

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

PINTEREST, INC.,

Petitioner,

v.

OPENTV, INC.

Patent Owner.

IP2026-00334

U.S. Patent No. 12,231,703

DECLARATION OF IMMANUEL FREEDMAN, PH.D., IN SUPPORT
OF PETITION FOR *INTER PARTES* REVIEW

TABLE OF CONTENTS

I.	Introduction.....	1
A.	Qualifications	1
B.	Preparation for this Declaration	4
II.	Legal Understanding.....	6
A.	Claim Construction	6
B.	Anticipation	6
C.	Obviousness.....	7
III.	The '703 patent	8
A.	Priority Date	8
B.	Specification.....	8
C.	Prosecution History	11
D.	Level of Ordinary Skill in the Art.....	12
E.	Claim Construction	13
IV.	Grounds.....	13
V.	Summary of Prior Art	14
A.	Orr.....	14
B.	Hsu.....	15
C.	Lim	15
D.	Hervey	16
VI.	Ground 1: Orr Renders Obvious Claims 1-4, 8-9, 11-14, and 20	17
A.	Limitation-By-Limitation Analysis.....	17
1.	Claim 1	17
2.	Claim 2	35
3.	Claim 3: The method of claim 2, further comprising: preserving an entirety of the playback pipeline in response to the audio decoder and the video decoder in	

	the playback pipeline being the types indicated by the codec information.	37
4.	Claim 4.....	38
5.	Claim 8.....	40
6.	Claim 9: The method of claim 1, wherein the receiving the selection, transitioning to the new piece of content, determining whether the new piece of content is of a same content type as the first piece of content, and determining whether to preserve is performed by a player of the digital receiver.	43
7.	Claim 11.....	44
8.	Claim 12: The system of claim 11, wherein the determining whether to preserve comprises: detecting codec information for the new piece of content, the codec information indicating types of decoders needed for the new piece of content; and determining whether the audio decoder and the video decoder in the playback pipeline are of the types indicated by the codec information.	46
9.	Claim 13: The system of claim 12, wherein the operations further comprise: preserving an entirety of the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being the types indicated by the codec information.	46
10.	Claim 14: The system of claim 12, wherein the operations further comprise: releasing the audio decoder and the video decoder in the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being different than the types indicated by the codec information; and obtaining a new audio decoder and a new video decoder based on the codec information.	47
11.	Claim 20.....	47
VII.	Ground 2: Orr in View of Hsu Renders Obvious Claims 6-7, 10, 16- 17, and 19.....	49
A.	Motivation to Combine	49
B.	Limitation-By-Limitation Analysis.....	51

1.	Claim 6: The method of any one of claim 1, wherein the transitioning to the new piece of content comprises initiating playback of the new piece of content using a bitrate used for playback of the first piece of content to maintain the bitrate for playback of the new piece of content.....	51
2.	Claim 7	52
3.	Claim 10: The method of claim 1, wherein the source element initiates the playback of the new piece of content using a same bitrate as the first piece of content and adjusts the bitrate in response to detecting that the new piece of content cannot continue playback at the bitrate.....	53
4.	Claim 16: The system of any one of claim 11, wherein the transitioning to the new piece of content comprises initiating playback of the new piece of content using a bitrate used for playback of the first piece of content to maintain the bitrate for playback of the new piece of content.....	54
5.	Claim 17: The system of claim 16, wherein the operations further comprise: during playback of the new piece of content using the bitrate, detecting whether the new piece of content can maintain playback at the bitrate; and in response to detecting that the new piece of content cannot continue playback at the bitrate, lowering the bitrate.....	55
6.	Claim 18: The system of claim 11, wherein the operations further comprise: based on the first piece of content and the new piece of content not being the same content type, deconstructing the playback pipeline by releasing the source element, the demultiplexer, the audio decoder, and the video decoder; and creating a new playback pipeline by acquiring other resources that are applicable to a content type of the new piece of content.....	55
7.	Claim 19: The system of claim 11, wherein the source element initiates the playback of the new piece of content using a same bitrate as the first piece of content and adjusts the bitrate in response to detecting that the new piece of content cannot continue playback at the bitrate.....	55

VIII. Ground 3: Orr in View of Lim Renders Obvious Claims 5 and 15	56
A. Motivation to Combine	56
B. Limitation-By-Limitation Analysis.....	59
1. Claim 5: The method of claim 2, wherein the detecting the codec information for the new piece of content comprises examining prefetched metadata for the new piece of content.	59
2. Claim 15.....	60
IX. Ground 4: Hervey Renders Obvious Claims 1-4, 8-9, 11-14, 18, and 20	61
A. Limitation-By-Limitation Analysis.....	61
1. Claim 1	61
2. Claim 2.....	72
3. Claim 3: The method of claim 2, further comprising: preserving an entirety of the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being the types indicated by the codec information.	74
4. Claim 4.....	75
5. Claim 8.....	77
6. Claim 9: The method of claim 1, wherein the receiving the selection, transitioning to the new piece of content, determining whether the new piece of content is of a same content type as the first piece of content, and determining whether to preserve is performed by a player of the digital receiver.	83
7. Claim 11	85
8. Claim 12: The system of claim 11, wherein the determining whether to preserve comprises: detecting codec information for the new piece of content, the codec information indicating types of decoders needed for the new piece of content; and determining whether the audio decoder and the video decoder in the playback pipeline are of the types indicated by the codec information.	89

9.	Claim 13: The system of claim 12, wherein the operations further comprise: preserving an entirety of the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being the types indicated by the codec information.	90
10.	Claim 14: The system of claim 12, wherein the operations further comprise: releasing the audio decoder and the video decoder in the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being different than the types indicated by the codec information; and obtaining a new audio decoder and a new video decoder based on the codec information.	90
11.	Claim 20.....	90
X.	Ground 5: Orr in View of Hsu Renders Obvious Claims 6-7, 10, 16-17, and 19.....	92
A.	Motivation to Combine	92
B.	Limitation-By-Limitation Analysis.....	94
1.	Claim 6: The method of any one of claim 1, wherein the transitioning to the new piece of content comprises initiating playback of the new piece of content using a bitrate used for playback of the first piece of content to maintain the bitrate for playback of the new piece of content.....	94
2.	Claim 7	95
3.	Claim 10: The method of claim 1, wherein the source element initiates the playback of the new piece of content using a same bitrate as the first piece of content and adjusts the bitrate in response to detecting that the new piece of content cannot continue playback at the bitrate.....	96
4.	Claim 16: The system of any one of claim 11, wherein the transitioning to the new piece of content comprises initiating playback of the new piece of content using a bitrate used for playback of the first piece of content to maintain the bitrate for playback of the new piece of content.....	98

5.	Claim 17: The system of claim 16, wherein the operations further comprise: during playback of the new piece of content using the bitrate, detecting whether the new piece of content can maintain playback at the bitrate; and in response to detecting that the new piece of content cannot continue playback at the bitrate, lowering the bitrate.....	98
6.	Claim 18: The system of claim 11, wherein the operations further comprise: based on the first piece of content and the new piece of content not being the same content type, deconstructing the playback pipeline by releasing the source element, the demultiplexer, the audio decoder, and the video decoder; and creating a new playback pipeline by acquiring other resources that are applicable to a content type of the new piece of content.	98
7.	Claim 19: The system of claim 11, wherein the source element initiates the playback of the new piece of content using a same bitrate as the first piece of content and adjusts the bitrate in response to detecting that the new piece of content cannot continue playback at the bitrate.	98
XI.	Ground 3: Hervey in View of Lim Renders Obvious Claims 5 and 15	99
A.	Motivation to Combine	99
B.	Limitation-By-Limitation Analysis.....	101
1.	Claim 5: The method of claim 2, wherein the detecting the codec information for the new piece of content comprises examining prefetched metadata for the new piece of content.	101
2.	Claim 15	103
XII.	Conclusion	103

