreted as a multimedia playlist file when requested. Finally, as the playlist usually may not vary for a period of time until a new request is received, the playlist can be kept in the local memory caching service 425.

As the requests arrive at the "Caching" unit from the multicasting units, the "Caching" unit can be configured to perform parallel encoding for all the different channels available from the unit. FIG. 5 illustrates an example method that can be implemented by the Mediaplugin, and in particular, by the "Caching" unit. At step 505, encoding flags or input parameters can be received which may include GOP (Group of Pictures) which is used by certain video encoders, resolution (e.g. 640x480) in pixels, bit rate in Kilobits per second, any encoding type of video (e.g. H.264), encoding type of audio (e.g. AC-3), duration of the video feed, etc. In addition, a "format" parameter can be received which refers to the output format or encapsulation, MPEG TS or MPEG4. The caching unit encodes audio and video, and can use the disk space in the Mediaplugin. Hence in 508 the method checks for sufficient space left in the memory or hard drive. This task can be run on a thread together with all the other tasks that are working in parallel in order to speed up the processes of encoding and transcoding.

Several indexes or identifiers by the name "ChID," (these integers preferably no more than an integer value from 1 to the maximum number of channels) begin (e.g. MAX could be 50) to be generated by the caching unit 510. This method can be performed for all the channels and all media linked to that particular channel identified by "MediaID" 515, in this particular case the "MediaID." The variable "MAX" in this context can refer to the maximum number of media files read (e.g. 500). The data captured from the media could be "Artist," "Author," or any other variable from an audio file. If a video file is used this could be "program name," "movie name," or any associated metadata information tied to the multimedia file and the Channel Identifier (ChID) associated with it "MediaID" file.

The caching unit can add a "Custom UI," or a customizable User Interface that is designed using Hypertext Markup Language (e.g. HTML) or a web service. A web service that matches a REST interface (Representation State Transfer) and uses Javascript Object Notation (JSON) or XML (Extensible Markup Language) to respond to a custom UI request can be used. Moreover, this custom UI can

US 10,123,074 B2

7

include JavaScript, CSS, and HTML content files that can be rendered using a browser using WebKit (e.g. PhantomJS or Safari) or any other browser-rendering engine compatible with HTML4/5 or any future HTML version. If a customizable UI is part of the configuration **518**, rendered images from a web service can be used to create overlapping video or replace any video in the video feed file. Likewise, this video or customized screen can be used when no video at all is defined in the multimedia file, e.g. Audio-only file. The web-service to be contacted may use the "PlaylistName" to be used for the Channel identified by the channel ID "ChID," retrieved using the "PlaylistID" function **520**. Each new PlaylistName may contain different identifies to retrieve new images. Therefore one or more screens will be retrieved from the rendering engine as such **522** can be used to retrieve a set of screens from the web-service being contacted.

Capturing screens **526** can be done using a tool similar to PhantomJS (Webkit-based headless engine for rendering images) or QtCapture, or simply capturing screen frame buffers from a machine running any software tool. Alternatively, the method can capture a video file from another rendering function, which can create a sequence of screens that can be used to generate a video file. Once all screens and video are captured, **530** the resulting output created can be a video file that then is either replaced or merged (multiplexed) with the original file. As an example, if an audio content file is used such as an MP3 file, this function can create a video file from all the screens captured by **526**. Those screens are generated by calling a URL where the web-service is associated and creating a video file with the input audio and the captured screens. The output format can be an MPEG Transport Stream file that can be retrievable or streamed to the multicasting embodiment.

Once the output file containing the screen shoot sequence of videos encoded in H.264 (for example, or MPEG2Video) is created, the audio can be multiplexed and added to one of the encoding channels of the resulting video output and saved into the caching unit storage disk **550**. The multiplexing can be done using tools such as FFMPEG or VLC or any proprietary tool that can multiplex audio and video files into a resulting MPEG Transport Stream (or MPEG-4 output, or any similar output).

FIG. 6 illustrates an example method **600** for providing multimedia content to a cable operator. At **602**, a device such as the Mediaplugin device receives a request from a cable/satellite operator for a media stream. A media stream can be formatted according to HTTP Live Streaming (HLS), an HTTP/RTSP playlist, or an RTSP stream. The media stream can include any number of multimedia items such as audio/video multimedia items. In some instances, the request can include a channel identifier that is associated with a particular category of multimedia items, such as a music genre corresponding to rock, classical, jazz, etc.

At **604**, contents corresponding to the multimedia items can be retrieved, downloaded, or otherwise obtained from one or more websites or cloud based servers. The contents can include data corresponding to either or both of the audio and video components of the multimedia items. For example, the data corresponding to the audio portion can include a music file in an MP3 format. Similarly, the data corresponding to the video portion can include a video that is obtained directly from the cloud. Alternatively, the video component can be assembled by using one or more screen captures from a website. The screen captures can include captures of still images presented by the website as well as captures of images from videos that are available on the

8

website. As illustrated in FIG. 5, the screen capture process can be repeated any number of times.

The Mediaplugin can then generate **606** multimedia items in a second format based on the content obtained from the web server or cloud. That is, the Mediaplugin can convert/encode the audio file into a format that is suitable for the cable operator and it can also assemble the various screen captures to generate the video component of the media item. Once the audio and video files are created, they can be combined, multiplexed, or encoded to a format that can be utilized by the multimedia stream. For example, the audio can be multiplexed and added to one of the encoding channels of the video output. The various multimedia items can be used **608** to assemble the media stream in a format corresponding with the request (e.g. HLS, HTTP/RTSP, RTSP stream). The media stream can be provided **610** to the content provider for broadcast on the broadcast media channel.

FIG. 7 illustrates a block diagram of an example architecture of the Caching Unit components. The figure shows **720** as the main operating system, which can be Linux, Windows, or any other operating system in use. The caching unit can contain two types of operating system services HTTP **710** and RTSP **712**. The RTSP service provides access so the media saved or created in the repository of the caching unit file system **730**. The RTSP server can be implemented using the Live555 RTSP server or similar. Alternatively, the HTTP Server can be implemented using "nginx" or "apache" web browsers, and the RTSP Servers can be implemented using Live555 services. In the case of the HTTP Server, this can also be used to get the Playlist of files generated from the method on FIG. 5 **715**.

The GetPlaylist **715** command can be implemented as follows:

- Receive and process a request for the name of the channel or playlist to be used,
- Retrieve all the filenames ordered by date of creation of all the files in a particular directory,
- Cache all the files into cache memory (e.g. Memcache or similar tool),
- Add type of HTTP header as M3U or PLS (e.g. a text file with a list of files or URLs to play),
- Return a list of HTTP or RTSP URLs to access each video generated or stored in the caching unit.

In some embodiments, the HTTP Server can also be used to clean up, update, and delete operations. The block unit called "CleanUp" **722** is in charge of keeping up to date caching information data and/or deleting files already used or that are not required to be used anymore by the multicast unit(s). Also the HTTP Server functions can include servicing M3U8 files corresponding to the HTTP Live Streaming format and all the segments created by the HTTP Live Streaming function. As such, a HTTP Live Streaming (HLS) generation program will produce the files and segment them as needed by the encoder and can then be serviced by the HTTP Server **710**. The Mediaplugin Parallel Encoder **705** corresponds to a block that can implement the method shown in FIG. 5. The resulting outputs from the Media Parallel Encoder **605** are stored in the file system **730**. The Mediaplugin Parallel encoder **605** can be implemented in Python and can rely on web services (such as Twisted or Django), PhantomJS, and FFMPEG. Monit **750** corresponds to a monitoring block that can ensure that the process running the "Media Parallel encoder" keeps running at all times and does not stop upon failure. Additional blocks that can be included in Mediaplugin are RSYNC **742** and SSH **744**. These blocks can be used to ensure synchronization and updates of new media files coming from the cloud. The

US 10,123,074 B2

9

RSYNC block uses the “rsync” protocol and program over ssh (Secured Socket Layer Shell) to download media from the “Mediamplify Cloud” and synchronize multimedia files that are newly updated and could be part of particular Channel ID and a Playlist Name. Each new file synchronized with the Mediamplify cloud has a Media ID value which the cloud can assign, and include additional metadata information such as “Artist” and “Author.” The “Mediamplify Parallel Encoder” also may receive requests from other components such as getStatus or StartEncoding, indicating what file is being encoded, its progress, and what other files are in the queue for encoding.

In some embodiments, the information from getStatus or StartEncoding can be generated using a JSON message retrieved upon a request via HTTP GET to the Mediamplify Parallel Encoder. A graphical user interface can then be used to plot the status of each song being encoded or provide previews of the videos being generated. Below is one non-limiting example of a JSON response made by the “Mediamplify Parallel Encoder” service:

```
{“song_encoded”: “ts”, “artist”: “Madonna”, “duration”: 0,
“exists_on_disk”: True, “genre”: “Pop”, “title”: “Like a Virgin”, “music_id”: “1082863”}
```

Response regarding the status for a song being encoded or a list of songs being encoded, as follows:

```
{“music_id”: “109281”, “encoding”: “78%”, “output_file”:
“/var/www/egllacast/pop/109281-abcdef-uuid.ts”, “thumbnail”:
http://mediamplify.com/images/abcdef-uuid.png”}
```

In some embodiments, there are two possible TCP/IP sockets open for listening used by the Caching component of the Mediaplugin. A) Port 80 (as an example), or the standard HTTP port, used to retrieve content, view video content generated, update/delete cached files, and access the playlists, and B) Port 9553 (as an example), or the standard Mediamplify Parallel encoder port, used to monitor and review status of the songs being encoded, the data being collected, and all other related activities with encoding. Via this port, encoding/transcoding can be started or stopped.

Similarly, the “Mediaplugin” components labeled “Multicast-01” and “Multicast-02” can also provide an HTTP interface to access the current song being played or being multicast via the HTTP Server 770. For example, the HTTP Server may run a PHP or Python script that will return the current song being played by the streamer. The streamer block 768 can connect to the caching server via HTTP or RTSP protocols. The implementation of this block can be done using a combination of FIG. 8 instructions and using multicasting services available in VLC or FFMPEG (e.g. multicast, or tsools/tsplay).

One function of the streamer 768 is to read the files encoded by the caching unit, process those for multicasting and creating the MPEG Transport Stream multicast packets sent via the Ethernet port. Additionally, the multicast-01 and multicast-02 blocks contain a “Recording” function 790, which is in charge of recording the multicasting streams and recording for later use in case of failure on the system. This operates by listening into the UDP (User Datagram Protocol) multicast address of each individual channels being broadcasted, and recording the output into a file. This file is then stored and made accessible for future use. Monit 764 can maintain the streamer process running at all times throughout a regular broadcasting cycle.

FIG. 8 illustrates an example method that can be performed by the streamer module in the Mediaplugin. In one embodiment, the streamer can perform functions that include: connect to the caching unit; retrieve generated videos per playlist; determine if unit is working in fault-

10

tolerant mode; playback pre-recorded audio if a failure is detected; and record units own output or any other streamer output if connected to switch or hub.

Turning to the method from FIG. 8, at step 805, the streamer reads the multicast address associated to the channel being broadcasted, in order to generate IGMP (Internet Group Management Protocol) messages and establish a multicast group (e.g. 239.0.0.1:8001). The channel name is also initialized at 810. This value can be used to retrieve the proper content from the Caching server. In one embodiment, a configuration table can be created with tuples: <ChID>, <Channel Name>, <Multicast Address>, <Multicast Port>, such that no duplicates are used or created. Automation software can create these configuration tables and automate the process of creating all streamer calls. A streamer method can be called (e.g. from/etc/init.d/) at boot time and monitored with a tool such as Monit. Once the configuration is known, the streamer will check 850 if there are already multicasting transport streams generated by any other multicast component and avoid collisions. If such multicast data is detected, a delay can be used to delay a new attempt to check on that particular multicast group 860. Hence this block indicates “Wait <Time>” or stop all processing for the amount of <Time> recommended by the specification. For example, the time for delay can be the “Main source” of input less the amount of time for receiving input from a backup server. During initialization, if the same multicast address is used to broadcast main and backup, the one with less amount of time delay will be first, followed by the “Backup.” In cases where a hot spare is used, or two different multicast addresses can be in use, one for main and the other one for backup, then this step may be optional.

Once it is determined that there are no elements multicasting at the same address, the streamer will check if the Caching server is “alive” 815. This can be performed in multiple ways, for example an HTTP Request must not generate a 404 or 403 error, but instead a successful 202 value. Likewise, ping requests can also be used before making an HTTP GET call to identify if the caching server is present and available. In the case that there are no ping responses, a timeout will trigger a failure and a pre-recorded content 820 will be played or sent to the multicast address and port of record. Recordings can be scheduled daily for an extended period of time (e.g. 8 hours or more). These recordings can store the recorded file in a temporary folder and then move that file over to the playback folder where streamer can read it.

If it is determined that the caching server is functioning properly, then there are three types of streams that can be sent to the multicast address and port:

HTTP Live Streaming (HLS) a multi-segment stream usually defined by extension M3U8, containing a set number of segments for a real-time stream source (Video/audio)

HTTP/RTSP file playlist, this means that one or many files are read from the caching server, usually defined by extension M3U or PLS

RTSP stream, this could be constant stream similar to HTTP Live Streaming (HLS) but using an RTSP protocol instead of HLS stream.

When multiple files are part of a playlist, the bit-rate may vary between playlist files. Different bit-rates among files in a playlist could cause problems in non-CBR (Constant Bit Rate) systems. To avoid such problems, the Mediaplugin can maintain the stream at the bit rate of the previous stream by adding null MPEG TS packets at the same rate as the previous media file was being sent over the multicast port. This will maintain appropriate bit-rate for playlists.

US 10,123,074 B2

11

In RTSP service, streaming occurs by writing the media content to a named pipe or a FIFO file and allowing the RTSP service to read from that file and generate an RTSP stream. As such, in HLS and RTSP Streaming, the caching server knows the filename of the file being streamed. When the file is streamed from via playlist, both the caching and multicast unit know the name of the file being streamed, as the multicast server can then request from the caching server the metadata associated with the file name being streamed.

Accordingly the present technology can accommodate both cases and retrieve the current file name being streamed as a JSON response. The response can identify the content displayed or played back at the end-user's set top box.

If HLS is used for streaming **818**, the streamer will connect to the proper M3U8 file **820** to pull all the segments corresponding to the live playlist file on the M3U8 stream. Programs and libraries such as FFMPEG or VLC can then be used to encapsulate the stream feed to an MPEG TS and multicast it to the pre-programmed multicast address and port. Similarly, if an RTSP stream **825** is used, the same process takes place at **830** using the RTSP as a protocol instead of HTTP.

If a list of files in a playlist **832** is in use "M3U," the list can be obtained using an HTTP GET request made to the caching server. Each file can then be streamed to the multicast address and port of record **838** until all elements of the playlist are completed and played back **835**.

In some instances, metadata can be added to the stream as an additional PID (Program Identifier) in the MPEG Transport Stream. If metadata is needed **841** then it will be inserted **842** here in accordance with FIG. 9.

For example, the metadata added could be as simple as a string containing Artist, Name, or other information, or as complex as DCII Text (DigiCipher version 2, as specified by Motorola), MPEG TS Control messages (Two-way), or any other metadata or PID that can be inserted into the stream being sent to the multicast address and port.

FIG. 9 illustrates an example method used to insert a special data packet into the stream being written to the multicast address, without making use of timing information. Therefore, the metadata can be inserted such that it does not have timing dependency. Alternatively, certain type of metadata such as subtitles or lyrics may need to be inserted only at certain times so that it is synchronized with the video data.

The sample MPEG TS packet shown in **902** and **905** corresponds to the metadata inside the packet. This matches standard MPEG TS packet structure and may vary depending on the type of metadata being added to the stream.

The method to insert the packet starts by reading the MPEG TS file retrieved from the caching server **910**. If metadata corresponds to only song information, this information can be added at uniform times. Non-uniform timing inclusions can also be done by adding a table that includes the time and the metadata content added to the stream **915**. This metadata can be retrieved from the Caching unit using an HTTP Get request directed to the caching server **918** and may include artist name, title of song, or any other information **920**.

Creation of the MPEG Packet(s) **925** to be inserted at the appropriate times, which will set the places with those packets, will be inserted in the MPEG TS. Modify the PMT (Program Management Table) of the generated file and include the PID being inserted into the MPEG TS **930**. At step **932**, an audio PID may include a metadata PID of type DCII Text, which is added to the stream.

12

For each timing or location where the PID needs to be inserted **935**, the method can insert those packets and generate a buffer that can be sent to the multicast address as in illustrated in FIG. 8.

As described above, the disclosed technology can be implemented using a set of modules included in the Mediaplug apparatus, which can be implemented using one or more servers accordingly. Alternatively, virtualization environments such as VMWare or Xen (FIG. 9) can include a system and architecture that can be used to host all the Mediaplug functionality in a virtualized environment using a single server with multiple GbE ports.

The illustration in FIG. 10 shows a configuration of three virtual machines used for implementing the Mediaplug. Caching **1030** and its respective Ethernet port **1032**, Multicast-01 **1020** with two virtual Ethernet ports **1022** and **1024**, and Multicast-02 **1010**, with two Ethernet ports **1012** and **1014**. There are three bridges that include **1007**, **1008**, and **1009**. The multicast Ethernet ports **1024** and **1014** are bridged with eth1 **1004** and eth2 **1003** respectively via bridge **1009** and bridge **1008**, respectively. That is, the same content generated at **1024** can be forwarded via the bridge **1009** to the physical interface eth1 **1004** in the hardware. The vSwitch **1045** or the virtual switch between the two bridged interfaces **1008** and **1009** permits the two virtual machines of Multicast-01 **1010** and Multicast-02 **1020** to record each other and monitor their activity.

To achieve concurrency and proper execution in a cloud environment, the Mediaplug can use the example method shown in FIG. 11. Once a create unit command is issued **1105**, provisioning of the metadata information and proper IP Address is provided for that particular instance **1110**. Once an IP Address and metadata has been chosen for the unit, the server can be allocated in the cloud and the instance can be created using CloudStack or OpenStack tools and services **1115**. The instance is then allocated and runs in the proper node in the cloud **1120**. Configuration of the assigned IP, Netmask, and VPN credentials are loaded at **1125**. Finally, certificate information is uploaded **1130** as well as all services run to enable the unit for execution of its caching functions, multicasting functions, or both.

FIG. 12 illustrates a conventional system bus computing system architecture **1200** such as can be used in Mediaplug **110**. Those of ordinary skill in the art will appreciate that other system architectures are also possible.

In computing system architecture **1200** the components of the system are in electrical communication with each other using a bus **1205**. Example system **1200** includes a processing unit (CPU or processor) **1210** and a system bus **1205** that couples various system components including the system memory **1215**, such as read only memory (ROM) **1220** and random access memory (RAM) **1225**, to the processor **1210**. The system **1200** can include a cache of high-speed memory connected directly with, in close proximity to, or integrated as part of the processor **1210**. The system **1200** can copy data from the memory **1215** and/or the storage device **1230** to the cache **1212** for quick access by the processor **1210**. In this way, the cache can provide a performance boost that avoids processor **1210** delays while waiting for data. These and other modules can control or be configured to control the processor **1210** to perform various actions. Other system memory **1215** may be available for use as well. The memory **1215** can include multiple different types of memory with different performance characteristics. The processor **1210** can include any general purpose processor and a hardware module or software module, such as module **1** **1232**, module **2** **1234**, and module **3** **1236** stored in storage device **1230**,

US 10,123,074 B2

13

configured to control the processor 1210 as well as a special-purpose processor where software instructions are incorporated into the actual processor design. The processor 1210 may essentially be a completely self-contained computing system, containing multiple cores or processors, a bus, memory controller, cache, etc. A multi-core processor may be symmetric or asymmetric.

To enable user interaction with the computing device 1200, an input device 1245 can represent any number of input mechanisms, such as a microphone for speech, a touch-sensitive screen for gesture or graphical input, keyboard, mouse, motion input, speech and so forth. An output device 1235 can also be one or more of a number of output mechanisms known to those of skill in the art. In some instances, multimodal systems can enable a user to provide multiple types of input to communicate with the computing device 1200. The communications interface 1240 can generally govern and manage the user input and system output. There is no restriction on operating on any particular hardware arrangement and therefore the basic features here may easily be substituted for improved hardware or firmware arrangements as they are developed.

Storage device 1230 is a non-volatile memory and can be a hard disk or other types of computer readable media which can store data that are accessible by a computer, such as magnetic cassettes, flash memory cards, solid state memory devices, digital versatile disks, cartridges, random access memories (RAMs) 1225, read only memory (ROM) 1220, and hybrids thereof.

The storage device 1230 can include software modules 1232, 1234, 1236 for controlling the processor 1210. Other hardware or software modules are contemplated. The storage device 1230 can be connected to the system bus 1205. In one aspect, a hardware module that performs a particular function can include the software component stored in a computer-readable medium in connection with the necessary hardware components, such as the processor 1210, bus 1205, display 1235, and so forth, to carry out the function.

It can be appreciated that example system 1200 can have more than one processor 1210 or be part of a group or cluster of computing devices networked together to provide greater processing capability.

For clarity of explanation, in some instances the present technology may be presented as including individual functional blocks including functional blocks comprising devices, device components, steps or routines in a method embodied in software, or combinations of hardware and software.

Any of the steps, operations, functions, or processes described herein may be performed or implemented by a combination of hardware and software modules, alone or in combination with other devices. In an embodiment, a software module can be software that resides in memory of a client device and/or one or more servers of a content management system and perform one or more functions when a processor executes the software associated with the module. The memory can be a non-transitory computer-readable medium.

In some embodiments the computer-readable storage devices, mediums, and memories can include a cable or wireless signal containing a bit stream and the like. However, when mentioned, non-transitory computer-readable storage media expressly exclude media such as energy, carrier signals, electromagnetic waves, and signals per se.

Methods according to the above-described examples can be implemented using computer-executable instructions that are stored or otherwise available from computer readable

14

media. Such instructions can comprise, for example, instructions and data which cause or otherwise configure a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Portions of computer resources used can be accessible over a network. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, firmware, or source code. Examples of computer-readable media that may be used to store instructions, information used, and/or information created during methods according to described examples include magnetic or optical disks, flash memory, USB devices provided with non-volatile memory, networked storage devices, and so on.

Devices implementing methods according to these disclosures can comprise hardware, firmware and/or software, and can take any of a variety of form factors. Typical examples of such form factors include laptops, smart phones, small form factor personal computers, personal digital assistants, and so on. Functionality described herein also can be embodied in peripherals or add-in cards. Such functionality can also be implemented on a circuit board among different chips or different processes executing in a single device, by way of further example.

The instructions, media for conveying such instructions, computing resources for executing them, and other structures for supporting such computing resources are means for providing the functions described in these disclosures.

Although a variety of examples and other information was used to explain aspects within the scope of the appended claims, no limitation of the claims should be implied based on particular features or arrangements in such examples, as one of ordinary skill would be able to use these examples to derive a wide variety of implementations. Further and although some subject matter may have been described in language specific to examples of structural features and/or method steps, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to these described features or acts. For example, such functionality can be distributed differently or performed in components other than those identified herein. Rather, the described features and steps are disclosed as examples of components of systems and methods within the scope of the appended claims.

What is claimed is:

1. A computer-implemented method comprising:

receiving, from a content provider, a request for at least one media stream for playback on a broadcast media channel, wherein the at least one media stream includes a plurality of multimedia items of different types;

obtaining content corresponding to the plurality of multimedia items from at least one source offering the content in at least one first format;

rendering a web page by a browser using the content; generating a temporal sequence of screen captures of the rendered web page, where each screen capture defines all the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time; assembling the at least one media stream using the temporal sequence of screen captures; and

providing the at least one media stream to the content provider for broadcast on the broadcast media channel.

2. The method of claim 1, wherein the obtaining comprises: retrieving, for each of the plurality of multimedia items, at least one audio file corresponding to an audio

US 10,123,074 B2

15

component of the multimedia item and a plurality of screen captures corresponding to a video component of the multimedia item.

3. The method of claim 2, wherein the generating comprises: combining the plurality of screen captures and the at least one audio file to create each of the plurality of multimedia items.

4. The method of claim 2, wherein the retrieving of the plurality of screen captures comprises obtaining the plurality of screen captures from a playback of a video on the at least one webpage.

5. The method of claim 4, wherein the combining of the plurality of screen captures comprises metadata corresponding to at least one of a song title, an artist, and a music genre.

6. The method of claim 1, further comprising: identifying the at least one webpage based on a channel identifier associated with the broadcast media channel.

7. The method of claim 1, further comprising: querying the content provider to determine the second format, wherein the second format corresponds to a Moving Picture Experts Group (MPEG) format.

8. The method of claim 3, further comprising:

detecting a change at the at least one webpage corresponding to at least one of the plurality of multimedia items;

in response to detecting the change, retrieving a new plurality of screen captures from the at least one webpage for the at least one of the plurality of multimedia items; and

reassembling the at least one of the plurality of multimedia items using the new plurality of screen captures to yield an updated multimedia item.

9. The method of claim 1, wherein the content provider is a cable television operator or a satellite television operator.

10. The method of claim 1, wherein providing the at least one media stream to the content provider comprises streaming the at least one media stream to an MPEG multiplexer associated with the content provider.

11. The method of claim 1, further comprising: provisioning, on at least one server, at least one virtual machine for performing the receiving, obtaining, generating, assembling, and providing.

12. A system comprising:

at least one processor;

a computer-readable storage medium having stored therein instructions which, when executed by the processor, cause the at least one processor to perform operations comprising:

receive, from a content provider, a request for at least one media stream for playback on a broadcast media channel, the at least one media stream including a plurality of multimedia items of different types;

retrieve, from a cloud based service, multimedia content corresponding to the plurality of multimedia items, the multimedia content in a first format;

render a web page by a browser using the multimedia content;

generate a temporal sequence of screen captures of the rendered web page, where each screen capture defines all the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time;

16

merge the temporal sequence screen captures with the at least one audio file to create the media stream; and send the media stream to the content provider.

13. The system of claim 12, further comprising: retrieve a video from the cloud based service, wherein the screen captures correspond to a plurality of static images obtained from the video.

14. The system of claim 12, further comprising: select the multimedia content from the cloud based service based on a music genre associated with the broadcast media channel.

15. The system of claim 12, wherein the request comprises a request for a plurality of multimedia streams for playback on a plurality of broadcast media channels.

16. The system of claim 12, wherein the media stream is sent to a broadcast internet protocol (IP) address associated with the content provider.

17. A non-transitory computer-readable storage medium having stored therein instructions which, when executed by a processor, cause the processor to perform operations comprising:

receiving, from a content provider, a request for at least one media stream for playback on a broadcast media channel, wherein the at least one media stream includes a plurality of multimedia items of different types;

obtaining content corresponding to the plurality of multimedia items from at least one source offering the content in at least one first format;

rendering a web page by a browser using the content;

generating a sequence of screen captures of the rendered web page, where each screen capture defines all of the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time;

assembling the at least one media stream using the sequence of screen captures; and

providing the at least one media stream to the content provider for broadcast on the broadcast media channel.

18. The non-transitory computer-readable storage medium of claim 17,

wherein the obtaining comprises retrieving, for each of the plurality of multimedia items, at least one audio file corresponding to an audio component of the multimedia item and a plurality of screen captures corresponding to a video component of the multimedia item; and wherein the assembling comprises combining the plurality of screen captures and the at least one audio file.

19. The non-transitory computer-readable storage medium of claim 17, wherein the at least one media stream corresponds to at least one of an HTTP live stream (HLS), an HTTP playlist, and a Real-time Streaming Transport (RTSP) stream.

20. The non-transitory computer-readable storage medium of claim 17, storing additional instructions which, when executed by the processor, result in operations comprising:

detect a network failure affecting connectivity with the at least one website; and

in response, provide a previously recorded media stream to the content provider.

21. The non-transitory computer-readable storage medium of claim 17, wherein assembling the at least one media stream comprises inserting at least one MPEG packet including metadata corresponding to the multimedia item.

* * * * *

Exhibit B



US010524002B2

(12) **United States Patent**
Hernandez-Mondragon

(10) **Patent No.:** **US 10,524,002 B2**

(45) **Date of Patent:** **Dec. 31, 2019**

(54) **METHOD, SYSTEM, AND APPARATUS FOR MULTIMEDIA CONTENT DELIVERY TO CABLE TV AND SATELLITE OPERATORS**

(71) Applicant: **Edwin A. Hernandez-Mondragon**,
Coral Springs, FL (US)

(72) Inventor: **Edwin A. Hernandez-Mondragon**,
Coral Springs, FL (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **16/152,606**

(22) Filed: **Oct. 5, 2018**

(65) **Prior Publication Data**

US 2019/0037271 A1 Jan. 31, 2019

Related U.S. Application Data

(63) Continuation of application No. 15/538,911, filed as application No. PCT/US2015/067464 on Dec. 22, 2015, now Pat. No. 10,123,074.

(Continued)

(51) **Int. Cl.**
H04N 21/4402 (2011.01)
H04N 21/2343 (2011.01)

(Continued)

(52) **U.S. Cl.**
CPC ... **H04N 21/440218** (2013.01); **H04N 21/233** (2013.01); **H04N 21/2387** (2013.01); **H04N 21/234309** (2013.01); **H04N 21/4312** (2013.01); **H04N 21/47217** (2013.01); **H04N 21/6118** (2013.01); **H04N 21/6143** (2013.01); **H04N 21/8106** (2013.01); **H04N 21/8133** (2013.01);

(Continued)

(58) **Field of Classification Search**

CPC H04N 21/440218; H04N 21/233; H04N 21/234309; H04N 21/2387; H04N 21/4312; H04N 21/47217; H04N 21/6118; H04N 21/6143; H04N 21/8106; H04N 21/8133; H04N 21/8153; H04N 21/84; H04N 21/8547; H04N 21/6405; H04N 21/64322; H04N 21/81; H04N 21/854

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,886,178 B1 4/2005 Mao et al.
7,020,888 B2 3/2006 Reynolds et al.
(Continued)

OTHER PUBLICATIONS

International Search Report and Written Opinion dated Apr. 20, 2016 for PCT/US15/67464.

(Continued)

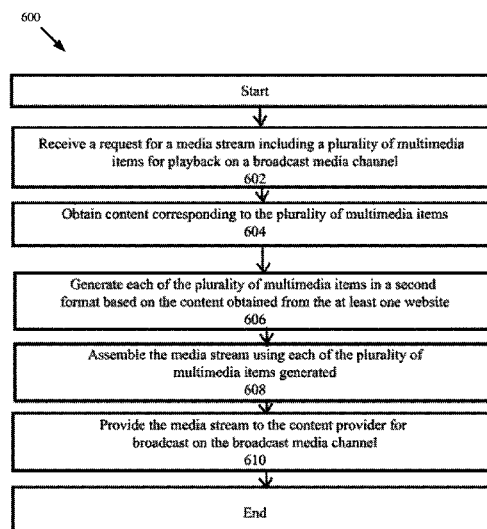
Primary Examiner — Rong Le

(74) *Attorney, Agent, or Firm* — Fox Rothschild LLP

(57) **ABSTRACT**

Systems, methods, and computer-readable media for delivering multimedia content from the cloud to cable operators are disclosed. A device located at the cable headend or implemented in the cloud can receive a request for at least one media stream for playback on a broadcast media channel. Content corresponding to a plurality of multimedia files in the media stream can be obtained from the internet or a cloud based service. The content can be used to generate the multimedia files in a format that is compatible with the cable operator. The multimedia files can be used to assemble the at least one media stream which can be provided to the cable operator for broadcast on the broadcast media channel.

13 Claims, 12 Drawing Sheets



US 10,524,002 B2

Page 2

Related U.S. Application Data

(60) Provisional application No. 62/095,504, filed on Dec. 22, 2014.

(51) **Int. Cl.**

H04N 21/84 (2011.01)
H04N 21/81 (2011.01)
H04N 21/2387 (2011.01)
H04N 21/233 (2011.01)
H04N 21/61 (2011.01)
H04N 21/472 (2011.01)
H04N 21/431 (2011.01)
H04N 21/8547 (2011.01)
H04N 21/643 (2011.01)
H04N 21/6405 (2011.01)

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

CPC *H04N 21/8153* (2013.01); *H04N 21/84* (2013.01); *H04N 21/8547* (2013.01); *H04N 21/6405* (2013.01); *H04N 21/64322* (2013.01)

(56) **References Cited****U.S. PATENT DOCUMENTS**

7,320,025	B1	1/2008	Steinberg et al.
8,769,602	B1	7/2014	Del Beccaro et al.
9,418,388	B1 *	8/2016	Young G06F 16/954
10,123,074	B2 *	11/2018	Hernandez-Mondragon H04N 21/84
2002/0138852	A1	9/2002	Reynolds et al.
2003/0115612	A1	6/2003	Mao et al.
2008/0040354	A1	2/2008	Ray et al.

2008/0098426	A1	4/2008	Candelore
2009/0178003	A1	7/2009	Fiedler
2010/0192178	A1	7/2010	Candelore
2012/0204217	A1	8/2012	Regis et al.
2013/0111057	A1 *	5/2013	Yoon H04N 21/2225 709/231
2014/0068690	A1 *	3/2014	Luthra H04N 21/234309 725/110
2014/0098815	A1	4/2014	Mishra et al.
2014/0143437	A1	5/2014	Mathur et al.
2014/0150019	A1 *	5/2014	Ma G06Q 30/0251 725/34
2014/0344880	A1 *	11/2014	Geller H04N 21/8586 725/115
2014/0351871	A1	11/2014	Bomfim et al.
2015/0319004	A1	11/2015	Kwan et al.
2015/0382081	A1 *	12/2015	Reese G11B 27/031 386/282

OTHER PUBLICATIONS

Justin Pot: "Save Videos From Any Site—Even Netflix—With Applian's Replay Capture Suite", Jun. 3, 2014 (Jun. 3, 2014), XP055472253, Retrieved from the Internet: URL:https://www.makeuseof.com/tag/savevideos- from-any-site-even-netflix-with-applians-replay-capture-suite/ [retrieved on May 3, 2018].
 Fawad Mir: "Schedule Auto Song Recording From Multiple Radio Streams With streamWriter", Jan. 9, 2013 (Jan. 9, 2013), XP055472533, Retrieved from the Internet: URL:https://www.addictivetips.com/windows-tips/schedule-auto-songrecording- from-multiple-radio-streams-streamwriter/ [retrieved on May 4, 2018].
 Chris Robley, How to make album art videos for your songs on YouTube, Jul. 16, 2014.