LIST OF EXHIBITS

Exhibit	Description
EX1001	U.S. Patent 12,231,703 (“703 Patent”)
EX1002	Declaration of Immanuel Freedman, Ph.D.
EX1003	Declaration of Edward Hervey
EX1004	File History of U.S. Patent 12,231,703
EX1005	U.S. Patent Application Publication 2009/0132725 (“Orr”)
EX1006	U.S. Patent Application Publication 2008/0056302A1
EX1007	U.S. Patent 8,516,144 (“Hsu”)
EX1008	U.S. Patent 9,628,543 (“Lim”)
EX1009	“ <i>decodebin3, Modern playback use-cases</i> ” presentation by Edward Hervey at the GStreamer Conference 2015, October 8 th , 2015 (“Hervey”) available at https://web.archive.org/web/20170226222859/https://gstreamer.freedesktop.org/data/events/gstreamer-conference/2015/Edward%20Hervey%20-%20decodebin3.pdf
EX1010	Declaration of Mina Ching
EX1011	“GStreamer Conference 2015,” available at https://web.archive.org/web/20160106210441/http://gstreamer.freedesktop.org/data/events/gstreamer-conference/2015/
EX1012	“Index of data/events/gstreamer-conference/2015,” available at https://web.archive.org/web/20160109011525/http://gstreamer.freedesktop.org/conference/2015/index.html
EX1013	“GStreamer Conference 2015, Schedule,” available at https://web.archive.org/web/20161018073419/https://gstreamer.freedesktop.org/conference/2015/schedule.html
EX1014	“GStreamer Conference 2015, Talks and Speakers,” https://web.archive.org/web/20160125231357/http://gstreamer.freedesktop.org/conference/2015/talks-and-speakers.html

Exhibit	Description
EX1015	LWN.net article “Upcoming work in and around GStreamer,” https://web.archive.org/web/20160330004434/https://lwn.net/Articles/661403/
EX1016	U.S. Pat. No. 8,289,370
EX1017	U.S. Pat. App. Pub. No. 2015/0104158
EX1018	“GstCaps” available at https://web.archive.org/web/20160111144202/http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gstreamer/html/GstCaps.html
EX1019	Intentionally Omitted
EX1020	Curriculum Vitae of Immanuel Freedman, Ph.D.

IPR of U.S. Patent No. 12,231,703

Decl. of Dr. Immanuel Freedman

I, Dr. Immanuel Freedman, declare as follows:

I. Introduction

1. I have been retained by Quinn Emanuel Urquhart & Sullivan, LLP on behalf of the Petitioner Pinterest, Inc. (“Petitioner”) as an independent expert in this *inter partes* review (this “Proceeding”) before the Patent Trial and Appeal Board of the United States Patent and Trademark Office (the “Board”) to review claims 1-20 (“the challenged claims”) of U.S. Patent No. 12,231,703 (“the ’703 patent”). I have been asked by the Petitioner to assist in evaluating the claims and the disclosure of the ’703 patent.

A. Qualifications

2. Ex. 1020 is a true and correct copy of my current CV, which describes my education, patents and publications, employment and research history, and professional activities and awards.

3. I received a Bachelor of Science and Ph.D. in Physics from the University of Durham in England in 1979 and 1986, respectively. I have over 30 years of industry experience, a substantial portion of which was spent working with video, imaging, modeling, simulation, systems analysis, design, development, and testing. I am a registered patent agent before the USPTO. Between obtaining my undergraduate and doctoral degree, I developed a microcomputer system for

IPR of U.S. Patent No. 12,231,703

Decl. of Dr. Immanuel Freedman

detecting coalmine fires and heatings as a scientist for the National Coal Board and worked as a software engineer for Laser-Scan Ltd. in Cambridge, England.

4. After obtaining my Doctorate, I served as a Research Assistant at University College London from September 1986 to June 1987, where I developed digital image processing algorithms to improve image and stereo-matching quality for a digital terrain modeling system, including software and algorithms for affine transformation, edge filtering, kriging interpolation, and image stereo-matching with sub-pixel acuity. I continued my work with digital image processing as a Research Associate at the University of Maryland, from June 1987 to September 1988. During my time at the University of Maryland, I designed algorithms for filtering, segmenting, clustering, and path planning based on digital images organized by quad-tree data structures.

5. I have worked as a chief systems engineer and have extensive experience in the scientific modeling and statistical data compression space. I received National Aeronautics and Space Administration Group Achievement Awards in 1990 and 1992.

6. I have also worked as a software engineer in contract and in senior positions, notably at the Federal National Mortgage Association in Washington, D.C., General Instrument Inc., Comstream Inc., and Armor Safe Technologies Inc.

7. From 1996-1997, I was the sole proprietor of Anugraha, where I researched and developed algorithms and processes to compress fine art photography at an image quality acceptable to artists based on the JPEG imaging standard implemented with image pre-processing and adaptive quantization. I then went on to work as an engineering contractor or consultant for various companies, working primarily on image processing systems and digital interactive television set-top boxes.

8. From 2000 to 2002, I worked as the chief systems engineer for Media Logic Systems Ltd. where I developed a live interactive television system for customers to communicate with human sales agents in video enabled call centers. I spent significant time researching and developing tools and encoder systems to improve image quality at prescribed latency and bit rate for distributing live video and audio streams encoded via low latency methods.

9. In addition to my over thirty years of relevant industry experience, I have authored many publications relating to video and imaging coding. In 2003, I authored a chapter entitled "Video Compression" for the *Internet Encyclopedia*. In 2004 I authored the chapter entitled "Video" for the *Berkshire Encyclopedia of Human-Computer Interaction*. And in 2007 I authored a chapter titled "Video Compression" for the *Handbook of Computer Networks*.

10. I am also a Senior Member of the Institute of Electrical and Electronics Engineers (“IEEE”) and currently chair the Philadelphia Chapter of the Communications & Information Theory Societies. I am a former Chair of the American Association of Pharmaceutical Scientists Pharmacology and Predictive Modeling Communities. I also served as the 2019 Vice Chair of the IEEE P2673 Intelligence Augmentation for Medical Imaging Standards Working Group, currently lead the Trusted Analytics Sub-Group of the IEEE P2795 Shared Analytics for Secured and Unsecured Networks Standards Working Group, and serve as a member of the IEEE Dynamic Spectrum Access Standards Machine Learning Standards Working Group.

B. Preparation for this Declaration

11. In forming my opinions, I have considered the ’703 patent specification, including the Abstract, the figures, and the claim language itself, as would have been understood by a person of ordinary skill in the art as of the priority date of the ’703 patent (a “POSITA”). My understanding of “POSITA” and “priority date” are set forth below. I have also reviewed the file history of the ’703 patent, the Exhibits that are listed in the list of Exhibits, and any other material cited in this declaration.