* cited by examiner

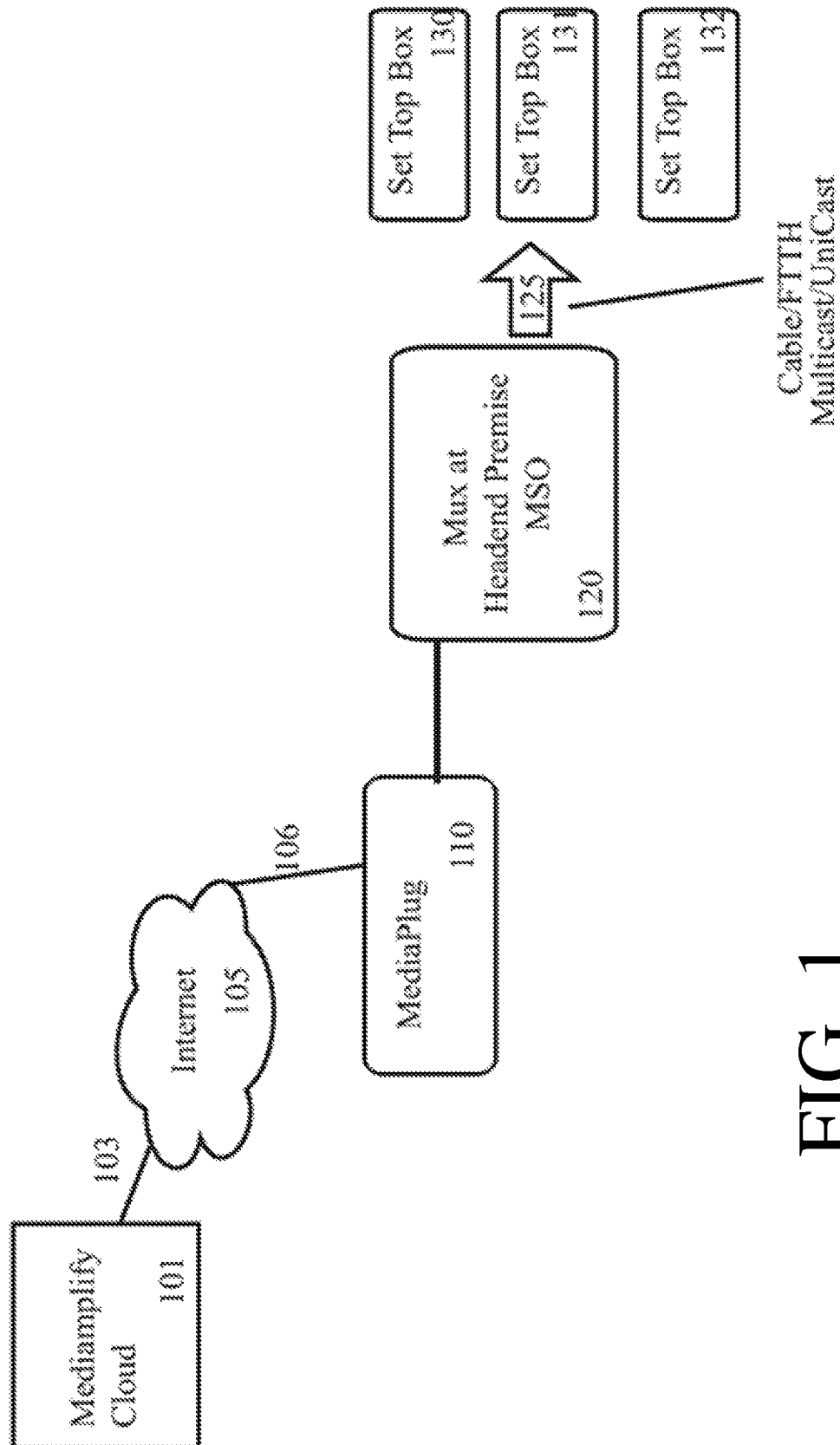


FIG. 1

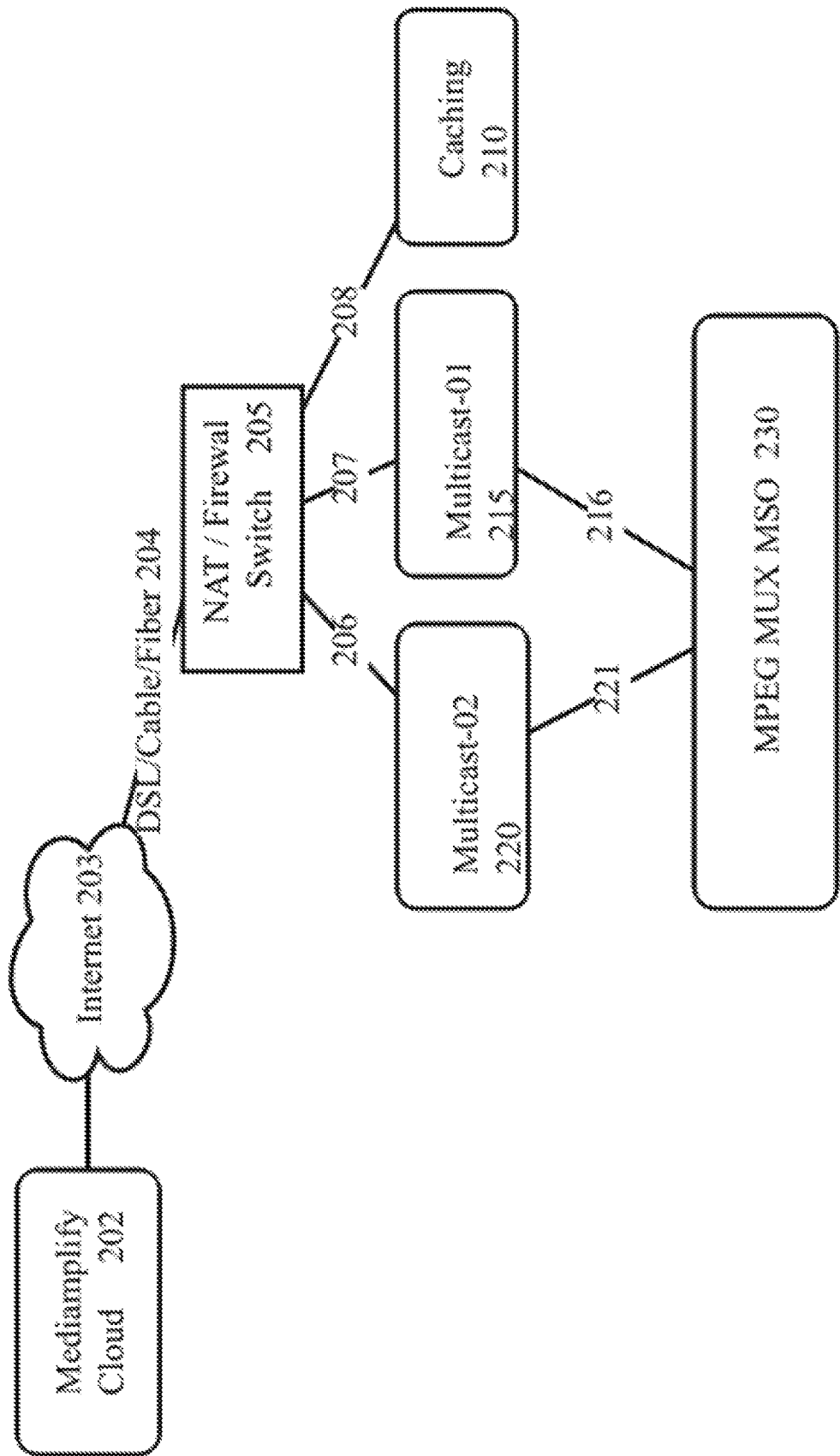


FIG. 2

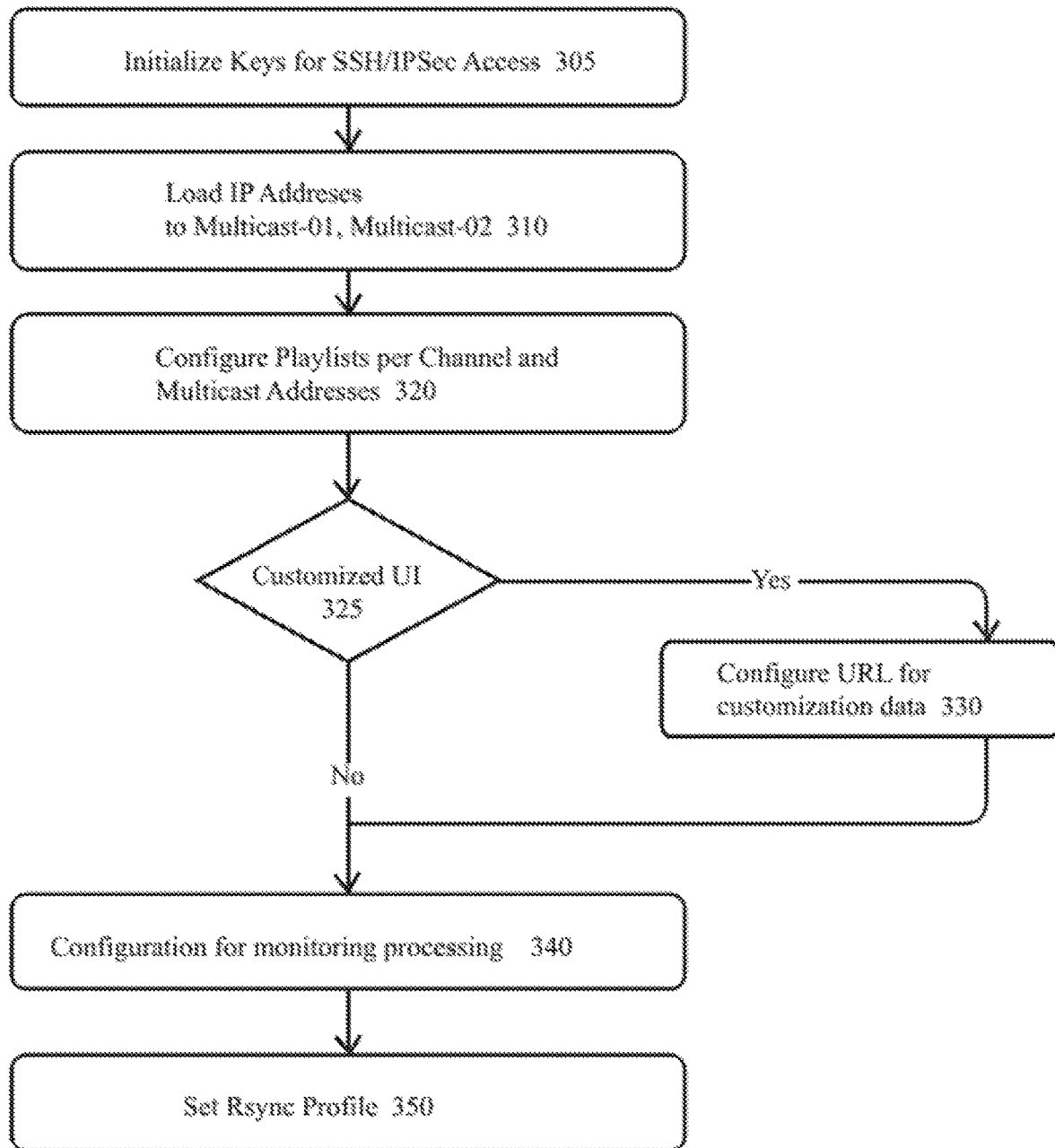


FIG. 3

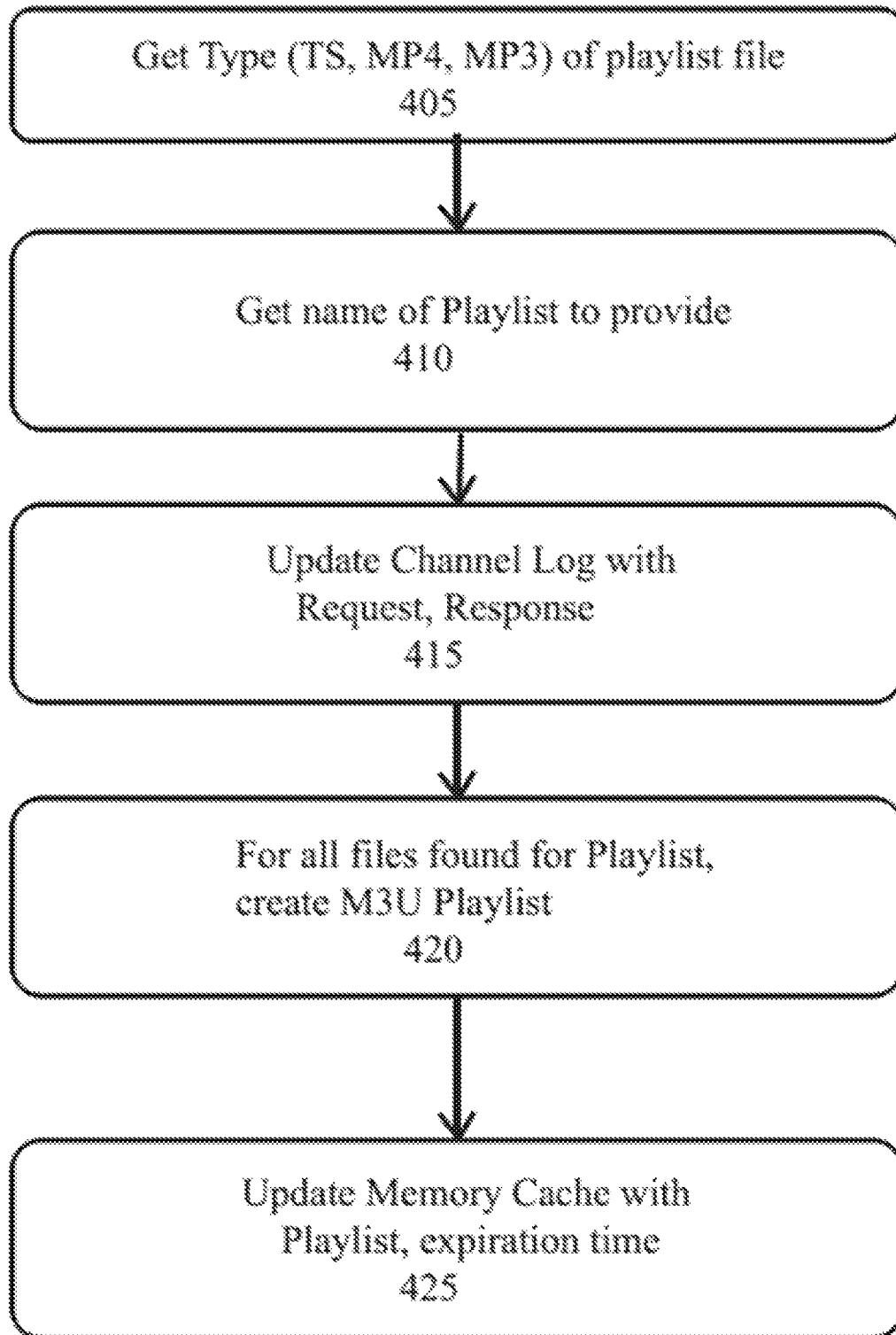


FIG. 4

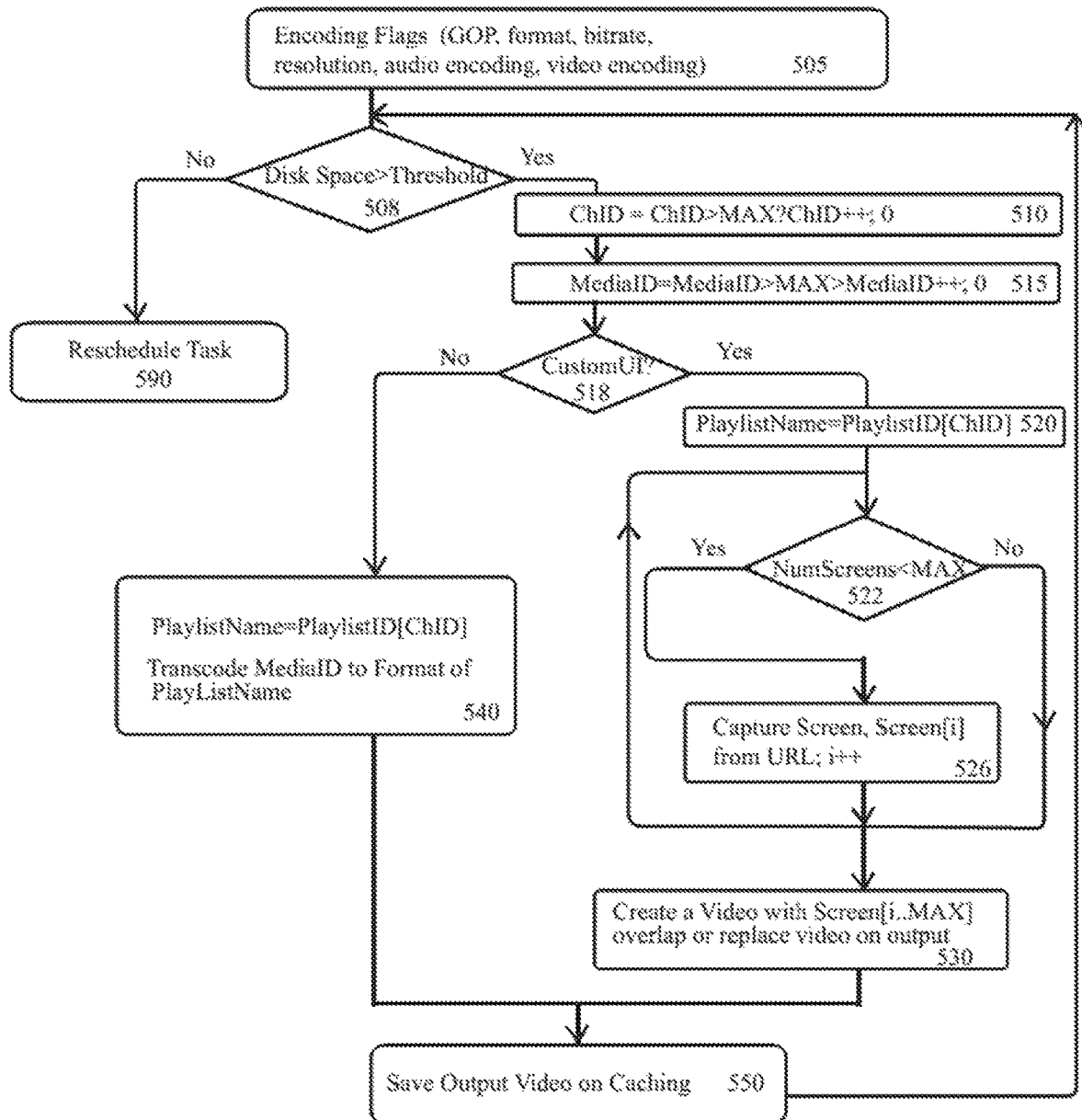


FIG. 5

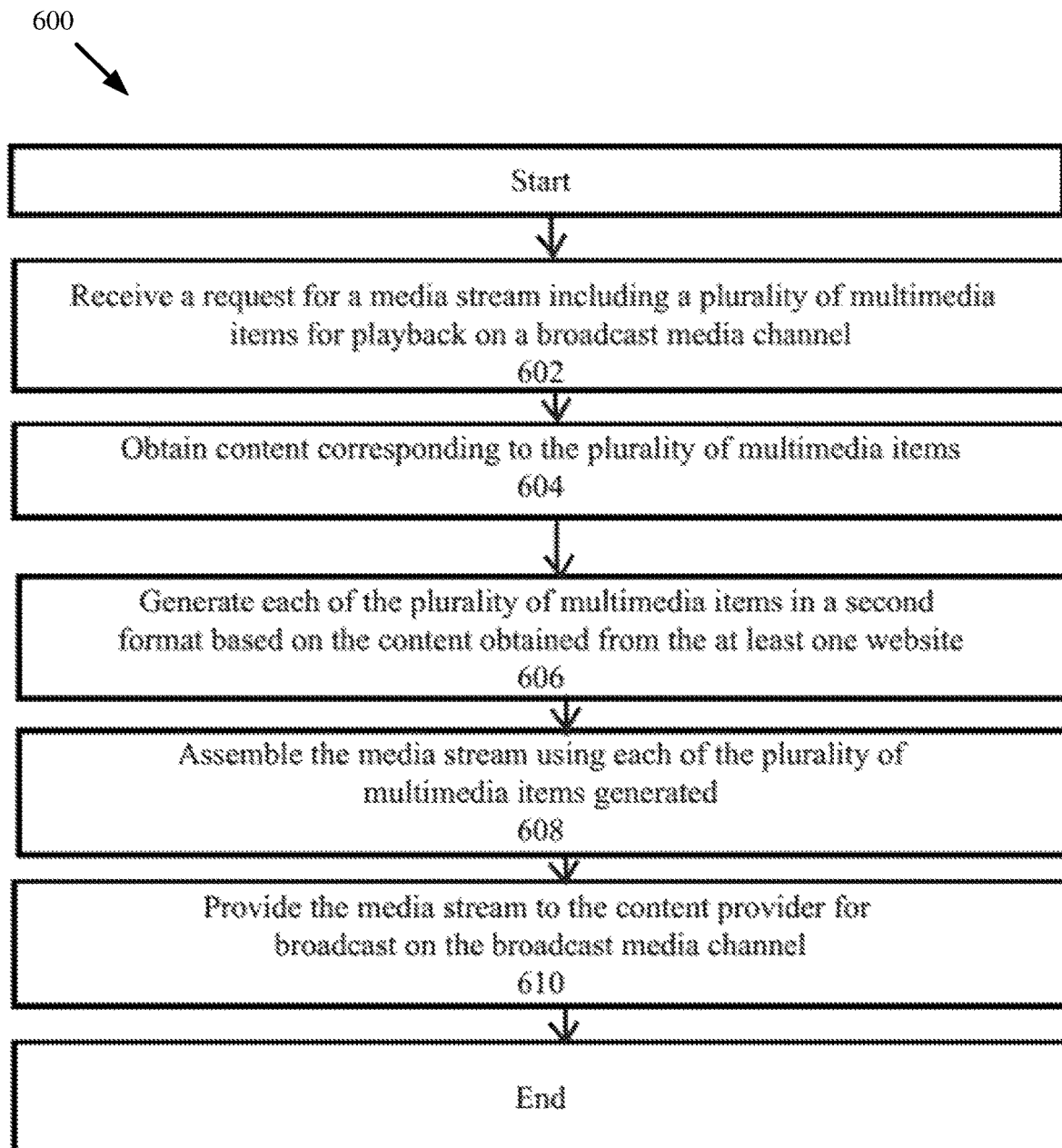
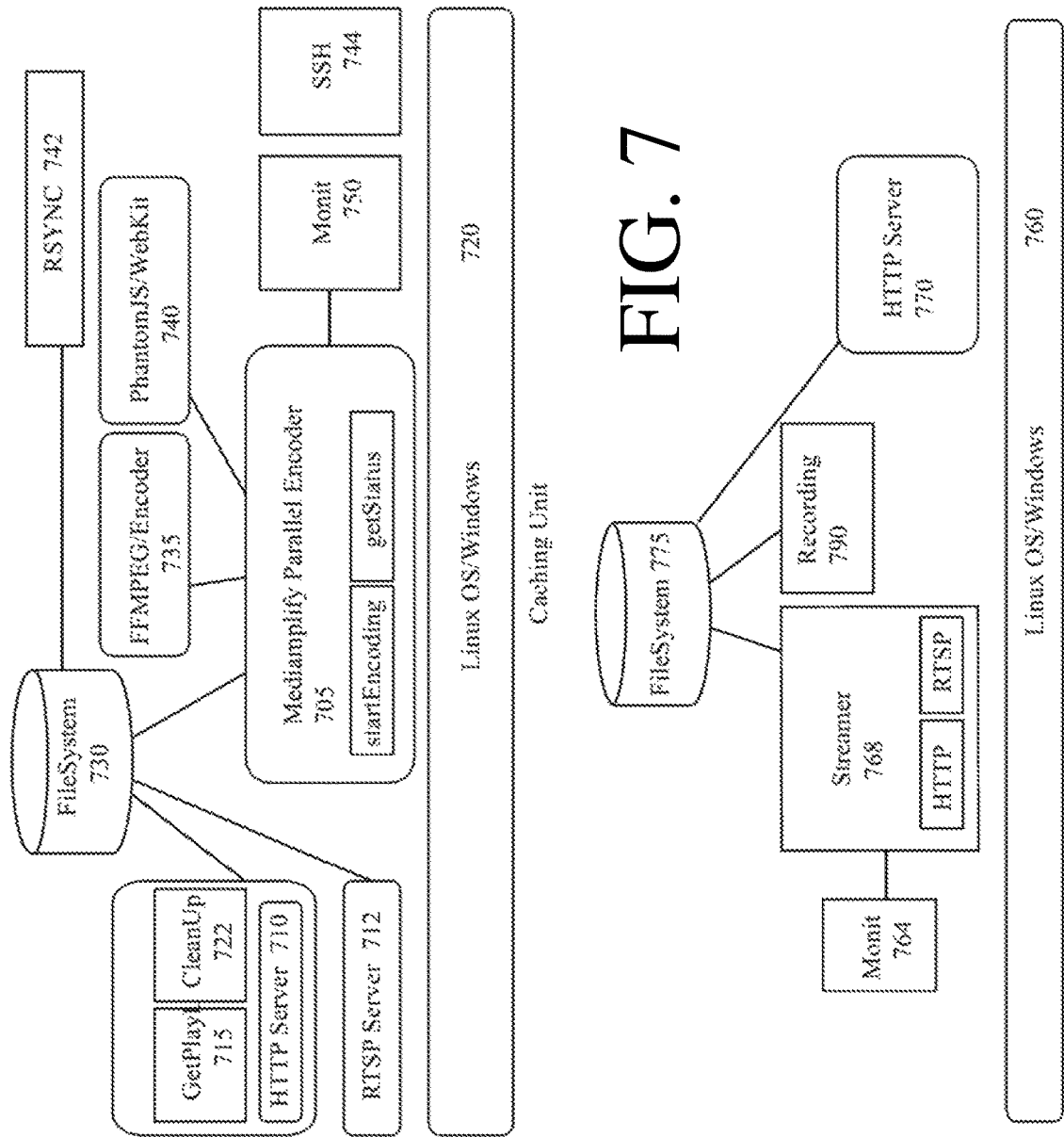


FIG. 6



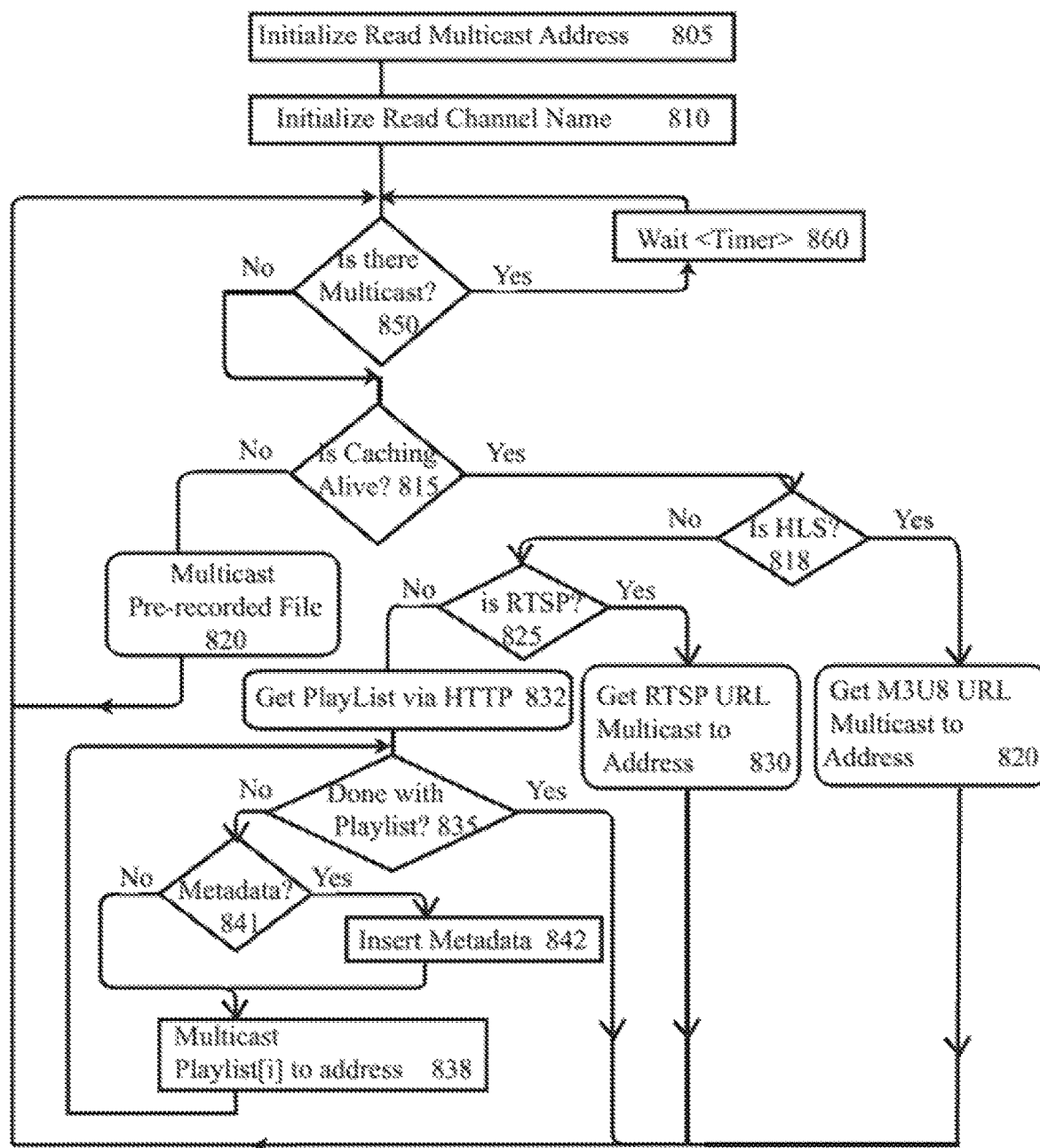


FIG. 8

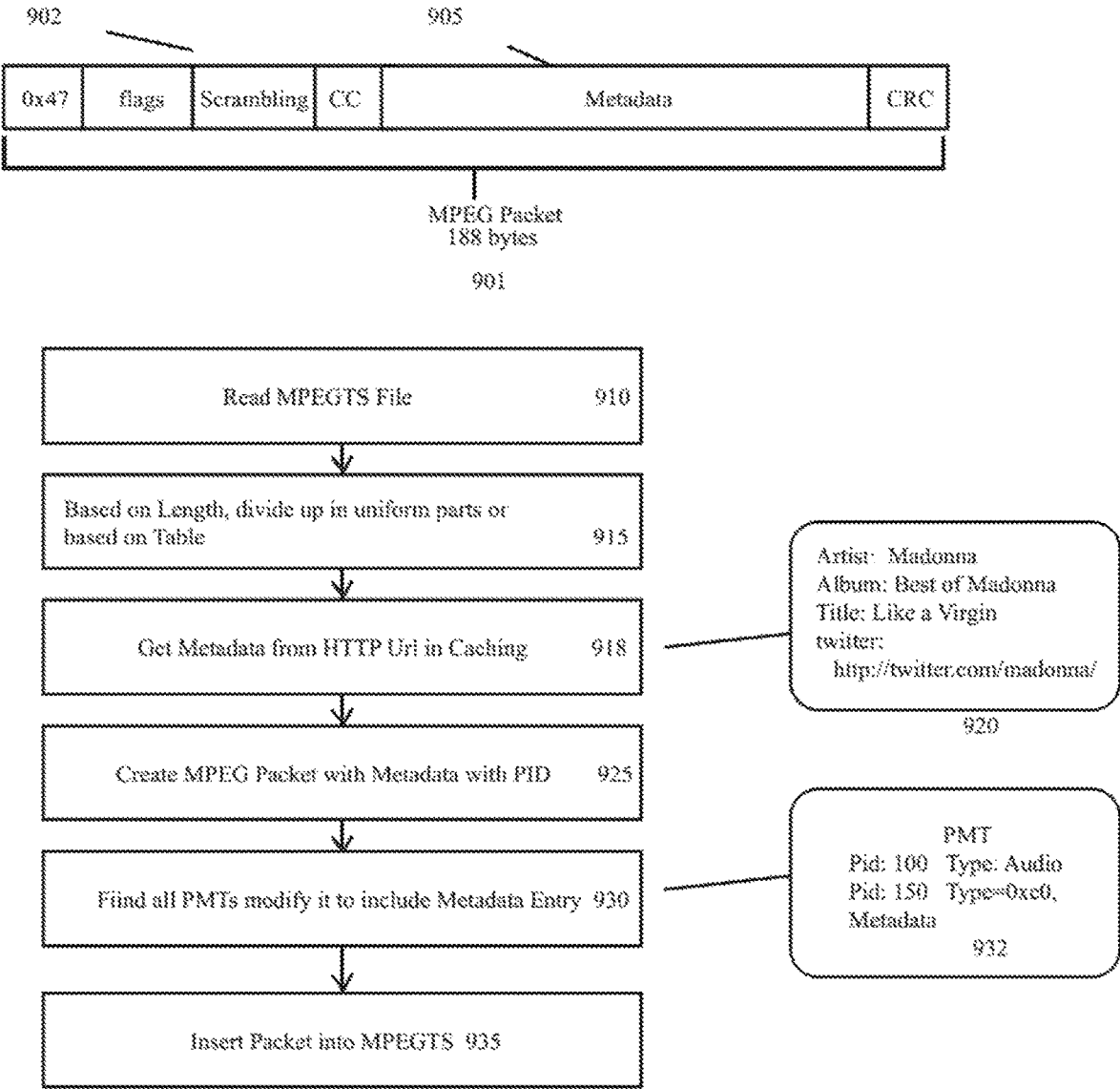


FIG. 9

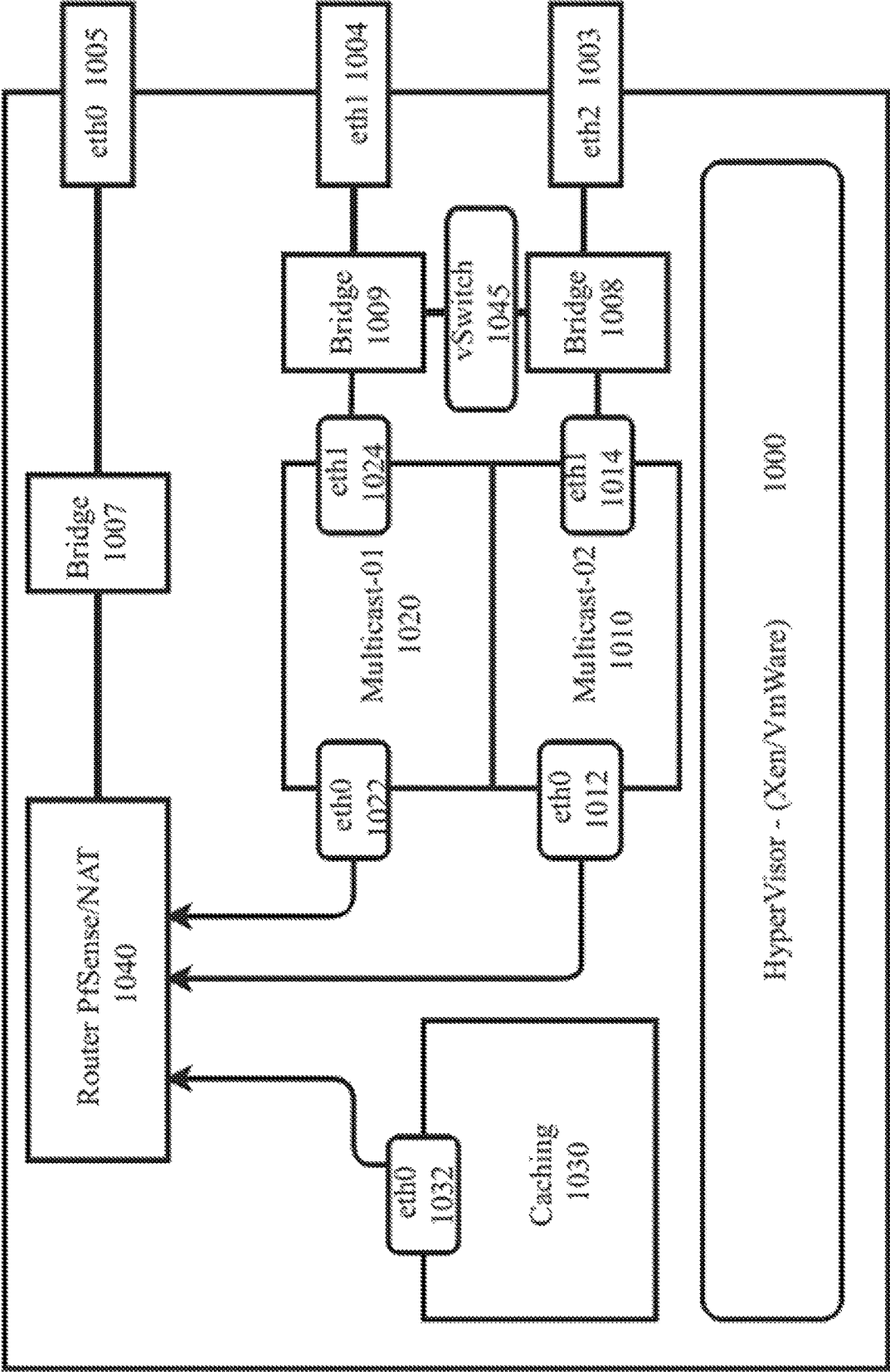


FIG. 10

U.S. Patent

Dec. 31, 2019

Sheet 11 of 12

US 10,524,002 B2

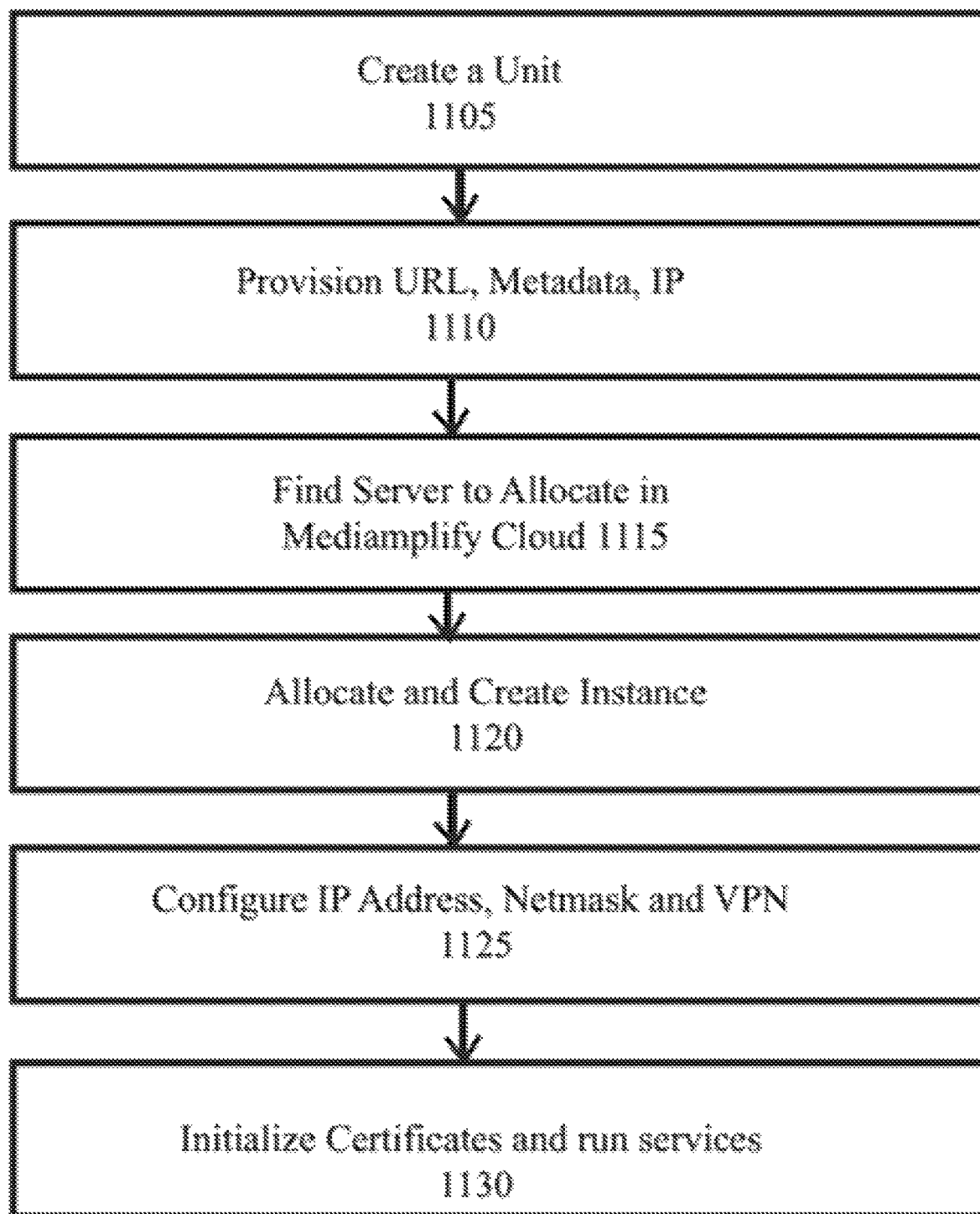
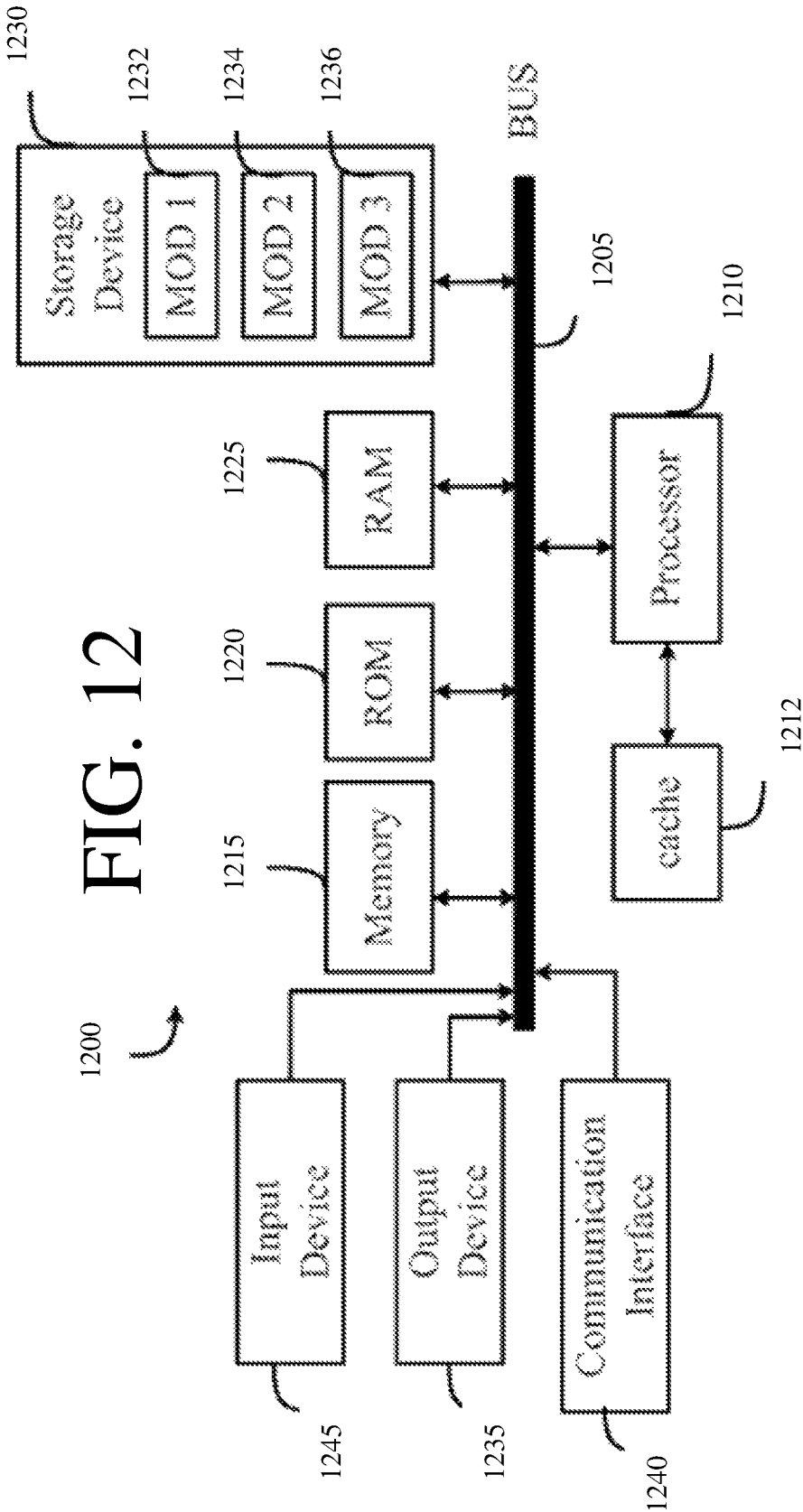


FIG. 11



US 10,524,002 B2

1

**METHOD, SYSTEM, AND APPARATUS FOR
MULTIMEDIA CONTENT DELIVERY TO
CABLE TV AND SATELLITE OPERATORS****CROSS-REFERENCE TO RELATED
APPLICATION**

This application is a continuation of U.S. patent application Ser. No. 15/538,911, which was filed Jun. 22, 2017, which is a National Stage Entry of International Patent Application No. PCT/US2015/067464, which was filed on Dec. 22, 2015, which claims priority to U.S. Provisional Application 62/095,504, which was filed on Dec. 22, 2014. The complete disclosures of which are incorporated herein by reference in their entirety.

FIELD OF THE DISCLOSURE

The present disclosure provides a mechanism for delivering multimedia content from a cloud-based system to cable or satellite content providers.

The disclosure describes a novel method and a system for continuous media delivery and secured synchronization of multimedia files that are broadcasted to set top boxes or subscriber units in the field that are connected to IPTV or Cable Systems. The present technology can securely connect to a cloud-based platform (e.g. OpenStack, Amazon EC2, Mediampify Cloud, etc.), creates a set of multimedia files (e.g. MPEG4 movies) into a secured repository (e.g. Secured File System, Linux/Windows, etc.), modifies those files according to certain rules as defined by the method herein, and reproduces those signals via a multicast video stream (e.g. User Datagram Protocol (UDP) multicast, etc.) encapsulated in a Transport Stream (TS) frame that can be a multiple transport stream (several transports in one TS frame) or a single transport stream (one transport stream according to the MPEG-2 Part 1 Specification or similar multimedia streaming that is compatible with MPEG-TS at a predetermined rate or at a variable bit rate).

BACKGROUND

Cable operators, Multiple System Operators (MSO), and Digital Satellite Systems (DSS) deliver media content by utilizing satellite, Radio Frequency (RF) and Integrated Receiver and Decoder (IRD) technologies, which provide outputs that can include Asynchronous Serial Interface (ASI) signal formats. Additionally, MSOs and DSSs use dedicated hardware encoders and transcoding systems if the source video/audio is incompatible with the set top boxes of the cable or satellite operator. In general, the video/audio content delivered via the satellite to the cable or satellite operator arrives via wireless link or receiver, which is processed by an IRD that receives the stream from the satellite and can ultimately broadcast that signal to the cable operator head end in several formats. In addition, some systems may include multicast units connected with IRDs that tune to the right transponder system.

Satellite systems are used to deliver data (e.g. TV, radio, Video-on-Demand, multimedia streams) from a particular source (e.g. third-party content provider) to cable head end systems or other satellite systems. Satellite delivery is performed via a broadcast signal containing digital pictures and audio inside a transponder frequency. Transponders are radio frequency (RF) space allocated in a particular satellite.

For example, a third-party content provider can maintain multimedia files in CD/DVDs, pre-recorded storage, or as

2

pre-recorded files that are reproduced in a particular audio or video encoding format using a video or audio encoder (e.g. MP3, H.264, AC3, and others). The audio and video is then made part of an RF signal that is broadcasted to the satellite system. This signal is then received by the cable or satellite operator using an antenna receptor and is decoded to generate either an analog signal (e.g. RCA Cables or any other analog interface) or digital signal (e.g. ASI, Asynchronous Serial Interface). If an analog signal is received, modern DVB (Digital Video Broadcasting) Cable and Satellite systems will encode this signal to their particular video/audio encoding requirements such that it can be broadcasted in digital packets via a multicast MPEG Transport Stream. The same occurs if a receiver decodes the signal in ASI. As such, this signal is converted from ASI to Ethernet, or simply redirected to a multicast MPEG Transport stream if arriving in a compatible encoding format for the receiving head end cable system.

When no satellite is being used, video, pictures and music, are read out from the DVD or a hard-drive containing MPEG-layer 3 or MPEG-layer 2 files that were encoded at a pre-determined bit rate according. Those video feeds can be reproduced at a synchronized timing using a multicast MPEG Transport Stream using RTP (Real-time Transport Protocol)-based multicast communication and RTCP (RTP Control Protocol)-based mechanisms, as well as standard UDP (User Datagram Protocol) multicast streams with proprietary timing.

Additionally, IPTV systems can provide mechanisms to broadcast a signal using the multicast protocol in MPEG transport streams to a customer premises with compatible set top boxes (same encoder, same streams, encryption keys, etc). A compatible set top box is such that when connected to the cable TV system it can properly decode video/audio from the cable operator's head end system. Some of these IPTV systems rely on CDN (Content Delivery Networks) that enable distribution of multimedia files to IP-enabled devices. For example, a mechanism can be provided whereby a head end content (MSO) that is being broadcasted to a cable subscriber is also available to an IP-enabled device such as a smartphone, PC, or tablet computer.

SUMMARY

Disclosed are systems, devices, methods, and non-transitory computer-readable media for media delivery from a cloud or IP-based distribution network, or CDN (Content Delivery Network) to MSO (Multiple System Operators) or head-ends that include cable and satellite delivery mechanisms. The present technology unifies cloud-based delivery (Mediampify cloud) with the cable-based mechanism (e.g. Comcast, Verizon FiOS).

An example computer-implemented method can receive, from a content provider such as a cable or satellite operator, a request for at least one media stream for playback on a broadcast media channel. The request for the at least one media stream can include a request for a plurality of multimedia items. Content corresponding to the plurality of multimedia items can be obtained from at least one website or cloud service that offers the content in at least one first format. Based on the obtained content, the plurality of multimedia files can be generated in a second format that is compatible with the content provider. The media stream can be assembled using each of the multimedia items generated in the second format. The media stream can be provided to the content provider for broadcast on the broadcast media channel.

US 10,524,002 B2

3

In some aspects of the present technology, obtaining the content can include retrieving at least one audio file corresponding to an audio component of the multimedia item and retrieving a plurality of images as screen captures that can correspond to a video component of the multimedia item. Generating the multimedia files in the second format can include combining the plurality of images and the at least one audio file to create each of the multimedia items.

BRIEF DESCRIPTION OF THE DRAWING

There are shown in the drawings, embodiments which are presently preferred, it being understood, however, that the disclosure is not limited to the precise arrangements and instrumentalities shown

FIG. 1 shows an example block diagram illustrating an exemplary head end (MSO);

FIG. 2 shows an example block diagram of the apparatus located at the MSO facility (mediaplug);

FIG. 3 shows an example method for initialization of Mediaplug components;

FIG. 4 shows an example method to obtain list of files to multicast to a cable operator MSO;

FIG. 5 shows an example method for encoding video and audio files;

FIG. 6 shows an example method for providing multimedia content to a cable operator;

FIG. 7 shows an example block diagram of example software components of the Mediaplug;

FIG. 8 shows an example method to multicast from a component in the Mediaplug;

FIG. 9 shows an example method to insert metadata into an MPEG Transport Stream;

FIG. 10 shows an example block diagram for a Virtualized Mediaplug

FIG. 11 shows an example method to initialize virtual mediaplug components; and

FIG. 12 shows an example possible system embodiment for implementing various embodiments of the present technology

While the present technology is amenable to various modifications and alternative forms, specifics thereof have been shown by way of example in the drawings and will be described in detail. It should be understood, however, that the intention is not to limit the present technology to the particular embodiments described. On the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present technology.

DETAILED DESCRIPTION

The following description should be read with reference of the drawings, in which like elements are numbered in similar fashion. The drawings are not necessarily at scale and are not intended to limit the scope of the disclosure. Those skilled in the art understand that there are alternatives that may be used as well which are also considered to be within the scope of the disclosure.

Generally, the disclosed technology pertains to a delivery mechanism for providing multimedia content to cable systems without the use of a satellite system in a fault-tolerant and reliable fashion. FIG. 1 illustrates a block diagram that includes a Mediaplugin Cloud 101. The cloud 101 can refer to a set of virtualized instances or non-virtualized instances of one or more servers that provide access to archives and digital files with media content stored in them. For example,

4

the cloud 101 can be implemented using VMWave or Xen and OpenStack, or simply reside in Amazon Web Services or Rackspace. In addition, special provisioning can be provided in cloud 101 for secured access.

The cloud 101 service files can be accessible via the internet 105 and can be stored in a cloud-based file system or cluster file system (e.g. GlusterFS or Hadoop, Amazon S3). These files can be multimedia files that include, as non-limiting examples, MPEG-4 Video files with H.264 encoded video and AAC encoded audio or MPEG-layer 3 audio as well (MPEG2 Video could also be used). These files can be accessed using a secured transport protocol such as, but not limited to, SSL or a synchronization tool as "rsync," commonly found in Linux/MacOS environments. The communication link 106 can be a DSL, Cable Modem, Wireless WAN, Fiber, Metro Ethernet, etc. and can be linked with the MediaPlug 110 unit. The Mediaplugin 110 can reside at the "MSO" facility and can be collocated with all the cable or satellite operator equipment. The MSO is a facility used by the Cable Operator to receive all TV and Radio Programming that the cable operator will then multiplex, re-encode, and distribute from the MSO to a set of subscriber units with set top boxes, or modulate the signal in TV analog channels (2-60+).

One function of the MediaPlug 110 is to provide multimedia content, e.g., video and audio, to an MPEG multiplexer at the headend 120. The content can be pre-formatted in MPEG Transport Streams that the multiplexer 120 (e.g. Arris TMX-2010) will then be able to capture, reformat in any way the operator requires, and deliver to the cable operator's subscribers via Coaxial Cable 125 or any other means suitable for a multicast or unicast delivery to its customers with "Set Top boxes" 130, 131, 132. A "Set Top Box" is customer premise equipment that will be able to decode and display in a TV set or other screen display, via a signal received by its HDMI or analog input, the contents of the TV programming the cable operator offers.

The Mediaplugin 110 can be a computer or server or set of computers of servers that can communicate with the Cloud 101 using a standard Local Area Network (e.g. Gigabit Ethernet). The Mediaplugin 110 can store, encode, and transmit multimedia files in the format that the cable operator at its MSO requires. This cable format may include audio, video, and metadata (subtitles, control data, and any additional component that can be packaged in an MPEG Packet that is part of the MPEG Transport Stream). In one embodiment, Mediaplugin 110 can include components such as a caching unit, and one or more multicasting servers.

FIG. 2 illustrates an example configuration of the hardware components of Mediaplugin 110 which includes three servers. A server is a computer device, usually rack mountable unit with volatile memory such as RAM, one or more micro-processors, and non-volatile memory such as a hard disk. However, one skilled in the art will recognize that the present technology is not limited to a particular configuration and can be varied according to different specifications. FIG. 2 illustrates Mediaplugin Cloud 202 which is connected to the Internet backbone 203. Link 204 (e.g. DSL, Cable Modem, Fiber) provides connectivity between cloud 202 and a Network Address Translator unit 205 (e.g. Cisco RVS4000, or any other NAT device) which can provide Gigabit Ethernet switching capabilities. The lines 206, 207, and 208 show Gigabit Ethernet links, however one skilled in the art will understand that Gigabit Ethernet or a Wireless LAN using 802.11g/n would be also a suitable interface to be used in a Mediaplugin configuration. The Caching unit 210 can be the main storage component in the system and it can

US 10,524,002 B2

5

contain all of the multimedia files in the format required by the cable operators, such as the Transport Stream format. The caching unit can communicate with Multicast-01 **215** and Multicast-02 **220**. The two Multicast units **215, 220** can pull or obtain information from the caching unit via TCP/IP or UDP, using protocols such as HTTP (Hypertext Transport Protocol) or RTSP (Real-time Streaming Transport). The caching unit can also encode and transcode the multimedia files it downloads from the Cloud **202**. The caching unit can also keep updated information of all the metadata and create customizable screens that can be converted into video formats compatible with the Transport Stream content that will be provided to the cable operator at its MSO. MPEG MUX MSO **230** can correspond to a program residing in a computer system that processes and handles the MPEG streams from multiple sources. The MUX **230** can also function as the bridge between the source of the signal and the customer's set top box units. A non-limiting example program would be the CherryPicker DM6400 by MOTOROLA. Each of the components or modules of the Mediaplugin can include software for performing functions and/or methods according to the present technology.

FIG. 3 illustrates an example method that can be used to configure and provision the units at boot-time. Each unit can initialize keys for SSH/IP Sec access at **305**. For example, the units can use IPSec and SSH Tunnel keys that can be stored in the administrator user account. The keys can be generated by the Mediaplugin Cloud that are private keys or certificates, shared secrets or similar components by which the Mediaplugin can connect securely with the Mediaplugin cloud. Each cable operator can provide custom-based IP addressing at **310** for the Gigabit Ethernet ports that connect to the operators' network. Changing the IP addresses of the Multicast-01 and/or Multicast-02 refers to the IP address by which the links **221** and **216** in FIG. 2 are linked to the MPEG MUX MSO unit. A change of the IP address can refer to the 'ifconfig' command which is used in Linux-based systems and that can be changed in an Ubuntu box at/etc/network/interfaces file. Similarly, the network routing can be changed if the multicast address cannot be routed to the default gateway which points to the **205** NAT, but to **221** and **216** respectively for Multicast-02 and Multicast-01. In general, the **239.*.*** network address may be routed to those interfaces.

At step **320**, the Mediaplugin can configure playlists per channel and multicast address. The playlist and channel configuration can be obtained from the cable/satellite operator and can include any number of channels for broadcasting multimedia content to end customers. As an example, the channels may include music channels corresponding to different music genres, e.g. rock, alternative, classical, jazz, hip hop, etc. Based on the particular genre, a corresponding playlist can be selected and/or configured. The Mediaplugin can also determine the multicast address for each of the channels.

In some configurations a customizable user interface (UI) can be added to the output video MPEG Transport stream. As such, if the "Custom UI" variable is then true and valid, **325**, a link or URL **330** is loaded into the Mediaplugin configuration that can be used when the video interface is created. The custom video interface can be created and fetched as needed or it can be created in real-time for a particular channel or for all the channels. The custom user interface variable can be uniquely configured per channel in the Mediaplugin. The mediaplugin can also monitor **340** all of the processes to ensure stability and continuous, uninterrupted operation. The initialization of the Mediaplugin can

6

also include setting the RSYNC profile **350**. The RSYNC profile can be used to update and synchronize media content with the cloud when the Mediaplugin determines that the content has changed or been modified.

As described above, the Mediaplugin can broadcast the multimedia MPEG stream using the programmed IP Address in multicast-01 and multicast-02 according to the method illustrated in FIG. 3. In FIG. 4, an example method is shown for obtaining the list of files to playback or broadcast to the MSO system. The file type or format requested by the multicasting embodiment, e.g. multicast-01 or multicast-02, can include Transport Stream (e.g. TS) and/or MPEG-4 Part 14 (e.g. MP4) values **405**. This request is executed from any of the multicasting embodiments, usually making use of the HTTP GET Request. The multicasting unit can connect to an HTTP Server and retrieve the list of files based on the type identified by the request. As the caching unit may handle multiple channels of different types and name, each channel can have a unique identifier corresponding to the name of the playlist. For example channel name "Cool" or "Rock," **410** are names that can identify the channel to be retrieved from the caching server. The activity log in the request can also be updated keeping track what step of the process is being completed **415**. At **420**, the caching server can generate a "playlist file" of mime type "audio/x-mpegurl" which will be interpreted as a multimedia playlist file when requested. Finally, as the playlist usually may not vary for a period of time until a new request is received, the playlist can be kept in the local memory caching service **425**.

As the requests arrive at the "Caching" unit from the multicasting units, the "Caching" unit can be configured to perform parallel encoding for all the different channels available from the unit. FIG. 5 illustrates an example method that can be implemented by the Mediaplugin, and in particular, by the "Caching" unit. At step **505**, encoding flags or input parameters can be received which may include GOP (Group of Pictures) which is used by certain video encoders, resolution (e.g. 640x480) in pixels, bit rate in Kilobits per second, any encoding type of video (e.g. H.264), encoding type of audio (e.g. AC-3), duration of the video feed, etc. In addition, a "format" parameter can be received which refers to the output format or encapsulation, MPEG TS or MPEG4. The caching unit encodes audio and video, and can use the disk space in the Mediaplugin. Hence in **508** the method checks for sufficient space left in the memory or hard drive. This task can be run on a thread together with all the other tasks that are working in parallel in order to speed up the processes of encoding and transcoding.

Several indexes or identifiers by the name "ChID," (these integers preferably no more than an integer value from **1** to the maximum number of channels) begin (e.g. MAX could be **50**) to be generated by the caching unit **510**. This method can be performed for all the channels and all media linked to that particular channel identified by "MediaID" **515**, in this particular case the "MediaID." The variable "MAX" in this context can refer to the maximum number of media files read (e.g. **500**). The data captured from the media could be "Artist," "Author," or any other variable from an audio file. If a video file is used this could be "program name," "movie name," or any associated metadata information tied to the multimedia file and the Channel Identifier (ChID) associated with it "MediaID" file.

The caching unit can add a "Custom UI," or a customizable User Interface that is designed using Hypertext Markup Language (e.g. HTML) or a web service. A web service that matches a REST interface (Representation State Transfer) and uses Javascript Object Notation (JSON) or

US 10,524,002 B2

7

XML (Extensible Markup Language) to respond to a custom UI request can be used. Moreover, this custom UI can include JavaScript, CSS, and HTML content files that can be rendered using a browser using WebKit (e.g. PhantomJS or Safari) or any other browser-rendering engine compatible with HTML4/5 or any future HTML version. If a customizable UT is part of the configuration **518**, rendered images from a web service can be used to create overlapping video or replace any video in the video feed file. Likewise, this video or customized screen can be used when no video at all is defined in the multimedia file, e.g. Audio-only file. The web-service to be contacted may use the "PlaylistName" to be used for the Channel identified by the channel ID "ChID," retrieved using the "PlaylistID" function **520**. Each new PlaylistName may contain different identifies to retrieve new images. Therefore one or more screens will be retrieved from the rendering engine as such **522** can be used to retrieve a set of screens from the web-service being contacted.

Capturing screens **526** can be done using a tool similar to PhantomJS (Webkit-based headless engine for rendering images) or QtCapture, or simply capturing screen frame buffers from a machine running any software tool. Alternatively, the method can capture a video file from another rendering function, which can create a sequence of screens that can be used to generate a video file. Once all screens and video are captured, **530** the resulting output created can be a video file that then is either replaced or merged (multiplexed) with the original file. As an example, if an audio content file is used such as an MP3 file, this function can create a video file from all the screens captured by **526**. Those screens are generated by calling a URL where the web-service is associated and creating a video file with the input audio and the captured screens. The output format can be an MPEG Transport Stream file that can be retrievable or streamed to the multicasting embodiment.

Once the output file containing the screen shoot sequence of videos encoded in H.264 (for example, or MPEG2Video) is created, the audio can be multiplexed and added to one of the encoding channels of the resulting video output and saved into the caching unit storage disk **550**. The multiplexing can be done using tools such as FFMPEG or VLC or any proprietary tool that can multiplex audio and video files into a resulting MPEG Transport Stream (or MPEG-4 output, or any similar output).

FIG. **6** illustrates an example method **600** for providing multimedia content to a cable operator. At **602**, a device such as the Mediaplugin device receives a request from a cable/satellite operator for a media stream. A media stream can be formatted according to HTTP Live Streaming (HLS), an HTTP/RTSP playlist, or an RTSP stream. The media stream can include any number of multimedia items such as audio/video multimedia items. In some instances, the request can include a channel identifier that is associated with a particular category of multimedia items, such as a music genre corresponding to rock, classical, jazz, etc.

At **604**, contents corresponding to the multimedia items can be retrieved, downloaded, or otherwise obtained from one or more websites or cloud based servers. The contents can include data corresponding to either or both of the audio and video components of the multimedia items. For example, the data corresponding to the audio portion can include a music file in an MP3 format. Similarly, the data corresponding to the video portion can include a video that is obtained directly from the cloud. Alternatively, the video component can be assembled by using one or more screen captures from a website. The screen captures can include

8

captures of still images presented by the website as well as captures of images from videos that are available on the website. As illustrated in FIG. **5**, the screen capture process can be repeated any number of times.

The Mediaplugin can then generate **606** multimedia items in a second format based on the content obtained from the web server or cloud. That is, the Mediaplugin can convert/encode the audio file into a format that is suitable for the cable operator and it can also assemble the various screen captures to generate the video component of the media item. Once the audio and video files are created, they can be combined, multiplexed, or encoded to a format that can be utilized by the multimedia stream. For example, the audio can be multiplexed and added to one of the encoding channels of the video output. The various multimedia items can be used **608** to assemble the media stream in a format corresponding with the request (e.g. HLS, HTTP/RTSP, RTSP stream). The media stream can be provided **610** to the content provider for broadcast on the broadcast media channel.

FIG. **7** illustrates a block diagram of an example architecture of the Caching Unit components. The figure shows **720** as the main operating system, which can be Linux, Windows, or any other operating system in use. The caching unit can contain two types of operating system services HTTP **710** and RTSP **712**. The RTSP service provides access so the media saved or created in the repository of the caching unit file system **730**. The RTSP server can be implemented using the Live555 RTSP server or similar. Alternatively, the HTTP Server can be implemented using "nginx" or "apache" web browsers, and the RTSP Servers can be implemented using Live555 services. In the case of the HTTP Server, this can also be used to get the Playlist of files generated from the method on FIG. **5 715**.

The GetPlaylist **715** command can be implemented as follows:

- Receive and process a request for the name of the channel or playlist to be used,
- Retrieve all the filenames ordered by date of creation of all the files in a particular directory,
- Cache all the files into cache memory (e.g. Memcache or similar tool),
- Add type of HTTP header as M3U or PLS (e.g. a text file with a list of files or URLs to play),
- Return a list of HTTP or RTSP URLs to access each video generated or stored in the caching unit.

In some embodiments, the HTTP Server can also be used to clean up, update, and delete operations. The block unit called "CleanUp" **722** is in charge of keeping up to date caching information data and/or deleting files already used or that are not required to be used anymore by the multicast unit(s). Also the HTTP Server functions can include servicing M3U8 files corresponding to the HTTP Live Streaming format and all the segments created by the HTTP Live Streaming function. As such, a HTTP Live Streaming (HLS) generation program will produce the files and segment them as needed by the encoder and can then be serviced by the HTTP Server **710**. The Mediaplugin Parallel Encoder **705** corresponds to a block that can implement the method shown in FIG. **5**. The resulting outputs from the Media Parallel Encoder **605** are stored in the file system **730**. The Mediaplugin Parallel encoder **605** can be implemented in Python and can rely on web services (such as Twisted or Django), PhantomJS, and FFMPEG. Monit **750** corresponds to a monitoring block that can ensure that the process running the "Media Parallel encoder" keeps running at all times and does not stop upon failure. Additional blocks that can be included in Mediaplugin are RSYNC **742** and SSH **744**.

US 10,524,002 B2

9

These blocks can be used to ensure synchronization and updates of new media files coming from the cloud. The RSYNC block uses the “rsync” protocol and program over ssh (Secured Socket Layer Shell) to download media from the “Mediamplify Cloud” and synchronize multimedia files that are newly updated and could be part of particular Channel ID and a Playlist Name. Each new file synchronized with the Mediamplify cloud has a Media ID value which the cloud can assign, and include additional metadata information such as “Artist” and “Author.” The “Mediamplify Parallel Encoder” also may receive requests from other components such as getStatus or StartEncoding, indicating what file is being encoded, its progress, and what other files are in the queue for encoding.

In some embodiments, the information from getStatus or StartEncoding can be generated using a JSON message retrieved upon a request via HTTP GET to the Mediamplify Parallel Encoder. A graphical user interface can then be used to plot the status of each song being encoded or provide previews of the videos being generated. Below is one non-limiting example of a JSON response made by the “Mediamplify Parallel Encoder” service:

```
{“song_encoded”: “ts”, “artist”: “Madonna”, “duration”: 0, “exists_on_disk”: True, “genre”: “Pop”, “title”: “Like a Virgin”, “music_id”: “1082863”}
```

Response regarding the status for a song being encoded or a list of songs being encoded, as follows:

```
{“music_id”: “109281”, “encoding”: “78%”, “output_file”: “/var/www/egllacast/pop/109281-abcdef-uuid.ts”, “thumbnail”: “http://mediamplify.com/images/abcdef-uuid.png”}
```

In some embodiments, there are two possible TCP/IP sockets open for listening used by the Caching component of the Mediaplugin. A) Port 80 (as an example), or the standard HTTP port, used to retrieve content, view video content generated, update/delete cached files, and access the playlists, and B) Port 9553 (as an example), or the standard Mediamplify Parallel encoder port, used to monitor and review status of the songs being encoded, the data being collected, and all other related activities with encoding. Via this port, encoding/transcoding can be started or stopped.

Similarly, the “Mediaplugin” components labeled “Multicast-01” and “Multicast-02” can also provide an HTTP interface to access the current song being played or being multicast via the HTTP Server 770. For example, the HTTP Server may run a PHP or Python script that will return the current song being played by the streamer. The streamer block 768 can connect to the caching server via HTTP or RTSP protocols. The implementation of this block can be done using a combination of FIG. 8 instructions and using multicasting services available in VLC or FFMPEG (e.g. multicast, or tsools/tsplay).

One function of the streamer 768 is to read the files encoded by the caching unit, process those for multicasting and creating the MPEG Transport Stream multicast packets sent via the Ethernet port. Additionally, the multicast-01 and multicast-02 blocks contain a “Recording” function 790, which is in charge of recording the multicasting streams and recording for later use in case of failure on the system. This operates by listening into the UDP (User Datagram Protocol) multicast address of each individual channels being broadcasted, and recording the output into a file. This file is then stored and made accessible for future use. Monit 764 can maintain the streamer process running at all times throughout a regular broadcasting cycle.

FIG. 8 illustrates an example method that can be performed by the streamer module in the Mediaplugin. In one embodiment, the streamer can perform functions that

10

include: connect to the caching unit; retrieve generated videos per playlist; determine if unit is working in fault-tolerant mode; playback pre-recorded audio if a failure is detected; and record units own output or any other streamer output if connected to switch or hub.

Turning to the method from FIG. 8, at step 805, the streamer reads the multicast address associated to the channel being broadcasted, in order to generate IGMP (Internet Group Management Protocol) messages and establish a multicast group (e.g. 239.0.0.1:8001). The channel name is also initialized at 810. This value can be used to retrieve the proper content from the Caching server. In one embodiment, a configuration table can be created with tuples: <ChID>, <Channel Name>, <Multicast Address>, <Multicast Port>, such that no duplicates are used or created. Automation software can create these configuration tables and automate the process of creating all streamer calls. A streamer method can be called (e.g. from /etc/init.d/) at boot time and monitored with a tool such as Monit. Once the configuration is known, the streamer will check 850 if there are already multicasting transport streams generated by any other multicast component and avoid collisions. If such multicast data is detected, a delay can be used to delay a new attempt to check on that particular multicast group 860. Hence this block indicates “Wait <Time>” or stop all processing for the amount of <Time> recommended by the specification. For example, the time for delay can be the “Main source” of input less the amount of time for receiving input from a backup server. During initialization, if the same multicast address is used to broadcast main and backup, the one with less amount of time delay will be first, followed by the “Backup.” In cases where a hot spare is used, or two different multicast addresses can be in use, one for main and the other one for backup, then this step may be optional.

Once it is determined that there are no elements multicasting at the same address, the streamer will check if the Caching server is “alive” 815. This can be performed in multiple ways, for example an HTTP Request must not generate a 404 or 403 error, but instead a successful 202 value. Likewise, ping requests can also be used before making an HTTP GET call to identify if the caching server is present and available. In the case that there are no ping responses, a timeout will trigger a failure and a pre-recorded content 820 will be played or sent to the multicast address and port of record. Recordings can be scheduled daily for an extended period of time (e.g. 8 hours or more). These recordings can store the recorded file in a temporary folder and then move that file over to the playback folder where streamer can read it.

If it is determined that the caching server is functioning properly, then there are three types of streams that can be sent to the multicast address and port:

HTTP Live Streaming (HLS) a multi-segment stream usually defined by extension M3U8, containing a set number of segments for a real-time stream source (Video/audio)

HTTP/RTSP file playlist, this means that one or many files are read from the caching server, usually defined by extension M3U or PLS

RTSP stream, this could be constant stream similar to HTTP Live Streaming (HLS) but using an RTSP protocol instead of HLS stream.

When multiple files are part of a playlist, the bit-rate may vary between playlist files. Different bit-rates among files in a playlist could cause problems in non-CBR (Constant Bit Rate) systems. To avoid such problems, the Mediaplugin can maintain the stream at the bit rate of the previous stream by adding null MPEG TS packets at the same rate as the

US 10,524,002 B2

11

previous media file was being sent over the multicast port. This will maintain appropriate bit-rate for playlists.

In RTSP service, streaming occurs by writing the media content to a named pipe or a FIFO file and allowing the RTSP service to read from that file and generate an RTSP stream. As such, in HLS and RTSP Streaming, the caching server knows the filename of the file being streamed. When the file is streamed from via playlist, both the caching and multicast unit know the name of the file being streamed, as the multicast server can then request from the caching server the metadata associated with the file name being streamed.

Accordingly the present technology can accommodate both cases and retrieve the current file name being streamed as a JSON response. The response can identify the content displayed or played back at the end-user's set top box.

If HLS is used for streaming **818**, the streamer will connect to the proper M3U8 file **820** to pull all the segments corresponding to the live playlist file on the M3U8 stream. Programs and libraries such as FFMPEG or VLC can then be used to encapsulate the stream feed to an MPEG TS and multicast it to the pre-programmed multicast address and port. Similarly, if an RTSP stream **825** is used, the same process takes place at **830** using the RTSP as a protocol instead of HTTP.

If a list of files in a playlist **832** is in use "M3U," the list can be obtained using an HTTP GET request made to the caching server. Each file can then be streamed to the multicast address and port of record **838** until all elements of the playlist are completed and played back **835**.

In some instances, metadata can be added to the stream as an additional PID (Program Identifier) in the MPEG Transport Stream. If metadata is needed **841** then it will be inserted **842** here in accordance with FIG. 9.

For example, the metadata added could be as simple as a string containing Artist, Name, or other information, or as complex as DCII Text (DigiCipher version 2, as specified by Motorola), MPEG TS Control messages (Two-way), or any other metadata or PID that can be inserted into the stream being sent to the multicast address and port.

FIG. 9 illustrates an example method used to insert a special data packet into the stream being written to the multicast address, without making use of timing information. Therefore, the metadata can be inserted such that it does not have timing dependency. Alternatively, certain type of metadata such as subtitles or lyrics may need to be inserted only at certain times so that it is synchronized with the video data.

The sample MPEG TS packet shown in **902** and **905** corresponds to the metadata inside the packet. This matches standard MPEG TS packet structure and may vary depending on the type of metadata being added to the stream.

The method to insert the packet starts by reading the MPEG TS file retrieved from the caching server **910**. If metadata corresponds to only song information, this information can be added at uniform times. Non-uniform timing inclusions can also be done by adding a table that includes the time and the metadata content added to the stream **915**. This metadata can be retrieved from the Caching unit using an HTTP Get request directed to the caching server **918** and may include artist name, title of song, or any other information **920**.

Creation of the MPEG Packet(s) **925** to be inserted at the appropriate times, which will set the places with those packets, will be inserted in the MPEG TS. Modify the PMT (Program Management Table) of the generated file and include the PID being inserted into the MPEG TS **930**. At

12

step **932**, an audio PID may include a metadata PID of type DCII Text, which is added to the stream.

For each timing or location where the PID needs to be inserted **935**, the method can insert those packets and generate a buffer that can be sent to the multicast address as in illustrated in FIG. 8.

As described above, the disclosed technology can be implemented using a set of modules included in the Mediaplugin apparatus, which can be implemented using one or more servers accordingly. Alternatively, virtualization environments such as VMWare or Xen (FIG. 9) can include a system and architecture that can be used to host all the Mediaplugin functionality in a virtualized environment using a single server with multiple GbE ports.

The illustration in FIG. 10 shows a configuration of three virtual machines used for implementing the Mediaplugin. Caching **1030** and its respective Ethernet port **1032**, Multicast-01 **1020** with two virtual Ethernet ports **1022** and **1024**, and Multicast-02 **1010**, with two Ethernet ports **1012** and **1014**. There are three bridges that include **1007**, **1008**, and **1009**. The multicast Ethernet ports **1024** and **1014** are bridged with eth1 **1004** and eth2 **1003** respectively via bridge **1009** and bridge **1008**, respectively. That is, the same content generated at **1024** can be forwarded via the bridge **1009** to the physical interface eth1 **1004** in the hardware. The vSwitch **1045** or the virtual switch between the two bridged interfaces **1008** and **1009** permits the two virtual machines of Multicast-01 **1010** and Multicast-02 **1020** to record each other and monitor their activity.

To achieve concurrency and proper execution in a cloud environment, the Mediaplugin can use the example method shown in FIG. 11. Once a create unit command is issued **1105**, provisioning of the metadata information and proper IP Address is provided for that particular instance **1110**. Once an IP Address and metadata has been chosen for the unit, the server can be allocated in the cloud and the instance can be created using CloudStack or OpenStack tools and services **1115**. The instance is then allocated and runs in the proper node in the cloud **1120**. Configuration of the assigned IP, Netmask, and VPN credentials are loaded at **1125**. Finally, certificate information is uploaded **1130** as well as all services run to enable the unit for execution of its caching functions, multicasting functions, or both.

FIG. 12 illustrates a conventional system bus computing system architecture **1200** such as can be used in Mediaplugin **110**. Those of ordinary skill in the art will appreciate that other system architectures are also possible.

In computing system architecture **1200** the components of the system are in electrical communication with each other using a bus **1205**. Example system **1200** includes a processing unit (CPU or processor) **1210** and a system bus **1205** that couples various system components including the system memory **1215**, such as read only memory (ROM) **1220** and random access memory (RAM) **1225**, to the processor **1210**. The system **1200** can include a cache of high-speed memory connected directly with, in close proximity to, or integrated as part of the processor **1210**. The system **1200** can copy data from the memory **1215** and/or the storage device **1230** to the cache **1212** for quick access by the processor **1210**. In this way, the cache can provide a performance boost that avoids processor **1210** delays while waiting for data. These and other modules can control or be configured to control the processor **1210** to perform various actions. Other system memory **1215** may be available for use as well. The memory **1215** can include multiple different types of memory with different performance characteristics. The processor **1210** can include any general purpose processor and a hardware

US 10,524,002 B2

13

module or software module, such as module 1 **1232**, module 2 **1234**, and module 3 **1236** stored in storage device **1230**, configured to control the processor **1210** as well as a special-purpose processor where software instructions are incorporated into the actual processor design. The processor **1210** may essentially be a completely self-contained computing system, containing multiple cores or processors, a bus, memory controller, cache, etc. A multi-core processor may be symmetric or asymmetric.

To enable user interaction with the computing device **1200**, an input device **1245** can represent any number of input mechanisms, such as a microphone for speech, a touch-sensitive screen for gesture or graphical input, keyboard, mouse, motion input, speech and so forth. An output device **1235** can also be one or more of a number of output mechanisms known to those of skill in the art. In some instances, multimodal systems can enable a user to provide multiple types of input to communicate with the computing device **1200**. The communications interface **1240** can generally govern and manage the user input and system output. There is no restriction on operating on any particular hardware arrangement and therefore the basic features here may easily be substituted for improved hardware or firmware arrangements as they are developed.

Storage device **1230** is a non-volatile memory and can be a hard disk or other types of computer readable media which can store data that are accessible by a computer, such as magnetic cassettes, flash memory cards, solid state memory devices, digital versatile disks, cartridges, random access memories (RAMs) **1225**, read only memory (ROM) **1220**, and hybrids thereof.

The storage device **1230** can include software modules **1232**, **1234**, **1236** for controlling the processor **1210**. Other hardware or software modules are contemplated. The storage device **1230** can be connected to the system bus **1205**. In one aspect, a hardware module that performs a particular function can include the software component stored in a computer-readable medium in connection with the necessary hardware components, such as the processor **1210**, bus **1205**, display **1235**, and so forth, to carry out the function.

It can be appreciated that example system **1200** can have more than one processor **1210** or be part of a group or cluster of computing devices networked together to provide greater processing capability.

For clarity of explanation, in some instances the present technology may be presented as including individual functional blocks including functional blocks comprising devices, device components, steps or routines in a method embodied in software, or combinations of hardware and software.

Any of the steps, operations, functions, or processes described herein may be performed or implemented by a combination of hardware and software modules, alone or in combination with other devices. In an embodiment, a software module can be software that resides in memory of a client device and/or one or more servers of a content management system and perform one or more functions when a processor executes the software associated with the module. The memory can be a non-transitory computer-readable medium.

In some embodiments the computer-readable storage devices, mediums, and memories can include a cable or wireless signal containing a bit stream and the like. However, when mentioned, non-transitory computer-readable storage media expressly exclude media such as energy, carrier signals, electromagnetic waves, and signals per se.

14

Methods according to the above-described examples can be implemented using computer-executable instructions that are stored or otherwise available from computer readable media. Such instructions can comprise, for example, instructions and data which cause or otherwise configure a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Portions of computer resources used can be accessible over a network. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, firmware, or source code. Examples of computer-readable media that may be used to store instructions, information used, and/or information created during methods according to described examples include magnetic or optical disks, flash memory, USB devices provided with non-volatile memory, networked storage devices, and so on.

Devices implementing methods according to these disclosures can comprise hardware, firmware and/or software, and can take any of a variety of form factors. Typical examples of such form factors include laptops, smart phones, small form factor personal computers, personal digital assistants, and so on. Functionality described herein also can be embodied in peripherals or add-in cards. Such functionality can also be implemented on a circuit board among different chips or different processes executing in a single device, by way of further example.

The instructions, media for conveying such instructions, computing resources for executing them, and other structures for supporting such computing resources are means for providing the functions described in these disclosures.

Although a variety of examples and other information was used to explain aspects within the scope of the appended claims, no limitation of the claims should be implied based on particular features or arrangements in such examples, as one of ordinary skill would be able to use these examples to derive a wide variety of implementations. Further and although some subject matter may have been described in language specific to examples of structural features and/or method steps, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to these described features or acts. For example, such functionality can be distributed differently or performed in components other than those identified herein. Rather, the described features and steps are disclosed as examples of components of systems and methods within the scope of the appended claims.

What is claimed is:

1. A computer-implemented method, comprising:
 - receiving, from a content provider, a request for at least one media stream for playback on a broadcast media channel, wherein the at least one media stream includes a plurality of multimedia items of different types;
 - obtaining content corresponding to the plurality of multimedia items from at least one source offering the content in at least one first format;
 - rendering a web page by a browser using the content;
 - generating a temporal sequence of screen captures of the rendered web page, where each screen capture defines all the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time;
 - assembling the at least one media stream using the temporal sequence of screen captures; and
 - providing the at least one media stream to the content provider for broadcast on the broadcast media channel;

US 10,524,002 B2

15

wherein the at least one media stream corresponds to at least one of an HTTP live stream (HLS), an HTTP playlist, and a Real-time Streaming Transport (RTSP) stream; and

wherein the web page is rendered in parallel in multiple threads. 5

2. The method according to claim 1, wherein the at least one media stream is interacted with MPEG two way control messages.

3. The method according to claim 1, wherein MPEG 10 messages are sent to a service provider via a unicast address.

4. A computer-implemented method, comprising:

receiving, from a content provider, a request for at least one media stream for playback on a broadcast media channel, wherein the at least one media stream includes a plurality of multimedia items of different types; 15

obtaining content corresponding to the plurality of multimedia items from at least one source offering the content in at least one first format;

rendering a web page by a browser using the content; 20

generating a temporal sequence of screen captures of the rendered web page, where each screen capture defines all the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time; 25

assembling the at least one media stream using the temporal sequence of screen captures; and

providing the at least one media stream to the content provider for broadcast on the broadcast media channel; wherein assembling the at least one media stream comprises inserting at least one MPEG packet including metadata corresponding to the multimedia item; and 30 wherein the web page is rendered in parallel in multiple threads.

5. The computer-implemented method of claim 4, 35 wherein the at least one media stream is interacted with MPEG two way control messages.

6. The computer-implemented method of claim 4, wherein MPEG messages are sent to a service provider via a unicast address. 40

7. A system, comprising:

a multicast server configured to generate parameters relating to at least one of video content, image content and audio content, and configured to output the parameters via MPEG outputs; 45

a caching server communicatively coupled to the multicast server, and configured to receive and store the parameters,

create a temporal sequence of screen captures of a rendered web page, where each said screen capture 50 defines content of the rendered web page at a given

16

time, and at least two adjacent screen captures in the temporal sequence illustrate a dynamic change of at least a portion of the video content, image content or audio content over time,

assemble the temporal sequence of screen captures, and provide at least one of the video content, image content and audio content to the multicast server for generating a multicast stream with MPEG encoded video and audio; and

a monitoring system configured to maintain generation of video and audio files, even in failure cases.

8. The system according to claim 7, further comprising multiple servers for load balancing and fault-tolerance.

9. The system according to claim 7, wherein the multicast server is selected to communicate a multicast stream available for a broadband network.

10. The system according to claim 7, wherein the video content is encoded in a video streaming format so as to generate a unicast stream.

11. The system according to claim 7, wherein the system generates audio MPEG files and add metadata in an MPEG frame.

12. The system according to claim 7, wherein the system generates MPEG metadata that is interacted with two way broadcast TV systems.

13. A computer-implemented method, comprising:

receiving, from a content provider, a request for at least one media stream for playback on a broadcast media channel, wherein the at least one media stream includes a plurality of multimedia items of different types;

obtaining content corresponding to the plurality of multimedia items from at least one source offering the content in at least one first format;

rendering a web page by a browser using the content;

generating a temporal sequence of screen captures of the rendered web page, where each screen capture defines all the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time; 40

assembling the at least one media stream using the temporal sequence of screen captures; and

providing the at least one media stream to the content provider for broadcast on the broadcast media channel;

wherein the at least one media stream corresponds to at least one of an HTTP live stream (HLS), an HTTP playlist, and a Real-time Streaming Transport (RTSP) stream; and

wherein the web page is rendered in parallel in virtual instances or virtual machines.

* * * * *

Exhibit C



US011140441B2

(12) **United States Patent**
Hernandez-Mondragon

(10) **Patent No.:** **US 11,140,441 B2**

(45) **Date of Patent:** ***Oct. 5, 2021**

(54) **METHOD, SYSTEM, AND APPARATUS FOR MULTIMEDIA CONTENT DELIVERY TO CABLE TV AND SATELLITE OPERATORS**

H04N 21/61 (2011.01)

H04N 21/6405 (2011.01)

H04N 21/643 (2011.01)

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

CPC ... *H04N 21/440218* (2013.01); *H04N 21/233*

(2013.01); *H04N 21/2387* (2013.01); *H04N*

21/234309 (2013.01); *H04N 21/4312*

(2013.01); *H04N 21/47217* (2013.01); *H04N*

21/6118 (2013.01); *H04N 21/6143* (2013.01);

H04N 21/8106 (2013.01); *H04N 21/8133*

(2013.01); *H04N 21/8153* (2013.01); *H04N*

21/84 (2013.01); *H04N 21/8547* (2013.01);

H04N 21/6405 (2013.01); *H04N 21/64322*

(2013.01)

(71) Applicant: **Edwin A. Hernandez-Mondragon,**
Coral Springs, FL (US)

(72) Inventor: **Edwin A. Hernandez-Mondragon,**
Coral Springs, FL (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(58) **Field of Classification Search**

CPC *H04N 21/233*; *H04N 21/234309*; *H04N*

21/2387; *H04N 21/4312*; *H04N*

21/440218; *H04N 21/47217*; *H04N*

21/6118; *H04N 21/6143*; *H04N 21/6405*;

H04N 21/64322; *H04N 21/8106*; *H04N*

21/8133; *H04N 21/8153*; *H04N 21/84*;

H04N 21/8547

See application file for complete search history.

(21) Appl. No.: **16/729,986**

(22) Filed: **Dec. 30, 2019**

(65) **Prior Publication Data**

US 2020/0322679 A1 Oct. 8, 2020

Related U.S. Application Data

(63) Continuation of application No. 16/152,606, filed on Oct. 5, 2018, now Pat. No. 10,524,002, which is a continuation of application No. 15/538,911, filed as application No. PCT/US2015/067464 on Dec. 22, 2015, now Pat. No. 10,123,074.

(60) Provisional application No. 62/095,504, filed on Dec. 22, 2014.

(51) **Int. Cl.**

H04N 21/4402 (2011.01)

H04N 21/84 (2011.01)

H04N 21/2343 (2011.01)

H04N 21/81 (2011.01)

H04N 21/8547 (2011.01)

H04N 21/233 (2011.01)

H04N 21/2387 (2011.01)

H04N 21/431 (2011.01)

H04N 21/472 (2011.01)

Primary Examiner — Rong Le

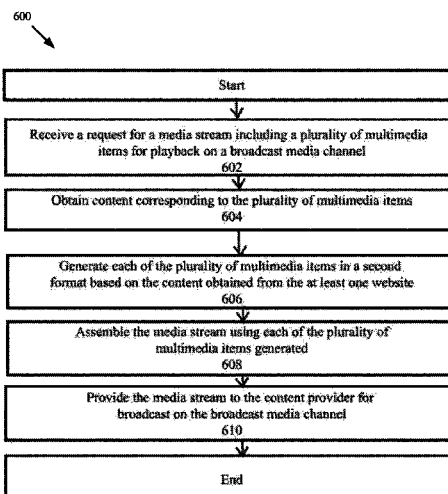
(74) Attorney, Agent, or Firm — The Concept Law Group, PA; Scott D. Smiley; Scott M. Garrett

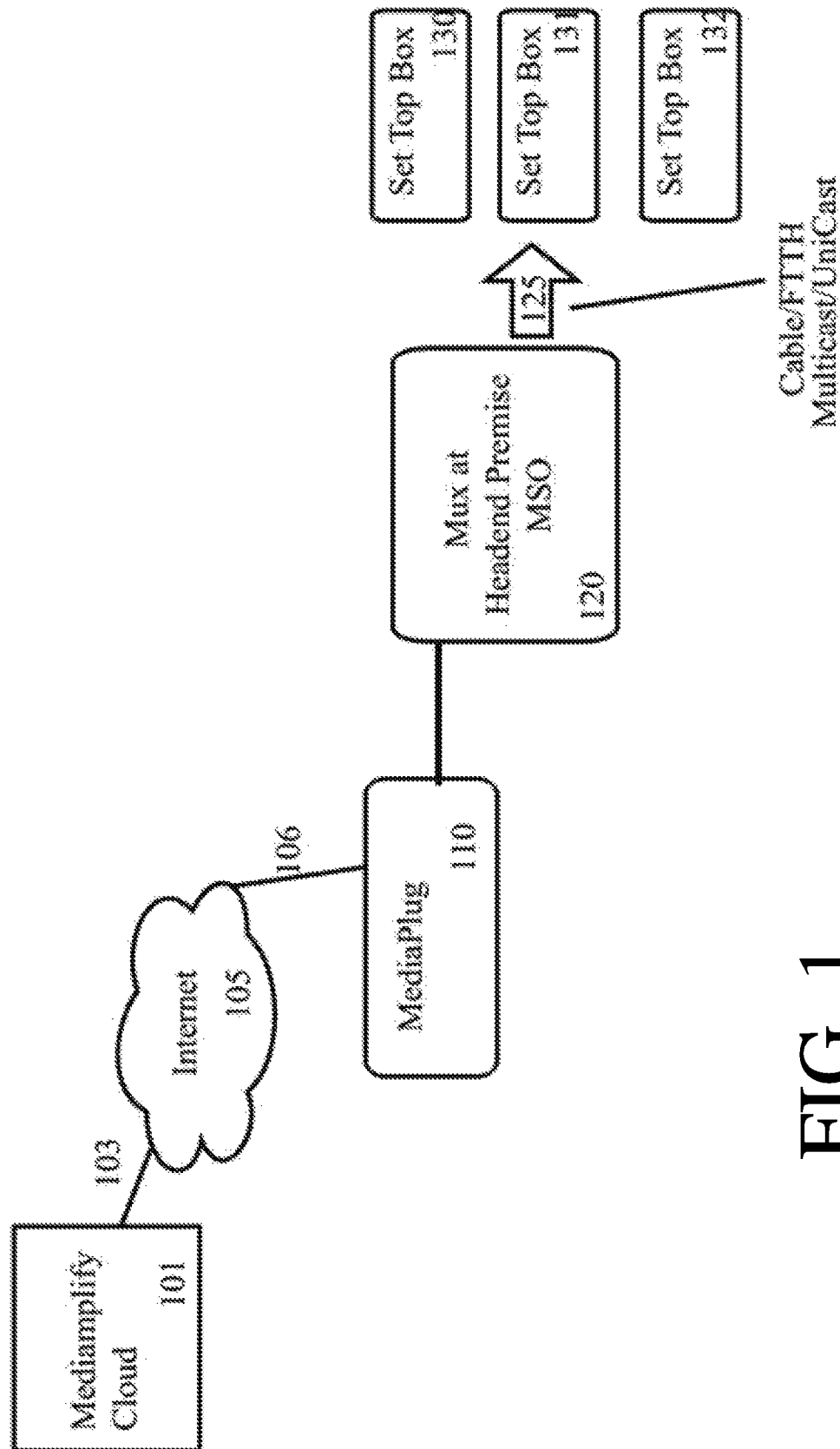
(57)

ABSTRACT

Systems, methods, and computer-readable media for delivering multimedia content from the cloud to cable operators are disclosed. A device located at the cable headend or implemented in the cloud can receive a request for at least one media stream for playback on a broadcast media channel. Content corresponding to a plurality of multimedia files in the media stream can be obtained from the internet or a cloud based service. The content can be used to generate the multimedia files in a format that is compatible with the cable operator. The multimedia files can be used to assemble the at least one media stream which can be provided to the cable operator for broadcast on the broadcast media channel.

26 Claims, 12 Drawing Sheets





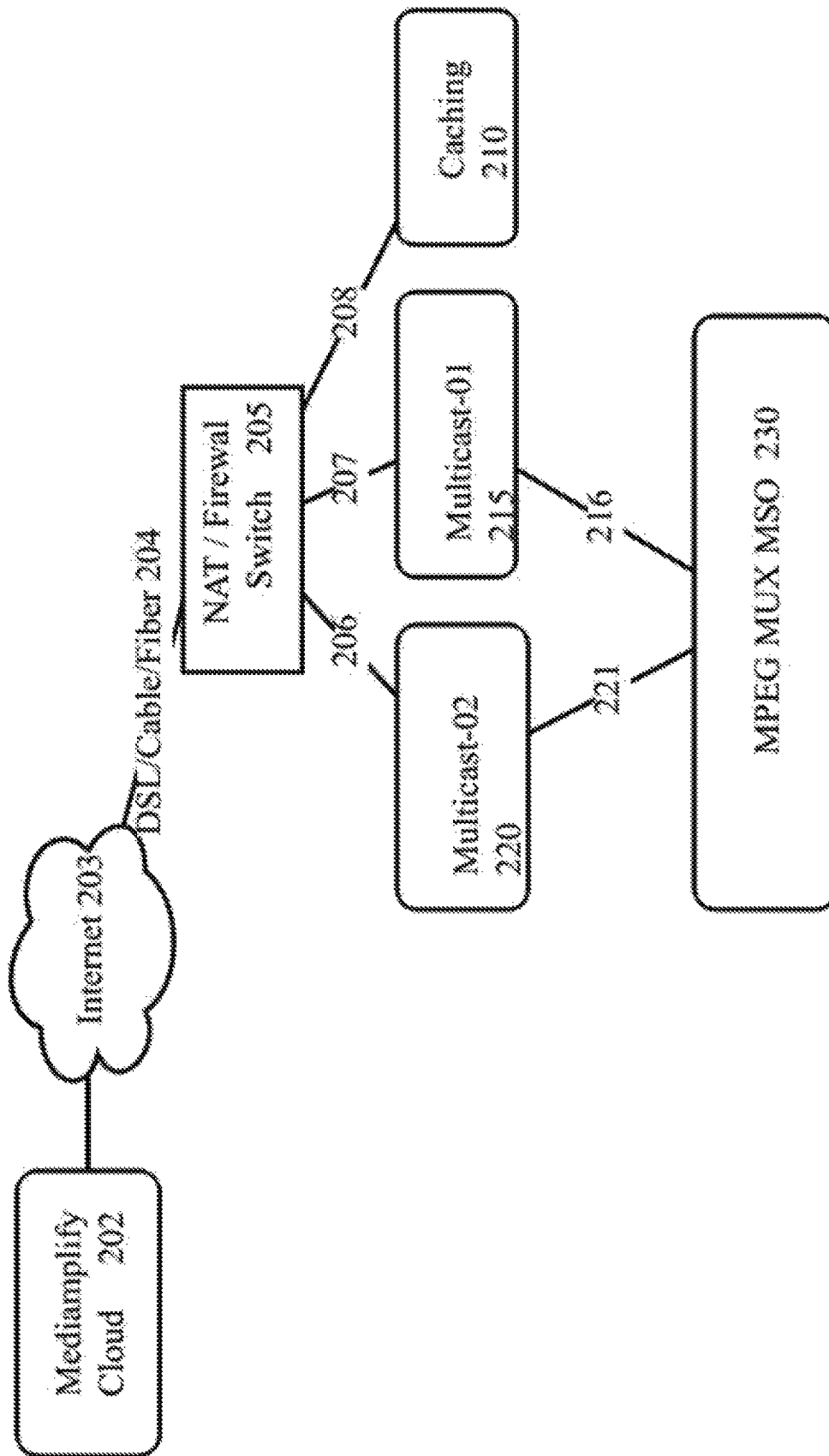


FIG. 2

U.S. Patent

Oct. 5, 2021

Sheet 3 of 12

US 11,140,441 B2

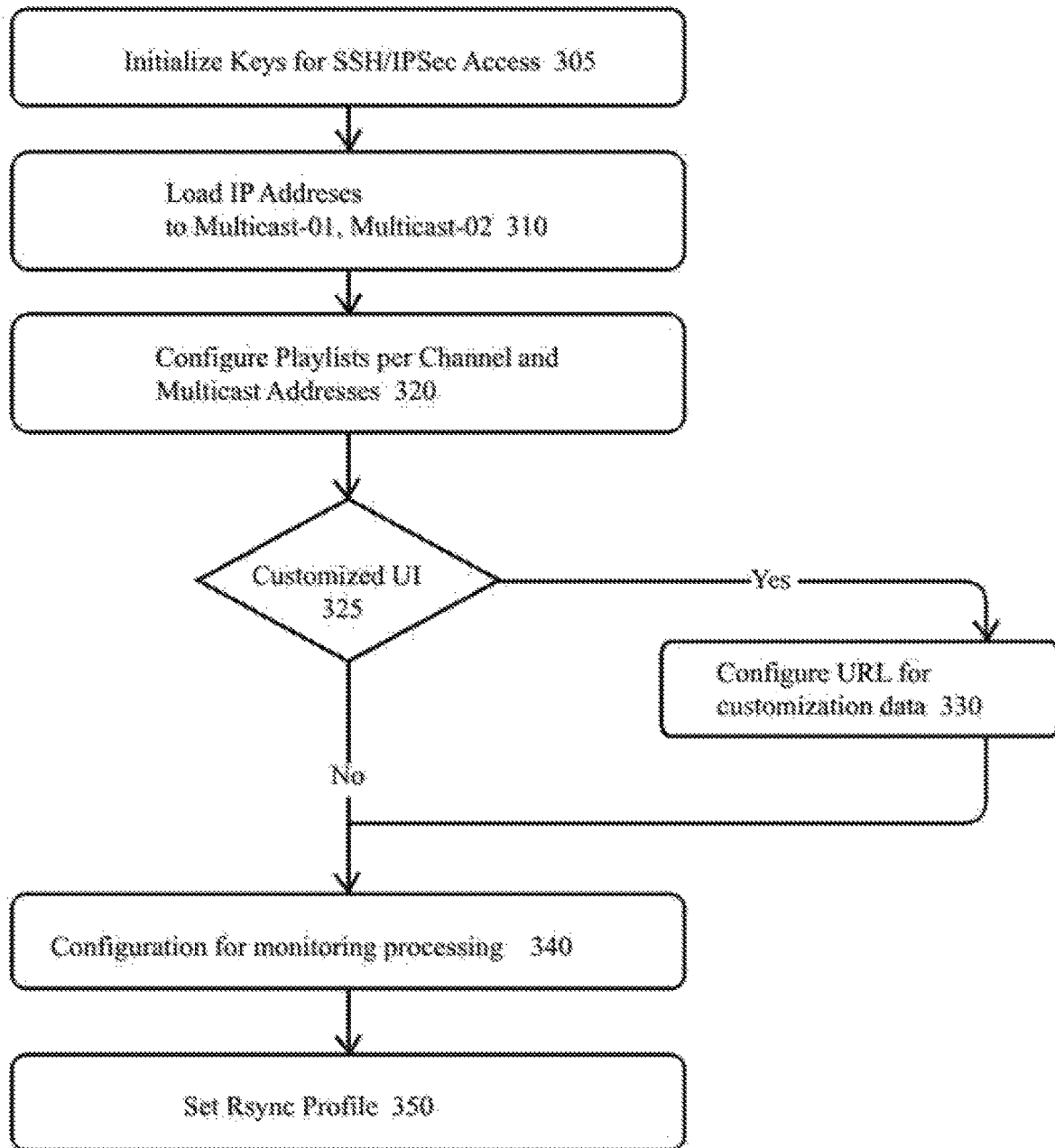


FIG. 3

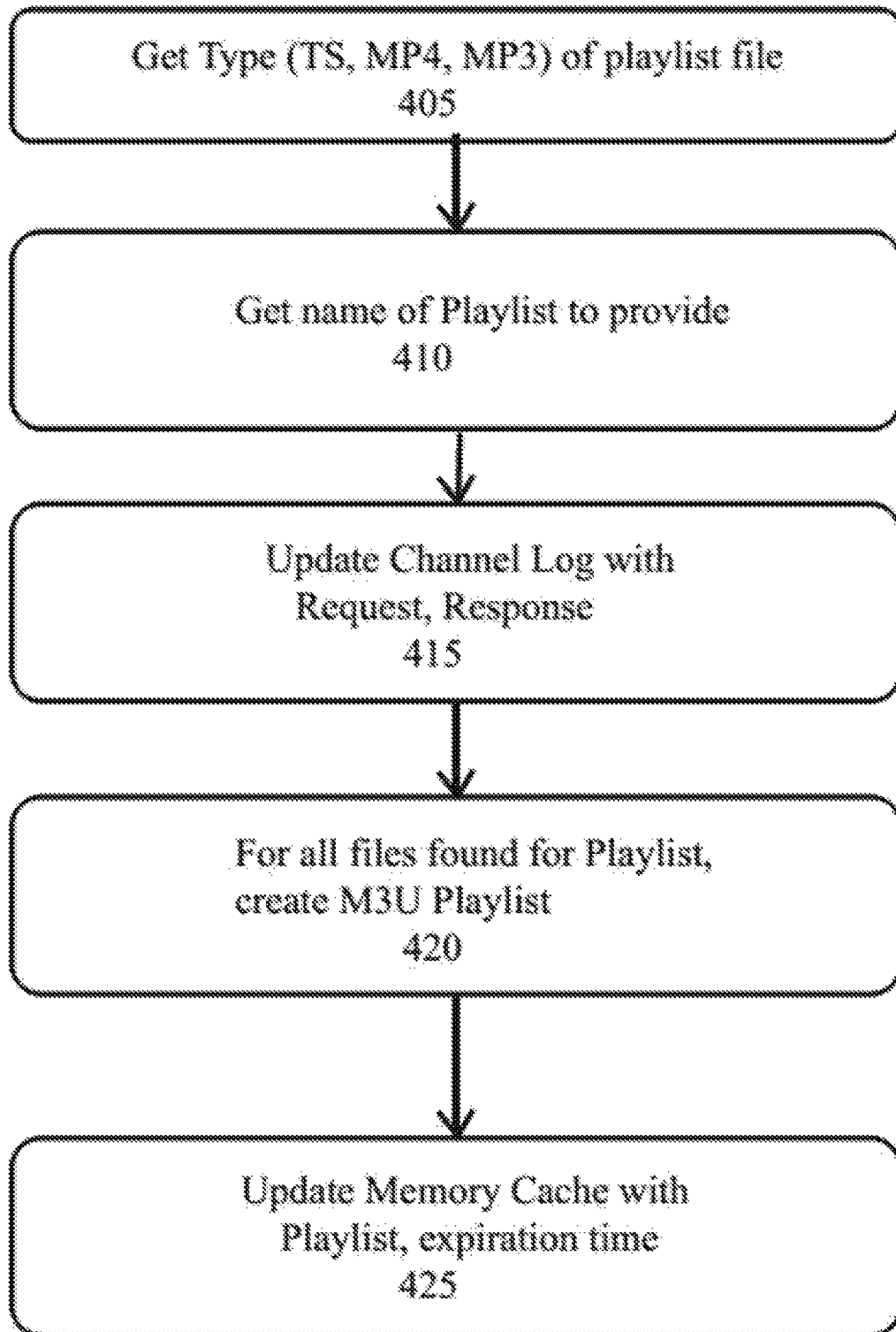


FIG. 4

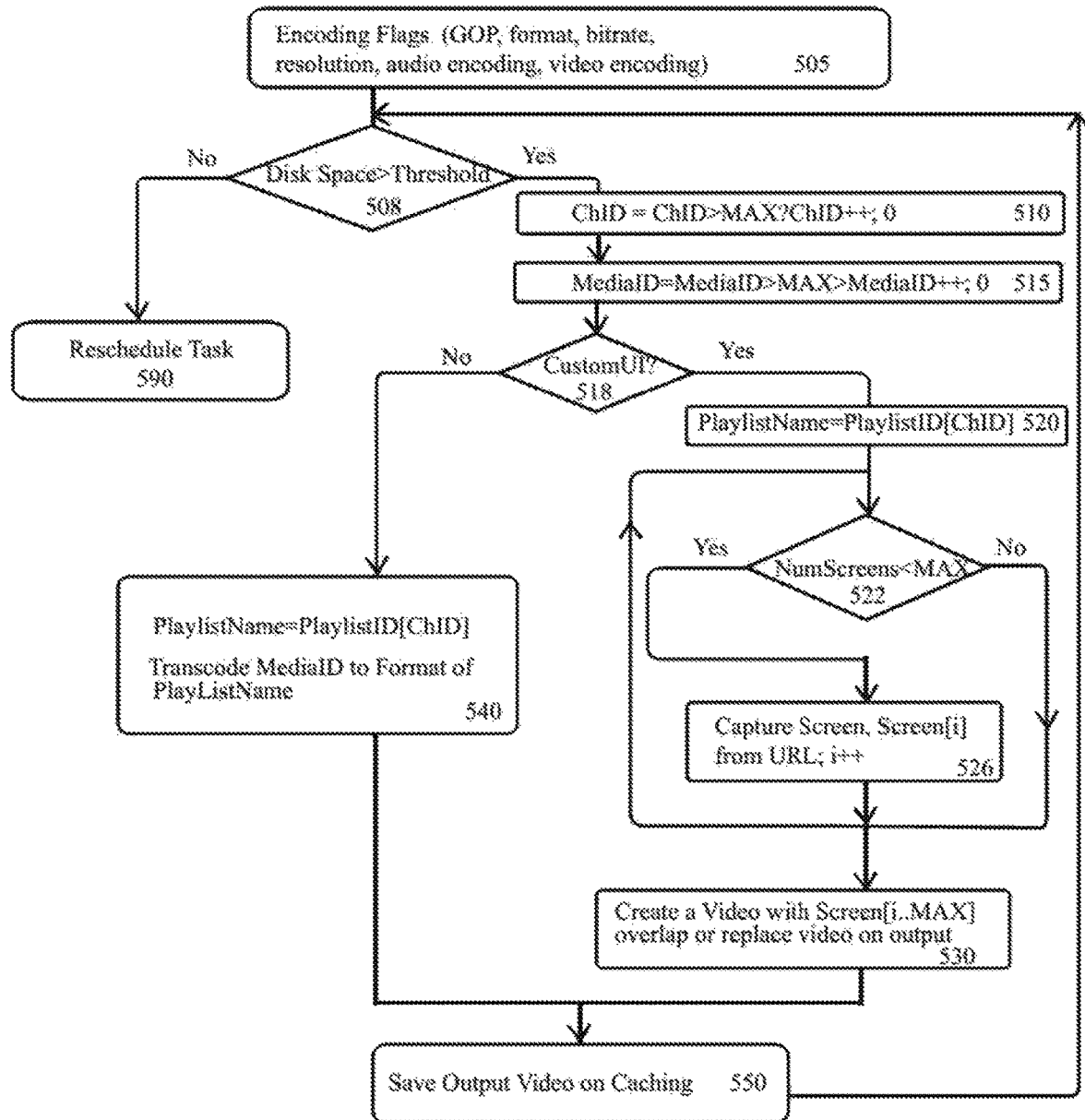


FIG. 5

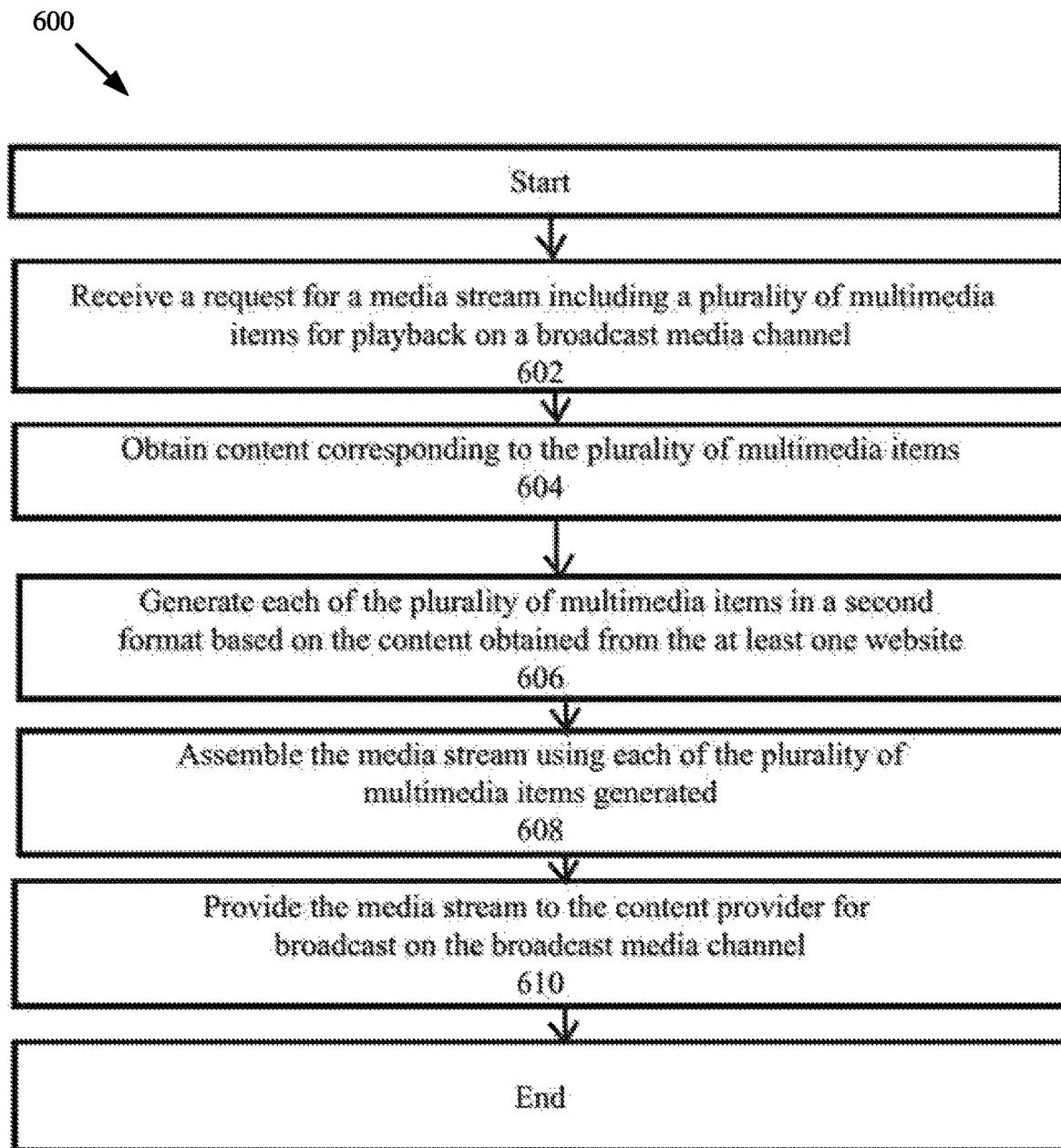


FIG. 6

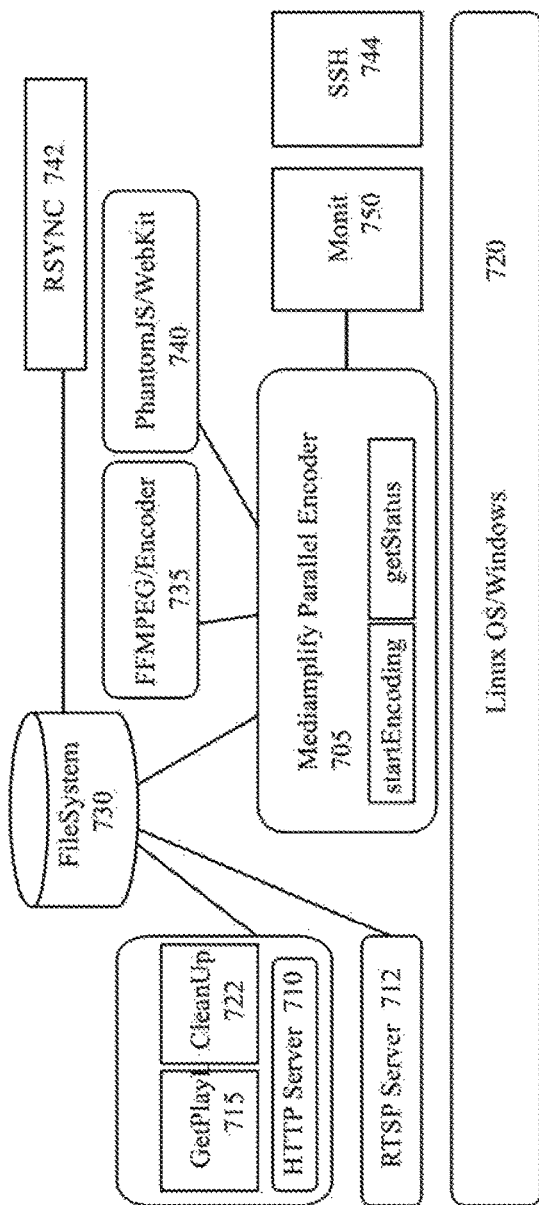
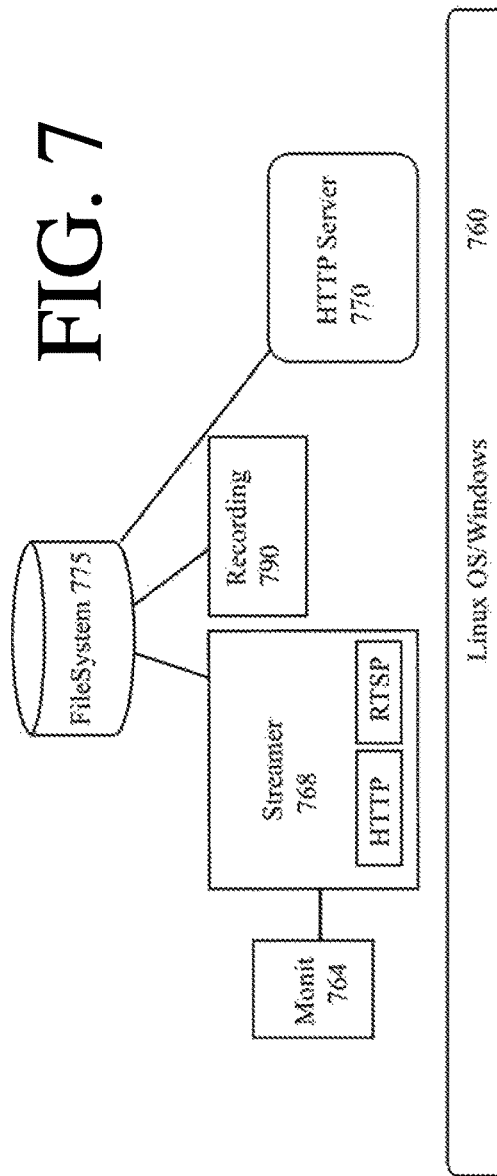