12. The list of exhibits are:

13. In forming my opinions, I have relied on my personal knowledge and professional experience, and on the documents and information referenced in this declaration.

14. This declaration explains, based on facts and information available to me to date, the subject matter and opinions related to this Proceeding. As such, I am prepared to provide expert testimony regarding opinions formed resulting from my analysis of the issues considered in this declaration if asked about those issues by the Board or by the private parties' attorneys.

15. Additionally, I may discuss my own work, teachings, and knowledge of the state of the art in the relevant time period. I may rely on handbooks, textbooks, technical literature, and the like to demonstrate the state of the art in the relevant period and the evolution of relevant technologies.

16. Throughout this declaration, I refer to specific pages of the '703 patent and other documents. The citations are intended to be exemplary and are not intended to convey that the citations are the only source of evidence to support the propositions for which they are cited.

17. I am being compensated for my time spent on this matter at a rate of \$850 per hour, and my compensation is in no way contingent upon the outcome of this matter or on the opinions I offer. All of the opinions expressed in this declaration are my own.

II. Legal Understanding

18. In this section, I describe my understanding of certain legal standards that I have relied upon in forming my opinions set forth in this declaration. I have been informed of these legal standards by Petitioner's attorneys. I am not an attorney and I have not thoroughly researched the law on patent invalidity. I am relying only on instructions from Petitioner's attorneys for these legal standards.

A. Claim Construction

19. I have been instructed by counsel that claim construction is a matter of law. I understand that in an *inter partes* review, claims are construed using the same claim construction standard that would be used to construe the claim in a civil action, namely according to their plain and ordinary meaning to a POSITA.

20. I understand that a patent may include two types of claims, independent claims and dependent claims. An independent claim stands alone and includes only the limitations it recites. A dependent claim can depend on an independent claim or another dependent claim. I understand that a dependent claim includes all the limitations that it recites in addition to the limitations recited in the claim from which it depends.

B. Anticipation

21. I understand that a patent claim is anticipated when a single piece of prior art describes every element of the claimed invention, either expressly or

inherently, arranged in the same way as in the claim. For inherent anticipation to be found, it is required that the missing descriptive material is necessarily present in the prior art. I understand that, for the purpose of an *inter partes* review, prior art that anticipates a claim can include both patents and printed publications from anywhere in the world.

C. Obviousness

22. I understand that a patent claim is unpatentable and invalid if the subject matter of the claim as a whole would have been obvious to a POSITA as of the time of the invention at issue. My understanding of a POSITA is set forth below. I understand that the following factors must be evaluated to determine whether the claimed subject matter is obvious: (1) the scope and content of the prior art; (2) the difference or differences, if any, between each claim of the patent and the prior art; and (3) the level of ordinary skill in the art at the time the patent was filed. Unlike anticipation, which allows consideration of only one item of prior art, I understand that obviousness may be shown by considering more than one item of prior art. Moreover, I have been informed and I understand that the so-called objective indicia of non-obviousness, also known as “secondary considerations,” are also to be considered when assessing obviousness. These include: (1) commercial success; (2) long-felt but unresolved needs; (3) copying of the invention by others in the field; (4) initial expressions of disbelief by experts in the field; (5) failure of others to solve

the problem that the inventor solved; and (6) unexpected results. I also understand that evidence of objective indicia of non-obviousness must be commensurate in scope with the claimed subject matter.

23. At this time, I am not aware of any evidence of secondary considerations of non-obviousness, have not seen anything identified by the Patent Owner, and do not think anything would overcome the strong showing of obviousness, but I reserve the right to respond if Patent Owner presents evidence or argument of secondary considerations of non-obviousness.

III. The '703 patent

24. The '703 patent is titled "Bitrate and pipeline preservation for content presentation."

25. The '703 patent was filed as U.S. Patent Application No. 18/480,918 on September 4, 2023, and issued on February 18, 2025.

A. Priority Date

26. The '703 patent claims priority to Indian application 201741031917, filed on September 8, 2017. For purposes of this declaration, I have been asked to assume that the '703 patent is entitled to this priority date.

B. Specification

27. The '703 patent is directed to "[s]ystems and methods for optimizing a content change process" in which a "digital receiver receives selection of a new piece

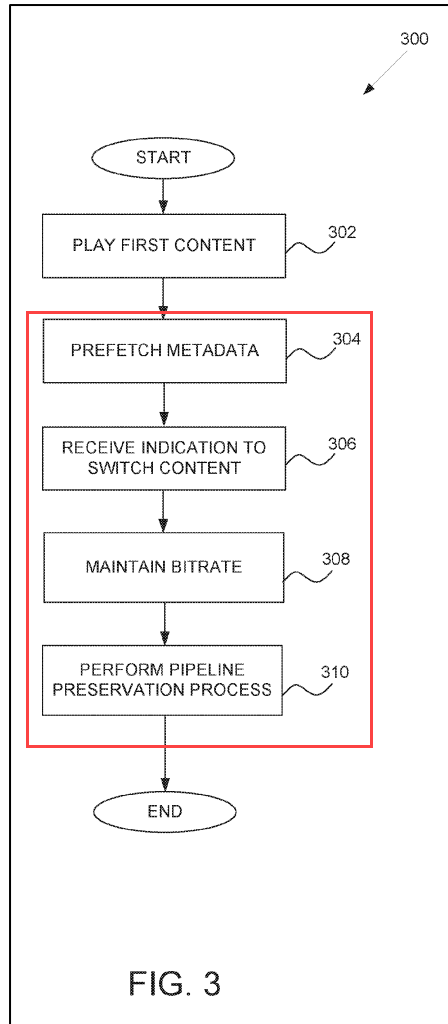
of content for playback,” and “maintains a bitrate used for playback of the first piece of content to initiate playback of the new piece of content.” EX1001, Abstract.

28. The patent identifies two conventional problems that arise at the moment of a content switch. First, with respect to bitrate, “every time a user switches from one content to the next, the playback will start with the lowest bitrate and slowly increase or improve according the capability of the network,” resulting in initially grainy video. (EX1001, 2:37-40.) Second, with respect to pipeline management, a content switch “typically causes a previous content delivery pipeline to be deconstructed (e.g., resources previously acquired are released),” requiring construction of an entirely new pipeline before playback of the new content can begin. (EX1001, 2:55-58.)

29. The alleged solution addresses the bitrate problem “by acquiring appropriate resources for obtaining and processing content data in order to playback selected content.” (EX1001, 5:13-15.) A pipeline 204 for playing content includes a source element 212 that “manages bitrate adaptation at the digital receiver 102” and “remembers a last bitrate that is used for playback of a current/previous piece of content, and continues to use that bitrate for playback of the new content.” (EX1001, 5:22-25.) If the new content cannot be delivered at that bitrate, the source element will lower it accordingly, but the system avoids the conventional penalty of always restarting from the lowest possible bitrate.

30. The pipeline preservation problem is addressed through a two-step analysis using prefetched metadata. The player first checks whether the new content is the same type as the current content (e.g., both HLS or both DASH), and if so, examines codec information to determine whether the same decoders can be reused. When both conditions are satisfied, “the entire pipeline 204 is preserved” and “the resources of the pipeline 204 are maintained and continued to be used to playback the new content.” (EX1001, 8:17–20.)