FIG. 7



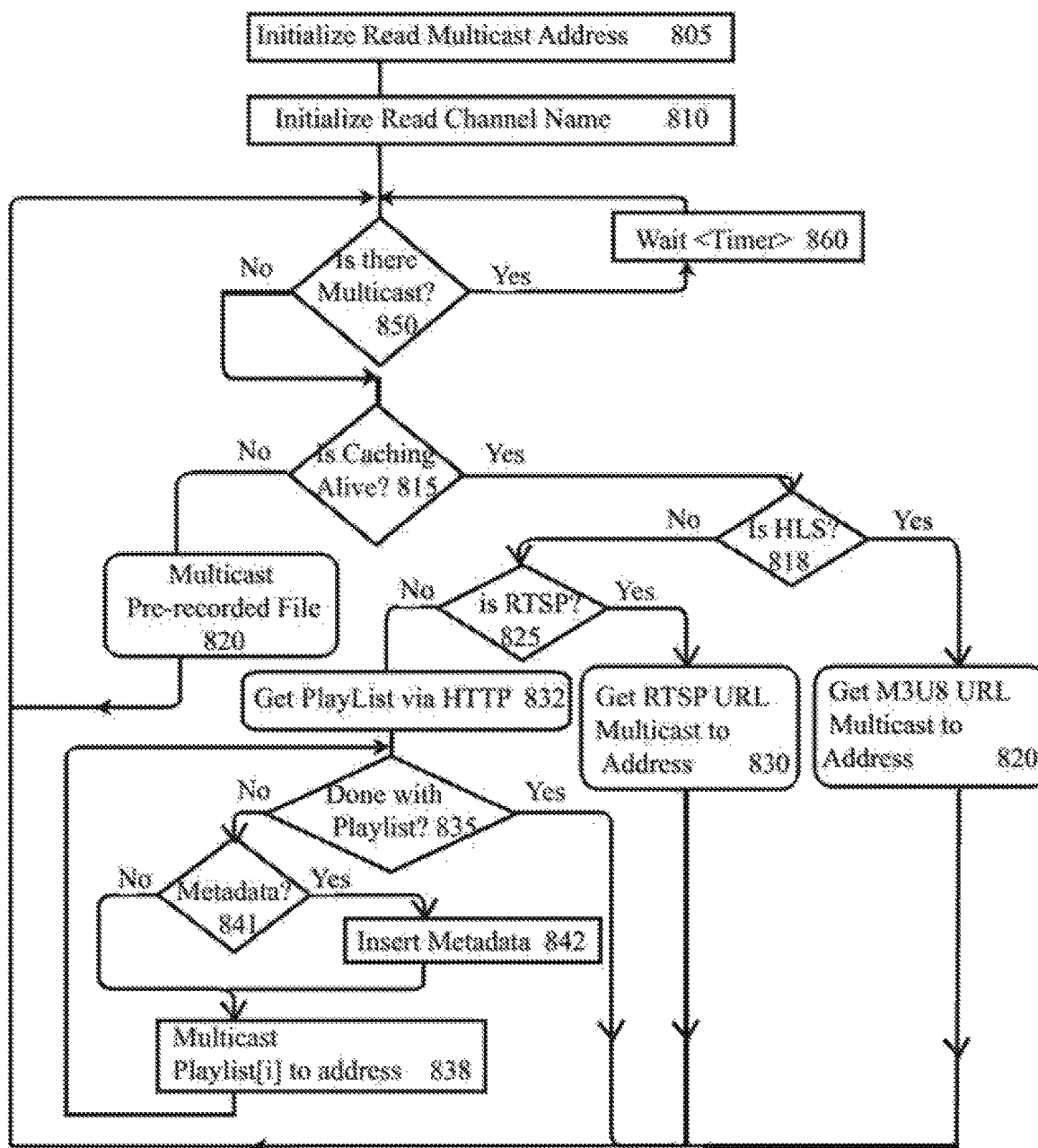


FIG. 8

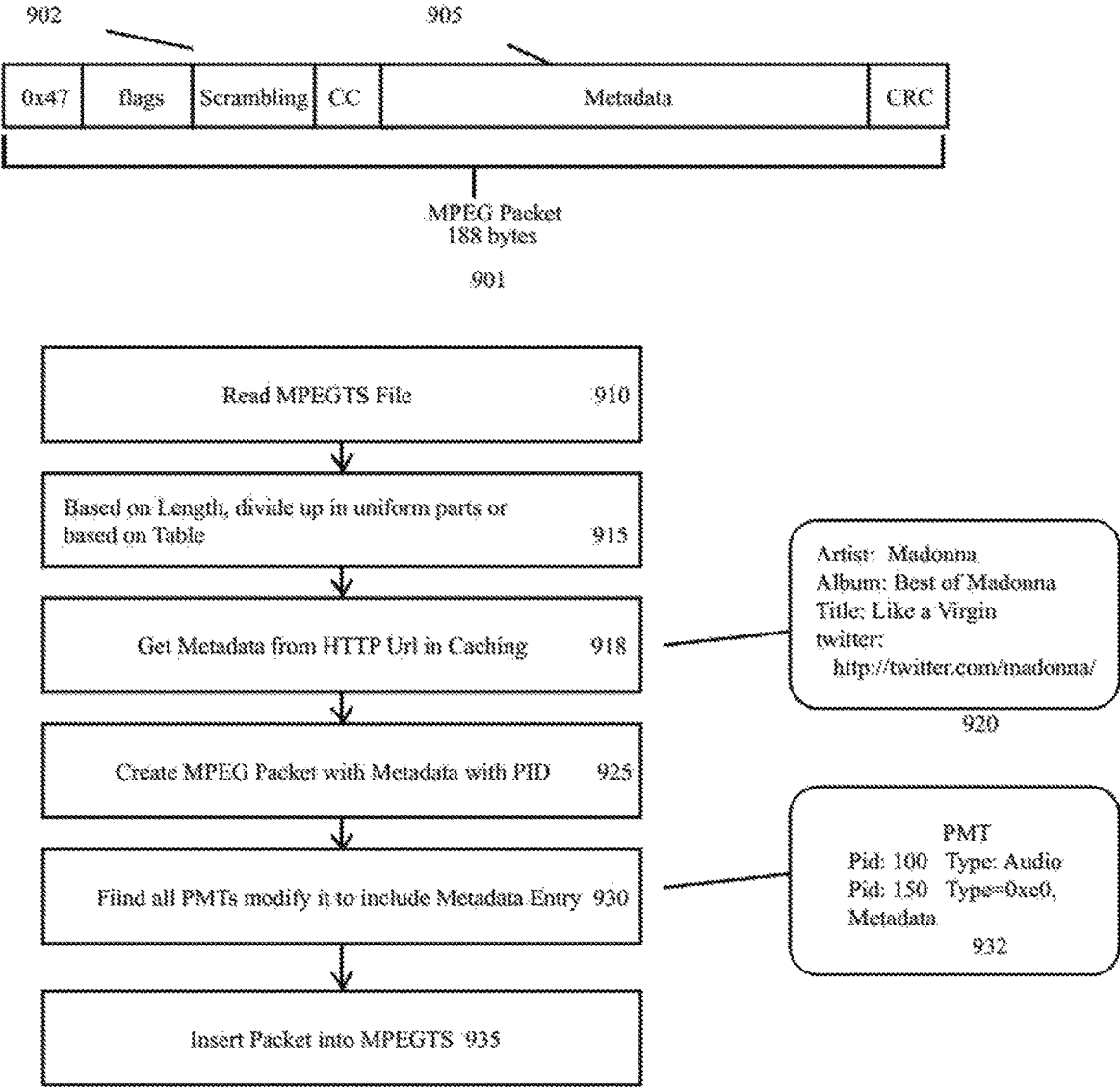


FIG. 9

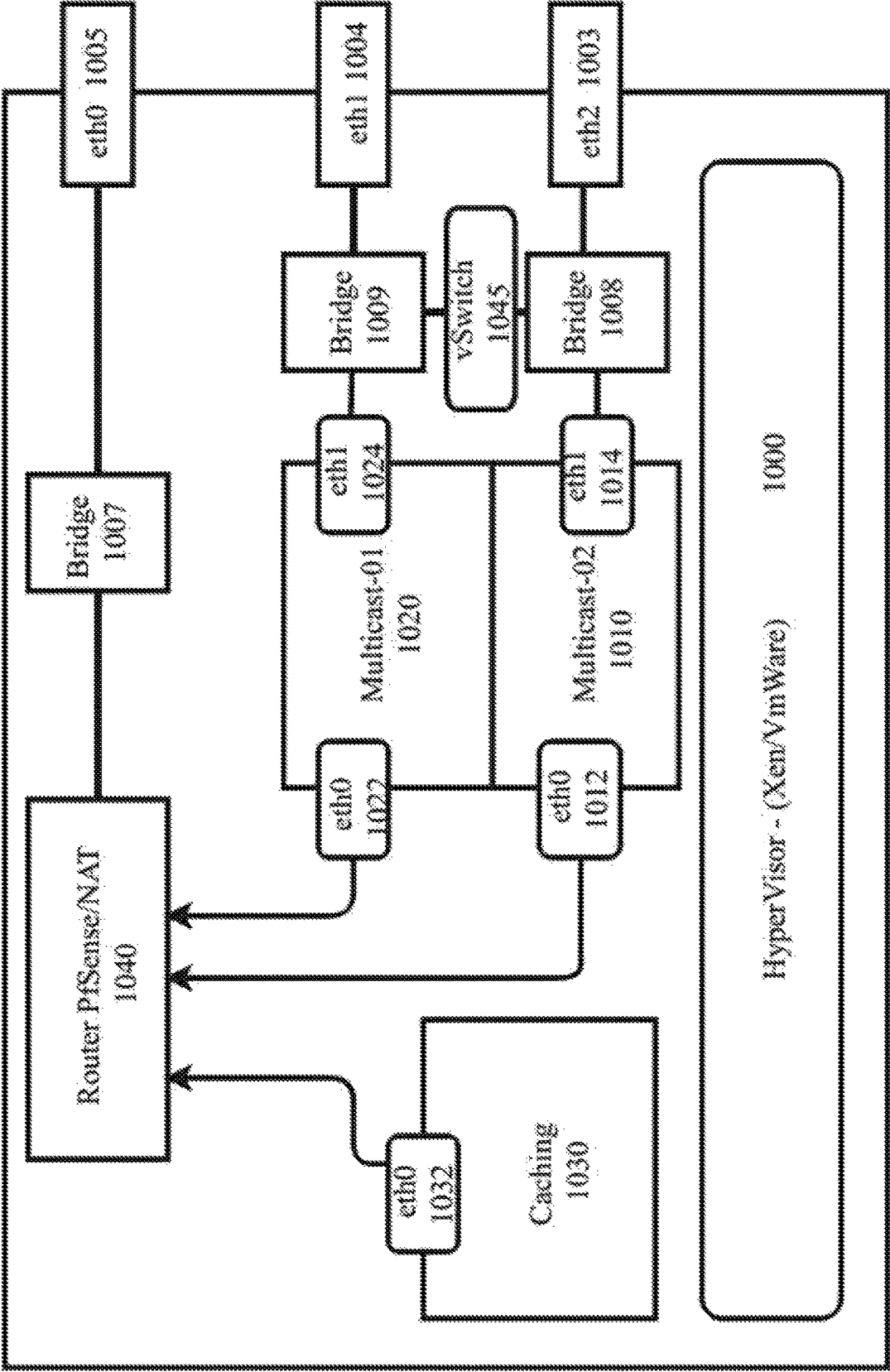


FIG. 10

U.S. Patent

Oct. 5, 2021

Sheet 11 of 12

US 11,140,441 B2

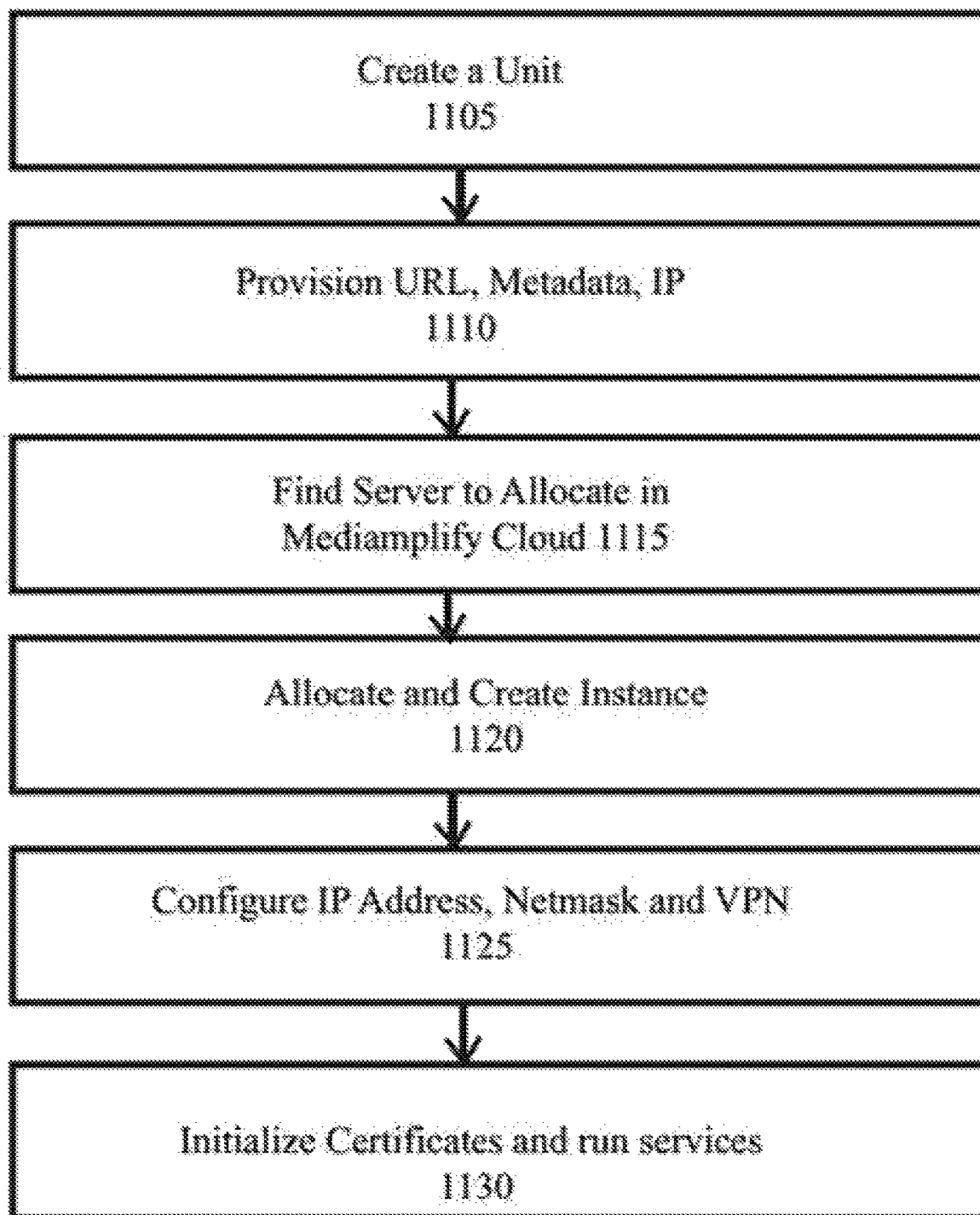
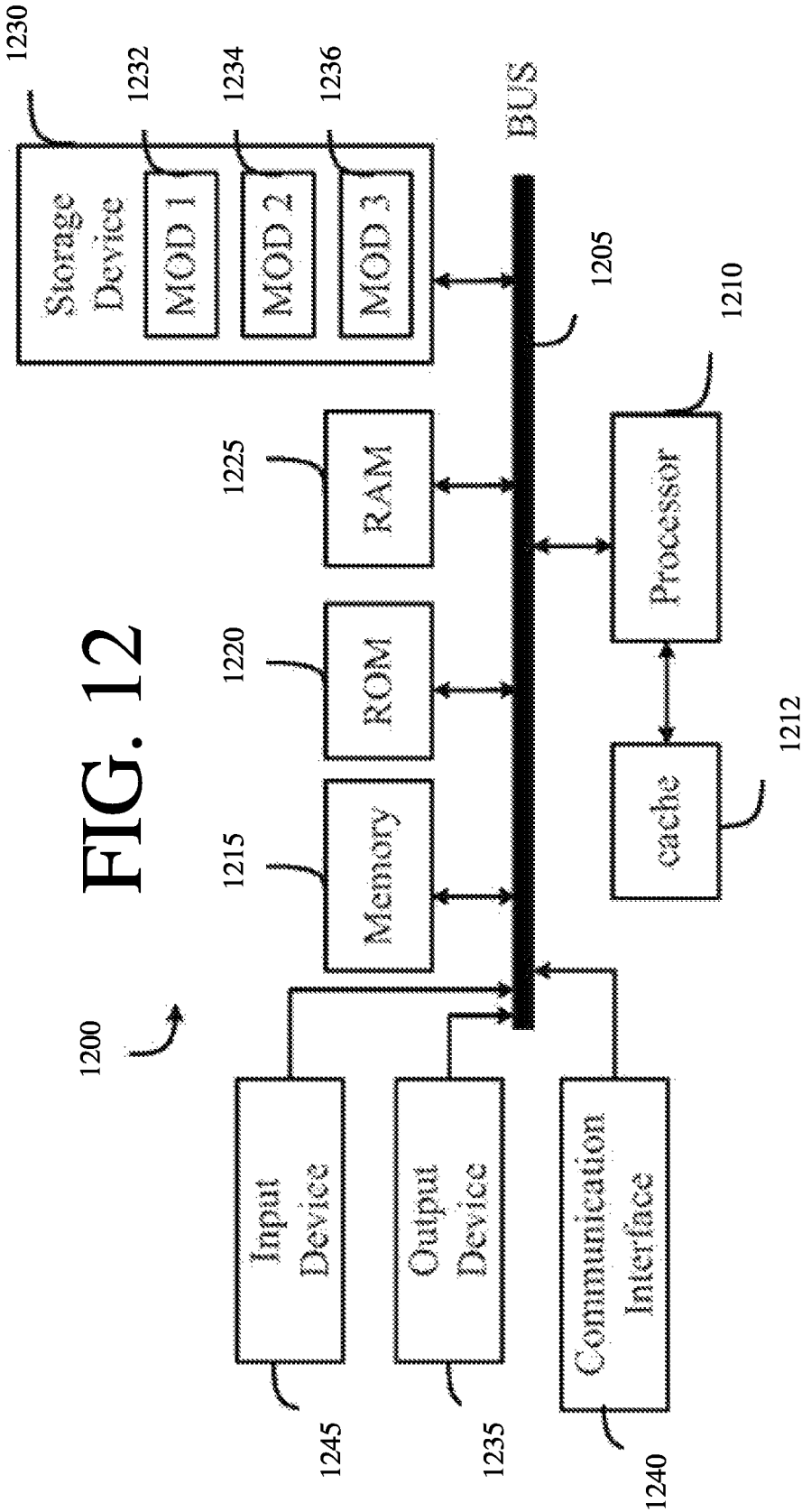


FIG. 11



US 11,140,441 B2

1

**METHOD, SYSTEM, AND APPARATUS FOR
MULTIMEDIA CONTENT DELIVERY TO
CABLE TV AND SATELLITE OPERATORS****CROSS-REFERENCE TO RELATED
APPLICATION**

This application is a continuation of U.S. patent application Ser. No. 16/152,606, which was filed Oct. 5, 2018, and is a continuation of U.S. patent application Ser. No. 15/538,911, which was filed Jun. 22, 2017, which is a National Stage Entry of International Patent Application No. PCT/US2015/067464, which was filed on Dec. 22, 2015, which claims priority to U.S. Provisional Application 62/095,504, which was filed on Dec. 22, 2014. The complete disclosures of which are hereby incorporated herein by reference in their entireties.

FIELD OF THE DISCLOSURE

The present disclosure provides a mechanism for delivering multimedia content from a cloud-based system to cable or satellite content providers.

The disclosure describes a novel method and a system for continuous media delivery and secured synchronization of multimedia files that are broadcasted to set top boxes or subscriber units in the field that are connected to IPTV or Cable Systems. The present technology can securely connect to a cloud-based platform (e.g. OpenStack, Amazon EC2, Mediamplify Cloud, etc.), creates a set of multimedia files (e.g. MPEG4 movies) into a secured repository (e.g. Secured File System, Linux/Windows, etc.), modifies those files according to certain rules as defined by the method herein, and reproduces those signals via a multicast video stream (e.g. User Datagram Protocol (UDP) multicast, etc.) encapsulated in a Transport Stream (TS) frame that can be a multiple transport stream (several transports in one TS frame) or a single transport stream (one transport stream according to the MPEG-2 Part 1 Specification or similar multimedia streaming that is compatible with MPEG-TS at a predetermined rate or at a variable bit rate).

BACKGROUND

Cable operators, Multiple System Operators (MSO), and Digital Satellite Systems (DSS) deliver media content by utilizing satellite, Radio Frequency (RF) and Integrated Receiver and Decoder (IRD) technologies, which provide outputs that can include Asynchronous Serial Interface (ASI) signal formats. Additionally, MSOs and DSSs use dedicated hardware encoders and transcoding systems if the source video/audio is incompatible with the set top boxes of the cable or satellite operator. In general, the video/audio content delivered via the satellite to the cable or satellite operator arrives via wireless link or receiver, which is processed by an IRD that receives the stream from the satellite and can ultimately broadcast that signal to the cable operator head end in several formats. In addition, some systems may include multicast units connected with IRDs that tune to the right transponder system.

Satellite systems are used to deliver data (e.g. TV, radio, Video-on-Demand, multimedia streams) from a particular source (e.g. third-party content provider) to cable head end systems or other satellite systems. Satellite delivery is performed via a broadcast signal containing digital pictures and audio inside a transponder frequency. Transponders are radio frequency (RF) space allocated in a particular satellite.

2

For example, a third-party content provider can maintain multimedia files in CD/DVDs, pre-recorded storage, or as pre-recorded files that are reproduced in a particular audio or video encoding format using a video or audio encoder (e.g. MP3, H.264, AC3, and others). The audio and video is then made part of an RF signal that is broadcasted to the satellite system. This signal is then received by the cable or satellite operator using an antenna receptor and is decoded to generate either an analog signal (e.g. RCA Cables or any other analog interface) or digital signal (e.g. ASI, Asynchronous Serial Interface). If an analog signal is received, modern DVB (Digital Video Broadcasting) Cable and Satellite systems will encode this signal to their particular video/audio encoding requirements such that it can be broadcasted in digital packets via a multicast MPEG Transport Stream. The same occurs if a receiver decodes the signal in ASI. As such, this signal is converted from ASI to Ethernet, or simply redirected to a multicast MPEG Transport stream if arriving in a compatible encoding format for the receiving head end cable system.

When no satellite is being used, video, pictures and music, are read out from the DVD or a hard-drive containing MPEG-layer 3 or MPEG-layer 2 files that were encoded at a pre-determined bit rate according. Those video feeds can be reproduced at a synchronized timing using a multicast MPEG Transport Stream using RTP (Real-time Transport Protocol)-based multicast communication and RTCP (RTP Control Protocol)-based mechanisms, as well as standard UDP (User Datagram Protocol) multicast streams with proprietary timing.

Additionally, IPTV systems can provide mechanisms to broadcast a signal using the multicast protocol in MPEG transport streams to a customer premises with compatible set top boxes (same encoder, same streams, encryption keys, etc). A compatible set top box is such that when connected to the cable TV system it can properly decode video/audio from the cable operator's head end system. Some of these IPTV systems rely on CDN (Content Delivery Networks) that enable distribution of multimedia files to IP-enabled devices. For example, a mechanism can be provided whereby a head end content (MSO) that is being broadcasted to a cable subscriber is also available to an IP-enabled device such as a smartphone, PC, or tablet computer.

SUMMARY

Disclosed are systems, devices, methods, and non-transitory computer-readable media for media delivery from a cloud or IP-based distribution network, or CDN (Content Delivery Network) to MSO (Multiple System Operators) or head-ends that include cable and satellite delivery mechanisms. The present technology unifies cloud-based delivery (Mediamplify cloud) with the cable-based mechanism (e.g. Comcast, Verizon FiOS).

An example computer-implemented method can receive, from a content provider such as a cable or satellite operator, a request for at least one media stream for playback on a broadcast media channel. The request for the at least one media stream can include a request for a plurality of multimedia items. Content corresponding to the plurality of multimedia items can be obtained from at least one website or cloud service that offers the content in at least one first format. Based on the obtained content, the plurality of multimedia files can be generated in a second format that is compatible with the content provider. The media stream can be assembled using each of the multimedia items generated in

US 11,140,441 B2

3

the second format. The media stream can be provided to the content provider for broadcast on the broadcast media channel.

In some aspects of the present technology, obtaining the content can include retrieving at least one audio file corresponding to an audio component of the multimedia item and retrieving a plurality of images as screen captures that can correspond to a video component of the multimedia item. Generating the multimedia files in the second format can include combining the plurality of images and the at least one audio file to create each of the multimedia items.

BRIEF DESCRIPTION OF THE DRAWING

There are shown in the drawings, embodiments which are presently preferred, it being understood, however, that the disclosure is not limited to the precise arrangements and instrumentalities shown

FIG. 1 shows an example block diagram illustrating an exemplary head end (MSO);

FIG. 2 shows an example block diagram of the apparatus located at the MSO facility (mediaplug);

FIG. 3 shows an example method for initialization of Mediaplug components;

FIG. 4 shows an example method to obtain list of files to multicast to a cable operator MSO;

FIG. 5 shows an example method for encoding video and audio files;

FIG. 6 shows an example method for providing multimedia content to a cable operator;

FIG. 7 shows an example block diagram of example software components of the Mediaplug;

FIG. 8 shows an example method to multicast from a component in the Mediaplug;

FIG. 9 shows an example method to insert metadata into an MPEG Transport Stream;

FIG. 10 shows an example block diagram for a Virtualized Mediaplug

FIG. 11 shows an example method to initialize virtual mediaplug components; and

FIG. 12 shows an example possible system embodiment for implementing various embodiments of the present technology

While the present technology is amenable to various modifications and alternative forms, specifics thereof have been shown by way of example in the drawings and will be described in detail. It should be understood, however, that the intention is not to limit the present technology to the particular embodiments described. On the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present technology.

DETAILED DESCRIPTION

The following description should be read with reference to the drawings, in which like elements are numbered in similar fashion. The drawings are not necessarily at scale and are not intended to limit the scope of the disclosure. Those skilled in the art understand that there are alternatives that may be used as well which are also considered to be within the scope of the disclosure.

Generally, the disclosed technology pertains to a delivery mechanism for providing multimedia content to cable systems without the use of a satellite system in a fault-tolerant and reliable fashion. FIG. 1 illustrates a block diagram that includes a Mediaplify Cloud 101. The cloud 101 can refer

4

to a set of virtualized instances or non-virtualized instances of one or more servers that provide access to archives and digital files with media content stored in them. For example, the cloud 101 can be implemented using VMWave or Xen and OpenStack, or simply reside in Amazon Web Services or Rackspace. In addition, special provisioning can be provided in cloud 101 for secured access.

The cloud 101 service files can be accessible via the internet 105 and can be stored in a cloud-based file system or cluster file system (e.g. GlusterFS or Hadoop, Amazon S3). These files can be multimedia files that include, as non-limiting examples, MPEG-4 Video files with H.264 encoded video and AAC encoded audio or MPEG-layer 3 audio as well (MPEG2 Video could also be used). These files can be accessed using a secured transport protocol such as, but not limited to, SSL or a synchronization tool as "rsync," commonly found in Linux/MacOS environments. The communication link 106 can be a DSL, Cable Modem, Wireless WAN, Fiber, Metro Ethernet, etc. and can be linked with the Mediaplug 110 unit. The Mediaplug 110 can reside at the "MSO" facility and can be collocated with all the cable or satellite operator equipment. The MSO is a facility used by the Cable Operator to receive all TV and Radio Programming that the cable operator will then multiplex, re-encode, and distribute from the MSO to a set of subscriber units with set top boxes, or modulate the signal in TV analog channels (2-60+).

One function of the Mediaplug 110 is to provide multimedia content, e.g., video and audio, to an MPEG multiplexer at the headend 120. The content can be pre-formatted in MPEG Transport Streams that the multiplexer 120 (e.g. Arris TMX-2010) will then be able to capture, reformat in any way the operator requires, and deliver to the cable operator's subscribers via Coaxial Cable 125 or any other means suitable for a multicast or unicast delivery to its customers with "Set Top boxes" 130, 131, 132. A "Set Top Box" is customer premise equipment that will be able to decode and display in a TV set or other screen display, via a signal received by its HDMI or analog input, the contents of the TV programming the cable operator offers.

The Mediaplug 110 can be a computer or server or set of computers of servers that can communicate with the Cloud 101 using a standard Local Area Network (e.g. Gigabit Ethernet). The Mediaplug 110 can store, encode, and transmit multimedia files in the format that the cable operator at its MSO requires. This cable format may include audio, video, and metadata (subtitles, control data, and any additional component that can be packaged in an MPEG Packet that is part of the MPEG Transport Stream). In one embodiment, Mediaplug 110 can include components such as a caching unit, and one or more multicasting servers.

FIG. 2 illustrates an example configuration of the hardware components of Mediaplug 110 which includes three servers. A server is a computer device, usually rack mountable unit with volatile memory such as RAM, one or more micro-processors, and non-volatile memory such as a hard disk. However, one skilled in the art will recognize that the present technology is not limited to a particular configuration and can be varied according to different specifications. FIG. 2 illustrates Mediaplify Cloud 202 which is connected to the Internet backbone 203. Link 204 (e.g. DSL, Cable Modem, Fiber) provides connectivity between cloud 202 and a Network Address Translator unit 205 (e.g. Cisco RVS4000, or any other NAT device) which can provide Gigabit Ethernet switching capabilities. The lines 206, 207, and 208 show Gigabit Ethernet links, however one skilled in the art will understand that Gigabit Ethernet or a Wireless

US 11,140,441 B2

5

LAN using 802.11g/n would be also a suitable interface to be used in a Mediaplugin configuration. The Caching unit **210** can be the main storage component in the system and it can contain all of the multimedia files in the format required by the cable operators, such as the Transport Stream format. The caching unit can communicate with Multicast-01 **215** and Multicast-02 **220**. The two Multicast units **215**, **220** can pull or obtain information from the caching unit via TCP/IP or UDP, using protocols such as HTTP (Hypertext Transport Protocol) or RTSP (Real-time Streaming Transport). The caching unit can also encode and transcode the multimedia files it downloads from the Cloud **202**. The caching unit can also keep updated information of all the metadata and create customizable screens that can be converted into video formats compatible with the Transport Stream content that will be provided to the cable operator at its MSO. MPEG MUX MSO **230** can correspond to a program residing in a computer system that processes and handles the MPEG streams from multiple sources. The MUX **230** can also function as the bridge between the source of the signal and the customer's set top box units. A non-limiting example program would be the CherryPicker DM6400 by MOTOROLA. Each of the components or modules of the Mediaplugin can include software for performing functions and/or methods according to the present technology.

FIG. 3 illustrates an example method that can be used to configure and provision the units at boot-time. Each unit can initialize keys for SSH/IP Sec access at **305**. For example, the units can use IPsec and SSH Tunnel keys that can be stored in the administrator user account. The keys can be generated by the Mediaplugin Cloud that are private keys or certificates, shared secrets or similar components by which the Mediaplugin can connect securely with the Mediaplugin cloud. Each cable operator can provide custom-based IP addressing at **310** for the Gigabit Ethernet ports that connect to the operators' network. Changing the IP addresses of the Multicast-01 and/or Multicast-02 refers to the IP address by which the links **221** and **216** in FIG. 2 are linked to the MPEG MUX MSO unit. A change of the IP address can refer to the 'ifconfig' command which is used in Linux-based systems and that can be changed in an Ubuntu box at/etc/network/interfaces file. Similarly, the network routing can be changed if the multicast address cannot be routed to the default gateway which points to the 205 NAT, but to **221** and **216** respectively for Multicast-02 and Multicast-01. In general, the 239.*.* network address may be routed to those interfaces.

At step **320**, the Mediaplugin can configure playlists per channel and multicast address. The playlist and channel configuration can be obtained from the cable/satellite operator and can include any number of channels for broadcasting multimedia content to end customers. As an example, the channels may include music channels corresponding to different music genres, e.g. rock, alternative, classical, jazz, hip hop, etc. Based on the particular genre, a corresponding playlist can be selected and/or configured. The Mediaplugin can also determine the multicast address for each of the channels.

In some configurations a customizable user interface (UI) can be added to the output video MPEG Transport stream. As such, if the "Custom UI" variable is then true and valid, **325**, a link or URL **330** is loaded into the Mediaplugin configuration that can be used when the video interface is created. The custom video interface can be created and fetched as needed or it can be created in real-time for a particular channel or for all the channels. The custom user interface variable can be uniquely configured per channel in

6

the Mediaplugin. The mediaplugin can also monitor **340** all of the processes to ensure stability and continuous, uninterrupted operation. The initialization of the Mediaplugin can also include setting the RSYNC profile **350**. The RSYNC profile can be used to update and synchronize media content with the cloud when the Mediaplugin determines that the content has changed or been modified.

As described above, the Mediaplugin can broadcast the multimedia MPEG stream using the programmed IP Address in multicast-01 and multicast-02 according to the method illustrated in FIG. 3. In FIG. 4, an example method is shown for obtaining the list of files to playback or broadcast to the MSO system. The file type or format requested by the multicasting embodiment, e.g. multicast-01 or multicast-02, can include Transport Stream (e.g. TS) and/or MPEG-4 Part 14 (e.g. MP4) values **405**. This request is executed from any of the multicasting embodiments, usually making use of the HTTP GET Request. The multicasting unit can connect to an HTTP Server and retrieve the list of files based on the type identified by the request. As the caching unit may handle multiple channels of different types and name, each channel can have a unique identifier corresponding to the name of the playlist. For example channel name "Cool" or "Rock," **410** are names that can identify the channel to be retrieved from the caching server. The activity log in the request can also be updated keeping track what step of the process is being completed **415**. At **420**, the caching server can generate a "playlist file" of mime type "audio/x-mpegurl" which will be interpreted as a multimedia playlist file when requested. Finally, as the playlist usually may not vary for a period of time until a new request is received, the playlist can be kept in the local memory caching service **425**.

As the requests arrive at the "Caching" unit from the multicasting units, the "Caching" unit can be configured to perform parallel encoding for all the different channels available from the unit. FIG. 5 illustrates an example method that can be implemented by the Mediaplugin, and in particular, by the "Caching" unit. At step **505**, encoding flags or input parameters can be received which may include GOP (Group of Pictures) which is used by certain video encoders, resolution (e.g. 640x480) in pixels, bit rate in Kilobits per second, any encoding type of video (e.g. H.264), encoding type of audio (e.g. AC-3), duration of the video feed, etc. In addition, a "format" parameter can be received which refers to the output format or encapsulation, MPEG TS or MPEG4. The caching unit encodes audio and video, and can use the disk space in the Mediaplugin. Hence in **508** the method checks for sufficient space left in the memory or hard drive. This task can be run on a thread together with all the other tasks that are working in parallel in order to speed up the processes of encoding and transcoding.

Several indexes or identifiers by the name "ChID," (these integers preferably no more than an integer value from 1 to the maximum number of channels) begin (e.g. MAX could be 50) to be generated by the caching unit **510**. This method can be performed for all the channels and all media linked to that particular channel identified by "MediaID" **515**, in this particular case the "MediaID." The variable "MAX" in this context can refer to the maximum number of media files read (e.g. **500**). The data captured from the media could be "Artist," "Author," or any other variable from an audio file. If a video file is used this could be "program name," "movie name," or any associated metadata information tied to the multimedia file and the Channel Identifier (ChID) associated with it "MediaID" file.

The caching unit can add a "Custom UI," or a customizable User Interface that is designed using Hypertext

US 11,140,441 B2

7

Markup Language (e.g. HTML) or a web service. A web service that matches a REST interface (Representation State Transfer) and uses Javascript Object Notation (JSON) or XML (Extensible Markup Language) to respond to a custom UI request can be used. Moreover, this custom UI can include JavaScript, CSS, and HTML content files that can be rendered using a browser using WebKit (e.g. PhantomJS or Safari) or any other browser-rendering engine compatible with HTML4/5 or any future HTML version. If a customizable UT is part of the configuration **518**, rendered images from a web service can be used to create overlapping video or replace any video in the video feed file. Likewise, this video or customized screen can be used when no video at all is defined in the multimedia file, e.g. Audio-only file. The web-service to be contacted may use the "PlaylistName" to be used for the Channel identified by the channel ID "ChID," retrieved using the "PlaylistID" function **520**. Each new PlaylistName may contain different identifies to retrieve new images. Therefore one or more screens will be retrieved from the rendering engine as such **522** can be used to retrieve a set of screens from the web-service being contacted.

Capturing screens **526** can be done using a tool similar to PhantomJS (Webkit-based headless engine for rendering images) or QtCapture, or simply capturing screen frame buffers from a machine running any software tool. Alternatively, the method can capture a video file from another rendering function, which can create a sequence of screens that can be used to generate a video file. Once all screens and video are captured, **530** the resulting output created can be a video file that then is either replaced or merged (multiplexed) with the original file. As an example, if an audio content file is used such as an MP3 file, this function can create a video file from all the screens captured by **526**. Those screens are generated by calling a URL where the web-service is associated and creating a video file with the input audio and the captured screens. The output format can be an MPEG Transport Stream file that can be retrievable or streamed to the multicasting embodiment.

Once the output file containing the screen shoot sequence of videos encoded in H.264 (for example, or MPEG2Video) is created, the audio can be multiplexed and added to one of the encoding channels of the resulting video output and saved into the caching unit storage disk **550**. The multiplexing can be done using tools such as FFMPEG or VLC or any proprietary tool that can multiplex audio and video files into a resulting MPEG Transport Stream (or MPEG-4 output, or any similar output).

FIG. 6 illustrates an example method **600** for providing multimedia content to a cable operator. At **602**, a device such as the Mediaplugin device receives a request from a cable/satellite operator for a media stream. A media stream can be formatted according to HTTP Live Streaming (HLS), an HTTP/RTSP playlist, or an RTSP stream. The media stream can include any number of multimedia items such as audio/video multimedia items. In some instances, the request can include a channel identifier that is associated with a particular category of multimedia items, such as a music genre corresponding to rock, classical, jazz, etc.

At **604**, contents corresponding to the multimedia items can be retrieved, downloaded, or otherwise obtained from one or more websites or cloud based servers. The contents can include data corresponding to either or both of the audio and video components of the multimedia items. For example, the data corresponding to the audio portion can include a music file in an MP3 format. Similarly, the data corresponding to the video portion can include a video that

8

is obtained directly from the cloud. Alternatively, the video component can be assembled by using one or more screen captures from a website. The screen captures can include captures of still images presented by the website as well as captures of images from videos that are available on the website. As illustrated in FIG. 5, the screen capture process can be repeated any number of times.

The Mediaplugin can then generate **606** multimedia items in a second format based on the content obtained from the web server or cloud. That is, the Mediaplugin can convert/encode the audio file into a format that is suitable for the cable operator and it can also assemble the various screen captures to generate the video component of the media item. Once the audio and video files are created, they can be combined, multiplexed, or encoded to a format that can be utilized by the multimedia stream. For example, the audio can be multiplexed and added to one of the encoding channels of the video output. The various multimedia items can be used **608** to assemble the media stream in a format corresponding with the request (e.g. HLS, HTTP/RTSP, RTSP stream). The media stream can be provided **610** to the content provider for broadcast on the broadcast media channel.

FIG. 7 illustrates a block diagram of an example architecture of the Caching Unit components. The figure shows **720** as the main operating system, which can be Linux, Windows, or any other operating system in use. The caching unit can contain two types of operating system services HTTP **710** and RTSP **712**. The RTSP service provides access so the media saved or created in the repository of the caching unit file system **730**. The RTSP server can be implemented using the Live555 RTSP server or similar. Alternatively, the HTTP Server can be implemented using "nginx" or "apache" web browsers, and the RTSP Servers can be implemented using Live555 services. In the case of the HTTP Server, this can also be used to get the Playlist of files generated from the method on FIG. 5 **715**.

The GetPlaylist **715** command can be implemented as follows:

- Receive and process a request for the name of the channel or playlist to be used,
- Retrieve all the filenames ordered by date of creation of all the files in a particular directory,
- Cache all the files into cache memory (e.g. Memcache or similar tool),
- Add type of HTTP header as M3U or PLS (e.g. a text file with a list of files or URLs to play),
- Return a list of HTTP or RTSP URLs to access each video generated or stored in the caching unit.

In some embodiments, the HTTP Server can also be used to clean up, update, and delete operations. The block unit called "CleanUp" **722** is in charge of keeping up to date caching information data and/or deleting files already used or that are not required to be used anymore by the multicast unit(s). Also the HTTP Server functions can include servicing M3U8 files corresponding to the HTTP Live Streaming format and all the segments created by the HTTP Live Streaming function. As such, a HTTP Live Streaming (HLS) generation program will produce the files and segment them as needed by the encoder and can then be serviced by the HTTP Server **710**. The Mediaplugin Parallel Encoder **705** corresponds to a block that can implement the method shown in FIG. 5. The resulting outputs from the Media Parallel Encoder **605** are stored in the file system **730**. The Mediaplugin Parallel encoder **605** can be implemented in Python and can rely on web services (such as Twisted or Django), PhantomJS, and FFMPEG. Monit **750** corresponds to a monitoring block that can ensure that the process

US 11,140,441 B2

9

running the “Media Parallel encoder” keeps running at all times and does not stop upon failure. Additional blocks that can be included in Mediaplug are RSYNC **742** and SSH **744**. These blocks can be used to ensure synchronization and updates of new media files coming from the cloud. The RSYNC block uses the “rsync” protocol and program over ssh (Secured Socket Layer Shell) to download media from the “Mediamplify Cloud” and synchronize multimedia files that are newly updated and could be part of particular Channel ID and a Playlist Name. Each new file synchronized with the Mediamplify cloud has a Media ID value which the cloud can assign, and include additional metadata information such as “Artist” and “Author.” The “Mediamplify Parallel Encoder” also may receive requests from other components such as getStatus or StartEncoding, indicating what file is being encoded, its progress, and what other files are in the queue for encoding.

In some embodiments, the information from getStatus or StartEncoding can be generated using a JSON message retrieved upon a request via HTTP GET to the Mediamplify Parallel Encoder. A graphical user interface can then be used to plot the status of each song being encoded or provide previews of the videos being generated. Below is one non-limiting example of a JSON response made by the “Mediamplify Parallel Encoder” service:

```
{“song_encoded”: “ts”, “artist”: “Madonna”, “duration”: 0,
“exists_on_disk”: True, “genre”: “Pop”, “title”: “Like a
Virgin”, “music_id”: “1082863”}
```

Response regarding the status for a song being encoded or a list of songs being encoded, as follows:

```
{“music_id”: “109281”, “encoding”: “78%”, “output_file”:
“/var/www/egllacast/pop/109281-abcdef-uuid.ts”,
“thumbnail”: “http://mediamplify.com/images/abcdef-
uuid.png”}
```

In some embodiments, there are two possible TCP/IP sockets open for listening used by the Caching component of the Mediaplug. A) Port 80 (as an example), or the standard HTTP port, used to retrieve content, view video content generated, update/delete cached files, and access the playlists, and B) Port 9553 (as an example), or the standard Mediamplify Parallel encoder port, used to monitor and review status of the songs being encoded, the data being collected, and all other related activities with encoding. Via this port, encoding/transcoding can be started or stopped.

Similarly, the “Mediaplug” components labeled “Multicast-01” and “Multicast-02” can also provide an HTTP interface to access the current song being played or being multicast via the HTTP Server **770**. For example, the HTTP Server may run a PHP or Python script that will return the current song being played by the streamer. The streamer block **768** can connect to the caching server via HTTP or RTSP protocols. The implementation of this block can be done using a combination of FIG. **8** instructions and using multicasting services available in VLC or FFMPEG (e.g. multicast, or ttools/tsplay).

One function of the streamer **768** is to read the files encoded by the caching unit, process those for multicasting and creating the MPEG Transport Stream multicast packets sent via the Ethernet port. Additionally, the multicast-01 and multicast-02 blocks contain a “Recording” function **790**, which is in charge of recording the multicasting streams and recording for later use in case of failure on the system. This operates by listening into the UDP (User Datagram Protocol) multicast address of each individual channels being broadcasted, and recording the output into a file. This file is then stored and made accessible for future use. Monit **764** can

10

maintain the streamer process running at all times through-out a regular broadcasting cycle.

FIG. **8** illustrates an example method that can be performed by the streamer module in the Mediaplug. In one embodiment, the streamer can perform functions that include: connect to the caching unit; retrieve generated videos per playlist; determine if unit is working in fault-tolerant mode; playback pre-recorded audio if a failure is detected; and record units own output or any other streamer output if connected to switch or hub.

Turning to the method from FIG. **8**, at step **805**, the streamer reads the multicast address associated to the channel being broadcasted, in order to generate IGMP (Internet Group Management Protocol) messages and establish a multicast group (e.g. 239.0.0.1:8001). The channel name is also initialized at **810**. This value can be used to retrieve the proper content from the Caching server. In one embodiment, a configuration table can be created with tuples: <ChID>, <Channel Name>, <Multicast Address>, <Multicast Port>, such that no duplicates are used or created. Automation software can create these configuration tables and automate the process of creating all streamer calls. A streamer method can be called (e.g. from/etc/init.d/) at boot time and monitored with a tool such as Monit. Once the configuration is known, the streamer will check **850** if there are already multicasting transport streams generated by any other multicast component and avoid collisions. If such multicast data is detected, a delay can be used to delay a new attempt to check on that particular multicast group **860**. Hence this block indicates “Wait <Time>” or stop all processing for the amount of <Time> recommended by the specification. For example, the time for delay can be the “Main source” of input less the amount of time for receiving input from a backup server. During initialization, if the same multicast address is used to broadcast main and backup, the one with less amount of time delay will be first, followed by the “Backup.” In cases where a hot spare is used, or two different multicast addresses can be in use, one for main and the other one for backup, then this step may be optional.

Once it is determined that there are no elements multicasting at the same address, the streamer will check if the Caching server is “alive” **815**. This can be performed in multiple ways, for example an HTTP Request must not generate a **404** or **403** error, but instead a successful **202** value. Likewise, ping requests can also be used before making an HTTP GET call to identify if the caching server is present and available. In the case that there are no ping responses, a timeout will trigger a failure and a pre-recorded content **820** will be played or sent to the multicast address and port of record. Recordings can be scheduled daily for an extended period of time (e.g. 8 hours or more). These recordings can store the recorded file in a temporary folder and then move that file over to the playback folder where streamer can read it.

If it is determined that the caching server is functioning properly, then there are three types of streams that can be sent to the multicast address and port:

HTTP Live Streaming (HLS) a multi-segment stream usually defined by extension M3U8, containing a set number of segments for a real-time stream source (Video/audio)

HTTP/RTSP file playlist, this means that one or many files are read from the caching server, usually defined by extension M3U or PLS

RTSP stream, this could be constant stream similar to HTTP Live Streaming (HLS) but using an RTSP protocol instead of HLS stream.

US 11,140,441 B2

11

When multiple files are part of a playlist, the bit-rate may vary between playlist files. Different bit-rates among files in a playlist could cause problems in non-CBR (Constant Bit Rate) systems. To avoid such problems, the Mediaplugin can maintain the stream at the bit rate of the previous stream by adding null MPEG TS packets at the same rate as the previous media file was being sent over the multicast port. This will maintain appropriate bit-rate for playlists.

In RTSP service, streaming occurs by writing the media content to a named pipe or a FIFO file and allowing the RTSP service to read from that file and generate an RTSP stream. As such, in HLS and RTSP Streaming, the caching server knows the filename of the file being streamed. When the file is streamed from via playlist, both the caching and multicast unit know the name of the file being streamed, as the multicast server can then request from the caching server the metadata associated with the file name being streamed.

Accordingly the present technology can accommodate both cases and retrieve the current file name being streamed as a JSON response. The response can identify the content displayed or played back at the end-user's set top box.

If HLS is used for streaming **818**, the streamer will connect to the proper M3U8 file **820** to pull all the segments corresponding to the live playlist file on the M3U8 stream. Programs and libraries such as FFMPEG or VLC can then be used to encapsulate the stream feed to an MPEG TS and multicast it to the pre-programmed multicast address and port. Similarly, if an RTSP stream **825** is used, the same process takes place at **830** using the RTSP as a protocol instead of HTTP.

If a list of files in a playlist **832** is in use "M3U," the list can be obtained using an HTTP GET request made to the caching server. Each file can then be streamed to the multicast address and port of record **838** until all elements of the playlist are completed and played back **835**.

In some instances, metadata can be added to the stream as an additional PID (Program Identifier) in the MPEG Transport Stream. If metadata is needed **841** then it will be inserted **842** here in accordance with FIG. 9.

For example, the metadata added could be as simple as a string containing Artist, Name, or other information, or as complex as DCII Text (DigiCipher version 2, as specified by Motorola), MPEG TS Control messages (Two-way), or any other metadata or PID that can be inserted into the stream being sent to the multicast address and port.

FIG. 9 illustrates an example method used to insert a special data packet into the stream being written to the multicast address, without making use of timing information. Therefore, the metadata can be inserted such that it does not have timing dependency. Alternatively, certain type of metadata such as subtitles or lyrics may need to be inserted only at certain times so that it is synchronized with the video data.

The sample MPEG TS packet shown in **902** and **905** corresponds to the metadata inside the packet. This matches standard MPEG TS packet structure and may vary depending on the type of metadata being added to the stream.

The method to insert the packet starts by reading the MPEG TS file retrieved from the caching server **910**. If metadata corresponds to only song information, this information can be added at uniform times. Non-uniform timing inclusions can also be done by adding a table that includes the time and the metadata content added to the stream **915**. This metadata can be retrieved from the Caching unit using an HTTP Get request directed to the caching server **918** and may include artist name, title of song, or any other information **920**.

12

Creation of the MPEG Packet(s) **925** to be inserted at the appropriate times, which will set the places with those packets, will be inserted in the MPEG TS. Modify the PMT (Program Management Table) of the generated file and include the PTD being inserted into the MPEG TS **930**. At step **932**, an audio PID may include a metadata PID of type DCII Text, which is added to the stream.

For each timing or location where the PID needs to be inserted **935**, the method can insert those packets and generate a buffer that can be sent to the multicast address as illustrated in FIG. 8.

As described above, the disclosed technology can be implemented using a set of modules included in the Mediaplugin apparatus, which can be implemented using one or more servers accordingly. Alternatively, virtualization environments such as VMWare or Xen (FIG. 9) can include a system and architecture that can be used to host all the Mediaplugin functionality in a virtualized environment using a single server with multiple GbE ports.

The illustration in FIG. 10 shows a configuration of three virtual machines used for implementing the Mediaplugin. Caching **1030** and its respective Ethernet port **1032**, Multicast-01 **1020** with two virtual Ethernet ports **1022** and **1024**, and Multicast-02 **1010**, with two Ethernet ports **1012** and **1014**. There are three bridges that include **1007**, **1008**, and **1009**. The multicast Ethernet ports **1024** and **1014** are bridged with eth1 **1004** and eth2 **1003** respectively via bridge **1009** and bridge **1008**, respectively. That is, the same content generated at **1024** can be forwarded via the bridge **1009** to the physical interface eth1 **1004** in the hardware. The vSwitch **1045** or the virtual switch between the two bridged interfaces **1008** and **1009** pehnits the two virtual machines of Multicast-01 **1010** and Multicast-02 **1020** to record each other and monitor their activity.

To achieve concurrency and proper execution in a cloud environment, the Mediaplugin can use the example method shown in FIG. 11. Once a create unit command is issued **1105**, provisioning of the metadata information and proper IP Address is provided for that particular instance **1110**. Once an IP Address and metadata has been chosen for the unit, the server can be allocated in the cloud and the instance can be created using CloudStack or OpenStack tools and services **1115**. The instance is then allocated and runs in the proper node in the cloud **1120**. Configuration of the assigned IP, Netmask, and VPN credentials are loaded at **1125**. Finally, certificate information is uploaded **1130** as well as all services run to enable the unit for execution of its caching functions, multicasting functions, or both.

FIG. 12 illustrates a conventional system bus computing system architecture **1200** such as can be used in Mediaplugin **110**. Those of ordinary skill in the art will appreciate that other system architectures are also possible.

In computing system architecture **1200** the components of the system are in electrical communication with each other using a bus **1205**. Example system **1200** includes a processing unit (CPU or processor) **1210** and a system bus **1205** that couples various system components including the system memory **1215**, such as read only memory (ROM) **1220** and random access memory (RAM) **1225**, to the processor **1210**. The system **1200** can include a cache of high-speed memory connected directly with, in close proximity to, or integrated as part of the processor **1210**. The system **1200** can copy data from the memory **1215** and/or the storage device **1230** to the cache **1212** for quick access by the processor **1210**. In this way, the cache can provide a performance boost that avoids processor **1210** delays while waiting for data. These and other modules can control or be configured to control the

US 11,140,441 B2

13

processor **1210** to perform various actions. Other system memory **1215** may be available for use as well. The memory **1215** can include multiple different types of memory with different performance characteristics. The processor **1210** can include any general purpose processor and a hardware module or software module, such as module **1 1232**, module **2 1234**, and module **3 1236** stored in storage device **1230**, configured to control the processor **1210** as well as a special-purpose processor where software instructions are incorporated into the actual processor design. The processor **1210** may essentially be a completely self-contained computing system, containing multiple cores or processors, a bus, memory controller, cache, etc. A multi-core processor may be symmetric or asymmetric.

To enable user interaction with the computing device **1200**, an input device **1245** can represent any number of input mechanisms, such as a microphone for speech, a touch-sensitive screen for gesture or graphical input, keyboard, mouse, motion input, speech and so forth. An output device **1235** can also be one or more of a number of output mechanisms known to those of skill in the art. In some instances, multimodal systems can enable a user to provide multiple types of input to communicate with the computing device **1200**. The communications interface **1240** can generally govern and manage the user input and system output. There is no restriction on operating on any particular hardware arrangement and therefore the basic features here may easily be substituted for improved hardware or firmware arrangements as they are developed.

Storage device **1230** is a non-volatile memory and can be a hard disk or other types of computer readable media which can store data that are accessible by a computer, such as magnetic cassettes, flash memory cards, solid state memory devices, digital versatile disks, cartridges, random access memories (RAMs) **1225**, read only memory (ROM) **1220**, and hybrids thereof.

The storage device **1230** can include software modules **1232**, **1234**, **1236** for controlling the processor **1210**. Other hardware or software modules are contemplated. The storage device **1230** can be connected to the system bus **1205**. In one aspect, a hardware module that performs a particular function can include the software component stored in a computer-readable medium in connection with the necessary hardware components, such as the processor **1210**, bus **1205**, display **1235**, and so forth, to carry out the function.

It can be appreciated that example system **1200** can have more than one processor **1210** or be part of a group or cluster of computing devices networked together to provide greater processing capability.

For clarity of explanation, in some instances the present technology may be presented as including individual functional blocks including functional blocks comprising devices, device components, steps or routines in a method embodied in software, or combinations of hardware and software.

Any of the steps, operations, functions, or processes described herein may be performed or implemented by a combination of hardware and software modules, alone or in combination with other devices. In an embodiment, a software module can be software that resides in memory of a client device and/or one or more servers of a content management system and perform one or more functions when a processor executes the software associated with the module. The memory can be a non-transitory computer-readable medium.

In some embodiments the computer-readable storage devices, mediums, and memories can include a cable or

14

wireless signal containing a bit stream and the like. However, when mentioned, non-transitory computer-readable storage media expressly exclude media such as energy, carrier signals, electromagnetic waves, and signals per se.

Methods according to the above-described examples can be implemented using computer-executable instructions that are stored or otherwise available from computer readable media. Such instructions can comprise, for example, instructions and data which cause or otherwise configure a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Portions of computer resources used can be accessible over a network. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, firmware, or source code. Examples of computer-readable media that may be used to store instructions, information used, and/or information created during methods according to described examples include magnetic or optical disks, flash memory, USB devices provided with non-volatile memory, networked storage devices, and so on.

Devices implementing methods according to these disclosures can comprise hardware, firmware and/or software, and can take any of a variety of form factors. Typical examples of such form factors include laptops, smart phones, small form factor personal computers, personal digital assistants, and so on. Functionality described herein also can be embodied in peripherals or add-in cards. Such functionality can also be implemented on a circuit board among different chips or different processes executing in a single device, by way of further example.

The instructions, media for conveying such instructions, computing resources for executing them, and other structures for supporting such computing resources are means for providing the functions described in these disclosures.

Although a variety of examples and other information was used to explain aspects within the scope of the appended claims, no limitation of the claims should be implied based on particular features or arrangements in such examples, as one of ordinary skill would be able to use these examples to derive a wide variety of implementations. Further and although some subject matter may have been described in language specific to examples of structural features and/or method steps, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to these described features or acts. For example, such functionality can be distributed differently or performed in components other than those identified herein. Rather, the described features and steps are disclosed as examples of components of systems and methods within the scope of the appended claims.

What is claimed is:

1. A computer implemented method, comprising:
 - creating a plurality of multimedia assets at a caching unit,
 - for each one of the plurality of multimedia assets:
 - determining that there is sufficient storage space for the multimedia asset at the caching unit;
 - generating a channel identifier for the multimedia asset;
 - retrieving from a cloud service a plurality of media files to be included in the multimedia asset including a media identifier for each one of the media files;
 - for each one of the media files, creating a custom hypertext transfer markup language (HTML) user interface that includes video;
 - encoding each of the media files and encapsulating them together using an MPEG transport stream format;

US 11,140,441 B2

15

storing each one of the multimedia assets at the caching unit;
 receiving a request at the caching unit for one of the multimedia assets from a broadcasting unit; and
 responsive to receiving the request, the caching unit providing the multimedia asset to the broadcasting unit in the MPEG transport stream format.

2. The method of claim 1, wherein receiving the requests for the multimedia assets comprising receiving the requests over a REST API, Or: wherein retrieving the plurality of media files is performed using a REST API.

3. The method of claim 1, wherein creating the custom HTML user interface for each media file is performed using Javascript.

4. The method of claim 1, wherein encoding each of the media files include encoding each of the media files at a specified bit rate.

5. The method of claim 1, wherein encoding each of the media files include encoding each of the media files using a H.264 encoder.

6. The method of claim 1, wherein the custom HTML user interfaces are configured to be rendered using a Webkit browser.

7. The method of claim 1, wherein the video for the custom HTML interfaces includes video taken from the media file.

8. The method of claim 1, wherein encoding the media file is performed using an AC-3 file format for audio content in the media file.

9. The method of claim 1, wherein the custom HTML user interfaces are configured so that multiple custom HTML user interfaces of the multimedia asset can be retrieved using a web rendering engine.

10. A computer-implemented method comprising:
 virtualizing a system including a virtualized caching server, wherein the virtualized caching server performs:

receiving from a content provider, a request for at least one media stream for playback on a broadcast media channel, wherein the at least one media stream includes a plurality of multimedia items of different types;

obtaining content corresponding to the plurality of multimedia items from at least one source offering the content in at least one first format;

rendering a web page by a browser using the content; generating a temporal sequence of screen captures of the rendered web page, where each screen capture defines all the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time;

assembling the at least one media stream using the temporal sequence of screen captures; and

providing the at least one media stream to the content provider for broadcast on the broadcast media channel.

11. The method of claim 10, wherein the obtaining comprises: retrieving, for each of the plurality of multimedia items, at least one audio file corresponding to an audio component of the multimedia item and a plurality of screen captures corresponding to a video component of the multimedia item.

12. The method of claim 11, wherein the generating comprises: combining the plurality of screen captures and the at least one audio file to create each of the plurality of multimedia items.

16

13. The method of claim 12, further comprising: the virtualized caching server detecting a change at the at least one webpage corresponding to at least one of the plurality of multimedia items; in response to detecting the change, retrieving a new plurality of screen captures from the at least one webpage for the at least one of the plurality of multimedia items; and reassembling the at least one of the plurality of multimedia items using the new plurality of screen captures to yield an updated multimedia item.

14. The method of claim 11, wherein the retrieving of the plurality of screen captures comprises obtaining the plurality of screen captures from a playback of a video on the at least one webpage.

15. The method of claim 14, wherein the combining of the plurality of screen captures comprises metadata corresponding to at least one of a song title, an artist, and a music genre.

16. The method of claim 10, further comprising: the virtualized caching server identifying the at least one webpage based on a channel identifier associated with the broadcast media channel.

17. The method of claim 10, further comprising: the virtualized server querying the content provider to determine the second format, wherein the second format corresponds to a Moving Picture Experts Group (MPEG) format.

18. The method of claim 10, wherein the content provider is a cable television operator or a satellite television operator.

19. The method of claim 10, wherein providing the at least one media stream to the content provider comprises streaming the at least one media stream to an MPEG multiplexer associated with the content provider.

20. The method of claim 10, further comprising: the virtualized caching server provisioning, on at least one server, at least one virtual machine for performing the receiving, obtaining, generating, assembling, and providing.

21. The method of claim 10, wherein virtualizing the system is performed using one of Docker, Xenserver, or a virtual machine.

22. The method of claim 10, wherein virtualizing the system includes virtualizing the system to include at least one virtualized multicasting server that perform the providing the at least one media stream to the content provider.

23. The method of claim 10, wherein virtualizing the system is performed for one channel, wherein the virtualized caching server is operable for one channel, wherein virtualizing the system is repeated for each one of a plurality of other channels.

24. The method of claim 10, wherein receiving the request is performed using a JSON API.

25. A computer-implemented method comprising:

receiving, from a content provider, a request for at least one media stream for playback on a broadcast media channel, wherein the at least one media stream includes a plurality of multimedia items of different types;

obtaining content corresponding to the plurality of multimedia items from at least one source offering the content in at least one first format, performed over a first TCP-IP socket of a caching unit;

rendering a web page by a browser using the content; generating a temporal sequence of screen captures of the rendered web page, where each screen capture defines all the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time;

assembling the at least one media stream using the temporal sequence of screen captures;

US 11,140,441 B2

17

providing the at least one media stream to the content
provider for broadcast on the broadcast media channel;
and
providing, at a second TCP/IP socket of the caching unit,
a status of the obtaining, generating, and assembling. 5
26. A computer-implemented method comprising:
at a caching unit:
receiving, from a content provider, a request for at least
one media stream for playback on a broadcast media
channel, wherein the at least one media stream 10
includes a plurality of multimedia items of different
types;
obtaining content corresponding to the plurality of
multimedia items from at least one source offering
the content in at least one first format; 15
rendering a web page by a browser using the content;
generating a temporal sequence of screen captures of
the rendered web page, where each screen capture

18

defines all the content of the web page at a given
time, and at least two adjacent screen captures illus-
trate a dynamic change of at least a portion of the
content over time;
assembling the at least one media stream using the
temporal sequence of screen captures;
at a multicast unit that is operable connected to the
caching unit:
providing the at least one media stream to the content
provider for broadcast on the broadcast media chan-
nel;
recording the media stream from the caching unit to
produce a recorded media stream; and
upon a fault being detected at the caching unit, provid-
ing at least a portion of the recorded media stream to
the content provider.

* * * * *

Exhibit D

**IN THE UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF FLORIDA
MIAMI DIVISION**

CASE No. 1:24-cv-21226-RAR

DR. EDWIN A. HERNANDEZ and
EGLA CORP.,

Plaintiffs,

v.

STINGRAY GROUP INC. f/k/a STINGRAY
DIGITAL GROUP, INC., STINGRAY
MUSIC USA, INC., MOOD MEDIA LLC
f/k/a MOOD MEDIA CORPORATION,
AT&T ENTERPRISES, LLC f/k/a AT&T
CORP., MILLICOM INTERNATIONAL
SERVICES, LLC, and BLUE STREAM
COMMUNICATIONS, LLC dba BLUE
STREAM FIBER,

Defendants.

JURY TRIAL DEMANDED

**DECLARATION OF DR. SAM MALEK
REGARDING CLAIM CONSTRUCTION**

Signed: _____



Dr. Sam Malek

February 6, 2025

I. Introduction

1. I have been retained as an expert on behalf of Defendant Stingray Group, Inc. (“Stingray”) in the above captioned cases to provide my professional opinions and conclusions relating to certain claim construction and indefiniteness issues involving U.S. Patent No. 10,123,074 (“the ’074 Patent”), U.S. Patent No. 10,524,002 (“the ’002 Patent”), and U.S. Patent No. 11,140,441 (“the ’441 Patent”) (collectively, “the Patents-at-Issue”).

II. Summary of Opinions

2. As detailed below, it is my opinion that a Person of Ordinary Skill in the Art (a “POSITA”)—in light of the intrinsic and extrinsic evidence and their own general knowledge—would understand the claim terms identified by Stingray from the Patents-at-Issue to have the following meanings:

3. The claim term “a web page” means “an HTML document that is accessible on the world wide web.”

4. The claim term “rendering a web page” means “converting a web page’s HTML code into displayable content.”

5. The claim term “rendered web page” means “displayable content generated from a document’s HTML code.”

6. The claim term “browser” means “an application that converts HTML code into displayable content.”

7. The claim term “a sequence of screen captures” means “two or more screen captures of the rendered web page taken at different points in time.”

8. The claim term “illustrate a dynamic change” means “illustrate the playback of video, animation or similar motion or change occurring on the rendered web page.”

9. Additionally, as detailed below, it is my opinion that the claim terms “output the parameters via MPEG outputs,” “multimedia assets,” and “maintain generation of video and audio files, even in failure cases” fail to inform those skilled in the art about the scope of the invention with reasonable certainty and are, therefore, indefinite.

III. Qualifications

10. I have summarized in this section my educational background, career history, publications, and other relevant qualifications. My full *curriculum vita* is attached as Exhibit A to this declaration.

11. I am a Full Professor with tenure in the Informatics Department within the School of Information and Computer Sciences at the University of California, Irvine. I am also the Director of the Software Engineering and Analysis Laboratory. My research interests lie broadly in the field of software engineering, with a focus on software analysis and testing, software security, distributed software systems, software architecture, and mobile/smartphone computing, among other areas. The underlying theme of my research has been to devise techniques and tools that aid with the construction, analysis, and maintenance of large-scale software systems.

12. I received my Ph.D. and M.S. degrees in Computer Science from the University of Southern California and my B.S. degree in Information and Computer Science from the University of California, Irvine. I have received numerous awards for my research contributions, including the Association for Computing Machinery Special Interest Group on Software Engineering Test of Time Award (2020), the National Science Foundation CAREER award (2013), GMU Emerging Researcher/Scholar/Creator award (2013), and GMU Computer Science Department Outstanding Faculty Research Award (2011). I have authored over 130 peer-reviewed research publications.

These publications have been highly cited (more than 11,500 times according to scholar.google.com).

13. I have received numerous research grants from a variety of organizations, including the National Science Foundation (NSF), Federal Bureau of Investigation (FBI), Air Force Office of Scientific Research, Department of Homeland Security (DHS), National Security Agency (NSA), Defense Advanced Research Projects Agency (DARPA), Office of the Director of National Intelligence, and Army Research Office among others.

14. I have several years of experience as a software engineer, software architect, and programmer at a variety of companies, including the Boeing Company, PricewaterhouseCoopers Consulting (later acquired by International Business Machines (“IBM”) Corporation), FieldCentrix, and Neural Computing Systems Lab. During my tenure in the industry, I gained professional experience in the design, development, and management of large-scale software engineering efforts.

15. My professional experience in industry is highly relevant to the technologies at issue in this case. As set forth above, during my tenure in industry, I contributed to the design and development of several large-scale enterprise and ecommerce software systems. In many of these systems, I designed and developed software systems that created or processed multimedia. For example, at the Boeing Corporation, I was part of the team that designed and developed autonomous aerial vehicles, nowadays commonly referred to as drones. Notably, I contributed to the development of technologies that allowed for creation, collection, and processing of multimedia content from drones. At the Neural Computing Systems Labs, I contributed to the development of several techniques for detection of obfuscated objects in satellite-captured images and videos as part of a military-sponsored project. At PricewaterhouseCoopers, I developed

several web-based applications, including energy management portals and ecommerce applications, that involved creation, storage, and processing of media content, such as images and videos. I used a variety of relevant technologies in these projects, including HTML, JavaScript, Java Applet, Java Servlet, and different forms of multimedia encoding.

16. My scholarly activities are highly relevant to the technologies at issue in this case. Over the past 20 years, I have conducted original research and published peer-reviewed manuscripts related to the technologies at issue in this case. For example, in collaboration with colleagues from NASA's Jet Propulsion Lab, I developed a middleware, called GLIDE, that aimed to facilitate the transfer, storage, and processing of media content (e.g., satellite images obtained from various telescopes) in highly distributed settings. In a paper published in the European Grid Conference in 2005, I illustrated and evaluated the utility of this technology using a mobile media sharing application. The mobile application allowed for users to query, search, locate, and retrieve MP3 resources across a set of mobile, distributed, and resource-constrained devices. Users would query mobile media servers for MP3 files by specifying values for genre and quality of the MP3, and if found, the MP3 files were streamed asynchronously to the requesting mobile media client.

17. My pedagogical activities are highly relevant to the technologies at issue in this case. I regularly teach courses at both undergraduate and graduate levels at my university. I have previously taught a course on Internet Applications Engineering, as part of which I cover topics related to web technologies, such as HTML, JavaScript, CSS, content delivery networks, multimedia types, etc. I have also taught several variations of a course on Distributed Software Architectures, as part of which I teach students about various architectural paradigms, including architectural styles that are suitable for streaming multimedia content over wide-area networks.

18. My service to the scientific community is highly relevant to the technologies at issue in this case. On more than 150 occasions, I have served as chair, committee member, and reviewer for numerous computer science journals, magazines, and conferences that publish cutting edge advancements in the field. I have served on the editorial board of the ACM Transactions on Software Engineering and Methodology, IEEE Transactions on Software Engineering, the ACM Transactions on Autonomous and Adaptive Systems, the Springer Journal of Automated Software Engineering, and the Springer Journal of Computing. In these roles, I have refereed thousands of articles submitted by the scientific community for their novelty and contribution to the existing body of knowledge. Many of these submissions have dealt with topics that are directly related to the technologies at issue in this case, including streaming technologies, web applications, and multimedia.

19. Because of my background, training, and experience, I am qualified as an expert in the technical areas relevant to the Patents-in-Suit.

20. I am being compensated for my time at my current standard rate of \$750 per hour for the services that I provide in connection with this case. That compensation is not contingent upon my performance, the outcome of the case, or any issues involved in or related to this case.

IV. Legal Standards

21. I am not an attorney. For the purposes of this report, I have been informed about certain aspects of the law that are relevant to my analysis and opinions.

22. I have been informed that, as a general rule, a claim term should be construed according to its plain and ordinary meaning, which is the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention.

23. I have been informed that determining the plain and ordinary meaning of a claim term involves considering the usage of the term in the context of the intrinsic evidence, including the claim language, patent specification and prosecution history.

24. I have been informed that extrinsic evidence such as technical dictionaries, treatises, and expert testimony may be considered to understand the underlying technology and the manner in which one skilled in the art might use the claim terms within that context, but extrinsic evidence may not be used for the purpose of altering the terms of the claim from their plain meaning nor to create contradictions in the terms of the claims.

25. I have been informed that the burden is on the accused infringer to prove indefiniteness by clear and convincing evidence. I understand that, to be valid, a patent's claims, viewed in light of the specification and prosecution history, must inform those skilled in the art about the scope of the invention with reasonable certainty.

V. Materials Considered

26. Among the materials I have reviewed in forming my opinions are the Patents-at-Issue, their prosecution file histories, and other materials cited in Defendants' Opening Claim Construction Brief.

VI. Person of Ordinary Skill in the Art

27. Based on the materials and information I have reviewed and on my experience in the technical areas relevant to the Patents-at-Issue, a person of ordinary skill in the art at the time of the invention would have at least a bachelor's or equivalent degree in computer science or a related field and at least 2 years of experience in multimedia processing or delivery using internet protocols. Furthermore, a person with less formal education but more experience, or more formal education but less experience, could have also met the relevant standard for a POSITA.

28. Based on my educational background, knowledge and experience, I was a POSITA as of December 22, 2014, the purported priority date for the Patents-at-Issue.

29. My opinions herein are based on my understanding of a POSITA at that time.

VII. Claim Constructions

A. “web page” / “rendering a web page” / “rendered web page” / “browser”

Patents and Claims	Claim Language	Construction
'074: 1-21 '002: 1-13 '441: 10-26	“web page”	“an HTML document that is accessible on the world wide web”
	“rendering a web page”	“converting a web page’s HTML code into displayable content”
	“rendered web page”	“displayable content generated from a document’s HTML code”
	“browser”	“an application that converts HTML code into displayable content”

30. All of the claims from the '074 Patent and the '002 Patent, along with claims 10-26 of the '441 Patent, require either “render[ing] a web page by a browser” and/or a “rendered web page.” For example, claim 1 of the '074 Patent recites “rendering a web page by a browser” and then “generating a temporal sequence of screen captures of the rendered web page.” See '074 Patent at Claim 1.

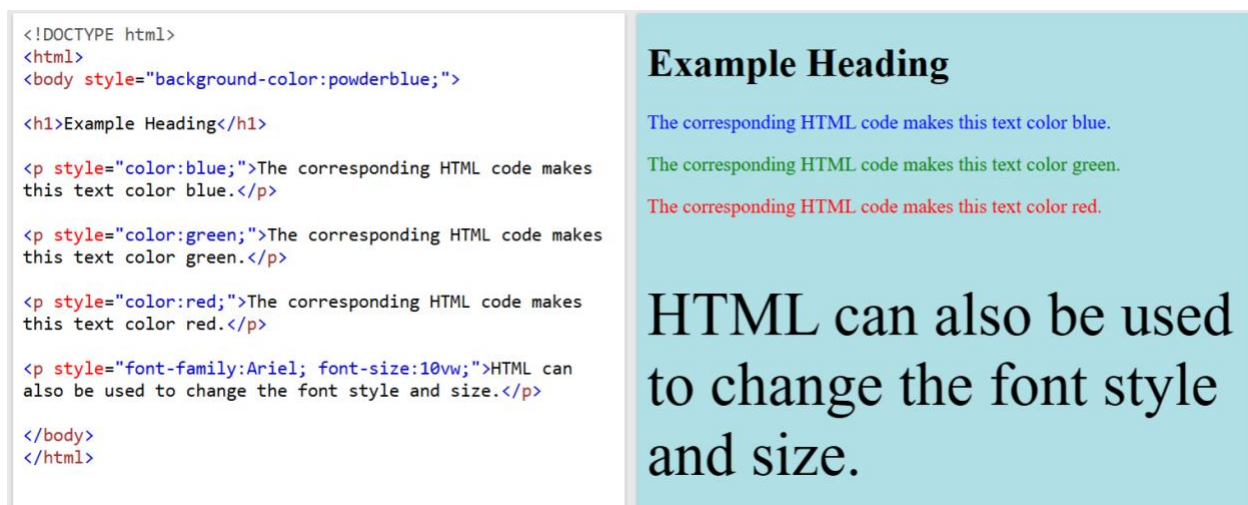
31. With respect to this process, the specifications¹ of the Patents-at-Issue disclose “JavaScript, CSS and HTML content files that can be rendered using a browser using WebKit (e.g. PhantomJS or Safari) or any other browser-rendering engine compatible with HTML4/5 or any future HTML version.” See '074 Patent at 7:1-4. Similarly, the patents disclose that “contents

¹ The Patents-at-Issue are continuations of the same patent application and share the same specification. For simplification, unless otherwise stated, my declaration cites to the portions of the '074 Patent.

corresponding to the multimedia items can be retrieved, downloaded, or otherwise obtained from one or more websites or cloud based servers” and that “the video component can be assembled by using one or more screen captures from a website.” *See* ’074 Patent at 7:55-65. The specifications state that “screen captures can include captures of still images presented by the website as well as captures of images from videos that are available on the website.” *See* ’074 Patent at 7:65-8:1. The specifications also state that “[c]apturing screens can be done using a tool similar to PhantomJS (Webkit-based headless engine for rendering images) or QtCapture, or simply capturing screen frame buffers from a machine running any software tool.” *See* ’074 Patent at 7:18-21. As a result, the intent of the purported invention of the Patents-at-Issue is be able to use pre-existing and commonly-known browser applications that are already “compatible with HTML” to render web pages into displayable content and then to use common screen capture software to capture that displayed content. Based on this, the patents’ reference to “rendering a web page with a browser” comports with a POSITA’s ordinary understanding of this concept.

32. Based on the Patents-at-Issue and the intended invention, the term “web page” is used as it would ordinarily be understood by a POSITA. Specifically, POSITAs would understand that a “web page” is a document on the world wide web that contains HTML code for defining the web page’s appearance. HTML, or hypertext markup language, is a standardized markup language used to display pages of content on the web. An HTML document contains HTML code that defines the structure and appearance of the page for display in a browser. For example, a web page designer may want the textual content of the page to be displayed in a specific font with a specific color and at a specific location in the display window. To accomplish this, the designer would include HTML code that defines the desired appearance of that page.

33. Because HTML is a standardized scripting language, web browsers—such as the “Safari” web browser used by Apple’s products and mentioned in the Patents-at-Issue—can then translate a page’s HTML into a graphical representation of the web page’s content, as intended by the web page designer. This process is known in the art as “rendering” and involves parsing the HTML document, which may also have embedded JavaScript or Cascading Style Sheets (“CSS”) code, to display content in the manner defined by the HTML document. As a simple example, the screen shot below shows HTML code on the left side—including code specifying the background color, text color, font style and size—and the rendered graphical representation of that HTML code on the right side, as it would be displayed in a browser:



34. POSITAs understand that a “browser” is an application that can read HTML code and generate a graphical representation of that HTML for display to a user. Browsers get their name by being able to browse the billions of pages on the web by rendering HTML. Importantly, not every application that can visualize content for a user is a “browser”—i.e., can browse the web—as that term is understood by a POSITA. For example, a document written in Microsoft Word is composed primarily of text and images. Microsoft’s Word software, which is a text editor, contains code that can read the Word file and generate a user interface with text and images, thus

allowing a user to edit the document. However, a text editor on its own cannot translate HTML code and generate displayable content as defined by that HTML code. For example, creating a Word file containing the HTML code from the image above would not generate the graphical representation shown on the right. Instead, loading such a Word file into the Microsoft Word text editor would merely show the **unrendered** HTML code, as follows:

```
<!DOCTYPE html>
<html>
<body style="background-color:powderblue;">

<h1>Example Heading</h1>

<p style="color:blue;">The corresponding HTML code makes this
text color blue.</p>

<p style="color:green;">The corresponding HTML code makes this
text color green.</p>

<p style="color:red;">The corresponding HTML code makes this
text color red.</p>

<p style="font-family:Ariel; font-size:10vw;">HTML can also be
used to change the font style and size.</p>

</body>
</html>
```

35. My review of additional information relating to this lawsuit confirms that Dr. Hernandez, the named inventor of the Patents-at-Issue, understood the phrase “rendering a web page by a browser” to have the meaning as discussed above. For example, prior to this lawsuit, I understand that Dr. Hernandez has attempted to license the Patents-at-Issue. In a licensing presentation for the ’074 Patent, he included the following slide describing his purported invention:

Analysis and Inventions on 10,123,074 Patent

- **Linear and VOD Streaming of Music, Music + Video, or Video/TV to Cable TV Operators**

- Rendering of HTML-based webpages from temporal screen captures and creating video streams with audio ready for broadcasting to MPEG Systems (aka Cable TV, DTH) for Linear or VOD Broadcasting
- Creating all screen captures and storing them in MPEG Format for 2-way interaction (VOD)
- Deployment and distribution using virtual machines or instances
- CDN integrated for Web, Mobile, and Cable TV broadcasting
- Fault-tolerance and detection



36. Dr. Hernandez’s description of his purported invention as “[r]endering of **HTML-based webpages**” shows that his understanding of the corresponding terms from the Patents-at-Issue is the same as an ordinary POSITA’s understanding. This also reflects the intent of his patents to capitalize on the standardized usage of HTML in webpages to generate displayable information using common web browsers and to then capture the resulting screens using available tools like QtCapture. See ’074 Patent at 7:18-21.

37. Based on the above, a POSITA would understand that a “web page” is an HTML document that is accessible on the world wide web. A POSITA would also understand that the act of “rendering a web page” means converting a web page’s HTML code into displayable content. Similarly, a POSITA would understand that a “rendered web page” is the displayable content generated from a document’s HTML code. A POSITA would further understand that a “browser” is an application that converts HTML code into displayable content.

B. “sequence of screen captures of the rendered web page”

Patents and Claims	Claim Language	Construction
’074: 1-21 ’002: 1-13 ’441: 10-26	“sequence of screen captures of the rendered web page”	“two or more screen captures of the rendered web page taken at different points in time”

38. All of the claims from the '074 Patent and the '002 Patent, along with claims 10-26 of the '441 Patent, further require a “sequence of screen captures of the rendered web page.” Continuing the discussion from above, POSITAs would understand that the term “screen captures of the rendered web page” is used in accordance with its ordinary meaning—i.e., two or more screen shots of what would be displayed on the screen following the rendering process.

39. The Patents-at-Issue further specify that a “sequence” of such screen captures are generated. The term “sequence” together with the plural form “screen captures” means that there are two or more screen captures taken one after another in series—i.e., at different times. In contrast, a single screen capture would neither be a “sequence” nor a plurality of “screen captures.” Similarly, taking a single screen capture at a single point of time and duplicating it would not generate a “sequence” of screen captures.

40. During prosecution of the '074 Patent, Dr. Hernandez amended the claims—including adding the requirement of a “sequence of screen captures” where “at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time.” Dr. Hernandez called this addition an “important distinction” over the cited prior art:

AMENDMENTS TO THE CLAIMS

Listing of Claims

1. (Currently Amended) A computer-implemented method, comprising:
 - receiving, from a content provider, a request for at least one media stream for playback on a broadcast media channel, wherein the at least one media stream includes a plurality of multimedia items of different types;
 - obtaining content corresponding to the plurality of multimedia items from at least one source website offering the content in at least one first format;
 - rendering a web page by a browser using the content;
 - generating a temporal sequence of each of the plurality of multimedia items in a second format based on the content obtained from the at least one website, wherein the second format screen captures of the rendered web page, where each screen capture defines all the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time is compatible with the at least one content provider;
 - assembling the at least one media stream using the temporal sequence of screen captures each of the plurality of multimedia items generated; and
 - providing the at least one media stream to the content provider for broadcast on the broadcast media channel.

As noted above, Reynolds fails to teach each and every limitation of claim 1. More particularly, Raynolds fails to disclose and/or suggest generating a temporal sequence of screen captures of a website rendered using content corresponding to a plurality of multimedia items of a media stream for playback, where (a) each screen capture defines all the content of the website at a given time and (b) at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time. None of the cited references cures this deficiency of Reynolds. There is no motivation in the cited references for modifying the inventions thereof so as to obtain the present solution.

This is an important distinction between the claimed solution and the cited references. In this regard, it should be understood that the present solution provides a more efficient and less computationally intensive process for providing content to set-top boxes. Also, the claimed solution allows for simplification of set-top boxes. This is not the case in any of the cited references. For example, in Reynolds, a special set-top box is required for decoding the broadcast stream and rendering the media stream for playback.

41. Based on this amendment, the Examiner allowed the claims, stating in the Notice of Allowability that “[w]hile it is known in the art to have multiple screen shots of a webpage (for

example, web.archive.org grabs entire webpages at different points in time),” that would not result in applicant’s invention. As stated by the Examiner, “[n]one of the art of record teaches the . . . sequence feature which includes a time and two adjacent times that illustrate a dynamic change of at least a portion of the content over time.”

Allowable Subject Matter

The following is an examiner’s statement of reasons for allowance:

Applicant’s arguments in light of the amendments are fully persuasive with respect to Reynolds.

While it is known in the art to have multiple screen shots of a webpage (for example web.archive.org grabs entire webpages at different points of time), and it is known to get a screen capture of content to create an index for an electronic program guide (for example US 2010/0192178 A1, US 2008/0098426 A1), but none of these would result in applicants invention. At best they would teach repetitively screen capturing an electronic program guide, or scanning channels and capturing on screen data and OCRing it to build a program guide.

None of the art of record teaches the requesting feature, the multimedia items of different types, the rendering by a browser feature, the temporal sequence feature which includes a time and two adjacent times that illustrate a dynamic change of least a portion of the content over time, and captures the entire web page, assembling the stream into a media stream using the temporal sequence, and providing the media stream for broadcast on the broadcast media channel.

42. As a result, the Examiner’s Notice of Allowability recognizes that a “sequence” of screen captures is composed of screen captures from different points in time.

43. The Patents-at-Issue further require that “at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time,” which I discuss in detail below. I note that this further bolsters the construction of “sequence of screen captures” as meaning two or more screen captures taken “at different points in time.” Given that there must be two adjacent screen captures of the rendered web page that illustrate a dynamic change of its content over time, those two screen captures must logically be taken at different points in time

rather than merely being duplicates of a single screen capture (which would not illustrate any change, let alone the required “dynamic change”).

44. Based on the above, a POSITA would understand that a “sequence of screen captures of the rendered web page” is two or more screen captures of the rendered web page taken at different points in time.

C. “illustrate a dynamic change”

Patents and Claims	Claim Language	Construction
'074: 1-21 '002: 1-13 '441: 10-26	“illustrate a dynamic change”	“illustrate the playback of video, animation or similar motion or change occurring on the rendered web page”

45. All of the claims from the '074 Patent and the '002 Patent, along with claims 10-26 of the '441 Patent, further require that “at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time.”

46. Neither the term “dynamic change” nor “dynamic” on its own appear anywhere in the Patents-at-Issue or the prosecution histories. There is no definition of the term “dynamic change” in the patent nor a definition of what it means for adjacent screen captures to “illustrate a dynamic change” of at least a portion of the content over time. In seeking to overcome the Examiner’s rejection, I note that Dr. Hernandez amended his claims to specifically require a “dynamic change.” Notably, Dr. Hernandez did not amend his claims to merely require “any change” or “a change”, and the Examiner’s Notice of Allowability reflects that screen captures illustrating changes in a rendered web page were commonly known in the art.

47. For example, in the Notice of Allowability, the Examiner explicitly stated that “it is known in the art to have multiple screen shots of a webpage (for example, web.archive.org grabs entire webpages at different points in time).” POSITAs are familiar with the publicly available

website Archive.org—also known as the Way Back Machine—which periodically crawls webpages, taking screen captures to preserve their content for future use. For example, the Way Back Machine may take a screen capture of a given web page this year and take a subsequent screen capture a month or even a year later. Indeed, the entire purpose of the Way Back Machine is to record **changes** in such web pages over time, forming a digital archive of the internet. Today, that digital archive comprises over 850 billion web pages and is commonly used by the public.

48. As a result, a “dynamic” change must mean something different from “any” change in the underlying web page’s content. However, because there is no disclosure of this concept in the Patents-at-Issue, a POSITA would not know with any reasonable certainty what this term means. Similarly, a POSITA would not be able to determine with any reasonable certainty when a mere change in the rendered web page becomes a “dynamic” change, such as would trigger infringement of the patent claims. Given that the term “dynamic” has no definite or measurable meaning in the context of the Patents-at-Issue, a POSITA would not know whether a change illustrated by two adjacent screen shots is an infringing “dynamic change” or merely a non-infringing “non-dynamic change.” Accordingly, it is my opinion that the term “dynamic change” is indefinite unless it is tied to the concept of “dynamic content.”

49. In the context of web design and mobile applications, a POSITA understands that “dynamic content” refers to the playback of animation, video or similar motion or change (such as content disappearing, reappearing or moving) that is actively occurring on the rendered web page or, in the context of mobile apps, in the mobile application. For example, an academic paper I co-authored, titled “Automated Accessibility Analysis of Dynamic Content Changes on Mobile Apps,” discusses that dynamic content includes visual changes occurring on the screen, such as appearing content, disappearing content, short-lived content, moving content, and content

modification. See Exhibit I to the Claim Construction Brief at 3. A POSITA understands that such dynamic content includes playback of video or animations in a web page. Dynamic content is included in a web page through the use of Dynamic HTML (“DHTML”), which uses JavaScript to actively modify HTML elements in the web page in real time, creating interactive elements like video and animations.

50. Although the patent specifications never use the term “dynamic change,” a POSITA’s understanding of dynamic content—such as the playback of video or animations—is consistent with the disclosed embodiment of the alleged invention. For example, I note that the patents describe taking screen captures from the playback of video on a web page, stating “the method can capture a video file from another rendering function, which can create a sequence of screens that can be used to generate a video file”. See ’074 Patent at 7:22-24. Similarly, the patents elsewhere state that the screen captures can include “captures of images from video that are available on the website.” See ’074 Patent at 7:65-8:1. Given that there is no explicit discussion of the term “dynamic change” in the patents, to the extent it has a definite meaning, it would be understood in light of a POSITA’s ordinary understanding of dynamic content. For example, “still images” from a static—*i.e.*, non-dynamic—web page are not dynamic content to a POSITA. Thus, screen capturing still images from a static web page would not “illustrate a dynamic change” of its content over time. Similarly, because the prior art Wayback Machine captures still images from web pages in order to archive any changes over time, a POSITA would understand from the Examiner’s Notice of Allowability that merely capturing “still images” from a web page would not satisfy the “dynamic change” language, even if that web page was modified or similarly rewritten in the intervening time period between screen captures. As a result, a POSITA would

understand “illustrate a dynamic change” in the context of the patents to mean illustrate the playback of video, animation or similar motion or change occurring on the rendered web page.

D. “output the parameters via MPEG outputs”

Patents and Claims	Claim Language	Construction
'002: 7-12	“output the parameters via MPEG outputs”	Indefinite.

51. Claims 7-12 of the '002 Patent require “a multicast server configured to generate parameters relating to at least one of video content, image content and audio content” and also “configured to output the parameters via MPEG outputs.” See '002 Patent at Claim 7. Those claims then require a “caching server” that is “configured to receive and store the parameters.” See '002 Patent at Claim 7.

52. The claim language is clear that the required parameters need to be outputted—i.e., transmitted—from the multicast server to the caching server, where they are received and stored. For example, the patents state that “encoding flags or input parameters can be received which may include GOP (Group of Pictures) which is used by certain video encoders, resolution (e.g. 640x480) in pixels, bit rate in Kilobits per second, any encoding type of video (e.g. H.264), encoding type of audio (e.g. AC-3), duration of the video feed, etc.” See '002 Patent at 6:35-40. “In addition, a ‘format’ parameter can be received which refers to the output format or encapsulation, MPEG TS or MPEG4.” See '002 Patent at 6:40-42.

53. Notably, the claims do not simply require outputting the parameters to the caching server for storage. Instead, those parameters must be output in a specific way in order for the claims to be infringed—i.e., “via MPEG outputs.” However, the patents fail to provide any explanation of what it means to “output the parameters via MPEG outputs,” as required by the claim language.

54. A POSITA would understand that “MPEG” refers to the “Moving Pictures Experts Group,” which is an alliance of working groups that sets standards for media coding. For example, MPEG has developed various encoding formats, such as MPEG-2 and MPEG-4, which are used to encode video and audio data. Similarly, every reference in the ’002 Patent to “output” that conceivably relates to “MPEG” refers to the output of *the encoded data* itself—not to the input or format parameters used to encode that data. For example, the patent references that “once all screens and video are captured, *the resulting output created can be a video file* that then is either replaced or merged (multiplexed) with the original file.” See ’002 Patent at 7:26-29. Elsewhere, the patent similarly states that “[t]he output format can be an MPEG Transport Stream file that can be retrievable or streamed to the multicasting embodiment” or that the system “can multiplex audio and video files into a resulting MPEG Transport Stream (or MPEG-4 output, or any similar output).” See ’002 Patent at 7:34-36 and 7:41-45.

55. In stark contrast, the term “output[ting] the parameters via MPEG outputs” does not have an understood meaning in the art, and a POSITA would not understand what the term “via MPEG outputs” means in relation to the parameters used to encode data. As mentioned above, Dr. Hernandez did not seek to broadly claim merely outputting parameters, but rather specified that the parameters had to be output by the multicast server to the caching server “via MPEG outputs” in order for there to be infringement. But the patents do not define or even mention “via MPEG outputs” or attempt to explain what the inventor meant by this term as it relates to the recited parameters.

56. For example, a POSITA would not know whether outputting parameters “via MPEG outputs” requires first encoding those parameters in accordance with some unidentified MPEG encoding standard. Alternatively, a POSITA would not know whether the term “via MPEG

outputs” instead simply refers to hardware outputs of a dedicated MPEG hardware encoder card. For example, the ‘002 Patent mentions that “MSOs and DSSs use dedicated hardware encoders” as opposed to software codecs running on general purpose computers. See ‘002 Patent at 1:48-48. Similarly, a POSITA would not know whether “via MPEG outputs” is meant to refer to some specific set of “virtual Ethernet ports” or “physical interfaces” and, if so, how those ports/interfaces become “MPEG outputs” for purposes of infringement. See ‘002 Patent at 12:15-22 and Fig. 10 (referencing various Ethernet ports and physical interfaces). Neither the patents nor the figures ever refer to any such port or interface—hardware or virtualized—as an “MPEG output.”

57. Similarly, To the extent Dr. Hernandez suggests that the parameters are encoded in an MPEG-2 or MPEG-4 format (or some other MPEG video/audio format) and then transmitted between the multicast server and the caching server, the ‘002 Patent fails to explain how mere *parameters* can be encoded by a standard that encodes audio and video data. Similarly, to the extent Dr. Hernandez instead suggests that the parameters are transmitted between the multicast server and the caching server via an MPEG Transport Stream, the ‘002 Patent also fails to explain how this is accomplished.

58. Given that the patents do not use the term “via MPEG outputs” and given that the term has no understood meaning, particularly in the context of outputting parameters, the mere use of the term is indefinite.

E. “multimedia assets”

Patents and Claims	Claim Language	Construction
’441: 1-9	“multimedia assets”	Indefinite.

59. Claims 1-9 of the ’441 Patent require “creating a plurality of multimedia assets.”

60. The Patents-at-Issue do not use the terms “multimedia asset” or “asset” at any point in their specifications, and these terms appear nowhere in the prosecution history. On its own, the term “multimedia asset” has no universal meaning to a POSITA. Adding to this problem, the patents reference a variety of different “multimedia” components, including (1) “multimedia content;” (2) “multimedia files;” (3) “multimedia items;” (4) “multimedia streams;” and (5) “multimedia playlists.”

61. For example, with respect to the term “multimedia content,” the patent discusses broadly “delivering multimedia content from a cloud-based system to cable or satellite content providers.” See ’441 Patent at 1:21-23. In contrast, the term “multimedia file” is more narrowly defined and described by the patents as “includ[ing], as non-limiting examples, MPEG-4 Video files with H.264 encoded video and AAC encoded audio or MPEG-layer 3 audio as well (MPEG2 Video could also be used). See ’441 Patent at 4:11-14. Likewise, the patents describe “multimedia items such as audio/video multimedia items” that are assembled “using the new plurality of screen captures.” See ’441 Patent at Claim 13. The patents also reference “multimedia streams” and state that “[o]nce the audio and video files are created, they can be combined, multiplexed, or encoded to a format that can be utilized by the multimedia stream.” See ’441 Patent at 8:13-16 and 6:8-10 (stating that “Mediaplug can broadcast the multimedia MPEG stream”) and 1:61-64 (equating “multimedia streams” with “TV, radio, [and] Video-on-Demand” programming). Lastly, the patents also reference “multimedia playlists” and state that “the caching server can generate a ‘playlist file’ . . . which will be interpreted as a multimedia playlist file when requested.” See ’441 Patent at 6:27-29.

62. Given the use of “multimedia” to refer to and modify different concepts—combined with the fact that there is no mention of “multimedia asset” anywhere in the intrinsic record—a POSITA would not know what the term “multimedia asset” refers to.

63. For example, claim 1 requires: (1) that the “multimedia asset” be created at a caching unit; (2) for each “multimedia asset” determining if there is sufficient storage space; (3) generating a channel identifier for the “multimedia asset”; (4) retrieving a plurality of media files to be included in the “multimedia asset”; (5) storing the “multimedia asset” at the caching unit; (6) receiving a request at the caching unit for the “multimedia asset” from the broadcasting unit; and (7) the caching unit providing the “multimedia asset” to the broadcasting unit in the MPEG transport stream format. However, those requirements apply equally without regard to whether the “multimedia asset” refers to a “multimedia file,” a “multimedia stream,” a “multimedia item,” a “multimedia playlist,” or to just multimedia content in the abstract. The claim language fails to indicate which (if any) of those possibilities the term applies to, and a POSITA would not understand what the term is intended to cover. As a result, it is my opinion that the term “multimedia asset” is indefinite.

F. “maintain generation of video and audio files, even in failure cases”

Patents and Claims	Claim Language	Construction
'002: 7-12	“maintain generation of video and audio files, even in failure cases”	Indefinite.

64. Claims 7-12 of the '002 Patent require “generating a multicast stream with MPEG encoded video and audio” and “a monitoring system configured to maintain generation of video and audio files, even in failure cases.”