31. When the content types match but different decoders are required, a partial preservation approach applies: the decoders are released while “a remaining portion of the pipeline 204 (e.g., the source element 212 and the demux 214) are preserved or maintained,” and a new set of compatible decoders is then acquired and added to the preserved portion of the pipeline. (EX1001, 8:27-33.) If the content types do not match at all, all pipeline resources are released and a new pipeline is constructed from scratch. This process is shown, for example, in Fig. 3:



C. Prosecution History

33. On June 21, 2024 an office action issued rejecting pending claims 16-35 on the ground of nonstatutory double patenting as being unpatentable over claims 1-8 and 13-19 of U.S. Patent No. 11,825,139. (EX1004, June 21, 2024 Office Action, pages 4-7). Claim 35 was also rejected under 35 U.S.C. § 101 because that claim was directed to non-statutory subject matter. (*Id.*)

34. On September 4, 2024, applicant filed a terminal disclaimer. (EX1004, September 4, 2024 Terminal Disclaimer.) Applicant also traversed the § 101

rejection. (EX1004, September 4, 2024 Applicant Remarks.) The claims were then allowed. (EX1004, October 15, 2024 Notice of Allowance.)

D. Level of Ordinary Skill in the Art

35. I understand that a person of ordinary skill in the art (“POSITA”) is a hypothetical person who is presumed to be aware of all pertinent art, thinks along conventional wisdom in the art, and is a person of ordinary creativity—not an automaton. In deciding the level of ordinary skill, I understand that the following factors may be considered:

- The levels of education and experience of persons working in the field;
- The types of problems encountered in the field; and
- The sophistication of the technology.

36. I understand that asserted claims must be evaluated from the perspective of a POSITA. I understand that the relevant point in time for determining the qualifications of a person of ordinary skill in the state-of-the-art is the time of the alleged invention, which I assume to be the earliest effective filing date for the patent. Here, I understand that the earliest alleged priority date is October 30, 2013.

37. In my opinion, a person of ordinary skill in the art at the time of the ’703 patent (“POSITA”) had at least a Bachelor of Science in electrical engineering or a similar field and at least two years of practical experience in the field of video

streaming applications. More education can supplement for less practical experience, and vice versa.

38. I meet these criteria now and met them at the time of the alleged invention. I have applied this level of skill in my analysis. My opinions would not change if a slightly higher or lower level of ordinary skill applied.

E. Claim Construction

39. I have been instructed by Petitioner to perform my technical analysis of the disclosures of the prior art by applying the plain and ordinary meaning of all claim terms in view of the specification and prosecution history. In my opinion, none of the limitations of the claims need to be expressly construed.

40. I reserve the right to provide additional opinions concerning claim construction or the application of certain claim constructions to the prior art, as appropriate, and to respond to any particular claim construction-related argument advanced by PO and/or its expert.

41. As described in more detail in the remainder of this declaration, the prior art discloses or renders obvious the challenged claims under any reasonable potential claim interpretation.

IV. Grounds

Ground	Basis	References	Claims
1	§103	Orr	1-4, 8-9, 11-14, 18-20

2	§103	Orr in combination with Hsu	6-7, 10, 16-17, 19
3	§103	Orr in combination with Lim	5, 15
4	§103	Hervey	1-4, 8-9, 11-14, 18-20
5	§103	Hervey in combination with Hsu	6-7, 10, 16-17, 19
6	§103	Hervey in combination with Lim	5, 15

V. Summary of Prior Art

A. Orr

42. Orr is directed to a channel switching system that reduces or eliminates the latency experienced when a user on a device switches between real-time content channels. (EX1005, Abstract; ¶[0003–05].) Orr identifies the same fundamental problem addressed by the '703 patent: conventional media players “reinitialize” when a channel change is detected, tearing down the existing rendering pathway and constructing a new one, a process that “may take more than a few seconds” and produces delays, blank screens, and distorted images. (EX1005, ¶[0004].)

43. Orr’s solution is to determine whether the new channel’s content is compatible with the existing rendering pathway and, if so, to feed the new content directly into the existing pathway without reinitialization, thereby achieving a fast channel switch. (EX1005, ¶[0015]; ¶[0021-22].)

B. Hsu

44. Hsu is directed to an adaptive bitrate streaming media player that tracks historical bitrate data across sessions and uses that history to determine an appropriate startup bitrate for new streaming sessions, rather than defaulting to the lowest available bitrate. (EX1007; Abstract, 5:28-40, 6:1-30.) Hsu expressly identifies the same content-switch quality problem addressed by the '703 patent: conventional systems “default to the lowest bitrate in the dynamic profile set,” resulting in users “playing a lower quality video than what the current bandwidth to that device allows,” with the time to reach optimal quality taking “as long as fifteen (15) seconds.” (EX1007; 5:28-40.)

C. Lim

45. Lim is directed to a content playback system for electronic devices that prefetches content metadata (including type, format, and codec-relevant information) for content the user has not yet selected, based on navigational context during playback of current content. (EX1008; Abstract, claim 1, [0026–36.].)] Samsung's prefetch manager module fetches “preview content clips, cover art, and metadata for all on-deck content” from a content source while the device is playing other content, making that metadata available locally for immediate use upon a user's content selection. (EX1008; claim 1, [0026].)

D. Hervey

46. Hervey is a PowerPoint document that was publicly presented by Edward Hervey at the GStreamer Conference 2015 on October 8th, 2015 in Dublin, Ireland.

47. Hervey is directed to the design of a new GStreamer pipeline element, decodebin3, intended to overcome the limitations of its predecessor, decodebin2, with respect to stream switching and media playback efficiency. (EX1009, Hervey, *e.g.*, slides 8-12.)

48. In my opinion, Exhibit 1009 was publicly available by 2015, or at least by February 2017.

49. An identical copy of Exhibit 1009 can be found on the Internet Archive. I am familiar with the Internet Archive and have reviewed the declaration of Ms. Ching regarding the operation of the Internet Archive. EX1010.

50. The Internet Archive exhibits are printouts of web pages or files that were archived by the Wayback Machine. Each link to the Wayback machine has a date and timestamp which indicates the date and time that the link was archived. In my experience this date and time an accurate reflection of when the Wayback Machine archives that particular page. Being familiar with the operation of the World Wide Web and crawlers like those used by the Wayback Machine, a page could not be archived by the Wayback Machine unless it was available to the public.

Moreover, crawlers generally operate by accessing pages that were linked by other pages, and therefore a page that was archived by the Wayback Machine is typically archived because it was linked from at least one other webpage. This further indicates that the archived webpage was publicly available at the time it was archived.

51. The Internet Archive link of Exhibit 1009 states that it was archived on February 26, 2017. Thus, in my opinion, Hervey was publicly available via the Internet Archive by February 2017.

VI. Ground 1: Orr Renders Obvious Claims 1-4, 8-9, 11-14, and 20¹

A. Limitation-By-Limitation Analysis

1. Claim 1

- a) *I[pre]: A method performed by a digital receiver, the method comprising:*

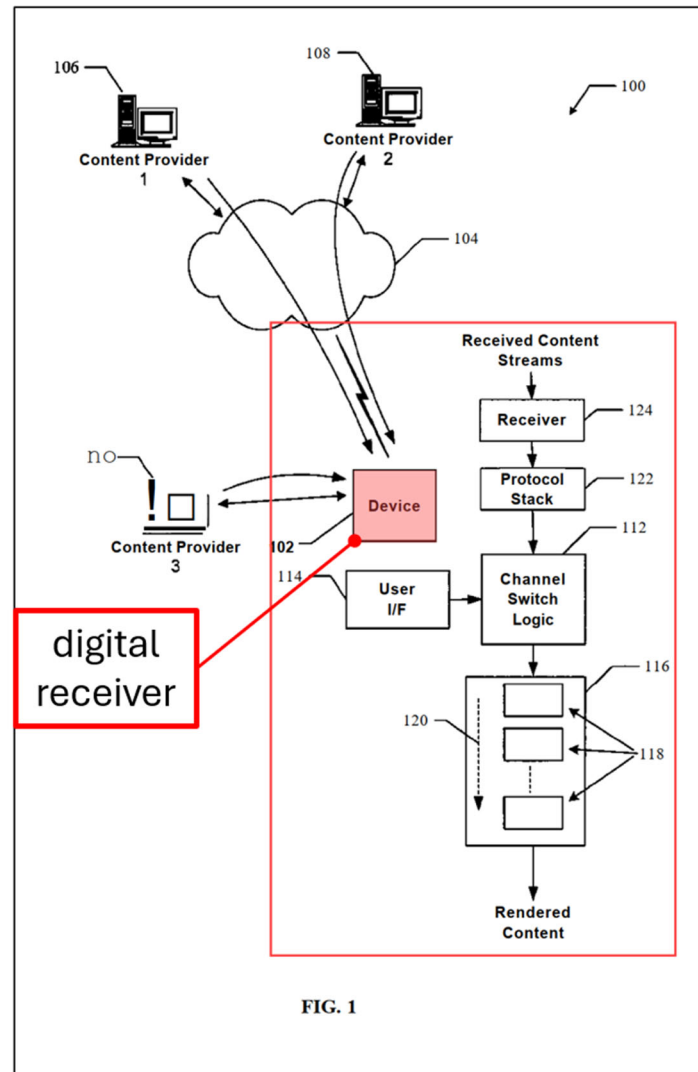
52. In my opinion, and to the extent limiting, Orr discloses the preamble of claim 1.

53. In my opinion, Orr discloses a digital receiver in the form of a device that receives digital media signals. Orr states that it is directed to methods and an apparatus for channel switching at a device that *receives* and renders real-time

¹ Unless noted otherwise, all emphases in quotes and annotations to figures from prior art references are added.

content. (EX1005, Title; Abstract.) Since Orr discloses that the content is delivered via a data network 104, the content is sent as a **digital** signal. (EX1005, ¶[0016]; ¶[0017] (“The content providers 106, 108 and 110 operate to provide content to the device 102.”); FIG. 1.) And Orr describes that the “circuits described in [Orr] may be implemented or performed with ... a digital signal processor (DSP),” further indicating that the signals are digital. (*Id.*, ¶[0053].)

54. As I have shown below in annotated Figure 1, Orr describes its device 102 as including a receiver 124, channel switch logic 112, a user interface 114, a protocol stack 122, and rendering logic 116. (*Id.*, ¶[0018]; FIG. 1.) Orr’s “device 120” is therefore a disclosure of claim 1’s “**digital receiver.**”



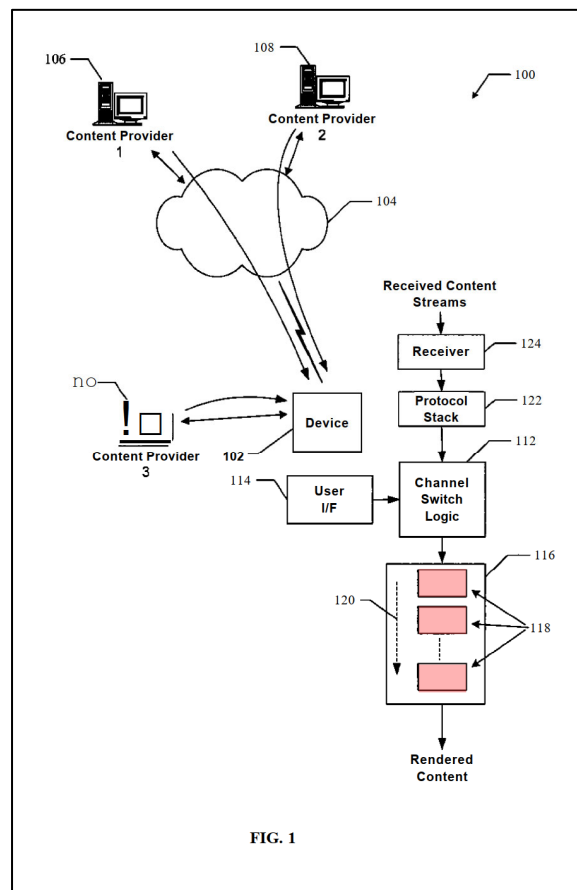
b) *1[a]: receiving a selection of a new piece of content for playback during playback of a first piece of content;*

55. In my opinion, Orr discloses this limitation.

56. Orr describes that device 102 (what I have mapped as the claimed “digital receiver”) *receives a selection of a new piece of content*. Specifically, Orr describes that “channel switch logic 112 [which is part of device 102] operates to select a particular content channel and inputs data from this content data to the

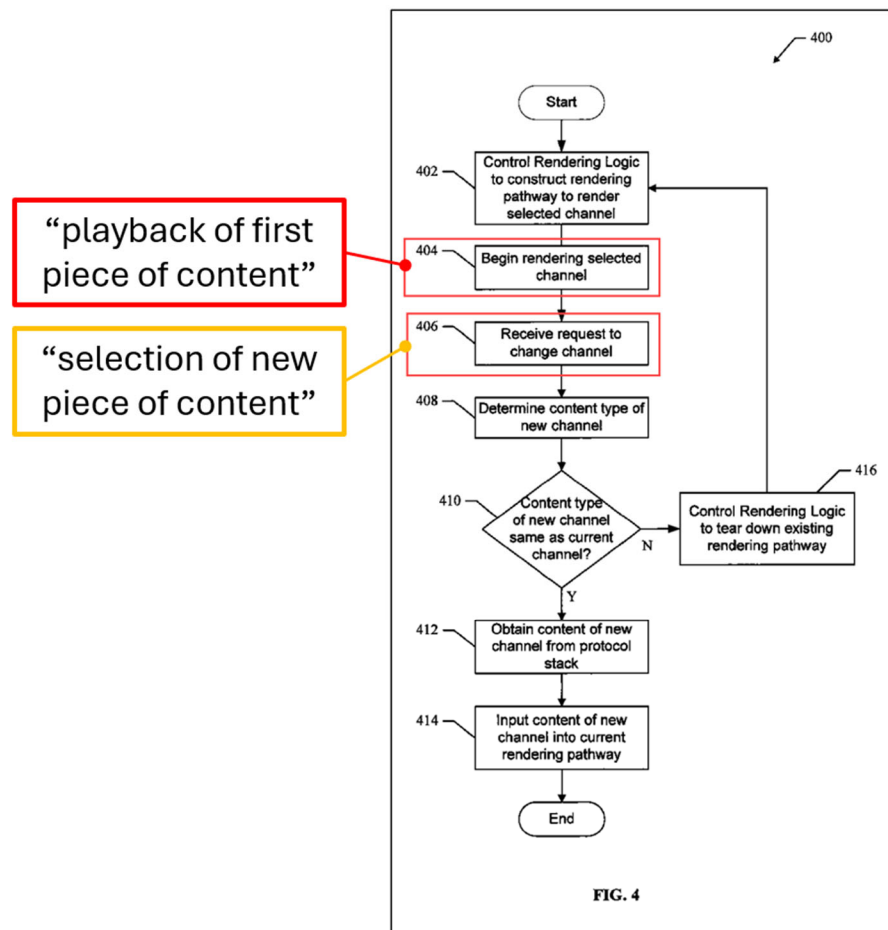
rendering logic 116.” (EX1005, ¶[0019].) Orr also describes that “[t]he channel switch logic 112 then operates to control rendering logic 116 to construct the functional elements 118 needed to decode and render the content [including] content filters [118], transform filters [118], decoders [118], and/or any other type of functional element needed to process the content flow of the selected channel for rendering at the device 102.” (EX1005, ¶[0019].