65. With respect to this requirement, the patents state that “mediaplug can also monitor all of the processes to ensure stability and continuous, uninterrupted operation.” See '002 Patent

at 5:65-67 and 8:63-66 (referencing a “monitoring block that can ensure that the process running the ‘Media Parallel encoder’ keeps running at all times and does not stop upon failure”).

66. As an initial matter, a POSITA would not know what it means to “maintain generation . . . even in failure cases.” A POSITA would understand that the term “failure case” generally means an unintended state that prevents the normal operation of a system. For example, intentionally shutting down a system for routine maintenance (i.e., entering an intended state) would not be considered a “failure case” by a POSITA. Similarly, a system designed with multiple tiers of redundancy, for example, would not be considered to have encountered a “failure case” when the system is simply load-balancing (as it is programmed and designed to do in normal operation), in order to maintain uninterrupted operation.

67. Conversely, a POSITA would understand that a system has encountered a “failure case” when it is (without intention) no longer able to do what it was designed to do. For example, a system operating on a single server or computer that experiences an electrical short or a sudden loss in power that shuts the system down would be viewed as encountering a failure case.

68. However, the fact that a video and audio distribution system is “maintaining generation of video and audio” would inherently mean that the system has not failed. In other words, as long as it is operating as intended—including through any system of redundancy or monitoring—there is no failure. Conversely, a POSITA would understand that a media distribution system has encountered a failure case when it is **not** able to maintain the generation of video and audio, as intended from its design. The entire point of designing a system, though, is to avoid encountering failure cases in which it would no longer be able to maintain the generation of video and audio files.

69. For example, with respect to the claim language at issue, a POSITA would not know whether their design of a multimedia distribution system with robust tiers of redundancy and/or monitoring is a system that simply **does not** fail easily or, on the other hand, is a system that is constantly “failing” but, nonetheless, is still maintaining the generation of audio/video files for streaming. As a result, the limitation “configured to maintain generation of video and audio files, even in failure cases” is indefinite.

70. Moreover, in my opinion, this limitation is also indefinite for an additional reason. Specifically, the limitation states that the system must be “configured to maintain generation of video and audio files, even in failure cases.” Notably, the limitation does not specify a discrete range or category of “failure cases” in which it must continue operation. Instead, on its face, the language requires that the system maintain generation in **all** failure cases, and Dr. Hernandez did not seek to limit the scope of this claim language. For example, under the plain language, the system would need to be able to maintain generation even in the event of widespread power outages or malware attacks. Conversely, if the system is unable to maintain generation in such an event, then it would not be “configured to maintain generation of video and audio files, even in failure cases,” as required by the claim language.

71. This problem is critical because a POSITA would have no way of knowing whether a given system is so configured. For example, even if a POSITA could test a given system under a variety of known failure cases, a POSITA would never be able to confirm that the system would maintain generation of video and audio in the limitless number of **unknown** failure cases. As a simple example, a POSITA may be able to design a system that is immune to a variety of known viruses or malware such that the system could continue operation even if it becomes infected. However, a POSITA could never say whether that system is immune to every possible virus or

malware that could be developed. As another example from our own technological past, entire computer systems had to be redesigned prior to New Year's Eve 2000 because of the "Y2K" computer bug, which resulted from prior programmers abbreviating date information in order to save memory space. However, for decades, the extent of that bug and the associated risks of failure were unknown to the programming world.

72. As a result, a POSITA would have no way of knowing if the system they were designing would be able to "maintain generation of video and audio files, even in failure cases"—i.e., they would have no way of knowing whether a system infringed the claims above.

73. Accordingly, for all of the above reasons, it is my opinion that the above claim language "maintain generation of video and audio files, even in failure cases" renders the corresponding claims indefinite.

I declare under penalty of perjury that the foregoing is true and correct.

Exhibit E

Docket No.: 162774.00102
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Edwin Hernandez-Mondragon

Application No.: 15/538,911

Confirmation No.: 8633

Filed: June 22, 2017

Art Unit: 2424

For: METHOD, SYSTEM, AND APPARATUS FOR
MULTIMEDIA CONTENT DELIVERY TO
CABLE TV AND SATELLITE OPERATORS

Examiner: Hunter B. Lonsberry

AMENDMENT

MS Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

INTRODUCTORY COMMENTS

In response to the non-final Office action dated January 26, 2018, please enter the following claims amendments and remarks into the file of the above-identified U.S. patent application.

Amendments to the Claims are reflected in the list of the claims which begins on page 2 of this paper.

Remarks/Arguments begin on page 7 of this paper.

Application No. 15/538,911
Amendment dated March 12, 2018
Reply to Non-Final Office Action mailed January 26, 2018

Docket No.: 162774.00102

AMENDMENTS TO THE CLAIMS

Listing of Claims

1. (Currently Amended) A computer-implemented method, comprising:
receiving, from a content provider, a request for at least one media stream for playback on a broadcast media channel, wherein the at least one media stream includes a plurality of multimedia items of different types;
obtaining content corresponding to the plurality of multimedia items from at least one ~~source website~~ offering the content in at least one first format;
rendering a web page by a browser using the content;
~~generating a temporal sequence of each of the plurality of multimedia items in a second format based on the content obtained from the at least one website, wherein the second format~~ screen captures of the rendered web page, where each screen capture defines all the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time ~~is compatible with the at least one content provider~~;
assembling the at least one media stream using the temporal sequence of screen captures ~~each of the plurality of multimedia items generated~~; and
providing the at least one media stream to the content provider for broadcast on the broadcast media channel.
2. (Original) The method of claim 1, wherein the obtaining comprises: retrieving, for each of the plurality of multimedia items, at least one audio file corresponding to an audio component of the multimedia item and a plurality of screen captures corresponding to a video component of the multimedia item.
3. (Original) The method of claim 2, wherein the generating comprises: combining the plurality of screen captures and the at least one audio file to create each of the plurality of multimedia items.

Application No. 15/538,911
Amendment dated March 12, 2018
Reply to Non-Final Office Action mailed January 26, 2018

Docket No.: 162774.00102

4. (Currently Amended) The method of claim 2, wherein the retrieving of the plurality of screen captures comprises obtaining the plurality of screen captures from a playback of a video on the at least one webpage~~website~~.
5. (Original) The method of claim 4, wherein the combining of the plurality of screen captures comprises metadata corresponding to at least one of a song title, an artist, and a music genre.
6. (Currently Amended) The method of claim 1, further comprising: identifying the at least one webpage~~website~~ based on a channel identifier associated with the broadcast media channel.
7. (Original) The method of claim 1, further comprising: querying the content provider to determine the second format, wherein the second format corresponds to a Moving Picture Experts Group (MPEG) format.
8. (Currently Amended) The method of claim 3, further comprising:
 - detecting a change at the at least one webpage~~website~~ corresponding to at least one of the plurality of multimedia items;
 - in response to detecting the change, retrieving a new plurality of -screen captures from the at least one webpage~~website~~ for the at least one of the plurality of multimedia items; and
 - reassembling the at least one of the plurality of multimedia items using the new plurality of screen captures to yield an updated multimedia item.
9. (Original) The method of claim 1, wherein the content provider is a cable television operator or a satellite television operator.

Application No. 15/538,911
 Amendment dated March 12, 2018
 Reply to Non-Final Office Action mailed January 26, 2018

Docket No.: 162774.00102

10. (Original) The method of claim 1, wherein providing the at least one media stream to the content provider comprises streaming the at least one media stream to an MPEG multiplexer associated with the content provider.

11. (Original) The method of claim 1, further comprising: provisioning, on at least one server, at least one virtual machine for performing the receiving, obtaining, generating, assembling, and providing.

12. (Currently Amended) A system comprising:
 at least one processor;
 a computer-readable storage medium having stored therein instructions which, when executed by the processor, cause the at least one processor to perform operations comprising:
 receive, from a content provider, a request for at least one multimedia stream~~content~~
 for playback on a broadcast media channel, the at least one media stream including a plurality of multimedia items of different types;
 retrieve, from a cloud based service, ~~offering the~~ multimedia content corresponding to the plurality of multimedia items, the multimedia content in a first format;
render a web page by a browser using the multimedia content;
generate a temporal sequence of that is not compatible with a required format associated with the content provider, a plurality of screen captures of the rendered web page, where each screen capture defines all the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time~~corresponding to a video component of the multimedia content and at least one audio file corresponding to an audio component of the multimedia content~~;
 merge the temporal sequence~~plurality of~~ screen captures with the at least one audio file to create the multimedia stream~~content in the required format~~; and
 send the multimedia stream~~content~~ to the content provider.

Application No. 15/538,911
Amendment dated March 12, 2018
Reply to Non-Final Office Action mailed January 26, 2018

Docket No.: 162774.00102

13. (Currently Amended) The system of claim 12, further comprising: retrieve a video from the cloud based service, wherein the ~~plurality of~~ screen captures correspond to a plurality of static images obtained from the video.

14. (Original) The system of claim 12, further comprising: select the multimedia content from the cloud based service based on a music genre associated with the broadcast media channel.

15. (Currently Amended) The system of claim 12, wherein the request ~~for the multimedia content~~ comprises a request for a plurality of multimedia streams for playback on a plurality of broadcast media channels.

16. (Currently Amended) The system of claim 12, wherein the ~~multimedia~~ stream~~content~~ is sent to a broadcast internet protocol (IP) address associated with the content provider.

17. (Currently Amended) A non-transitory computer-readable storage medium having stored therein instructions which, when executed by a processor, cause the processor to perform operations comprising:

receiving, from a content provider, a request for at least one media stream for playback on a broadcast media channel, wherein the at least one media stream includes a plurality of multimedia items of different types;

obtaining content corresponding to the plurality of multimedia items from at least one ~~source~~ website offering the content in at least one first format;

rendering a web page by a browser using the content;

generating a sequence of screen captures of the rendered web page, where each screen capture defines all of the plurality of multimedia items in a second format based on the content of the web page at a given time, and at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time ~~obtained from the at least one website, wherein the second format is compatible with the at least one content provider;~~

Application No. 15/538,911
Amendment dated March 12, 2018
Reply to Non-Final Office Action mailed January 26, 2018

Docket No.: 162774.00102

assembling the at least one media stream using ~~each of the~~ sequence of screen captures~~plurality of multimedia items generated~~; and

providing the at least one media stream to the content provider for broadcast on the broadcast media channel.

18. (Currently Amended) The non-transitory computer-readable storage medium of claim 17, wherein the obtaining comprises retrieving, for each of the plurality of multimedia items, at least one audio file corresponding to an audio component of the multimedia item and a plurality of screen captures corresponding to a video component of the multimedia item; and wherein the ~~assembling~~generating comprises combining the plurality of screen captures and the at least one audio file ~~to create each of the plurality of multimedia items~~.

19. (Original) The non-transitory computer-readable storage medium of claim 17, wherein the at least one media stream corresponds to at least one of an HTTP live stream (HLS), an HTTP playlist, and a Real-time Streaming Transport (RTSP) stream.

20. (Original) The non-transitory computer-readable storage medium of claim 17, storing additional instructions which, when executed by the processor, result in operations comprising:
detect a network failure affecting connectivity with the at least one website; and
in response, provide a previously recorded media stream to the content provider.

21. (Original) The non-transitory computer-readable storage medium of claim 17, wherein assembling the at least one media stream comprises inserting at least one MPEG packet including metadata corresponding to the multimedia item.

22. – 23. (Canceled)

Application No. 15/538,911
 Amendment dated March 12, 2018
 Reply to Non-Final Office Action mailed January 26, 2018

Docket No.: 162774.00102

REMARKS

These remarks are in response to the non-final Office action mailed January 26, 2018, ("Office Action"). This reply is timely filed. At the time of the Office Action, claims 1-23 were pending in the application.

I. Status of the Claims

Claims 1-4, 6-10, 12, 13, 15, 17, 18 and 21 have been rejected under 35 U.S.C. §102. Claims 5, 14, 11, 16, 19, 20, 23 and 24 have been rejected under 35 U.S.C. §103. The rejections are set forth in detail below.

Claims 1, 4, 6, 8, 9, 11, 13 and 15-18 have been amended. No new subject matter has been added by the claim amendments. Support for the amendments made to the independent claims can be found at least in FIGS. 5-7 and paragraphs [0030], [0031], [0038]-[0043], [0047], [0048] of the present application. Claims 22 and 23 have been canceled without prejudice or disclaimer to the subject matter therein.

II. Claims Rejections Under 35 U.S.C. §102

In the Office Action, claims 1-4, 6-10, 12, 13, 15, 17, 18 and 21 were rejected under 35 U.S.C. §102 as being anticipated by U.S. Patent Publication No. 2002/0138852 to Reynolds et al. ("Reynolds"). Applicant respectfully traverses the Examiner's rejection.

Reynolds generally discloses systems and methods for generating and transmitting a broadcast stream. The methods involve: obtaining a set of disparate media content elements (e.g., audio, video and metadata) from respective sources **102, 104, 106**¹; generating a plurality of disparate data streams by converting a format of each media content element to a transmission format²; packaging together the disparate data streams into an omnimedia package³; and transmitting the omnimedia package in a broadcast stream to a set-top box⁴.

¹ See paragraph [0022] of Reynolds.

² See paragraph [0049] of Reynolds.

³ See Abstract of Reynolds.

⁴ See paragraph [0007] of Reynolds.

Application No. 15/538,911
Amendment dated March 12, 2018
Reply to Non-Final Office Action mailed January 26, 2018

Docket No.: 162774.00102

Although Reynolds discloses systems and methods in which an omnimedia package is generated and transmitted to a set-top box, it fails to teach the present solution recited in amended independent claim 1. More particularly, Reynolds fails to disclose and/or suggest generating a temporal sequence of screen captures of a web page rendered by a browser using content corresponding to a plurality of multimedia items of a media stream for playback, where (a) each screen capture defines all the content of the web page at a given time and (b) at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time.

Therefore, independent claim 1 is not anticipated by Reynolds. As such, claim 1 is believed to be in condition for allowance. Independent claims 12 and 17 are similar to claim 1, albeit different in some ways. Accordingly, claims 12 and 17 are also believed to be allowable at least for the same reasons as claim 1 discussed herein. Each of the dependent claims is allowable at least by virtue of its dependence on an allowable base claim.

III. Claim Rejections Under 35 U.S.C. §103

In the Office Action, claims 5 and 14 were rejected under 35 U.S.C. §103 as being unpatentable over Reynolds in view of U.S. Patent Publication No. 2009/0178003 to Fiedler (“Fiedler”). Claim 11 was rejected under 35 U.S.C. §103 as being unpatentable over Reynolds in view of U.S. Patent Publication No. 2008/0040354 to Ray et al. (“Ray”). Claim 18 was rejected under 35 U.S.C. §103 as being unpatentable over Reynolds in view of U.S. Patent Publication No. 2014/0068690 to Luthra et al. (“Luthra”). Claim 20 was rejected under 35 U.S.C. §103 as being unpatentable over Reynolds in view of U.S. Patent Publication No. 2014/0143437 to Mathur et al. (“Mathur”). Claims 23 and 24 were rejected under 35 U.S.C. §103 as being unpatentable over Ray in view of U.S. Patent Publication No. 2014/0098815 to Mishra et al. (“Mishra”) and U.S. Patent Publication No. 2015/0319004 to Kwan et al. (“Kwan”). Applicant respectively traverses the Examiner’s rejections.

As noted above, Reynolds fails to teach each and every limitation of claim 1. More particularly, Reynolds fails to disclose and/or suggest generating a temporal sequence of screen captures of a website rendered using content corresponding to a plurality of multimedia items of a

Application No. 15/538,911
Amendment dated March 12, 2018
Reply to Non-Final Office Action mailed January 26, 2018

Docket No.: 162774.00102

media stream for playback, where (a) each screen capture defines all the content of the website at a given time and (b) at least two adjacent screen captures illustrate a dynamic change of at least a portion of the content over time. None of the cited references cures this deficiency of Reynolds. There is no motivation in the cited references for modifying the inventions thereof so as to obtain the present solution.

This is an important distinction between the claimed solution and the cited references. In this regard, it should be understood that the present solution provides a more efficient and less computationally intensive process for providing content to set-top boxes. Also, the claimed solution allows for simplification of set-top boxes. This is not the case in any of the cited references. For example, in Reynolds, a special set-top box is required for decoding the broadcast stream and rendering the media stream for playback.

In view of the forgoing, amended independent claim 1 is non-obvious in view of the cited references and the suggested combinations thereof. As such, amended independent claim 1 is believed to be in condition for allowance. Amended independent claim 12, 17 and 22 are the same as amended independent claim 1, albeit different in some ways. Accordingly, amended independent claims 12, 17 and 22 are also believed to be in condition allowance. Each of the dependent claims is allowable at least by virtue of its dependence on an allowable base claim.

IV. Conclusion

Applicant has made every effort to present claims that distinguish over the prior art, and believes that all claims are in condition for allowance. Nevertheless, Applicant invites the Examiner to call the undersigned if (s)he believes that a telephonic interview would expedite the prosecution of the application to an allowance. In view of the forgoing remarks, Applicant respectfully requests reconsideration and prompt allowance of the pending claims. Applicant believes no fee is due with this response. However, if a fee is due, please charge our Deposit Account No. 50-1943 under matter no. 162774.00101 from which the undersigned is authorized to draw.

Application No. 15/538,911
Amendment dated March 12, 2018
Reply to Non-Final Office Action mailed January 26, 2018

Docket No.: 162774.00102

Dated: March 12, 2018

Respectfully submitted,

Electronic Signature: /Carol E. Thorstad-Forsyth/
Carol E. Thorstad-Forsyth
Registration No.: 56,455
FOX ROTHSCHILD LLP
997 Lenox Drive, Building 3
Lawrenceville, New Jersey 08648-2311
(561) 804-4438
(609) 896-1469 (Fax)
Attorneys/Agents For Applicant

Exhibit F

Notice of Allowability	Application No. 15/538,911		Applicant(s) HERNANDEZ-MONDRAGON, Edwin A.	
	Examiner HUNTER B LONSBERRY		Art Unit 2424	AIA Status Yes

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to 3/15/18.

☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on _____.

2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

3. ☒ The allowed claim(s) is/are 1-21. As a result of the allowed claim(s), you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to **PPHfeedback@uspto.gov**.

4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

Certified copies:

a) ☐ All b) ☐ Some *c) ☐ None of the:

1. ☐ Certified copies of the priority documents have been received.

2. ☐ Certified copies of the priority documents have been received in Application No. _____.

3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

* Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file areply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.

☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☒ Notice of References Cited (PTO-892) 2. ☒ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____ 3. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material _____ 4. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____	5. ☐ Examiner's Amendment/Comment 6. ☒ Examiner's Statement of Reasons for Allowance 7. ☐ Other _____
--	---

/HUNTER B LONSBERRY/
 Primary Examiner, Art Unit 2424

Application/Control Number: 15/538,911
Art Unit: 2424

Page 2

DETAILED ACTION

Notice of Pre-AIA or AIA Status

The present application, filed on or after March 16, 2013, is being examined under the first inventor to file provisions of the AIA.

Allowable Subject Matter

The following is an examiner's statement of reasons for allowance:

Applicant's arguments in light of the amendments are fully persuasive with respect to Reynolds.

While it is known in the art to have multiple screen shots of a webpage (for example web.archive.org grabs entire webpages at different points of time), and it is known to get a screen capture of content to create an index for an electronic program guide (for example US 2010/0192178 A1, US 2008/0098426 A1), but none of these would result in applicant's invention. At best they would teach repetitively screen capturing an electronic program guide, or scanning channels and capturing on screen data and OCRing it to build a program guide.

None of the art of record teaches the requesting feature, the multimedia items of different types, the rendering by a browser feature, the temporal sequence feature which includes a time and two adjacent times that illustrate a dynamic change of least a portion of the content over time, and captures the entire web page, assembling the stream into a media stream using the temporal sequence, and providing the media stream for broadcast on the broadcast media channel.

The examiner has fully considered the claims with respect to the 101 statute and finds them fully compliant.

Claims 1-21 are allowed.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Application/Control Number: 15/538,911
Art Unit: 2424

Page 3

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to HUNTER B LONSBERRY whose telephone number is (571)272-7298. The examiner can normally be reached on M-F 8:00-21:00.

Examiner interviews are available via telephone, in-person, and video conferencing using a USPTO supplied web-based collaboration tool. To schedule an interview, applicant is encouraged to use the USPTO Automated Interview Request (AIR) at <http://www.uspto.gov/interviewpractice>.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Pankaj Kumar can be reached on (571)272-3011. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

HUNTER B. LONSBERRY
Primary Examiner
Art Unit 2424

/HUNTER B LONSBERRY/
Primary Examiner, Art Unit 2424

Exhibit G



Licensing Opportunity

Licensing to: Cloud to Cable Software: US
Patents: 10123074 , 16/152,606
(Allowed)., European Patent: EP3238457

By **Dr. Edwin A. Hernandez Mondragon**
Inventor and Owner
edwinhm@eglacorp.com

September 5th, 2019



US010123074B2

(12) **United States Patent**
Hernandez-Mondragon

(10) **Patent No.:** **US 10,123,074 B2**
(45) **Date of Patent:** **Nov. 6, 2018**

(54) **METHOD, SYSTEM, AND APPARATUS FOR MULTIMEDIA CONTENT DELIVERY TO CABLE TV AND SATELLITE OPERATORS**

(58) **Field of Classification Search**
CPC H04N 21/440218; H04N 21/233; H04N 21/234309; H04N 21/2387;
(Continued)

(71) Applicant: **Edwin A. Hernandez-Mondragon**,
Coral Springs, FL (US)

(72) Inventor: **Edwin A. Hernandez-Mondragon**,
Coral Springs, FL (US)

(56) **References Cited**
U.S. PATENT DOCUMENTS

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

6,886,178 B1 4/2005 Mao et al.
7,020,888 B2 3/2006 Reynolds et al.
(Continued)

(21) Appl. No.: **15/538,911**

(22) PCT Filed: **Dec. 22, 2015**

(86) PCT No.: **PCT/US2015/067464**

§ 371 (c)(1),
(2) Date: **Jun. 22, 2017**

(87) PCT Pub. No.: **WO2016/106360**

PCT Pub. Date: **Jun. 30, 2016**

(65) **Prior Publication Data**

US 2017/0374417 A1 Dec. 28, 2017

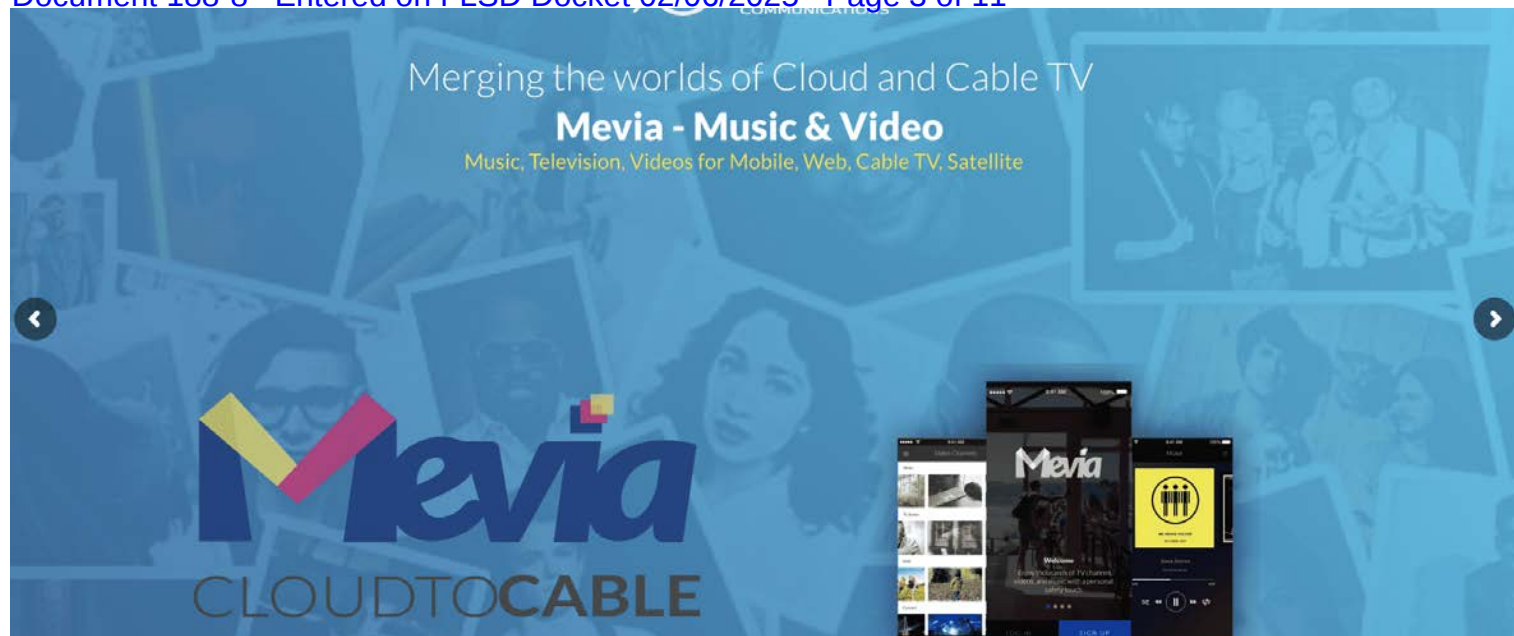
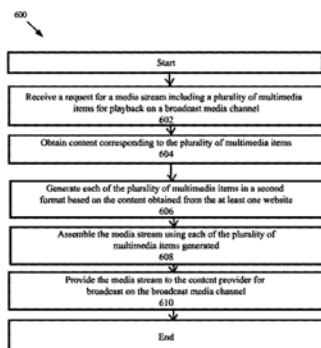
Related U.S. Application Data

(60) Provisional application No. 62/095,504, filed on Dec. 22, 2014.

(51) **Int. Cl.**
H04N 21/4402 (2011.01)
H04N 21/2343 (2011.01)
(Continued)

(52) **U.S. Cl.**
CPC ... **H04N 21/440218** (2013.01); **H04N 21/233** (2013.01); **H04N 21/2387** (2013.01);
(Continued)

21 Claims, 12 Drawing Sheets



Solution

Licensing of the 10,123,074 Portfolio of patents 'and software

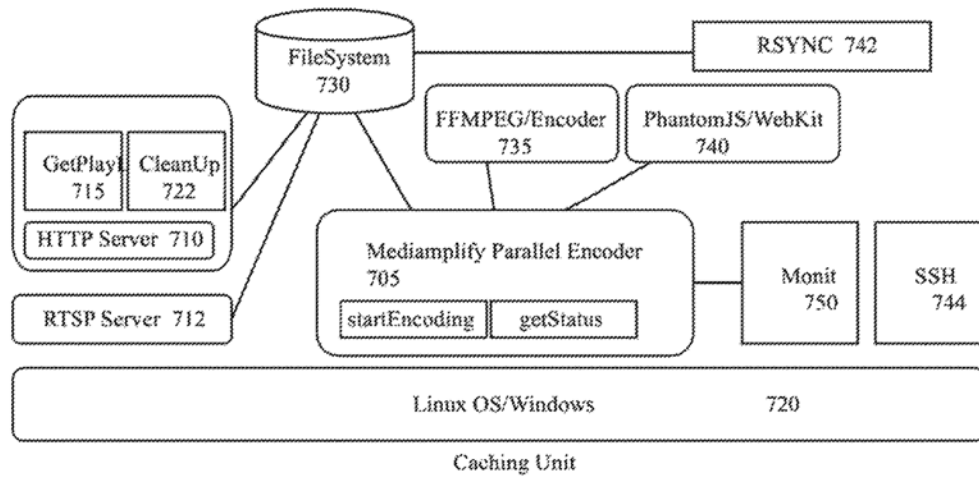
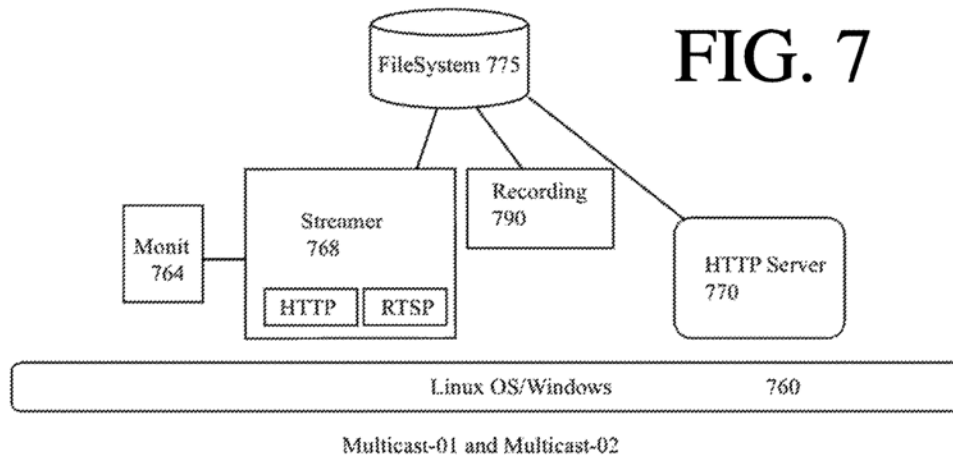


FIG. 7



Cloud to Cable
provides a system for
multimedia
broadcasting in the
cloud – web elements
with video & audio



Multicasting for IPTV and CABSAT Systems integrated with web-base interfaces and multimedia elements

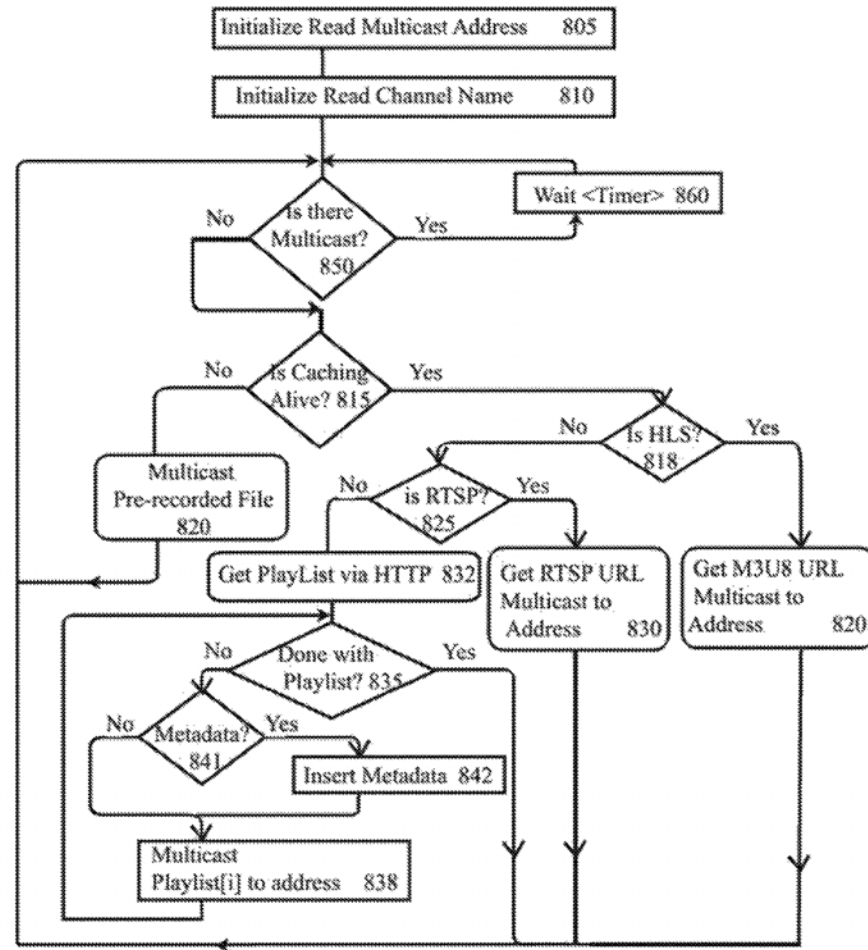


FIG. 8

Analysis and Inventions on 10,123,074 Patent

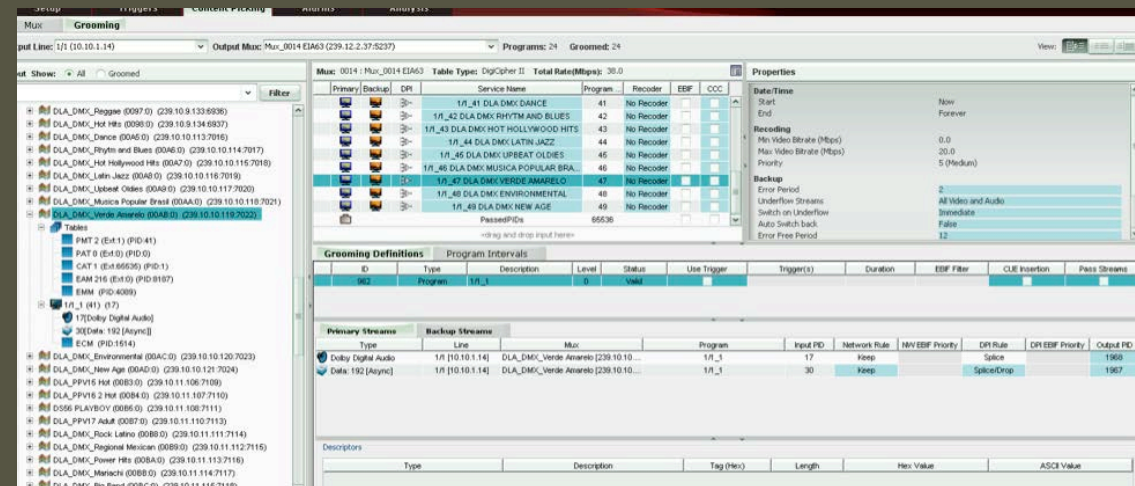
- **Linear and VOD Streaming of Music, Music + Video, or Video/TV to Cable TV Operators**
- **Rendering of HTML-based webpages** from temporal screen captures and creating video streams with audio ready for broadcasting to MPEG Systems (aka Cable TV, DTH) for Linear or VOD Broadcasting
- Creating all screen captures and storing them in MPEG Format for 2-way interaction (VOD)
- Deployment and distribution using virtual machines or instances
- CDN integrated for Web, Mobile, and Cable TV broadcasting
- Fault-tolerance and detection
- Multiple Streaming protocols supported
- **MPEG 2-way broadcasting, MPEG multicasting, HTTP Live Streaming**



Additional Intellectual Property – Software & Implementation



- Software for Cloud to Cable TV Appliance
 - Tested in Head ends in LATAM and US
- APIs and software for system and implementation
- Technical Assistance and Engineering Support



Customers

- 700k subscribers with CABLE COLOR in Honduras, Nicaragua and Guatemala



www.cablecolor.hn

CABLE COLOR

TE TRAE NUEVOS CANALES EN HD



discovery kds^{HD} | ID^{HD} INVESTIGATION DISCOVERY | h&h^{HD} | ANIMAL PLANET^{HD}

Discovery^{HD} | Discovery^{HD} WORLD | ESPN2^{HD}

Nuestros canales cuentan con:

DOLBY DIGITAL | SAP

Canal 702 ESPN 2 HD
Canal 729 Discovery Kids HD
Canal 773 Investigation Discovery HD
Canal 774 Discovery World HD
Canal 775 H&H HD
Canal 776 Discovery Channel HD
Canal 777 Animal Planet HD

f YouTube in Instagram

CONTRÁTALO LLAMANDO AL **2540-1234 ó 2262-0000**

US Patent
10,123,074
Cites Patents
in dispute
Stingray v.
Music Choice

Patent Citations (14)

Publication number	Priority date	Publication date	Assignee	Title
US20020138852A1	2000-11-27	2002-09-26	Steven Reynolds	System and method for providing an omnimedia package
US20030115612A1	1998-04-01	2003-06-19	Morecom, Inc.	Digital TV system with synchronized World Wide Web content
US7320025B1	2002-03-18	2008-01-15	Music Choice	Systems and methods for providing a broadcast entertainment service and an on-demand entertainment service
US20080040354A1	2006-08-10	2008-02-14	Qualcomm Incorporated	System and method for media content delivery
US20080098426A1 *	2006-10-23	2008-04-24	Candelore Brant L	Decoding multiple remote control code sets
US20090178003A1	2001-06-20	2009-07-09	Recent Memory Incorporated	Method for internet distribution of music and other streaming content
US20100192178A1 *	2009-01-26	2010-07-29	Candelore Brant L	Capture of stylized TV table data via OCR
US20120204217A1	2010-10-14	2012-08-09	Activevideo Networks, Inc.	Streaming Digital Video between Video Devices Using a Cable Television System
US20140068690A1	2012-08-31	2014-03-06	General Instrument Corporation	Broadcast content to http client conversion
US20140098815A1 *	2012-10-10	2014-04-10	Telefonaktiebolaget L M Ericsson (Publ)	Ip multicast service leave process for mpls-based virtual private cloud networking
US20140143437A1	2012-11-16	2014-05-22	Adobe Systems Incorporated	Converting Live Streaming Content to Video-on-Demand Streaming Content
US8769602B1	2001-08-28	2014-07-01	Music Choice	System and method for providing an interactive, visual complement to an audio program
US20140351871A1	2013-05-22	2014-11-27	Microsoft Corporation	Live media processing and streaming service
US20150319004A1 *	2007-08-31	2015-11-05	At&T Intellectual Property I, L.P.	System and method of monitoring video data packet delivery

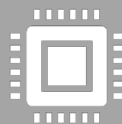
- Visual Component Patent
- VOD Patents

Music Choice filed its original Complaint against the Corporation on June 6, 2016, asserting infringement of four U.S. patents, namely, U.S. Patent Nos. 8,769,602 (the '602 Patent), 9,357,245 (the '245 Patent), 7,320,025 (the '025 Patent) and 9,351,045 (the '045 Patent).

Licensing of an entire media streaming portfolio



US Patent: 10,123,074



US Patent application: 16/152,606



European Patent Application : EP3238457
(A1)



Technology: Cloud to
Cable TV Software

Used by DMX Media
during the acquisition
with Stingray Digital

Thank you
Feel free to
contact US

www.eglacorp.com

Dr. Edwin A. Hernandez

sales@eglacorp.com

[561.306.4996](tel:561.306.4996)



Exhibit H



path{fill:none;stroke:#333333;stroke-width:1.5px;}
\$0.00
path{fill:none;stroke:#000;stroke-miterlimit:10;stroke-width:1.5px;}

MEVIA Patent Portfolio

[Home](#) [MEVIA Patent Portfolio](#)



MEVIA uses our patented technology platform. MEVIA brings together video and music all in one mobile experience. MEVIA connects web and mobile experiences with Cable TV broadcasting, IPTV systems, and Smart TVs. Our technologies are unique having our target markets in Latin America and the United States.

Our technology is protected by several patents US10123074, US10524002, US11140441, US12075116 and EP3238457.

This portfolio consists in 5 issued patents (4 US and 1 European, **US Patents 10,123,074, 10,524,002, 11,140,441, 12,075,116 European Patent EP3238457A1**) and 1 continuation patent **US 18/800,685**, which are critical to a rapidly developing new technology and new market

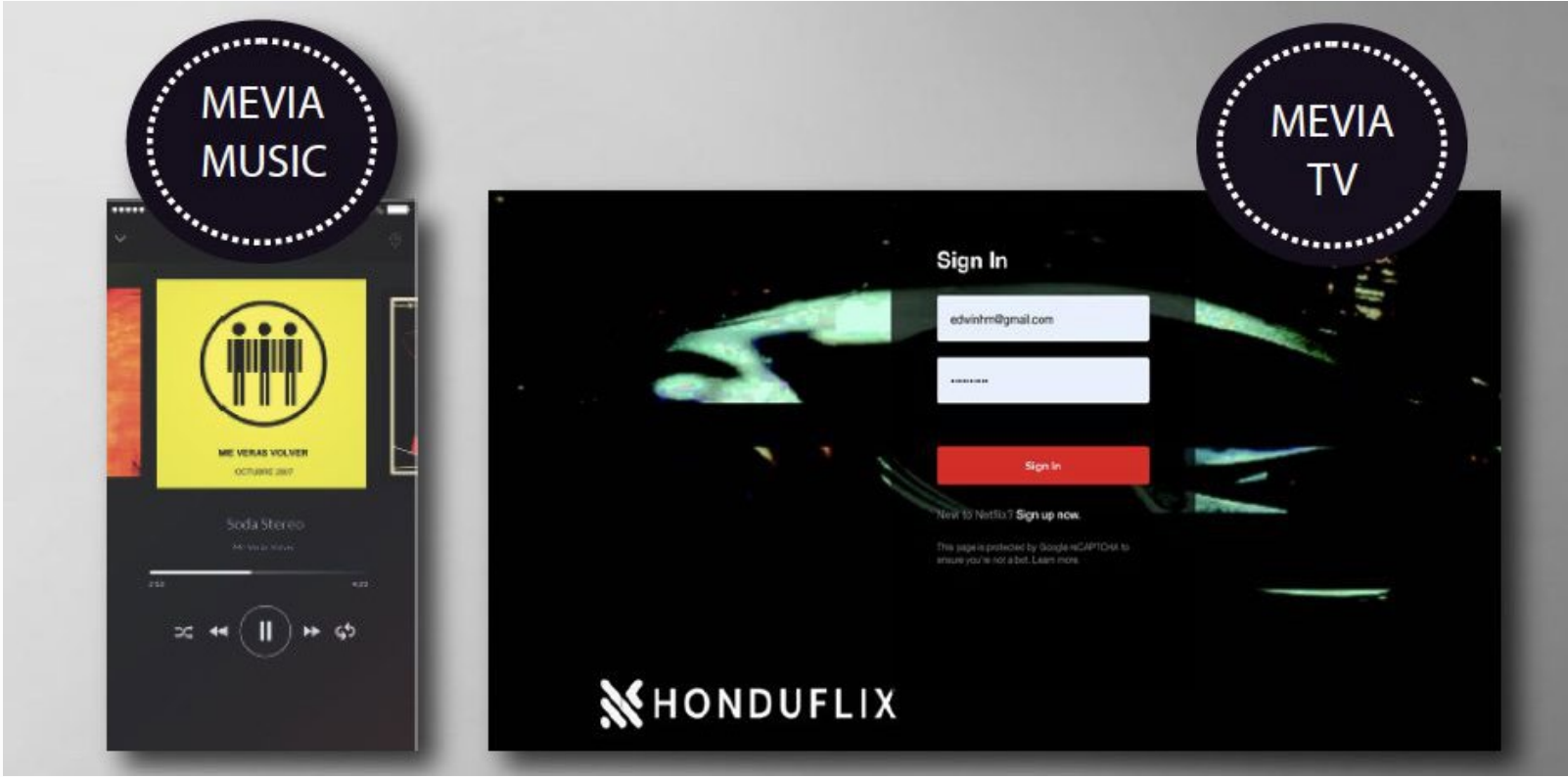
called NextGen TV (ATSC 3.0), Smart TV broadcasting, IPTV, and even in-car vehicle technologies.

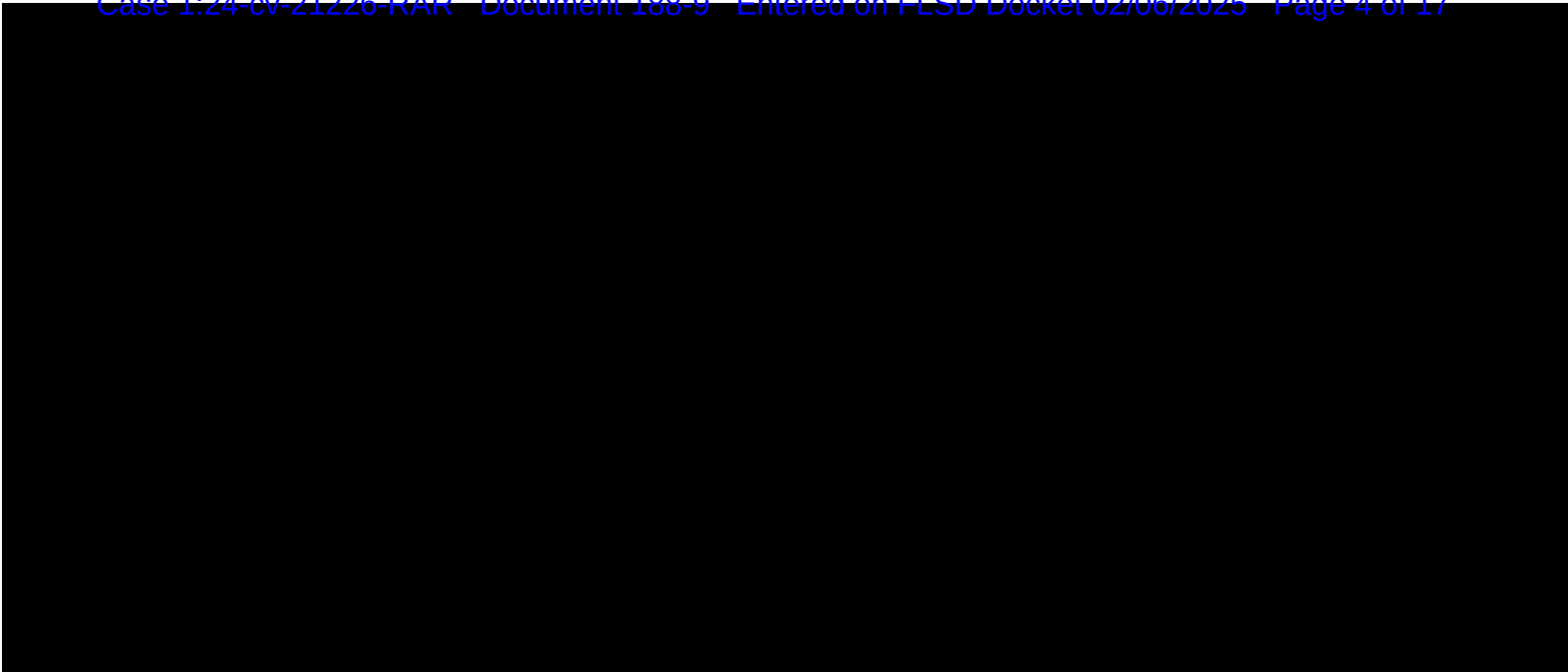
Dr. Edwin A. Hernandez PhD created these patent several years ago, and it was only recently he began to understand the importance of the patents and the long-term value proposition of the patents.

The original patents were developed when EGLA CORP was working on a project, and Dr. Hernandez’s interest in improving upon how "Text Boxes" and Metadata information were integrated into a final TV or video feed by Cable TV systems (e.g. DOCSIS) and other video broadcasting technologies.

The older technology and the solutions available were more hardware-based and was not HTML-enabled. Imagine the integration of Web-based stock tickers, news feeds, animation, ads, interstitials or any other type of final video stream you can view with integrated parts of the video screen.

[





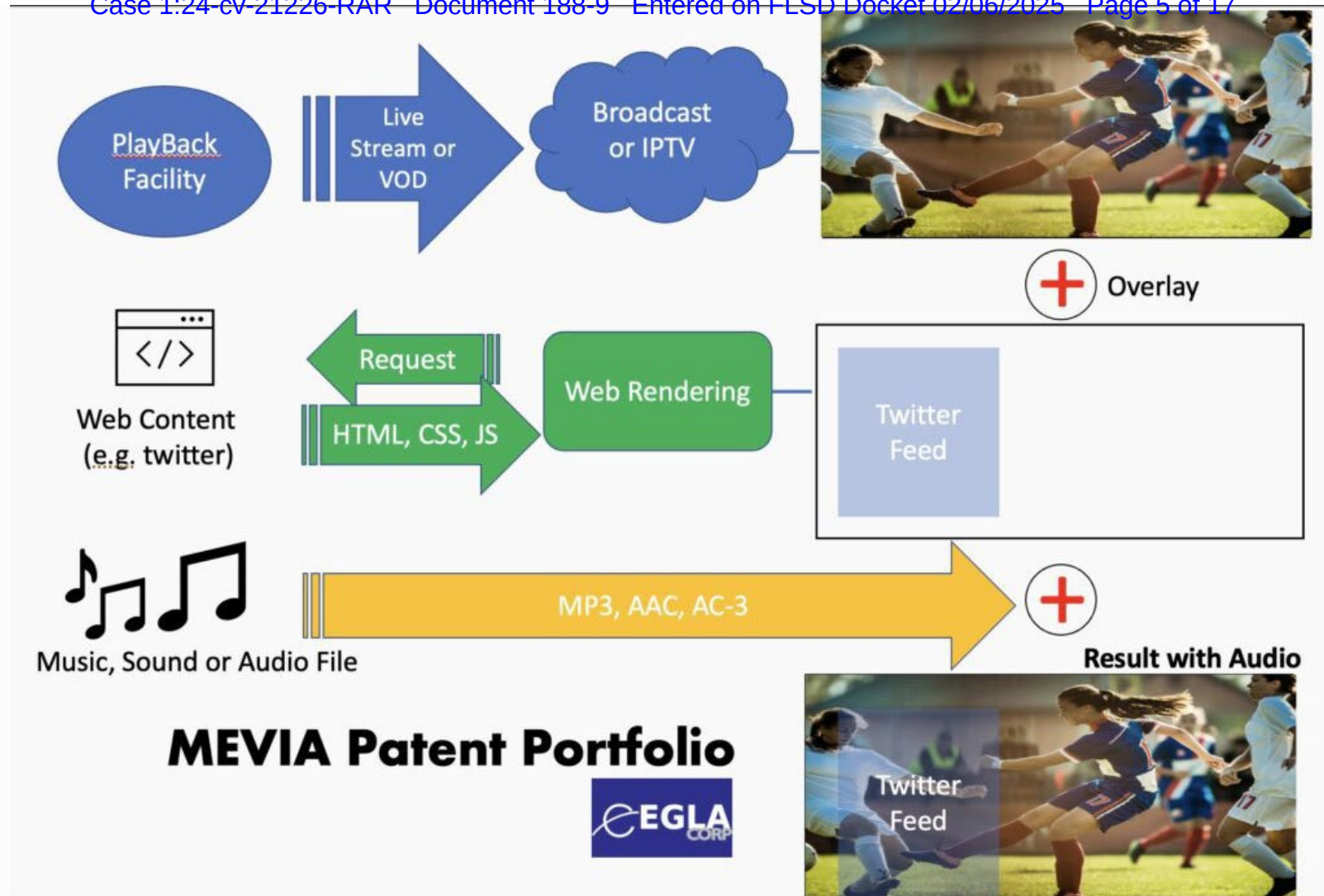
The previous technology used physical encoders and hard-coded graphics to one final video and many other encoders. MEVIA brings the solution to software, HTML web-pages, and interactive TV is modernized. EGLA CORP's prototypes and patents allow the integration of fully customizable "User Experiences" and even a customization in a per-user basis.