57. Thus, Orr explicitly describes that the device receives a selection of a new channel, which Orr describes contains a new *piece of content*. I have highlighted these elements in the below annotated Figure 1.



58. Orr also discloses that the new channel selection is made *during playback of the first piece of content*: “[d]uring operation, the device user may request a channel *change* to render content from a *different* channel.” (EX1005, ¶[0020]; ¶[0042] (“[A] channel request is received to render content on a *different* channel.”).) The fact that Orr says there is a “channel change” means that the user was watching a first channel with first content when the selection for a different channel with different content is made. Thereafter, the user interface receives a channel selection indicator from the user and passes it to the channel switch logic during active playback. (EX1005, ¶[0042]; claim 1.)

59. In addition, as I have annotated below, Figure 4 depicts that block 406 (“Receive request to change channel”) appears in the flow after block 404 (“Begin rendering selected channel”) confirming that the selection for a new piece of content is received while the first channel is actively being rendered. (EX1005, FIG. 4.)

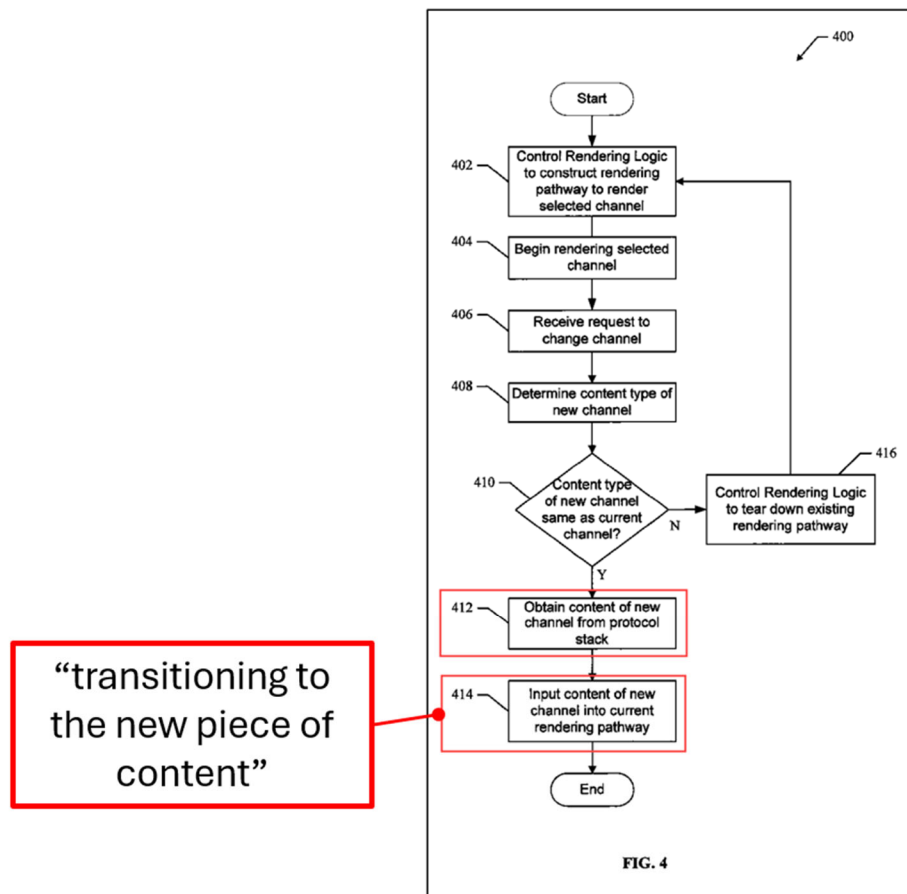


c) *1[b]: in response to the receiving of the selection, transitioning to the new piece of content;*

60. In my opinion, Orr discloses this limitation.

61. Orr teaches that upon receiving the channel selection indicator, the channel switch logic performs a series of operations that eventually results in the device obtaining the new content from the protocol stack and either inputting it into the existing rendering pathway or constructing a new pathway, both of which constitute the device ***transitioning to the new content***. (EX1005, ¶[0022]; ¶[0046-47]; FIG. 4, blocks 412–414.) Orr’s Claim 1 also recites “inputting the second