MEVIA Patents Portfolio : Graphical Representation

As shown below, a media provider usually owns a "play back" facility. This facility contains all videos that are generating a Linear TV stream which is being broadcasted to Cable TV, Satellite, or Over-the-Air. In my invention, the mediaplug can perform all these steps in a single box without requiring of a different facility. Hence, a video can be broadcasted to a DOCSIS 3.0 system directly from the mediaplug, as the mediaplug counts with a caching unit and multicasting units that encode and transcode all streams.

In the case of ATSC 3.0, this is done as an over the air transmission system. In, many TV broadcast medium, specially those over IP use a "multicast network." The mediaplug is then equipped to handle any multicast network. In this "multicast network," thousands if not more subscribers are connected and watching several streams.

The current MEVIA implementation uses software to connect to the web, retrieve HTML content and render that content to be overlaid on top of the originated stream from the broadcast. This widget is then ready to be presented to the user. At that point, once the overlay is on top of the video, = special music or audio streams available can be added for the user to have a richer experience. Those are audio assets are made available for the rendered video to create a resulting video that ultimately the user will see.



ATSC 3.0 and Streaming

Currently, the **ATSC 3.0 standard**, which uses an HTML-like video delivery platform, is exactly what the patents are designed to describe. There is 1 issued patent that reaffirms the ATSC 3.0 technology in Europe and that increases the value of all the patents and their legal standing. Besides ATSC 3.0, our patented technology has other uses in broadcasting for Smart TVs, IPTV, Cable TV systems, and even in-vehicle technologies.

Streaming for CarPlay and Android Auto

The main players for in-vehicle systems are Apple® with Carplay and Google's with Android Auto. These technologies are used in million of vehicles globally, and use for example navigation technologies that use extensive HTML and rendering engines. Also, information presented in a driver's console is derived from web widgets and assets

MEVIA OS - A Novel Distributed Operating System

MEVIA OS is new way to interact with User Interfaces and Smart TVs. In MEVIA OS, smart phones and tablets are used as controllers, and all Smart TVs with web-based interfaces are displays. In a way, an application has: a) A Controller, for example a keyboard, a game controller, or others, b) The Application selects the controller and is designed to work with the commands sends from a phone. c) the MEVIA OS provides storages, memory, data transports, for all the messages from the controllers to applications.

For more information you can review [here](#)

Method, system, and apparatus for multimedia
content delivery to cable TV and satellite
operators

US Patent 10,123,074

Systems, methods, and computer-readable media for delivering multimedia content
from the cloud to cable operators are disclosed. A device located at the cable

headend or implemented in the cloud can receive a request for at least one media stream for playback on a broadcast media channel. Content corresponding to a plurality of multimedia files in the media stream can be obtained from the internet or a cloud based service. The content can be used to generate the multimedia files in a format that is compatible with the cable operator. The multimedia files can be used to assemble the at least one media stream which can be provided to the cable operator for broadcast on the broadcast media channel.

[Read more](#)

METHOD, SYSTEM, AND APPARATUS FOR MULTIMEDIA CONTENT DELIVERY TO CABLE TV AND SATELLITE OPERATORS

US Pat 10,524,002

Continuation from the '074 Patent.

[Read more](#)

METHOD, SYSTEM, AND APPARATUS FOR MULTIMEDIA CONTENT DELIVERY TO CABLE TV AND SATELLITE OPERATORS

US Patent 11,140,441

Continuation of the '074 Patent

[Read more](#)

METHOD, SYSTEM, AND APPARATUS FOR

MULTIMEDIA CONTENT DELIVERY TO CABLE TV AND SATELLITE OPERATORS

US Patent 12,075,116

Continuation of the '074 Patent

[Read more](#)

METHOD, SYSTEM, AND APPARATUS FOR MULTIMEDIA CONTENT DELIVERY TO CABLE TV AND SATELLITE OPERATORS

European Patent EP3238457A1

Systems, methods, and computer-readable media for delivering multimedia content from the cloud to cable operators are disclosed. A device located at the cable headend or implemented in the cloud can receive a request for at least one media stream for playback on a broadcast media channel.

Content corresponding to a plurality of multimedia files in the media stream

can be obtained from the internet or a cloud based service. The content can be used to generate the multimedia files in a format that is compatible with the cable operator. The multimedia files can be used to assemble the at least one media stream which can be provided to the cable operator for broadcast on the broadcast media channel.

[Read more](#)

Continuation Patents

- US 18/800,685 - Patent Pending
- MEVIA OS Patent filed - The Next Generation Platform for MEVIA streaming platform [Here](#)

Attorneys: **Dickinson&Wright, Zeisler Law, Machat Law. Gray Robinson and Jeffer Mangels Butler & Mitchell.**

[Read more](#)

Licensing and Ownership

Dr. Edwin A. Hernandez is the owner of all IP related to MEVIA, and EGLA CORP is the exclusive licensee of his technologies.

Evidence of Use

Available under NDA, use cases in Web plus broadcasting.

[Read more](#)

Platform in Use

MEVIA uses these patents in their software and the platform The software platform is also available for licensing and is being in use at CABLE COLOR Honduras..

[Read more](#)

Power



July 1, 2023 ▶

June 19, 2023 ▶

October 13, 2021 ▶

June 10, 2021 ▶

EGLA maintains its leadership position in Cloud TV and Streaming

[Read more](#)

LSTM and Accelerometer

[Read more](#)

mevia.tv

[Read more](#)

New Issued Patent to Portfolio

[Read more](#)



Tags

4G 4G LTE 5g AI Artificial Intelligence
broadcasting cable tv cable tv music
cloud Cloud Computing cloudtocable
cloud to cable DVB
EGLA EGLA COMMUNICATIONS

Newsletter

Join Our
Newsletter

EGLA CORP wants to keep you
posted on new ventures, startups,

Categories

- 5G (7)
- AI (8)
- Android (1)

Newsletter

Join Our
Newsletter

EGLA CORP wants to keep you
posted on new ventures, startups,

.path{fill:none;stroke:#000;stroke-
miterlimit:10;stroke-width:1.5px;}

Enter your search

Tags

Eglavator Expert witness
handover innovation
intellectual property
internet TV IPR litigation machine
learning media Mediampify
Media Streaming Mevia
music choice music for cable
music platform music streaming
MVPD Neural Networks Patent
Cases patents PTAB
startups stingray digital
streaming
Technology trade secrets
uspto video platform wireless

Newsletter

Join Our
Newsletter

EGLA CORP wants to keep you
posted on new ventures, startups,
and intellectual property

Newsletter

Join our mailing list!

Enter your email

Submit

and intellectual property

Newsletter

Join our mailing list!

Enter your email

Submit

Menu

- Home
- IP & Products
 - Streaming
 - MEVIA Streaming Platform
 - MEVIA Music
 - Cloud to Cast – Demonstrations with Social Media and Web
 - mevia.tv
 - AI
 - Artificial

- Augmented Reality (3)
- Big Data (3)
- Business (2)
- Claim Construction (2)
- Cloud (26)
- cloud to cable (13)
- CloudTV (1)
- Crypto (2)
- Damages Expert (6)
- Damages Report (6)
- Daubert Challenge (4)
- Edwin (1)
- EGLA (14)
- EGLA MEDIA PLATFORM (3)
- EGLAVATOR (6)
- Entrepreneurship (13)
- Expert Witness (13)
- FinTech (1)
- Hacking (2)
- Hadoop (3)
- Health Technologies (1)
- Intellectual Property (23)

4G 4G LTE 5G AI Artificial Intelligence
broadcasting cable tv cable tv music
cloud Cloud Computing cloudtocable
cloud to cable DVB
EGLA EGLA COMMUNICATIONS
Eglavator Expert witness
handover innovation
intellectual property
internet TV IPR litigation machine
learning media Mediampify
Media Streaming Mevia
music choice music for cable
music platform music streaming
MVPD Neural Networks Patent
Cases patents PTAB
startups stingray digital
streaming
Technology trade secrets
uspto video platform wireless

Newsletter

Join our mailing list!

Enter your email

Submit

	Intelligence	■ Inventions (16)
	and	
	Machine	■ IOT (2)
	Learning	■ IPR (6)
	■ LSTM and Accelerometer	■ IPTV (7)
		■ J2ME (1)
	■ Using AI in Podcasts	■ Legal (2)
		■ Linux (3)
	■ MedTech	■ Litigation (10)
		■ LTE (2)
	■ Medical Systems	■ Machine Learning (6)
		■ Media Projects (9)
	■ Eco-Friendly Medical Waste disposal	■ Mediampify (11)
		■ Mevia (21)
	■ Wireless	■ Mobile Development (8)
		■ Mobile Device (7)
	■ Mobility Workx and Verizon Reach Agreement for Wireless Tech	■ Mobile World Congress (5)
		■ Mobility (5)
		■ Music for Cable TV (12)
		■ Neural Networks (2)
	■ Access Control	■ Open CV (1)
		■ Patents (21)
	■ Access Control and Nextgen Spaces	■ Products (2)
		■ PTAB (1)

EGLA CORP dba EGLA COMMUNICAITONS is a Florida-based tech company with a consulting division and the creator of Cloud to Cable, the platform that merges the worlds of cloud and Cable TV.

Our focus is to bring our media products to mobile, MSOs, and broadcast content in all forms from SD to 4K.

We are computer engineers and experts that can assist your company!

- Consulting
 - Expert Witness

- Code Reviews

Assisted Cases:
Expert Witness &
Damages Expert

Audit & Expert
Witness Services

WIFI Expert

Research Labs at
EGLA

Startups

Startups Assistance
– EGLAVATOR

Job Dashboard

Consultants &
Software Engineers

EGLAVATOR
Startups in Boca
Raton

EGLA CORP
CAPITAL

EGLAVATOR

Jobs

Jobs

Submit a Job

About Us

- What's ECLA
- CORP
- Our Team
- News and Updates
- Contact Us
- Partner Sites
- EGLACOMM
- CLOUD to CABLE
- Music for Cable
- MEDIAMPLIFY
- Dr. Edwin Hernandez



Exhibit I

Automated Accessibility Analysis of Dynamic Content Changes on Mobile Apps

Forough Mehralian
University of California, Irvine
fmehrali@uci.edu

Ziyao He
University of California, Irvine
ziyaoh5@uci.edu

Sam Malek
University of California, Irvine
malek@uci.edu

Abstract—With mobile apps playing an increasingly vital role in our daily lives, the importance of ensuring their accessibility for users with disabilities is also growing. Despite this, app developers often overlook the accessibility challenges encountered by users of assistive technologies, such as screen readers. Screen reader users typically navigate content sequentially, focusing on one element at a time, unaware of changes occurring elsewhere in the app. While dynamic changes to content displayed on an app’s user interface may be apparent to sighted users, they pose significant accessibility obstacles for screen reader users. Existing accessibility testing tools are unable to identify challenges faced by blind users resulting from dynamic content changes. In this work, we first conduct a formative user study on dynamic changes in Android apps and their accessibility barriers for screen reader users. We then present TIMESTUMP, an automated framework that leverages our findings in the formative study to detect accessibility issues regarding dynamic changes. Finally, we empirically evaluate TIMESTUMP on real-world apps to assess its effectiveness and efficiency in detecting such accessibility issues.

Index Terms—Android, Accessibility, Screen Reader, Dynamic Content Changes

I. INTRODUCTION

Dynamically changing visual content of screen (e.g., through animation) is a commonly used technique for enhancing the visual aesthetics of an app and to guide users’ attention to specific parts of the app. However, these visually appealing techniques should not come at the cost of making apps inaccessible. In adherence to legal frameworks [1], [2], established guidelines [3]–[5], and ethical principles, the digital realm should be inclusive and accessible to all. This is especially crucial for the approximately 15% of the global population with some form of disability, including more than 300 million users that are blind or visually impaired [6].

Visually impaired users rely on assistive technologies like screen readers to interact with mobile apps. These tools enable users to navigate to a specific element on the screen and listen to the content in focus. However, the tunnel-like focus provided by screen readers may lead to unawareness of dynamic changes occurring elsewhere on the screen. A known example of such dynamic content is error notifications. When an app assesses user inputs and provides feedback, such as an error message through a notification, these changes may go unnoticed by screen reader users. Mobile platforms

let developers designate these dynamically changing parts of a screen as “live regions”, assisting screen readers to detect and announce such changes to users. Unfortunately, developers often neglect using proper accessibility attributes, posing significant accessibility challenges for the blind.

Earlier studies and guidelines addressing software accessibility have only scratched the surface of this critical issue. The related accessibility guidelines on this matter primarily center on designating live regions for screen reader announcements. Specifically, in scenarios involving error messages, Web Content Accessibility Guidelines (WCAG) success criterion 3.3.1 emphasizes the crucial need for users to be informed about errors and comprehend what went wrong and recommends techniques such as annotating error notifications as live regions [7]. However, the challenge extends beyond these scenarios. Dynamic changes have been neglected from prior studies and tools that rely on screen captures from an app to detect accessibility issues [8]–[10]. GUI crawlers and app explorers typically capture screenshots of an app under test after it is in stable conditions by waiting for certain amount of time [11], [12]. Unfortunately, these approaches fail to capture app states during the entire rendering process, overlooking changes that occur over time on the screen. Consequently, they are not capable of detecting accessibility issues caused by dynamic contents.

To bridge this gap, we initiated a formative study aimed at identifying various types of dynamic changes and assessing their impact on screen reader users. This study revealed characteristics of accessibility issues related to dynamic content changes that negatively impact blind users. Building on these insights, we developed TIMESTUMP, an automated framework designed to detect such issues in Android apps. TIMESTUMP comprises an automated crawler, randomly exploring diverse app states and capturing data before, during, and after each action. Subsequently, this data undergoes processing using the identified patterns from our initial study to pinpoint problematic dynamic changes for screen reader users. The identified issues are then reported and visualized for developers.

This paper makes the following contributions:

- The first study on accessibility issues arising from dy-

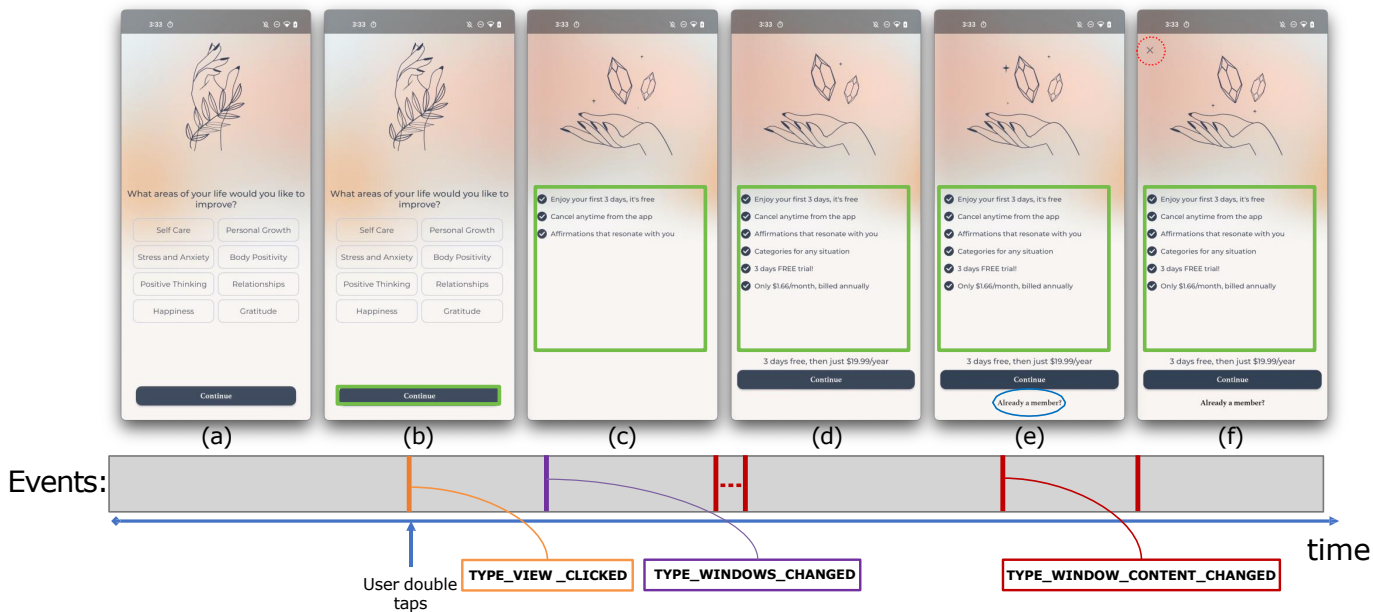


Fig. 1. Evolution of content loading on the screen across various states over time: (a) represents the initial screen state before the user initiates an action, (b) captures the moment when the user interacts with the app by clicking on a button, and (c) to (f) illustrate the gradual appearance of new screen content over time. Notably, in (f), the close button, indicated by a dashed red circle, appears above the accessibility focus. Since it is not tagged with `liveRegion` attribute, it is also not announced, and a screen reader user does not notice it.

dynamic content changes in Android apps.

- The introduction of the first automated crawler capable of capturing dynamic content changes, complemented by the creation of the initial dataset cataloging such behaviors.
- The development and public release of the first automated tool, named `TIMESTUMP`, designed for localizing and detecting accessibility issues related to dynamic content changes in Android apps [13].
- An empirical evaluation on real-world apps, corroborating the effectiveness of `TIMESTUMP` in detecting accessibility issues induced by dynamic screen changes.
- A user study involving blind participants to assess the impacts of dynamic screen change on app accessibility.

The remainder of this paper is organized as follows. Section II provides the background information. Section III describes our formative user study that motivated this work. Section IV presents `TIMESTUMP`, an automated approach for detection of problematic dynamic content changes. Section V details the evaluation of `TIMESTUMP` on real-world apps and in collaboration with blind participants. The paper concludes with a discussion of threats to validity, related research, and future work.

II. BACKGROUND

Mobile platforms offer the possibility of dynamic content changes, allowing developers to alter the screen content in real-time. Figure 1 displays an Android app called “I Am” [14] that provides daily affirmations for users and has more than

5 million downloads. When the screen reader focuses on the continue button as shown in Figure 1(b), the user can double-tap to perform the click gesture. Soon after clicking the button, the window changes, and some promotional content appears gradually, such as text, buttons, and other elements. For example, the “already a member” button, dotted in blue in Figure 1(e), and the close button, dashed red circle in Figure 1(f), appear after the bullet points are displayed.

This kind of screen rendering can pose severe challenges to screen reader users. Blind users utilize screen readers to interact with apps, and when encountering an unfamiliar app, they navigate through the on-screen elements sequentially to understand the app’s layout. The swipe right and left gestures allow the screen reader to move to the next or previous element, respectively, highlighting it with a green box as shown in Figure 1(b). When an element is focused, the screen reader vocalizes its textual description, enabling blind users to gauge its functionality in a manner analogous to how sighted users depend on the visual cues of an element. Should the textual description align with their expectations, blind users execute a double-tap, mirroring the single-tap action typical of sighted users. The following example illustrates the challenges blind users can face when dealing with dynamic changes. In Figure 1, as the user navigates to screen (c), the top element, which is the text view component, receives the accessibility focus. Screen reader users explore the screen by moving through the elements sequentially from top to bottom using a swipe-right gesture. However, the close button annotated in

dashed red is not recognizable as it appears on top of the screen and users are less likely to traverse backward, to the area they already visited. Such barriers can lead to unintentional interactions with ads or difficulties navigating away from them.

In Android, the guidelines suggest using an attribute called `liveRegion` to help screen readers recognize the appeared content. When an element is annotated as `liveRegion`, it is announced by the screen reader. Android system utilizes an event-based model to inform screen readers of changes in live regions. A GUI element emits an `AccessibilityEvent` when there are changes to its state, which is received by assistive technologies such as a screen reader. Figure 1 illustrates several different types of events that can be triggered during the loading of app content, with each color representing a distinct event. For example, `TYPE_VIEW_CLICKED` events occur after a view is clicked, `TYPE_WINDOWS_CHANGED` events happen when the app transitions to a different window, and `TYPE_WINDOW_CONTENT_CHANGED` events take place after the content inside a window changes.

Assistive technologies can also identify the element that is the source of events. In Android, GUI elements are represented by a tree of `AccessibilityNodeInfo` objects that mirror the XML hierarchy of elements and their attributes.

`AccessibilityNodeInfo` tree can be likened to the Document Object Model (DOM) tree in the case of web pages, offering a hierarchical representation of rendered elements on a web page. Prior studies on exploring various states of web apps for testing purposes have characterized dynamic content changes as modifications to the DOM that occur without reloading the page [15], [16]. These changes include updating or disappearing content [17], reordering elements [18], and inserting specific elements [19], as outlined in the WCAG guidelines. Similar to WCAG guidelines, Android suggests using accessibility attributes to notify screen reader users of such changes [20].

However, these guidelines only scratch the surface of the issues that may arise as a result of dynamic contents. For instance, when a temporary button, like an undo button, pops up on the bottom of the screen, users may struggle to locate it within the brief time frame of its visibility. This challenge intensifies when content disappears before users become aware of its existence. Merely relying on accessibility attributes does not fully resolve this issue.

III. FORMATIVE STUDY

We conducted a formative study to investigate the impact of various dynamic content changes on screen reader users.

A. Study Design

Prior studies and guidelines on Web defined dynamic content changes as modifications to the DOM that occur without reloading the page [15], [16]. Consequently, in Android apps, dynamic content changes encompasses any modifications to

the hierarchical representation of elements, i.e., a tree of `AccessibilityNodeInfos`, in a rendered window. These modifications include adding, removing, or changing attributes of elements. To have a better understanding of different types of dynamic content changes in Android, two authors conducted an empirical analysis of 50 Android apps. These apps were randomly chosen from the Google Play Store, representing various app categories. Additionally, to ensure the significance and popularity of the apps studied, each app selected had a minimum of 1 million downloads. For each app, two authors manually explored the app screen using a combination of actions, such as clicking, scrolling, and typing, to observe if that would trigger dynamic content changes. If so, a recording from the screen is taken with a brief description of the dynamic content change. Following this, two authors engaged in an iterative open-coding process to categorize the types of dynamic content changes identified.

Through our empirical analysis, we identified 5 types of dynamic content changes.

Appearing Content. This content change type is characterized by an element that initially is not present on a loaded window, but appears a few moments later and remains on the screen. For example, in Figure 1(f), the close button, marked by a red circle, appears after a few seconds.

Disappearing Content. This content change type describes an element that disappears either after a set period or as a result of user interaction. For example, in Figure 2(i), the navigation bar at the bottom including element **a** as well as the more button on top, element **b**, disappear when users navigate through the list of items.

Short-Lived Content. This content change type relates to an element that initially is not present on the screen but appears and remains on the screen only for a brief duration. Due to its transient nature, we refer to this as short-lived content. For example, as illustrated in Figure 2(iii), when users save a restaurant, a notification message annotated as **e** appears to notify them that they have successfully saved the store and to offer an option to view all saved stores. This message disappears after a few seconds.

Moving Content. This content change type refers to an element that is initially visible on the screen but subsequently gets relocated to a different part of the screen. For instance, in Figure 2(ii), the app-related information, marked as **c**, is shifted to the bottom of the screen and goes out of screen bounds after user presses the install button. Users must locate that information at a different position within the sequence of elements on the screen, as perceived by screen readers.

Content Modification. This content change type pertains to an element that remains on the screen but its attributes change. For example, in Figure 2(ii), the `TextView` annotated as **d**, continuously refreshes its text to display the progress of the app installation process.

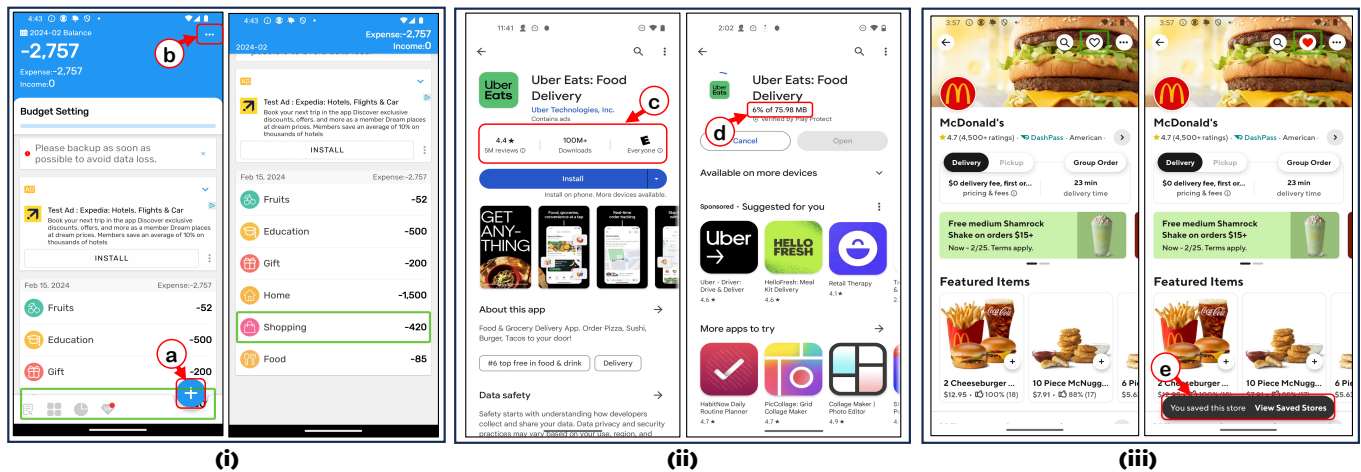


Fig. 2. Examples of dynamic content changes: (i) the add button (annotated as a) and the more button (annotated as b) disappear when users continue exploring the screen, (ii) the app information (marked as c) moves to the bottom of the screen after hitting the Install button; the text (annotated as d) constantly changes to indicate installation progress, (iii) the short-lived notification (annotated as e at the bottom) after saving a restaurant.

Having identified different types of dynamic content changes in Android, our objective was to understand their impacts on screen reader users. To this end, we selected 4 apps that collectively represented all identified types of dynamic content changes. We then designed 5 specific tasks that would involve interactions with these dynamic changes. Our objective was to understand whether the users can perform the tasks, whether they can perceive the dynamic changes in the apps, and to generally develop a better understanding of how the dynamic changes impact app accessibility. To recruit participants, we leveraged the Fable platform [21], which connects tech companies with disabled users for accessibility testing. Each user interview session was conducted over a one-hour period. During these sessions, we requested participants to share their phone screens and perform our designed tasks while vocalizing their thoughts and actions, aka think aloud [22]. This think-aloud method, combined with in situ questioning, enabled us to observe their understanding of the dynamic content changes and assess their ability to complete the tasks successfully. Our 3 blind participants included one female and two males, all of whom demonstrated a high proficiency in using the TalkBack.

B. Results

The user interviews focused on all the apps depicted in Figures 1 and 2. To generate a comprehensive list of accessibility issues related to dynamic content changes, two authors thoroughly examined each interview session. Initially, they independently identified areas where screen reader users faced confusion and attempted to ascertain the underlying reasons. Then, they engaged in discussions to reach a consensus. The following list outlines the dynamic content changes that proved challenging for screen reader users in our study.

Latent Appearing Content. When content appears without being annotated as a live region and is situated in an area previously explored by the user, it remains latent or unknown. For instance, the close button on an app's promotional page, as shown in Figure 1(f), emerges after a few seconds without alerting blind users. During the interview, blind users had already navigated past it, possibly interacting with elements located lower on the screen. This led to confusion when attempting to exit the promotional page, requiring users to employ various strategies such as using the TalkBack back gesture or re-navigating the screen.

Latent Disappearing Content. Content that vanishes before the user explores that region of the app stays undiscovered. In Figure 2(i), annotated buttons **b** and **a** disappear as users swipe through the list of items. The disappearance of the *add* button (element a) presented specific challenges for blind participants, as the button disappears in an unexplored area. As a result, they were unable to locate the element to add a new cost entry to the list. Conversely, the *more* button (element b) remains accessible, as users had already visited that element before its disappearance and when navigating backward, the more button becomes visible again.

Latent Short-Lived Content. When an element appears temporarily, it may be inaccessible to the user, especially if the element is actionable. The time it takes for the user to navigate to that element and perform the action might exceed the visibility period of short-lived content, leading to accessibility issues. In Figure 2(iii), we observe a brief notification annotated as **e** that emerges after users save a restaurant. Our user interviews revealed that participants were aware of this notification, understanding that they had successfully saved the restaurant and that the app offered an option to view all saved stores. However, this notification disappeared within a

few seconds. For this type of dynamic content, participants only partially grasped the situation. While they recognized the appearance of the notification, thanks to the live region annotation, they did not fully understand its transient nature and were unable to click on the *View Saved Stores* button. One interviewee expressed, “It was a flash or pop up, and it went away. I would expect to be able to navigate to that button, but when I move back and forth, it is gone.”

Latent Moving Content. When the location of a previously visited element changes, it can cause confusion for blind users. As illustrated in Figure 2(ii), the app-related information, including app rating and download number, relocates to the bottom of the screen after pressing the install button. In our study, blind users were assigned the task of installing an app and finding its download number. After installation, they navigated backward, relying on the previous announcement by TalkBack about the download number during their journey to the install button. However, to their surprise, upon navigating back, the information they sought was no longer present, resulting in confusion and a period of being stuck on the page. None of the participants completed the task, with one participant believing that he could eventually locate the download number by navigating further down, acknowledging it would take more time.

Latent Content Modification. Changes in attributes of elements that are noticeable by sighted users may go unnoticed by users relying on screen readers. During the formative study, the `TextView` (element **d**) in Figure 2(ii) continuously updated its text to reflect the progress of the app installation. The proper implementation of the `liveRegion` feature ensured that changes in the `TextView` content were effectively announced to blind participants, enabling them to accurately comprehend the status of the installation process. Conversely, this suggests that if the `liveRegion` feature is not correctly implemented, it becomes challenging for screen reader users to understand content modification, such as the installation status. In addition, changes in attributes such as size and color, which are not perceptible by screen readers, do not impact their perception of the app.

IV. APPROACH

Relying on the insights gained from formative interviews with screen reader users, we developed an automated framework called TIMESTUMP, designed to identify accessibility issues associated with dynamic content changes. Figure 3 provides an overview of TIMESTUMP, highlighting the three distinct phases of its operation. In the initial phase, we install an Android app on a Virtual Machine (VM) and utilize a GUI crawler to automatically explore the app, generating a diverse set of states in an app. The Snapshot Recorder tracks app states and records snapshots of distinct screens. In the second phase, we extract the list of actionable elements

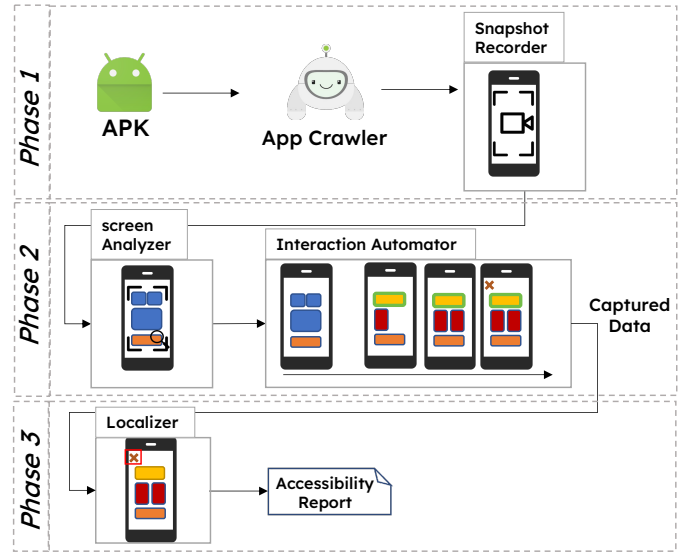


Fig. 3. TIMESTUMP's approach overview.

in each recorded snapshot. Then, the Interaction Automator systematically executes each action, capturing information before, during, and after the action. This rich dataset is passed to the third phase, where the Localizer component assesses the gathered information, precisely flagging accessibility issues stemming from dynamic content changes.

We now describe the details of each phase.

A. Phase 1: Capturing Unique App Screens

The main goal of this phase is to navigate through an app and explore its different states for subsequent examination. Interacting with the app leads to various state changes, ranging from subtle modifications in attributes, such as selecting a checkbox on the screen, to more significant changes like transitioning to an entirely new screen, resulting in a completely different hierarchical structure of elements. In this phase, our focus is on identifying a diverse set of screens from each app that have undergone significant changes. Detailed assessment of minor changes is reserved for subsequent phases.

To facilitate the testing of GUI apps, a variety of tools, such as STOAT [23], Monkey [24], Spaienze [25], and APE [26], have been specifically designed to traverse the expansive domain of an app's different states. TIMESTUMP seamlessly integrates with any existing app exploration tool, providing the flexibility to traverse various app states. Additionally, TIMESTUMP allows manual testers to explore the app with diverse scenarios in mind or leverage existing GUI test cases.

The Snapshot Recorder plays a pivotal role in tracking alterations. As the crawler interacts with the app, Snapshot Recorder keeps track of the screen structure and *Activities*, i.e., an Android component representing a single screen. It then captures VM snapshots from app states involving changes to activity names or the hash value of hierarchical structure of

elements. Similar to previous studies [11], the hash function excludes nodes that are not important for accessibility, i.e., those not notified by screen readers, as well as attributes such as `checked` or `enabled` that do not contribute to recognizing a different screen in an app. VM Snapshots enable us to load the app from the exact state and perform further analysis without concerns about the impacts of prior interactions.

B. Phase 2: Monitoring Apps in Action

During this phase, each potential interaction within a given app snapshot is automatically executed, all while monitoring the app for dynamic content changes.

To achieve this, we first load a VM snapshot. The Screen Analyzer employs an accessibility service to extract and dump the hierarchical structure of elements on the screen. The outcome is a tree structure wherein each node represents an element, accompanied by various attributes such as clickability. Parsing this node tree, the Screen Analyzer identifies all interactable elements and enumerates the types of actions they support, such as click, type, or swipe.

Every element is uniquely identified by its resource-id and a set of other attributes such as text, content description, and class name. The resource-id serves as a distinctive marker for locating each element. In instances where developers have not assigned a resource-id, the combination of other attributes can help in locating the element.

The Interaction Automator receives the comprehensive list of actions identified by the Screen Analyzer and executes them on the app while collecting certain data before, during, and after each action. The Interaction Automator consists of two main components: Controller and Accessibility Service.

The Controller functions as a server, sending commands to the client—the Accessibility Service, which operates in the background. The Accessibility Service is responsible for interacting with elements on the device. The client captures two frames of the app: *first frame* and *last frame*. The first frame corresponds to the initial state of the window, while the last frame corresponds to the app's state once all the content has finished rendering. The first frame primarily reflects the state of the app before any action takes place. However, if an action triggers a window change, this first frame then denotes the app's status immediately following the action. To accomplish this, TIMESTUMP tracks accessibility events that signal either the loading of a new window or shifts in accessibility focus. When a window change occurs, the first frame is recorded immediately after detecting the event that indicates a change in the window. For example, in Figure 1, the `TYPE_WINDOWS_CHANGED` event, highlighted in purple, signifies a window transition. Consequently, Figure 1(c) is identified as the first frame. In the absence of such events, the first frame defaults to the state of the app before executing the action.

For the last frame, TIMESTUMP listens for accessibility events indicating window content changes and, if none occur for more than 5 seconds, it captures that final state. For instance, Figure 1(f) is designated as the last frame, indicating that all changes on the app screen have been finalized. This practice is common in prior studies [11] and automation tools [27], ensuring app stability before data capture. The waiting time can be configured to be as long as necessary, or even adaptive to the specific app to optimize efficiency [12]. In instances of continual changes, such as animations or ads, a timeout period is implemented to bypass waiting.

The Controller stores these frames, as well as real-time logs of accessibility events generated by the Accessibility Service throughout the entire action execution period. Leveraging this extensive dataset empowers the Localizer to pinpoint accessibility issues arising from dynamic content changes.

C. Phase 3: Localizing Problematic Dynamic Changes

TIMESTUMP analyzes the collected data to identify various categories of latent content changes for screen reader users. This analysis is conducted across captured frames of the app, as well as the accessibility events captured during the execution of each action on the screen.

When a screen element undergoes a change, it triggers an event of type `Window_Content_Changed`. TIMESTUMP identifies sources of such events within the captured frames of the app, forming the initial set of candidate elements within a window that undergo a problematic change. The Localizer then compares elements in the final frame against those in the first frame to pinpoint the problematic changes.

As explained in Section IV-B, if the execution of an action results in a window transition, the first frame is the newly loaded window, i.e., the frame captured immediately after performing the action, similar to Figure 1(c). To detect window transitions, Localizer examines accessibility events of type `Windows_Changed` and `Window_State_Changed`, which are also used in the Android source code to detect the appearance of a new window [28]. Subsequently, we elaborate on the logic employed for detecting various types of problematic changes.

Due to space limits, we provide an intuitive explanation of how TIMESTUMP localizes each issue here, and provide the detailed algorithms on the companion website [13].

Latent Appearing Content: As explained in Section II and Section III, if certain content appears in previously explored areas (i.e., above the accessibility focus in default navigation order) and is not designated as a live region, it remains unknown to the screen reader user. The Localizer classifies elements in the final frame that trigger a content change event as latent appearing content if (1) they do not appear in the first frame, (2) are not designated as live regions, and (3) are positioned before the current accessibility focus.

Latent Disappearing Content: When an element disappears from the screen, a change event is triggered, similar to the case of appearing content. However, in this scenario, the event's source is the container of the vanishing element. For example, if a button within a linear layout disappears, the event source will be the linear layout, potentially covering the entire screen. Consequently, the Localizer evaluates all the children of an event publisher node in the first frame to verify their presence in the final frame. A child node is categorized as latent disappearing content if (1) it is not observed in the last frame, (2) it is not designated as a live region, and (3) it is positioned after the current accessibility focus.

Latent Short-lived Content: Elements that appear and disappear have a brief visibility period. Even if this content is announced, navigating to them and interacting with them using screen readers is challenging. The Localizer identifies these elements by searching for pairs of change events and localizing their sources denoted as $\langle S1, S2 \rangle$ in two consecutive frames, checking whether $S2$ is the container of $S1$. For any such found pair, an element $S1$ is categorized as latent short-lived content if (1) $S1$ is not present in the first frame, (2) its container $S2$ is observed in the second frame, and (3) $S1$ is not designated as a live region or is actionable.

Latent Moving Content: The Localizer examines elements displaying a shift in their position on the screen across different frames while maintaining consistent identifiers such as resource-id, and content description. Elements are flagged as problematic if their changed position is above the accessibility focus or if they move beyond the screen boundaries.

Latent Content Modification: The change of attributes in an element refers to any modification in the properties that define an element's behavior, or metadata within a UI. Localizer uses a hash function to detect such modifications across different frames. The hash function encodes the element attributes that are important in exploring the app with screen readers. A discrepancy in the hash values of an element between any two frames signifies a modification in the element's content. When the `liveRegion` attribute is absent, the change remains unknown to screen reader users.

V. EVALUATION

We evaluated TIMESTUMP on real-world apps and with the help of several blind users to answer the following questions:

- RQ1.** How accurate is TIMESTUMP in detecting dynamic content changes and different categories of accessibility issues?
- RQ2.** How do the issues reported by TIMESTUMP impact the screen reader users?
- RQ3.** What is the performance of TIMESTUMP?

A. RQ1. Accuracy of TIMESTUMP

1) *Experimental Setup:* For this experiment, we utilized STOAT [23] as the app exploration tool. We evaluated our

approach on 30 real-world Android apps. Our test set consists of two groups of apps: (group1) 10 apps with manually verified dynamic content changes from different categories of Google play store, (group2) 20 randomly selected apps with known accessibility issues from a prior study [11]. For apps in group1, the authors installed top rated apps in different categories of Google play store and manually explored each app, looking for dynamic content. They captured a VM snapshot of each app at that state, with the accessibility focus set on the target element such that performing the action on the target element results in the dynamic change. For apps in group2, we set the crawler to automatically get VM snapshots from two random unique states of the app. The tool then explores the actionable elements in each state to get the real-time data.

The precision of the captured data directly impacts the ability of TIMESTUMP to detect changes in dynamic content. Before evaluating TIMESTUMP's effectiveness, we manually reviewed the captured data from a test set to confirm alignment with our definitions of the first and last frames, and encountered no issues. For this experiment, we chose 15 unique app states from 5 random apps from a prior study [12], encompassing a range of app transitions such as implicit loading, explicit loading, and transitions. Additional information about this study can be found on the companion website [13].

All experiments were conducted on a typical computer setup for development (MacBook Pro, Apple M1 Max, 32 GB memory). We used the most recent distributed Android OS (SDK34), and the latest versions of Android screen reader.

2) *Results:* To answer this question, the authors manually examined the issues reported by TIMESTUMP and tagged them as False Positive (FP) if the reported issue is not correct, and True Positive (TP) if the reported issue correctly detects and categorizes problematic dynamic content changes. The authors used an emulator to load the captured snapshot and manually interacted with the app using TalkBack. This process allowed for the identification of legitimate dynamic elements in exploring the app with screen readers. We then report precision as the ratio of the number of TPs to the number of all detected issues. As shown in Table I, the overall precision over all the elements in 130 actions of apps is 0.94. To compute recall, we manually reviewed apps in both group 1 and group 2 to identify dynamic changes and establish the ground truth. In group 1, snapshots were captured during manual exploration of the app, revealing states with dynamic content changes. In contrast, for group 2 apps, we manually inspected automatically captured snapshots of random app states for dynamic content changes. Dynamic elements missed by TIMESTUMP were identified and manually labeled as False Negatives (FN). The overall recall across 130 actions is 0.92, as depicted in Table I.

Figure 4 presents examples of problematic dynamic elements detected by TIMESTUMP. In Figure 4(a), the four

elements highlighted by orange boxes emerge above the accessibility focus after tapping on the *plus* button. Figure 4(b) illustrates short-lived elements, including a button, indicated by blue boxes, appearing after adding a song to favorites. As all of those elements are actionable, TIMESTUMP reports them as problematic. In Figure 4(c), a `TextView`, annotated by the black box, is updated following the tap on the *CALCULATE* button. Since this element is not tagged as live region, it remained unannounced while exploring the app with TalkBack. Thus, TIMESTUMP reports it as problematic.

We analyzed the failures of TIMESTUMP and identified issues falling into two main themes, resulting in both false positives and false negatives.

The first pattern relates to inadequate identifiers assigned by developers to elements. For instance, in the Fuelio app, a `FrameLayout` serves as a container for its child elements, lacking essential identifiers such as `resource_id`, `text`, and `content_description`. As a result, its unique identification relies solely on its screen bounds. However, when an action triggers a layout modification, the element's screen bounds also change, leading TIMESTUMP to mistakenly interpret this as an appearing element, thus generating a false positive. Moreover, the absence of sufficient identifiers can result in false negatives. In the ESPN app, certain elements possess identifiers that are neither empty nor unique. TIMESTUMP primarily depends on these identifiers to match an element, but since they are not distinctive, TIMESTUMP fails to differentiate between elements on the screen, consequently failing to report associated issues.

Another category of failures occurs when the screen displays multiple windows, such as step-by-step guidelines overlaid on the main app. ADB allows us to capture the `AccessibilityNodeTree` of the foremost window only, thereby missing content in other windows. This limitation contributes to both false positives and false negatives.

B. RQ2. Qualitative Study

To assess the impact of the issues detected by TIMESTUMP on screen reader users, we conducted 15 user studies. We randomly selected three apps from RQ1, representing various types of dynamic content changes, corresponding to IDs *P8*, *G1*, and *G7*, as listed in Table I. Our qualitative study consisted of 10 self-guided tasks and 5 user interviews with blind testers, recruited through the Fable platform [21]. In the self-guided tasks, testers were given concise task descriptions to execute offline on apps while recording their screens and articulating their thoughts aloud. This approach, without a moderator present during sessions, helped mitigate interviewer bias. Additionally, user interviews were conducted to explore incidents in different states of one app, *G7*, and ask follow-up questions. Due to the limited size of the tester pool on the platform, some tasks involving different apps were assigned

to the same tester. However, no tester evaluated the same app multiple times. In total, 8 distinct testers participated in this study: 6 males, 2 females, with 7 identifying as White and 1 as Asian. Participant ages ranged from 20 to 45. Below, we first outline the issues that users confirmed. We then discuss the observed shortcomings and insights gained.

1) *User Confirmed Issues*: Of the 30 issues identified, users directly confirmed 25, yielding a confirmation rate of 0.83.

Appearing elements in explored areas. Figure 4(a) depicts an instance from this category. Blind testers were tasked with locating the *Gas* entry button, which appears after tapping the *Plus* button. The target button, along with three other elements highlighted with orange boxes on Figure 4(a), emerged in areas previously explored by users, without any notification. Consequently, users felt as if nothing had changed after tapping the *Plus* button. One participant expressed, *“It’s very confusing and disorienting when the screen changes without any audio feedback from the screen readers”* Another noted, *“Usually, new elements appear below [the current TalkBack focus], but in this case, they appeared above.”* The appearance of elements in previously explored areas caused confusion for screen reader users, resulting in longer times to locate the desired element. Two out of five interviewees were unable to find the targeted button and complete the task, while others had to explore the screen multiple times to do so. A similar issue in self-guided tasks resulted in confusion for all the participants. For elements that appear dynamically, blind testers recommended setting the `liveRegion` attribute appropriately. They suggested moving the TalkBack focus to the first new element on the screen in cases of significant window changes. For minor window changes, introduce dynamic elements in unexplored screen areas.

Disappearing elements in unexplored areas. In one of the test apps, activating a switch at the top caused some form entries to disappear. Testers interacting with the switch were not informed of the changes and were confused as to why they could not find certain elements. Four out of five interviewees were unable to complete the task, concluding that the required element was not present on the screen. Conversely, one interviewee managed to find the desired element by turning off a switch, leveraging his prior experience with such controls. Among the interviewees who failed to perform the task, one person remarked, *“[I] Thought the Recurrence section was either not on the screen or not visible to TalkBack. I just couldn’t find it.”* The tester expressed a preference for receiving a notification indicating that *“new controls are available or shown”* once the checkbox is ticked. This would enable them to recognize that the layout of the app has changed on the same screen and understand how to revert the layout to its original configuration.

Short-lived buttons. As depicted in Figure 4(b), when users

TABLE I
THE ACCURACY OF TIMESTAMP ON SUBJECT APPS.

ID	App	Category	#Installs	# Issues	TP	FP	FN	Precision	Recall
P1	Autozone	Auto & Vehicles	>5M	5	5	1	0	0.83	1
P2	Duolingo	Education	>500M	19	18	1	1	0.94	0.94
P3	Forest	Productivity	>10M	7	5	0	2	1	0.71
P4	Gratitude	LifeStyle	>1M	8	8	0	0	1	1
P5	Motivation	Health & Fitness	>5M	3	3	0	0	1	1
P6	Starbucks	Food & Drink	>10M	4	4	1	0	0.8	1
P7	TicketMaster	Events	>10M	1	1	0	0	1	1
P8	Spotify	Music & Audio	>1B	5	4	0	1	1	0.8
P9	H&M	Lifestyle	>50M	1	1	0	0	1	1
P10	File Manager	Tools	>1B	20	11	0	9	1	0.55
G1	Booking.com	Travel & Local	>500M	39	36	0	3	1	0.92
G2	Easy Bills Reminder	Finance	>100K	2	2	0	0	1	1
G3	Burn	Education	NA	6	2	0	4	1	0.66
G4	Dictionary.com	Books & Reference	>10M	4	4	0	0	1	1
G5	ESPN	Sports	>50M	3	2	0	1	1	0.66
G6	Calorie Counter by FatSecret	Health & Fitness	>50M	58	58	5	0	0.92	1
G7	Fuelio	Auto & Vehicle	>1M	101	92	5	9	0.94	0.91
G8	Life360	Lifestyle	>100M	4	4	0	0	1	1
G9	Master Lock Vault Enterprise	Lifestyle	>100K	4	1	0	3	1	0.25
G10	Nike	Shopping	>50M	25	24	0	1	1	0.96
G11	Weee! Asian Grocery Delivery	Food & Drink	>1M	8	8	0	0	1	1
G12	Norton Secure VPN	Tools	>10M	2	2	0	0	1	1
G13	TripIt	Travel & Local	>5M	7	7	1	0	0.87	1
G14	ToonMe photo cartoon maker	Photography	>50M	2	2	0	0	1	1
G15	Vimeo	Entertainment	>10M	6	6	0	0	1	1
G16	Yelp	Food & Drink	>50M	1	1	0	0	1	1
G17	The Clock	Productivity	>10M	56	55	11	1	0.83	0.98
G18	King James Bible	Books & Reference	>50M	23	21	2	2	0.91	0.91
G19	Lyft	Maps & Navigation	>50M	11	11	0	0	1	1
G20	To-Do List - Schedule Planner	Productivity	>10M	50	47	2	3	0.95	0.94
Overall				485	445	29	40	0.94	0.92

add a song to their favorites, a notification pops up and let them revert the action by tapping on the `Change` button. Three of the users became aware that they could potentially use this element. Two participants missed the button because TalkBack simultaneously announced three short-lived elements, which overwhelmed them. Moreover, during self-guided tasks, none of the participants could interact with the `Change` button as it disappeared quickly. As a result, three participants could not accomplish the task for removing a song from the favorites. Two participants were able to remove the song through an alternative method, using the ticked button, annotated by a green box in Figure 4(b). Therefore, short-lived elements should not overwhelm blind users with excessive information. Additionally, it is recommended to avoid including clickable elements in a short-lived manner, as blind users navigating sequentially with a screen reader are likely to miss them before they vanish.