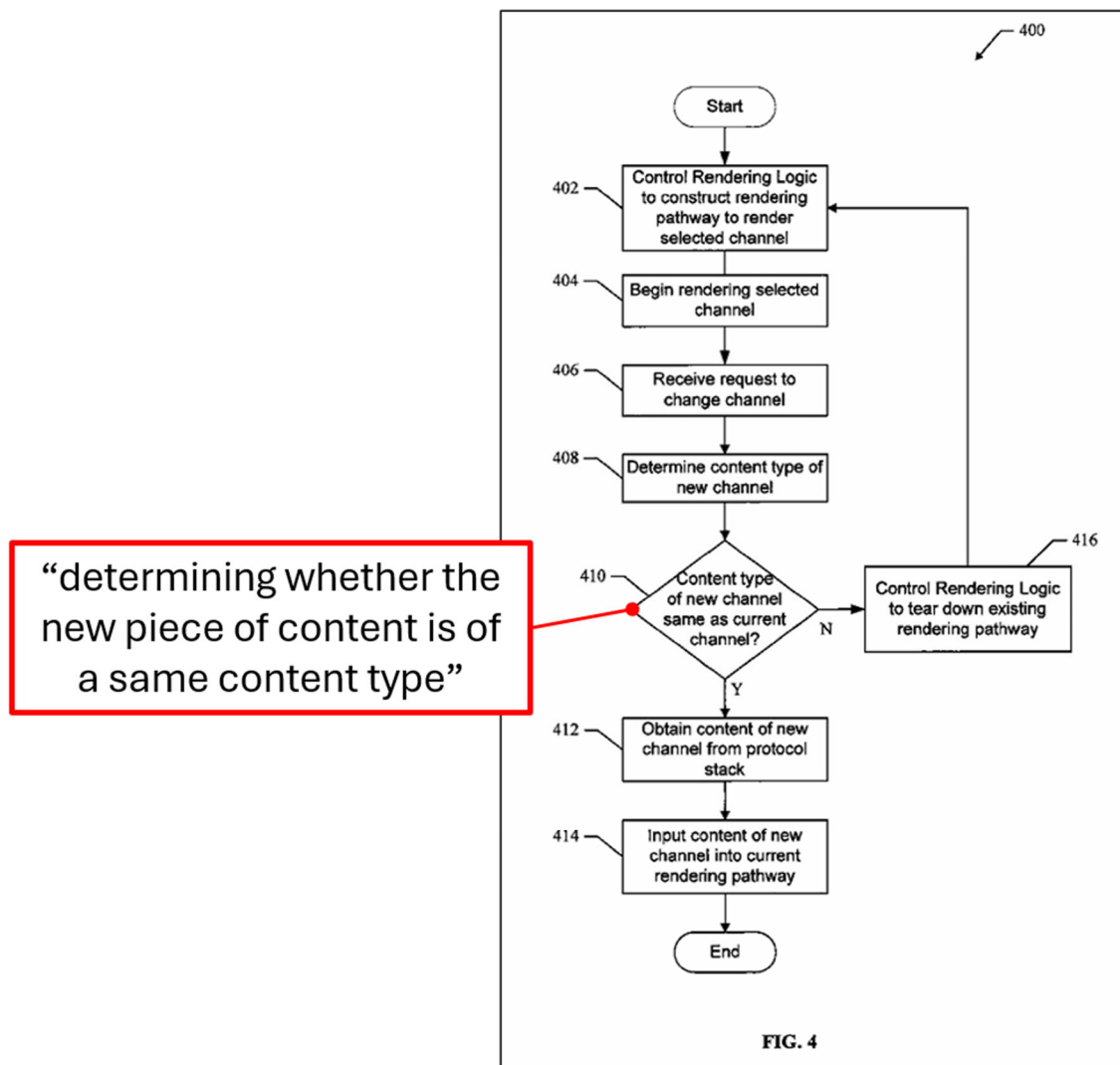
content into the first rendering pathway if it is determined that the second content can be rendered using the first rendering pathway.” As shown in the annotated figure below, FIG. 4, after receiving a “request to change channel” at step 406, Orr’s device performs several steps (as I describe below for limitations 1[c] and 1[d]) that results in “[o]btaining content of new channel” at step 412 and playing the content at step 414. (EX1005, ¶¶[0046-47].)



- d) 1[c]: determining whether the new piece of content is of a same content type as the first piece of content; and

62. In my opinion, Orr discloses this limitation.

63. Orr teaches that the channel switch logic “determines the type of content on the new channel to be rendered” and then assesses “if a current rendering pathway can be used to render the content on the new channel.” (EX1005, ¶[0021]; *see also* ¶[0043]–¶[0044].) Orr describes that the determination of the content-type involves “determin[ing] if the content on the new channel *has the same format* as the content currently being rendered.” (EX1005, ¶[0026].) As I have shown below in annotations, Figure 4 depicts this determination explicitly at block 410: “Content type of new channel *same as current channel?*” (EX1005, FIG. 4, block 410.)

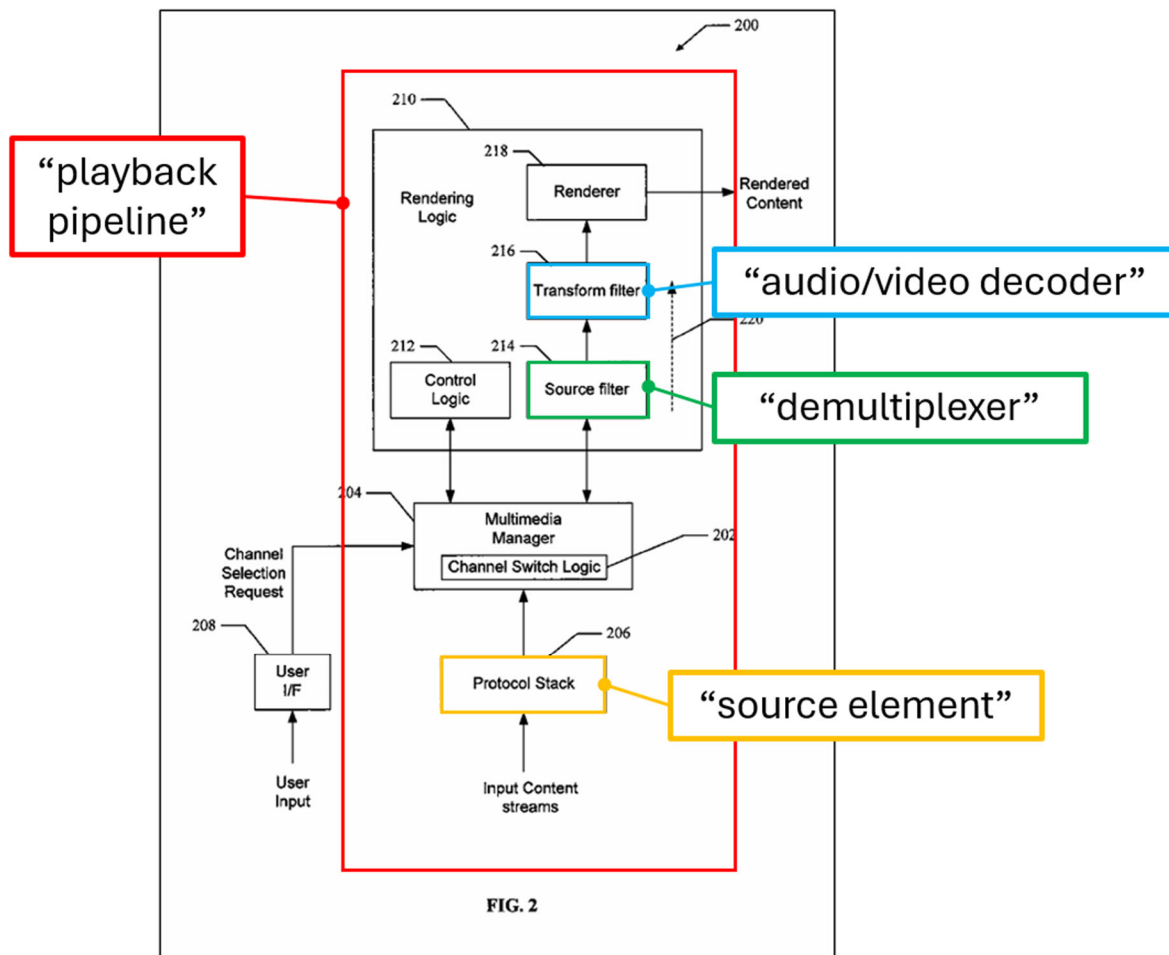


- e) *1[d]: based on the new piece of content being the same content type as the first piece of content, determining whether to preserve at least a portion of a playback pipeline used to play back the first piece of content, the playback pipeline comprising a source element to manage the downloading of content data for the new piece of content, a demultiplexer, an audio decoder, and a video decoder, the preserving of at least the portion of the playback pipeline comprising preserving one or more of the source element, the demultiplexer,*

the audio decoder, or the video decoder used to obtain the first piece of content for the new piece of content.