Unannounced Short-lived Elements. In the Spotify app, when users removed a song from their favorites, a short-lived notification, with the text *Removed from Liked Songs*, appeared signaling this change, providing immediate feedback to sighted users. However, TalkBack did not announce the change to the screen reader user, leading to confusion.

Only two participants advanced to the step of removing a song from the favorites in our self-guided tasks. For those screen reader users, they were uncertain if the song had been successfully removed and felt compelled to navigate through the entire screen to verify that. When a song has not been added to favorites, the `Plus` button is labeled with the content description *Add Item*. Conversely, when the song is in the favorites, its description changes to *Item Added*. As a result, the blind users need to navigate the screen to check if the content description has reverted to *Add Item*. In order to confirm that their action was successful. Their experience suggested that the `liveRegion` attribute should be appropriately configured for short-lived notifications.

Unannounced Content Modification In Figure 4(c), tapping the `CALCULATE` button triggers an update of the result, indicated by the black box, appearing above the button. However, this change is not communicated to screen reader users, forcing them to navigate back to check the calculation result. Although all participants in our self-guided tasks managed to find the calculation result by navigating back and forth, they reported it as confusing and inconvenient. Additionally, if users press the `CALCULATE` button without providing any input for prior entries, they receive an error

message advising them to input values before proceeding. One participant noted, *"For the sake of consistency, having the calculation result announced just like the error message would not disappoint me."* Proper utilization of the `liveRegion` attribute would alleviate such issues.

2) *Observed Shortcomings*: The user study also shed light on `TIMESTUMP`'s shortcomings and enhancement opportunities.

Reverting Changes. `TIMESTUMP` evaluates the changes resulting from each action. However, a series of actions may have counteractive impacts. For instance, For example, in the Spotify app, top views move up as users navigate toward the bottom. As soon as they attempt to navigate back to those top elements, the views are restored to their original position, and users do not perceive any problem. Our manual exploration of our test set reveals 8 elements with similar issues that may not be problematic for users. However, for users who rely on alternative interaction modes, such as *explore by touch* where `TalkBack` shifts its focus to the coordinates of the touch gesture, these cases can still be confusing.

Severity of Issues. The dynamic elements identified as problematic exhibit varying degrees of severity and impact on blind users, with `TIMESTUMP` unable to prioritize them by severity. Factors such as the frequency of the issue among different apps and users' familiarity with it contribute to its severity. For instance, during interactions with the Booking.com app, switching tabs changes the content of the window without altering the screen reader focus or providing any notification. While this issue caused confusion for participants, they relied on their intuition and manually adjusted the `TalkBack` focus to the top of the screen to access the new content. However, all participants expressed that it would be helpful if the focus were automatically moved to the newly appeared content. Another factor influencing the severity of the issues is the distance of the changed element from the accessibility focus. In Figure 4 (c), navigating one element back and forth could help users find the results, while if the appearance of the result is far from the current focus, it may become impossible for users to locate it.

Navigation Order. `TIMESTUMP` relies on the default navigation order of elements for `TalkBack` to determine if a dynamic change is problematic. However, users may have their own interaction preferences when using screen readers. In our study, some users rely on their prior knowledge and tap on specific parts of the app to find the requested element. One interviewee mentioned that it would be helpful if the change was announced, but he could still locate the dynamically updated `TextView`. Additionally, although customization of the navigation order of elements was not observed in the apps used in our experiments, it is important to note that developers can override the default `TalkBack` navigation order, e.g., allowing the topmost element to be designated as

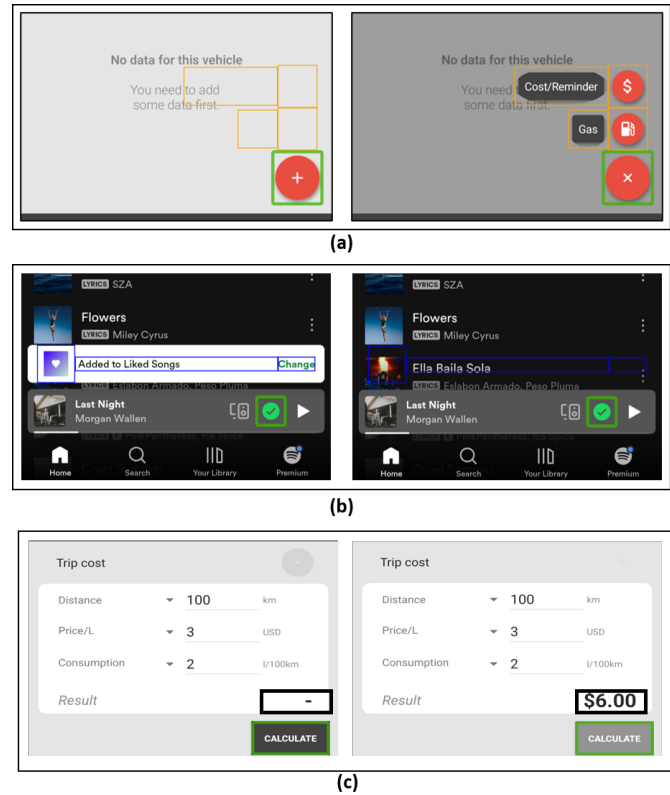


Fig. 4. Examples of detected issues by `TIMESTUMP`: (a) Appearing Content, (b) Short-Lived Content, and (c) Content Modification

the last element focused by `TalkBack` on a screen. If so, the button in Figure 1(e), circled in red, would not trouble blind users as it would be in an unexplored area.

C. RQ3. Performance

The performance evaluation of our tool, `TIMESTUMP`, is structured into three phases: app screen capturing (Phase 1), monitoring apps in action (Phase 2), and localizing problematic dynamic changes (Phase 3). In RQ1, Phase 1 involves automated capturing of two distinct app states from group2 apps, requiring an average of 167 seconds using `STOAT` [23]. Phase 2's analysis of each state for the number of actions is rapid; however, executing each action, capturing data, and transferring it to the server consumes about 66 seconds per action on average. The bulk of this time, approximately 20 seconds, is dedicated to dumping and transferring data, especially the captured video for each action. Developers can opt to disable video capturing in the tool, relying instead on screenshots, to significantly improve the tool's performance. In Phase 3, the post-analysis of the technique includes analysis of collected frames as well as the accessibility events, completing in approximately 7.5 seconds for each action.

VI. THREATS TO VALIDITY

External Validity. In this study, we examined dynamic content changes following action execution or screen transition. However, ad-related pop-ups or random rating requests may appear without user actions. Investigating these requires analyzing the source code and library calls to find them. Future research could focus on identifying and understanding these instances.

Another concern is the completeness of our work, both in terms of the types of dynamic content changes and the challenges they pose. We carefully selected and manually explored a diverse range of apps to identify the types of dynamic change, ensuring these aligned with web testing definitions related to element structure and attribute modifications. Moreover, we utilized interviews to pinpoint scenarios where different types of dynamic content change might pose issues for screen reader users. Although our initial findings were extracted from three interviews, the subsequent user study in RQ2 reaffirms the validity of our conclusions.

Similarly, a concern related to RQ1 is the completeness of the identified accessibility issues. The ground truth was manually created due to the lack of preexisting datasets. To validate the manual construction of the ground truth, two authors independently reviewed the snapshots, including the `AccessibilityNodeInfo` tree and `AccessibilityEvents`, to identify problematic dynamic changes. Subsequently, they engaged in discussions to ensure agreement in their evaluations.

Internal Validity. `TIMESTUMP` integrates various libraries and tools, including `Stoat`, `ADB`, `AVD`, and `AccessibilityService`, raising potential risks of defects. Additionally, there is a possibility of defects in our prototype's implementation. To counteract, we utilized the latest version of third-party tools, conducted Github code reviews, and tested on varied apps. We also assessed data capture accuracy on apps with different transitions, detailed on our website [13]. For rigorous testing, we used different sets of apps for our formative studies, accuracy evaluations, and tool assessments.

VII. RELATED WORKS

Automated accessibility testing includes various tools and studies for both web and mobile app accessibility analysis.

Web Pages: Web accessibility testing primarily relies on the WCAG guidelines [3]. These guidelines have led to the development of tools assessing web page accessibility compliance [29]–[34]. However, the guidelines overlook various accessibility challenges encountered by assistive technology users, especially in the context of dynamic changes. While a few criteria mandate developers to ensure dynamically displayed error/success messages are accessible to all, they fail to address other issues arising from dynamic content changes. Existing tools [29]–[34] cover only a fraction of the standards, thus inadequately detecting these issues on web pages.

To address limitations of guidelines, dynamic techniques have been proposed to assess apps while interacting with them. They resulted in studies that detect accessibility issues during interactions with web pages [35]–[37] or evaluate and infer correct accessibility attributes, like ARIA labels [38], for web content [39], [40]. In the evaluation of interaction issues, recent studies have attempted to utilize assistive technologies, similar to how an end user explores the app. They also account for changes introduced by JavaScript by evaluating multiple states that a single web page can take [41]–[47]. These studies focus on interaction failures which are only a subset of the challenges posed by dynamic changes. While exploring the app dynamically, they overlook real-time changes, like unannounced buttons appearing in previously explored areas. Furthermore, these studies miss changes like content modification that does not involve altering the DOM structure.

Mobile Apps: Similar to web accessibility testing, various automated tools are designed for mobile apps to assess specific app states and report their adherence to accessibility guidelines [48]–[57]. Recognizing the limitations of accessibility guidelines and the unique interaction modes of assistive technologies, recent studies have focused on identifying inaccessible content by utilizing assistive technologies to navigate various app states and comparing it with exploring the app without assistive technologies [11], [58]–[63]. However, no technique tackles the challenges of dynamic content changes.

VIII. CONCLUSION

The broad impacts of dynamic content changes on accessibility issues have not been thoroughly examined in prior research. We presented `TIMESTUMP`, an automated framework identifying accessibility issues due to dynamic screen changes. `TIMESTUMP` navigates through app states, collects data before, during and after each action, and applies a set of rules to detect dynamic screen changes that may lead to accessibility problems for the blind. An empirical study on real-world apps and a user study with blind participants prove its efficacy.

Future directions involve extending our work to ad-related pop-ups and unexpected rating requests that may appear without user actions, and expanding our implementation to other platforms, such as Web and iOS.

Our research artifacts are available publicly [13].

ACKNOWLEDGMENT

This work has been supported, in part, by award numbers 2211790, 1823262, and 2106306 from the National Science Foundation. We thank Fable for their collaboration in making this research possible. We are grateful for the detailed feedback from the anonymous reviewers of this paper, which helped improve this work.

REFERENCES

- [1] U. D. of Justice, “Americans with disabilities act,” <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX>
- [2] —, “A guide to disability rights laws,” <https://www.ada.gov/cguide.htm>, U.S. Department of Justice, 2020, Last Accessed: February 12, 2024.
- [3] W3C, “Wcag 2 overview,” <https://www.w3.org/WAI/standards-guidelines/wcag/>, W3C, 2023, last Accessed: December 5, 2023.
- [4] Apple, “Accessibility on ios,” <https://developer.apple.com/accessibility/ios/>, Apple, 2022, last Accessed: May 6, 2021.
- [5] Android, “Build more accessible apps,” <https://developer.android.com/guide/topics/ui/accessibility>, Google, 2022, last Accessed: May 6, 2022.
- [6] WHO, “World report on disability,” <https://www.who.int/news-room/fact-sheets/detail/disability-and-health>, WHO, 2023, Last Accessed: July 18, 2024.
- [7] WAI, “Understanding sc 3.3.1: Error identification (level a),” <https://www.w3.org/WAI/WCAG21/Understanding/error-identification.html#:~:text=Providing%20information%20about%20input%20errors,icons%20and%20other%20visual%20cues.>, W3C, 2024, last Accessed: March 5, 2024.
- [8] T. F. Liu, M. Craft, J. Situ, E. Yumer, R. Mech, and R. Kumar, “Learning design semantics for mobile apps,” in *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, 2018, pp. 569–579.
- [9] A. Mathur, G. Acar, M. J. Friedman, E. Lucherini, J. Mayer, M. Chetty, and A. Narayanan, “Dark patterns at scale: Findings from a crawl of 11k shopping websites,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, pp. 1–32, 2019.
- [10] B. Deka, Z. Huang, C. Franzen, J. Hibschan, D. Afegan, Y. Li, J. Nichols, and R. Kumar, “Rico: A mobile app dataset for building data-driven design applications,” in *Proceedings of the 30th annual ACM symposium on user interface software and technology*, 2017, pp. 845–854.
- [11] N. Salehnamadi, F. Mehralian, and S. Malek, “Groundhog: An automated accessibility crawler for mobile apps,” in *2022 37th IEEE/ACM International Conference on Automated Software Engineering*, IEEE, Rochester, Michigan, USA: ACM New York, NY, USA, 2022.
- [12] S. Feng, M. Xie, and C. Chen, “Efficiency matters: Speeding up automated testing with gui rendering inference,” in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 906–918.
- [13] F. Mehralian and Z. He, “Timestamp companion website,” <https://github.com/seal-hub/Timestamp>, SEAL, 2024, Last Accessed: Aug 16, 2024.
- [14] Monkey Taps LLC, “I am - Daily Affirmations,” https://play.google.com/store/apps/details?id=com.hrd.iam&hl=en_US&gl=US, 2024, accessed: 2024-02-12.
- [15] N. F. Malik, A. Nadeem, and M. A. Sindhu, “Achieving state space reduction in generated ajax web application state machine,” *Intelligent Automation & Soft Computing*, vol. 33, no. 1, 2022.
- [16] K. J. Koswara and Y. D. W. Asnar, “Improving vulnerability scanner performance in detecting ajax application vulnerabilities,” in *2019 International Conference on Data and Software Engineering (ICoDSE)*. IEEE, 2019, pp. 1–5.
- [17] W3C, “Understanding success criterion 2.2.2 — understanding wcag 2.0,” <https://www.w3.org/TR/UNDERSTANDING-WCAG20/time-limits-pause.html>, W3C, 2023, Last Accessed: December 7, 2023.
- [18] —, “Reordering page sections using the document object model,” <https://www.w3.org/WAI/WCAG22/Techniques/client-side-script/SCR27.html>, W3C, 2023, Last Accessed: December 7, 2023.
- [19] —, “Inserting dynamic content into the document object model immediately following its trigger element,” <https://www.w3.org/WAI/WCAG22/Techniques/client-side-script/SCR26.html/>, W3C, 2023, Last Accessed: December 7, 2023.
- [20] WAI, “Using aria role=alert or live regions to identify errors,” <https://www.w3.org/WAI/WCAG21/Techniques/aria/ARIA19>, W3C, 2024, last Accessed: March 5, 2024.
- [21] F. T. Labs, “Fable — digital accessibility, powered by people with disabilities,” <https://makeitfable.com/>, Fable Tech Labs, 2023, Last Accessed: December 7, 2023.
- [22] M. Van Someren, Y. F. Barnard, and J. Sandberg, “The think aloud method: a practical approach to modelling cognitive,” *London: Academic Press*, vol. 11, pp. 29–41, 1994.
- [23] T. Su, G. Meng, Y. Chen, K. Wu, W. Yang, Y. Yao, G. Pu, Y. Liu, and Z. Su, “Guided, stochastic model-based gui testing of android apps,” in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. Paderborn, Germany: ACM New York, NY, USA, 2017, pp. 245–256.
- [24] Google, “Ui/application exerciser monkey,” <https://developer.android.com/studio/test/monkey>, Google, 2022, Last Accessed: May 6, 2022.
- [25] K. Mao, M. Harman, and Y. Jia, “Sapienz: Multi-objective automated testing for android applications,” in *Proceedings of the 25th International Symposium on Software Testing and Analysis*. Saarbrücken, Germany: ACM New York, NY, USA, 2016, pp. 94–105.
- [26] T. Gu, C. Sun, X. Ma, C. Cao, C. Xu, Y. Yao, Q. Zhang, J. Lu, and Z. Su, “Practical gui testing of android applications via model abstraction and refinement,” in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, IEEE, Montreal, Canada: IEEE, 2019, pp. 269–280.
- [27] Google, “Dumpcommand,” <https://android.googlesource.com/platform/frameworks/testing/+jb-mr2-release/uiautomator/cmds/uiautomator/src/com/android/commands/uiautomator/DumpCommand.java#87>, Google, 2024, Last Accessed: Feb 18, 2024.
- [28] —, “clickandwaitfornewwindow,” <https://android.googlesource.com/platform/frameworks/base/+refs/heads/main/cmds/uiautomator/library/core-src/com/android/uiautomator/core/InteractionController.java#252>, Google, 2024, Last Accessed: Feb 18, 2024.
- [29] G. Broccia, M. Manca, F. Paternò, and F. Pulina, “Flexible automatic support for web accessibility validation,” *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. EICS, pp. 1–24, 2020.
- [30] G. Gay and C. Q. Li, “Achecker: open, interactive, customizable, web accessibility checking,” in *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A)*. Raleigh, USA: Association for Computing Machinery, 2010, pp. 1–2.
- [31] A. M. Agency, “Access monitor plus,” <https://accessmonitor.acessibilidade.gov.pt/>, Administrative Modernization Agency, 2021, Last Accessed: December 5, 2023.
- [32] C. Benavidez, “Examinator,” 2015. [Online]. Available: <http://examinator.net/>
- [33] WebAIM, “Wave web accessibility evaluation tool,” <https://wave.webaim.org/>, WebAIM, 2023, last Accessed: December 5, 2023.
- [34] accessiBe, “accessscan - website accessibility checker - free & instant - accessibe,” <https://accessibe.com/accessscan>, accessiBe, 2023, last Accessed: December 5, 2023.
- [35] H. Takagi, C. Asakawa, K. Fukuda, and J. Maeda, “Accessibility designer: visualizing usability for the blind,” *ACM SIGACCESS accessibility and computing*, no. 77-78, pp. 177–184, 2003.
- [36] J. P. Bigham, J. T. Brudvik, and B. Zhang, “Accessibility by demonstration: enabling end users to guide developers to web accessibility solutions,” in *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility*. Orlando, USA: Association for Computing Machinery, 2010, pp. 35–42.
- [37] F. Durgam, J. Grigera, and A. Garrido, “Dynamic detection of accessibility smells,” *Universal Access in the Information Society*, pp. 1–12, 2023.
- [38] W3C, “Accessible rich internet applications (wai-aria) 1.2,” World Wide Web Consortium (W3C), Tech. Rep., 2014. [Online]. Available: <https://www.w3.org/TR/wai-aria/>
- [39] C. Duarte, A. Salvado, M. E. Akpinar, Y. Yeşilada, and L. Carriço, “Automatic role detection of visual elements of web pages for automatic accessibility evaluation,” ser. W4A ’18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: <https://doi.org/10.1145/3192714.3196827>

- [40] M. Bajammal and A. Mesbah, "Semantic web accessibility testing via hierarchical visual analysis," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1610–1621.
- [41] W. M. Watanabe, R. P. Fortes, and A. L. Dias, "Acceptance tests for validating aria requirements in widgets," *Universal Access in the Information Society*, vol. 16, pp. 3–27, 2017.
- [42] N. Fernandes, D. Costa, S. Neves, C. Duarte, and L. Carriço, "Evaluating the accessibility of rich internet applications," in *Proceedings of the International Cross-Disciplinary Conference on Web Accessibility*, 2012, pp. 1–4.
- [43] N. Fernandes, D. Costa, C. Duarte, and L. Carriço, "Evaluating the accessibility of web applications," *Procedia Computer Science*, vol. 14, pp. 28–35, 2012.
- [44] L. Sensiate, H. Lidio Antonelli, W. Massami Watanabe, and R. Pontin de Mattos Fortes, "A mechanism for identifying dynamic components in rich internet applications," in *Proceedings of the 38th ACM International Conference on Design of Communication*, ser. SIGDOC '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: <https://doi.org/10.1145/3380851.3416785>
- [45] P. T. Chiou, A. S. Alotaibi, and W. G. J. Halfond, "Detecting and localizing keyboard accessibility failures in web applications," ser. ESEC/FSE 2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 855–867. [Online]. Available: <https://doi.org/10.1145/3468264.3468581>
- [46] P. T. Chiou, A. S. Alotaibi, and W. G. Halfond, "Bagel: An approach to automatically detect navigation-based web accessibility barriers for keyboard users," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–17.
- [47] —, "Detecting dialog-related keyboard navigation failures in web applications," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1368–1380.
- [48] Android, "Improve your code with lint checks," <https://developer.android.com/studio/write/lint?hl=en>, Google, 2023, last Accessed: December 7, 2023.
- [49] —, "Accessibility scanner - apps on google play," https://play.google.com/store/apps/details?id=com.google.android.apps.accessibility.auditor&hl=en_US, Google, 2023, last Accessed: December 7, 2023.
- [50] S. Hao, B. Liu, S. Nath, W. G. Halfond, and R. Govindan, "Puma: programmable ui-automation for large-scale dynamic analysis of mobile apps," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. Bretton Woods, New Hampshire, USA: ACM New York, NY, USA, 2014, pp. 204–217.
- [51] Android, "Espresso : Android developers," <https://developer.android.com/training/testing/espresso>, Google, 2023, last Accessed: December 7, 2023.
- [52] Robolectric, "Android unit testing framework," <https://github.com/robolectric/robolectric>, Robolectric, 2023, Last Accessed: December 7, 2023.
- [53] M. M. Eler, J. M. Rojas, Y. Ge, and G. Fraser, "Automated accessibility testing of mobile apps," in *2018 IEEE 11th International Conference on Software Testing, Verification and Validation*. Västerås, Sweden: ICST, 2018, pp. 116–126.
- [54] S. Chen, C. Chen, L. Fan, M. Fan, X. Zhan, and Y. Liu, "Accessible or not an empirical investigation of android app accessibility," *IEEE Transactions on Software Engineering*, vol. 48, pp. 3954–3968, 2021.
- [55] KIF, "Keep it functional - an ios functional testing framework," <https://github.com/kif-framework/KIF>, KIF, 2023, last Accessed: December 7, 2023.
- [56] H. N. da Silva, S. R. Vergilio, and A. T. Endo, "Accessibility mutation testing of android applications," *Journal of Software Engineering Research and Development*, vol. 10, pp. 8–1, 2022.
- [57] L. Li, R. Wang, X. Zhan, Y. Wang, C. Gao, S. Wang, and Y. Liu, "What you see is what you get? it is not the case! detecting misleading icons for mobile applications," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 538–550.
- [58] M. Taeb, A. Swearngin, E. School, R. Cheng, Y. Jiang, and J. Nichols, "Axnav: Replaying accessibility tests from natural language," *arXiv preprint arXiv:2310.02424*, 2023.
- [59] F. Mehralian, N. Salehnamadi, S. F. Huq, and S. Malek, "Too much accessibility is harmful! automated detection and analysis of overly accessible elements in mobile apps," in *2022 37th IEEE/ACM International Conference on Automated Software Engineering*, IEEE. Rochester, Michigan, USA: ACM New York, NY, USA, 2022.
- [60] N. Salehnamadi, A. Alshayban, J.-W. Lin, I. Ahmed, S. Branham, and S. Malek, "Latte: Use-case and assistive-service driven automated accessibility testing framework for android," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. Virtual, Okohama, Japan: ACM New York, NY, USA, 2021, pp. 1–11.
- [61] A. S. Alotaibi, P. T. Chiou, and W. G. Halfond, "Automated detection of talkback interactive accessibility failures in android applications," in *2022 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2022, pp. 232–243.
- [62] A. Alshayban and S. Malek, "Accessitext: automated detection of text accessibility issues in android apps," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 984–995.
- [63] N. Salehnamadi, Z. He, and S. Malek, "Assistive-technology aided manual accessibility testing in mobile apps, powered by record-and-replay," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–20.

Exhibit J

Data Sheet: H-AV-MPEG2-E

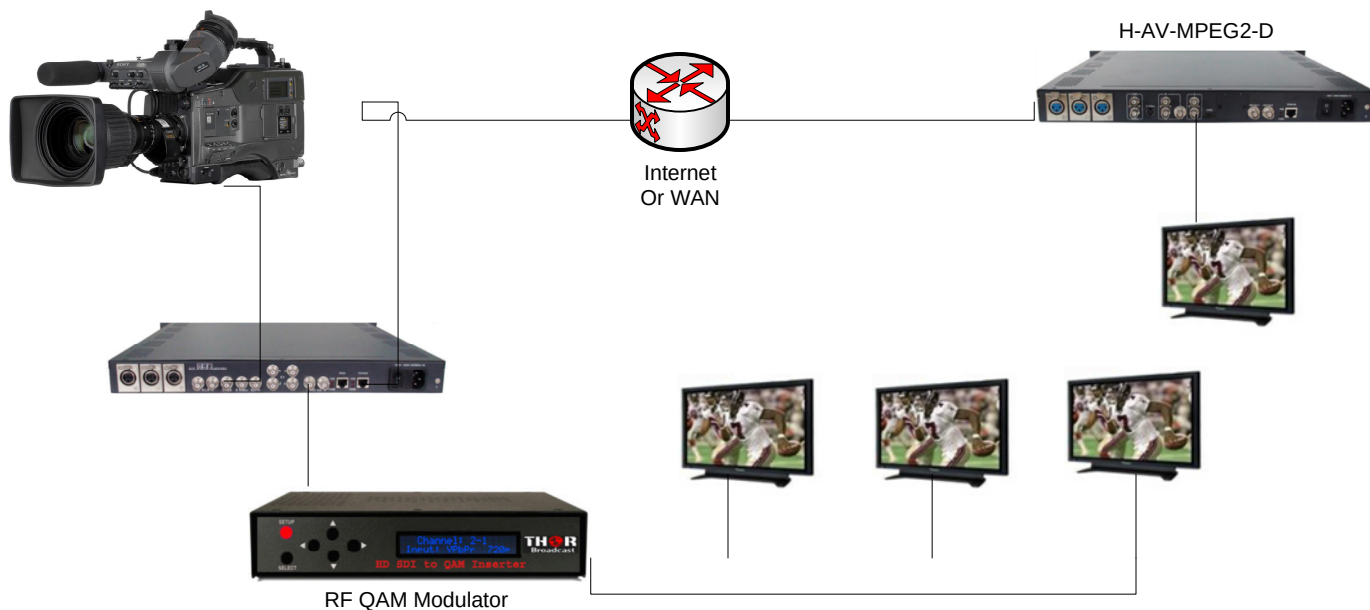


The H-AV-MPEG2-E Encoder is an SD-SDI or Baseband Video/Audio **MPEG-2 encoder** with either IP or ASI output.

This unit is popular in both the digital video broadcast industry and in CCTV security systems. The unit features a high quality output while maintaining a reasonable bit rate of 1.5 to 15 Mbps. The H-AV-MPEG2-E is a **complete hardware encoding solution** with time based correction circuitry that adapts to signal sources that are low in quality. Featuring remote management via SNMP as well as remote network control. The H-AV-MPEG2-E is an excellent choice for all your SD-SDI or baseband video/audio encoding needs.



Typical H-AV-MPEG2-E Application



Technical Specifications

Standards: NTSC, PAL, SECAM
 Input: SD-SDI, Baseband A/V
 Output: DVB-ASI EN 50083-9 ASI
 MPEG2 over IP TS
 Bit Rate: 1.5 – 15 Mbps
 Format: 188 bytes
 IP Protocol: TCP/IP
 Control/IP Output: 10/100 RJ45 UDP/RTP/IP
 Remote Control: Web, SNMP, and Software
 Encoding Feature: ISO 1172(MPEG1) ISO 13818
 Encoding Support: MPEG-2 4:2:0MP@ML
 Sampling Rates: 32KHz, 44.1KHz, 48KHz

PAL 720×576, 704×576, 640×576, 544×576,
 480×576, 384×576, 352×576
NTSC 720×480, 704×480, 640×480, 544×480,
 480×480, 384×480, 352×480
 Operating Temp: 0 – 50 (°C)
 Storage Temp: -20 – 70 (°C)
 Operating Humidity: 0 – 90 (%)
 Storage Humidity: 0 – 90 (%)
 Power Requirements: 110/240V Auto Sense
 Dimensions: 480mmX225mmX44mm

Exhibit K



The AMERICAN HERITAGE[®] dictionary of the English Language

Search



HOW TO USE THE DICTIONARY

To look up an entry in *The American Heritage Dictionary of the English Language*, use the search window above. For best results, after typing in the word, click on the "Search" button instead of using the "enter" key.

Some compound words (like *bus rapid transit*, *dog whistle*, or *identity theft*) don't appear on the drop-down list when you type them in the search bar. For best results with compound words, place a quotation mark before the compound word in the search window.

[GUIDE TO THE DICTIONARY](#)



THE USAGE PANEL

The Usage Panel is a group of nearly 200 prominent scholars, creative writers, journalists, diplomats, and others in occupations requiring mastery of language. Annual surveys have gauged the acceptability of particular usages and grammatical constructions.

[THE PANELISTS](#)

se·quence (sē kwens, -kwěns')

Share: [Tweet](#)

n.

1. A following of one thing after another; succession.
2. An order of succession; an arrangement.
3. A related or continuous series. See [Synonyms at series](#).
4. Games Three or more playing cards in consecutive order and usually the same suit; a run.
5. A series of related shots that constitute a complete unit of action in a movie.
6. Music A melodic or harmonic pattern successively repeated at different pitches with or without a key change.
7. Roman Catholic Church A hymn sung between the gradual and the Gospel.
8. Mathematics An ordered set of quantities, as x , $2x^2$, $3x^3$, $4x^4$.
9. Biochemistry The order of constituents in a polymer, especially the order of nucleotides in a nucleic acid or of the amino acids in a protein.

tr.v. **se·quenced**, **se·quenc·ing**, **se·quenc·es**

1. To organize or arrange in a sequence.
2. To determine the order of constituents in (a polymer, such as a nucleic acid or protein molecule).

[Middle English, a type of hymn, from Old French, from Medieval Latin *sequentia*, hymn, that which follows (from its following the alleluia), from Late Latin, from Latin *sequēns*, *sequent-*, present participle of *sequi*, to follow; see [sek^w](#)-¹ in the Appendix of Indo-European roots.]

The American Heritage[®] Dictionary of the English Language, Fifth Edition copyright ©2022 by HarperCollins Publishers. All rights reserved.

Indo-European & Semitic Roots Appendices

Thousands of entries in the dictionary include etymologies that trace their origins back to reconstructed proto-languages. You can obtain more information about these forms in our online appendices:

[Indo-European Roots](#)

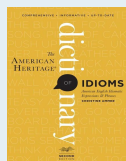
[Semitic Roots](#)

The Indo-European appendix covers nearly half of the Indo-European roots that have left their mark on English words. A more complete treatment of Indo-European roots and the English words derived from them is available in our [Dictionary of Indo-European Roots](#).

American Heritage Dictionary Products



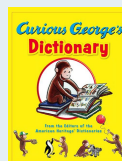
The American Heritage Dictionary, 5th Edition



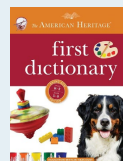
The American Heritage Dictionary of Idioms



The American Heritage Desk Dictionary and Thesaurus



Curious George's Dictionary



The American Heritage Children's Dictionary



AMERICAN HERITAGE DICTIONARY APP

The new American Heritage Dictionary app is now available for [iOS](#) and [Android](#).



THE AMERICAN HERITAGE DICTIONARY BLOG

The articles in our [blog](#) examine new words, revised definitions, interesting images from the fifth edition, discussions of usage, and more.

THE 100 WORDS[®]

See word lists from the best-selling 100 Words Series!

[FIND OUT MORE!](#)



INTERESTED IN DICTIONARIES?

Check out the Dictionary Society of North America at <http://www.dictionariesociety.com>

CONTACT US

Customer Service

Make Me An Author

Ebooks Help with Glose Reader

ABOUT US

Company Profile

Leadership Team

Corporate Social Responsibility

HarperCollins Careers

HarperCollins Imprints

HarperGreen

Social Media Directory

Accessibility

FOR READERS

Browse Reading Guides

FOR AUTHORS

Submit a Manuscript

Report Piracy

Agent Portal

MEDIA

Publicity Contacts

Press Room

SERVICES

HarperCollins Speakers Bureau

Library Services

Academic Services

Desk & Exam Copies

Review Copies

OpenBook API

Marketing Partnerships

COVID-19 RESOURCES & PERMISSIONS

Permissions for Adult Online Readings

Permissions for Kids Online Readings

SALES & RIGHTS

Booksellers & Retailer Ordering

HarperCollins Catalogs

Permissions

Subsidiary Rights

Media Rights and Content Development

GLOSE APP

iPhone

Android

GLOBAL DIVISIONS

HarperCollins US

HarperCollins Canada

HarperCollins Christian

HarperCollins Australia

HarperCollins India

HarperCollins UK



News Corp

Terms of Use • Terms of Sale • Your Ad Choices • Privacy Policy • California Privacy Policy

Do Not Sell My Personal Information

Copyright 2022 HarperCollins Publishers All rights reserved.

*This website is best viewed in Chrome, Firefox, Microsoft Edge, or Safari. Some characters in pronunciations and etymologies cannot be displayed properly in Internet Explorer.

https://ahdictionary.com/word/search.html?q=sequence[2/6/2025 11:17:19 AM]



The AMERICAN HERITAGE® dictionary of the English Language

Search



HOW TO USE THE DICTIONARY

To look up an entry in *The American Heritage Dictionary of the English Language*, use the search window above. For best results, after typing in the word, click on the "Search" button instead of using the "enter" key.

Some compound words (like *bus rapid transit*, *dog whistle*, or *identity theft*) don't appear on the drop-down list when you type them in the search bar. For best results with compound words, place a quotation mark before the compound word in the search window.

[GUIDE TO THE DICTIONARY](#)



THE USAGE PANEL

The Usage Panel is a group of nearly 200 prominent scholars, creative writers, journalists, diplomats, and others in occupations requiring mastery of language. Annual surveys have gauged the acceptability of particular usages and grammatical constructions.

[THE PANELISTS](#)

se·ries (sîr ēz)

Share: [Tweet](#)

n. pl. **series**

1. A number of objects or events arranged or coming one after the other in succession.
2. A set of stamps, coins, or currency issued in a particular period.
3. Physics & Chemistry A group of objects related by linearly varying successive differences in form or configuration: a radioactive decay series; the paraffin alkane series.
4. Mathematics The sum of a sequentially ordered finite or infinite set of terms.
5. Geology A group of rock formations closely related in time of origin and distinct as a group from other formations.
6. Grammar A succession of coordinate elements in a sentence.
7.
 - a. A succession of publications that present an extended narrative, such as a comic book series, or that have similar subjects or similar formats, such as a series of cookbooks.
 - b. A succession of individual programs presented as parts of a unified whole, such as the set of episodes of a television show or a podcast.
8.
 - a. Sports A number of games played by the same two teams, often in succession.
 - b. Baseball The World Series.
9. Linguistics A set of vowels or diphthongs related by ablaut, as in sing, sang, sung, and song.

Idiom:

in series

In an arrangement that forms a series.

[Latin *seriēs*, from *serere*, to join; see **ser-**² in the Appendix of Indo-European roots.]

Synonyms: series, chain, progression, sequence, string, succession

These nouns denote a number of things placed or occurring one after the other: a series of days, a series of facts; a chain of command, a chain of proof; a progression of courses toward a degree, a progression of prime numbers; a sequence of a chemical reactions, the sequence of events leading to the accident; a string of islands, a string of questions; a succession of failures, a succession of actors auditioning for the play.

Usage Note: *Series* is both a singular and a plural form. When it has the singular sense of "one set," it takes a singular verb, even when *series* is followed by *of* and a plural noun: *A series of lectures is scheduled*. When it has the plural sense of "two or more sets," it takes a plural verb: *Two series of lectures are scheduled: one for experts and one for laypeople*.

The American Heritage® Dictionary of the English Language, Fifth Edition copyright ©2022 by HarperCollins Publishers. All rights reserved.

Indo-European & Semitic Roots Appendices

Thousands of entries in the dictionary include etymologies that trace their origins back to reconstructed proto-languages. You can obtain more information about these forms in our online appendices:

[Indo-European Roots](#)



AMERICAN HERITAGE DICTIONARY APP

The new American Heritage Dictionary app is now available for [iOS](#) and [Android](#).



THE AMERICAN HERITAGE DICTIONARY BLOG

The articles in our [blog](#) examine new words, revised definitions, interesting images from the fifth edition, discussions of usage, and more.

THE 100 WORDS®

See word lists from the best-selling 100 Words Series!

[FIND OUT MORE!](#)



INTERESTED IN DICTIONARIES?

Check out the Dictionary Society of North America at <http://www.dictionariesociety.com>

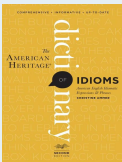
Semitic Roots

The Indo-European appendix covers nearly half of the Indo-European roots that have left their mark on English words. A more complete treatment of Indo-European roots and the English words derived from them is available in our **Dictionary of Indo-European Roots**.

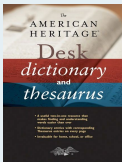
American Heritage Dictionary Products



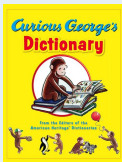
The American Heritage Dictionary, 5th Edition



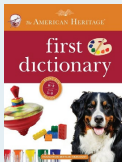
The American Heritage Dictionary of Idioms



The American Heritage Roget's Thesaurus



Curious George's Dictionary



The American Heritage Children's Dictionary

CONTACT US

- Customer Service
- Make Me An Author
- Ebooks Help with Glose Reader

ABOUT US

- Company Profile
- Leadership Team
- Corporate Social Responsibility
- HarperCollins Careers
- HarperCollins Imprints
- HarperGreen
- Social Media Directory
- Accessibility

FOR READERS

- Browse Reading Guides

FOR AUTHORS

- Submit a Manuscript
- Report Piracy
- Agent Portal

MEDIA

- Publicity Contacts
- Press Room

SERVICES

- HarperCollins Speakers Bureau
- Library Services
- Academic Services
- Desk & Exam Copies
- Review Copies
- OpenBook API
- Marketing Partnerships

COVID-19 RESOURCES & PERMISSIONS

- Permissions for Adult Online Readings
- Permissions for Kids Online Readings

SALES & RIGHTS

- Booksellers & Retailer Ordering
- HarperCollins Catalogs
- Permissions
- Subsidiary Rights
- Media Rights and Content Development

GLOSE APP

- iPhone
- Android

GLOBAL DIVISIONS

- HarperCollins US
- HarperCollins Canada
- HarperCollins Christian
- HarperCollins Australia
- HarperCollins India
- HarperCollins UK



Terms of Use • Terms of Sale • Your Ad Choices • Privacy Policy • California Privacy Policy
Do Not Sell My Personal Information

Copyright 2022 HarperCollins Publishers All rights reserved.

*This website is best viewed in Chrome, Firefox, Microsoft Edge, or Safari. Some characters in pronunciations and etymologies cannot be displayed properly in Internet Explorer.

Exhibit L

Microsoft Press

Computer Dictionary

Third Edition

Microsoft Press

PUBLISHED BY

Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Copyright © 1997 by Microsoft Corporation

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Library of Congress Cataloging-in-Publication Data pending.

ISBN 1-57231-743-4

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9 QMQM 2 1 0 9 8 7

Distributed to the book trade in Canada by Macmillan of Canada, a division of Canada Publishing Corporation.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office. Or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at mspress.microsoft.com.

Macintosh, Power Macintosh, QuickTime, and TrueType fonts are registered trademarks of Apple Computer, Inc. Intel is a registered trademark of Intel Corporation. DirectInput, DirectX, Microsoft, Microsoft Press, MS-DOS, Visual Basic, Visual C++, Win32, Win32s, Windows, Windows NT, and XENIX are registered trademarks and ActiveMovie, ActiveX, and Visual J++ are trademarks of Microsoft Corporation. Java is a trademark of Sun Microsystems, Inc. Other product and company names mentioned herein may be the trademarks of their respective owners.

Acquisitions Editor: Kim Fryer

Project Editors: Maureen Williams Zimmerman, Anne Taussig

Technical Editors: Dail Magee Jr., Gary Nelson, Jean Ross, Jim Fuchs, John Conrow,
Kurt Meyer, Robert Lyon, Roslyn Lutsch

.wav \dot W A-V\ *n.* The file extension that identifies sound files stored in waveform (WAV) audio format. *See also* WAV.

WAV \wāv, W A-V\ *n.* A file format in which Windows stores sounds as waveforms. Such files have the extension .wav. Depending on the sampling frequency, on whether the sound is monaural or stereo, and on whether 8 or 16 bits are used for each sample, one minute of sound can occupy as little as 644 kilobytes or as much as 27 megabytes of storage. *See also* sampling (definition 2), waveform.

wave \wāv\ *n.* **1.** Any disturbance or change that has an oscillatory, periodic nature, for example, a light or sound wave. *See also* waveform. **2.** In electronics, the time-amplitude profile of an electrical signal.

waveform \wāv'fɔrm\ *n.* The manner in which a wave's amplitude changes over time. *See also* period, phase, wavelength.

wavelength \wāv'lenkth, wāv'length\ *n.* The distance between successive peaks or troughs in a periodic signal that is propagated through space. Wavelength is symbolized by the Greek letter lambda (λ) and can be calculated as speed divided by frequency.

wavelet \wāv'lət\ *n.* A mathematical function that varies over a limited extent of time. Wavelets are coming into increasing use for analyzing signals (such as sound). They have limited duration and sudden changes in frequency and amplitude rather than the infinite duration and constant amplitude and frequency of the sine and cosine functions. *Compare* Fourier transform.

WBEM \W B-E-M\ *n.* Acronym for **Web-Based Enterprise Management**. A protocol that links a Web browser directly to a device or application that monitors a network. *See also* communications protocol.

WDEF \W D-E-F\ *n.* *See* window definition function.

WDL \W D-L\ *n.* *See* Windows Driver Library.

weak typing \wēk tī'pēng\ *n.* A characteristic of a programming language that allows the program to change the data type of a variable during program execution. *See also* data type, variable. *Compare* strong typing.

web \web\ *n.* A set of interlinked documents in a document system, the user interface to the web, through a home page. *See also* World Wide Web.

Web \web\ *n.* *See* World Wide Web.

Web address \web a'dres, ə-dres\ *n.* *See* URL.

Web-Based Enterprise Management \web'bāsd en'tər-prīz man'əj-mənt\ *n.* *See* WBEM.

Web browser \web' brou'zər\ *n.* A client application that enables a user to view HTML documents on the World Wide Web, another network, or the user's computer; follow the hyperlinks among them; and transfer files. Text-based Web browsers, such as Lynx, can serve users with shell accounts but show only the text elements of an HTML document; most Web browsers, however, require a connection that can handle IP packets but will also display graphics that are in the document, play audio and video files, and execute small programs, such as Java applets or ActiveX controls, that can be embedded in HTML documents. Some Web browsers require helper applications or plug-ins to accomplish one or more of these tasks. In addition, most current Web browsers permit users to send and receive e-mail and to read and respond to newsgroups. *Also called* browser. *See also* ActiveX controls, helper application, hyperlink, Internet Explorer, Java applet, Lynx, Mosaic, Netscape Navigator, plug-in.

WebCrawler \web'krä'lər, krō'lər\ *n.* A World Wide Web search engine operated by America Online. *See also* search engine.

Web development \web de-vel'əp-mənt\ *n.* The design and coding of World Wide Web pages.

Web directory \web dər-ek'tər-ē\ *n.* A list of Web sites, giving the URL and a description of each. *See also* URL.

Web index \web in'deks\ *n.* A Web site intended to enable a user to locate other resources on the Web. The Web index may include a search facility or may merely contain individual hyperlinks to the resources indexed.

Webmaster or **webmaster** \web'ma'stər\ *n.* A person responsible for creating and maintaining a World Wide Web site. A Webmaster is often responsible for responding to e-mail, ensuring the site is operating properly, creating and updating Web pages, and maintaining the overall structure

webmistress

and design of the site. *Also called* webmistress, webweaver.

webmistress \web'mis-trəs\ *n.* See webmaster.

Web page \web'pāj\ *n.* A document on the World Wide Web. A Web page consists of an HTML file, with associated files for graphics and scripts, in a particular directory on a particular machine (and thus identifiable by a URL). Usually a Web page contains links to other Web pages. *See also* URL.

Web phone \web'fōn\ *n.* See Internet telephone.

Web server \web'sər-vər\ *n.* See HTTP server.

Web site \web'sīt\ *n.* A group of related HTML documents and associated files, scripts, and databases that is served up by an HTTP server on the World Wide Web. The HTML documents in a Web site generally cover one or more related topics and are interconnected through hyperlinks. Most Web sites have a home page as their starting point, which frequently functions as a table of contents for the site. Many large organizations, such as corporations, will have one or more HTTP servers dedicated to a single Web site. However, an HTTP server can also serve several small Web sites, such as those owned by individuals. Users need a Web browser and an Internet connection to access a Web site. *See also* home page, HTML, HTTP server (definition 1), Web browser.

Web terminal \web'tər-mə-nəl\ *n.* A system containing a central processing unit (CPU), RAM, a high-speed modem or other means of connecting to the Internet, and powerful video graphics, but no hard disk, intended to be used solely as a client to the World Wide Web rather than as a general-purpose computer. *Also called* network computer.

Web TV \web' T-V\ *n.* A system for accessing the World Wide Web and displaying Web pages on a television screen using a set-top box.

webweaver \web'wē-vər\ *n.* See webmaster.

webzine \web'zēn\ *n.* An electronic publication distributed primarily through the World Wide Web, rather than as an ink-on-paper magazine. *See also* ezine.

weighted code \wā'təd kōd\ *n.* A data representation code in which each bit position has a specified inherent value, which might or might not be

Whetstone

included in the interpretation of the data, depending on whether the bit is on or off.

welcome page \wel'kəm pāj\ *n.* See home page.

WELL \wel, W-Ē-L-L\ *n.* Acronym for **Whole Earth 'Lectronic Link**. A conferencing system based in San Francisco, California, that is accessible through the Internet and through dial-up access points in many major cities. The WELL attracts many computer professionals, along with other people who enjoy participating in one of the Internet's most successful virtual communities. Because of the number of journalists and other prominent people who participate in the WELL, it has substantial influence beyond its own relatively small number of subscribers.

well-behaved \wel'bē-hāvd\ *adj.* **1.** Of, pertaining to, or characteristic of a program that performs properly even when given extreme or erroneous input values. **2.** Obeying the rules of a particular programming environment.

well-mannered \wel'man'ərd\ *adj.* *See* well-behaved.

wetware \wet'wār\ *n.* Slang for living beings and their brains, as part of the environment that also includes hardware and software.

"what-if" evaluation \hwət-if ē-val'yōō-ā'shən, wət-if\ *n.* A kind of spreadsheet evaluation in which certain values in a spreadsheet are changed in order to reveal the effects of those changes. For example, a spreadsheet user can use "what-if" evaluation to try different mortgage rates and terms to see the effect on monthly payments and on total interest paid over the life of a loan.

whatis \hwət-iz', wət-iz'\ *n.* **1.** A UNIX utility for obtaining a summary of a keyword's documentation. *See also* man pages. **2.** An Archie command for locating software whose description contains desired words.

What You See Before You Get It \hwət' yōō sē' bə-fōr' yōō get' it, wət'\ *adj.* *See* WYSBYGI.

What You See Is What You Get \hwət' yōō sē' iz hwət' yōō get', wət'\ *adj.* *See* WYSIWYG.

wheel printer \hwēl' prīn'tər, wēl'\ *n.* *See* daisy-wheel printer.

Whetstone \hwet'stōn, wet'stōn\ *n.* A benchmark test that attempts to measure the speed and efficiency with which a computer carries out floating-

WHIR

point
units
mark
incon
marks
tosthe
of Era
WHIRL
tal co
Massa
and u
duced
and r
memb
found
See al
whiteb
that a
work
neous
thoug
whitel
white
tains c
the fre
by an
all the
trum,
produ
nel ov
white
nition
white
paper
ficatio
ificatio
whois
vided
e-mail
listed
mand
that di
netwo
whois
as the
to acc
dresse

IN THE UNITED STATES DISTRICT COURT
FOR THE SOUTHERN DISTRICT OF FLORIDA
MIAMI DIVISION

CASE No. 1:24-cv-21226-RAR

DR. EDWIN A. HERNANDEZ and
EGLA CORP.,

Plaintiffs,

V.

STINGRAY GROUP INC. f/k/a STINGRAY DIGITAL GROUP, INC., STINGRAY MUSIC USA, INC., MOOD MEDIA LLC f/k/a MOOD MEDIA CORPORATION, AT&T ENTERPRISES, LLC f/k/a AT&T CORP., MILLICOM INTERNATIONAL SERVICES, LLC, and BLUE STREAM COMMUNICATIONS, LLC dba BLUE STREAM FIBER,

Defendants.

JURY TRIAL DEMANDED

CLAIM CONSTRUCTION PROPOSED ORDER

BEFORE THE COURT is the Defendants' Opening Claim Construction Brief. Having fully considered the Brief, applicable law, and evidence, the Court hereby construes the identified claim terms from the Patents-in-Suit as follows:

Claim Language	Court’s Construction
“web page”	“an HTML document that is accessible on the world wide web”
“rendering a web page”	“converting a web page’s HTML code into displayable content”

“rendered web page”	“displayable content generated from a document’s HTML code”
“browser”	“an application that converts HTML code into displayable content”
“sequence of screen captures of the rendered web page”	“two or more screen captures of the rendered web page taken at different points in time”
“illustrate a dynamic change”	“illustrate the playback of video, animation or similar motion or change occurring on the rendered web page”
“output the parameters via MPEG outputs”	The claim term is indefinite.
“multimedia assets”	The claim term is indefinite.
“maintain generation of video and audio files, even in failure cases”	The claim term is indefinite.

IT IS SO ORDERED.

SIGNED on ____ day of _____, 2025.

HONORABLE JUDGE RODOLFO A. RUIZ, II
UNITED STATES DISTRICT JUDGE