64. In my opinion, Orr discloses or renders obvious this limitation.

65. I will first discuss the limitation of a ***playback pipeline that comprises a source element, a demultiplexer, an audio decoder, and a video decoder***. Orr discloses that the device includes several components that are arranged in a series to render and play content, which is a ***playback pipeline***. Figure 2 depicts these components, which include a “protocol stack 206,” a “multimedia manager 204,” and “rendering logic 210.” (EX1005, ¶¶[0023-28]; FIG. 2.) These components arranged in a series constitute a ***playback pipeline*** and comprise the claimed source element, demultiplexer, audio decoder, and video decoder. Below, I have annotated Figure 2 to indicate which of these components meet which of the claimed limitations.



66. For the source element to manage the downloading of content data for the new piece of content, in my opinion, Orr discloses that its protocol stack 206 and protocol stack interface 310 meets this limitation. (EX1005, ¶¶[0025], [0033].) The protocol stack “receive[s] channel content from a variety of sources and operates to make this channel content available to the protocol stack interface 310,” including “content data associated with a selected content channel,” *i.e.* the ***new piece of content***. (EX1005, ¶[0033]; *see also* EX1005, ¶[0023] (“[T]he protocol stack 206 may receive the content streams from a network and/or from another communication

channel.”). Because the protocol stack receives content from a network or communication channel, it is *downloading content*. The protocol stack and interface further *manages the downloading of the new piece of content*, because the protocol stack interface “provide[s] protocol stack control signal to request selected content data [*i.e. new piece of content*] from a protocol stack.” (EX1005, ¶[0033].) A POSITA would understand that managing when the new content should be downloaded is *managing the downloading of the new piece of content*, because when content comes in would be understood as managing the reception (*i.e. downloading*) of that content.

67. In the alternative, it is also my opinion that a POSITA would have understood and considered it obvious that the protocol stack manages the downloading of the content while it is being downloaded. Orr describes that content is received by the protocol stack using “a real time transport protocol (RTP).” (EX1005, ¶[0017].) RTP is a well-known standardized protocol, and is described in, for example, U.S. Pat. Pub. No. 2008/0056302 (“Erdal”), which was published March 6, 2008. A POSITA would understand that when content is being downloaded using RTP, *management of the downloading of the content* is performed. For example, downloading content using RTP involves receiving individual RTP packets that each carry a sequence number. (*See, e.g.*, EX1006, ¶[0037].) Packets are not necessarily received in the correct order, and so the protocol stack reorders

the packets according to their sequence number. (*Id.*, ¶[0039] (“The sequence number 314 may be used by a receiver to detect packet loss and to restore proper packet sequence.”)) Each RTP packet also has a synchronization source identifier (SSRC) which identifies the source of the RTP stream. (*Id.*, ¶¶[0037], [0038] (“[T]he SSRC identifier 318 identifies to which RTP stream the particular RTP packet 300 belongs.”)) Orr describes that the protocol stack can “receive[] content streams from one or more sources.” (EX1005, ¶[0017]; [0025] (“The multimedia manager 204 communicates with the protocol stack 206 to receive real time content and also functions to multiplex in streams from other transports if needed.”)) Thus, in situations where the protocol stack is receiving multiple streams of content, it will use the SSRC to assign the packets to their respective stream. All of these are examples of managing the downloading of the content.

68. In my opinion, a POSITA would have known of and considered it obvious to combine these teachings about RTP headers with Orr. A POSITA would have been motivated to do so because Orr expressly describes the use of RTP to download content and a POSITA would understand that Orr is therefore disclosing the use of RTP headers, as such headers are defined as part of the RTP protocol. A POSITA would have been motivated to use RTP headers in order to properly implement what Orr has taught. In addition, a POSITA would be motivated by the benefits in using RTP headers, including the sequence number and SSRC. A

POSITA would be motivated to use the sequence number header because it allows the protocol stack to rearrange packets in the correct order if they are received out of order. It is common for packets to be received out of order and if they are not properly ordered, the content stream will be incomprehensible to the viewer. A POSITA would also be motivated to use the SSRC header in order to handle receiving multiple streams, which Orr expressly teaches and which a POSITA would understand would be beneficial. A POSITA would have had a reasonable expectation of success because implementing the well-known RTP protocol, including the sequence number and SSRC headers, was well-within the capabilities of an ordinary artisan by the earliest filing date of the '703 patent.

69. For the demultiplexer, Orr discloses that the playback pipeline includes a demultiplexer, which is a component that separates the video and audio signal. (EX1001, 5:36-38 (“The demux demultiplexes the content data ... to obtain an audio signal and a video signal.”). Orr teaches that content “compris[es] video, audio, captioning, and overlays etc.” (EX1005, ¶[0018].) Orr further discloses **demultiplexing** of the audio and video performed by a “source filter 214” that is part of the “rendering logic 210,” *i.e.* a portion of the **playback pipeline**. (EX1005, ¶[0024].) Specifically, the source filter “extracts media data for presentation to the transform filter 216.” (EX1005, ¶[0024].) Orr explains that the “transform filter 216 is a pluggable filter” and uses, for example, “an H264 decoder that takes in data from

the source filter 214 and decompresses it.” (EX1005, ¶[0024].) H.264 is a known *video* compression/decompression algorithm—and decoders generally decompress audio or video but not both. (EX1016, 2:44-47 (“H.264 standard video codecs”).) Thus, because the decoders can only operate on either a video or audio signal, but not both, a POSITA would understand that the “source filter” “extracts media data” in the form of a separate video and audio signal for decoding by the transform filter using respective video and audio decoders. (EX1005, ¶[0024].) This is further confirmed by Orr’s disclosure that the transform filter sends the decompressed signal to “the renderer 218 [which] operates to render *video, audio*, and overlays that are included in the content data.” (EX1005, ¶[0024].)

70. In addition, it would have been obvious to include a demultiplexer in Orr’s playback pipeline. Demultiplexers for extracting digital audio and video signals have been known since the 1950s, and are well-attested in the prior art. (*See, e.g.*, EX1017, ¶52 (“The stream demultiplexer 634 separates the AV bit stream AVS into a video bit stream VBS and an audio bit stream ABS”)). Orr discloses that its content signals contain both a video and an audio signal (EX1005, ¶[0018]) as well as the use of software decoders (such as H264), which decode audio or video signals, but not both (EX1005, ¶[0024]). Thus, a POSITA would be motivated to demultiplex the audio and video signal in order that they can be decoded and rendered to the user. Without such demultiplexing, the decoders would be unable to decode the signal. A

POSITA would have had a reasonable expectation of success because demultiplexers for digital signals have been in use for decades and well-within the skill of an ordinary artisan.

71. For the audio decoder, in my opinion, Orr discloses that its playback pipeline includes an audio decoder. Orr teaches that content data includes “audio” and that the “rendering logic” includes elements “needed to decode and render the content.” (EX1005, ¶¶[0018]-[0019].) In addition, Orr discloses that the rendering logic includes a “transform filter 216,” which “is a pluggable filter” that “decompresses” (*i.e. decodes*) media content using a “decoder.” (EX1005, ¶[0024].) The pluggable filter sends the decoded signals to the “renderer” which “render[s] ... audio.” (EX1005, ¶[0024].) In other words, Orr teaches that content includes audio and video, further teaches the rendering logic renders the content, and teaches that a decoder is used. Since an audio decoder is needed to decode the audio, a POSITA would therefore understand that the transform filter of the rendering logic includes an *audio decoder*.

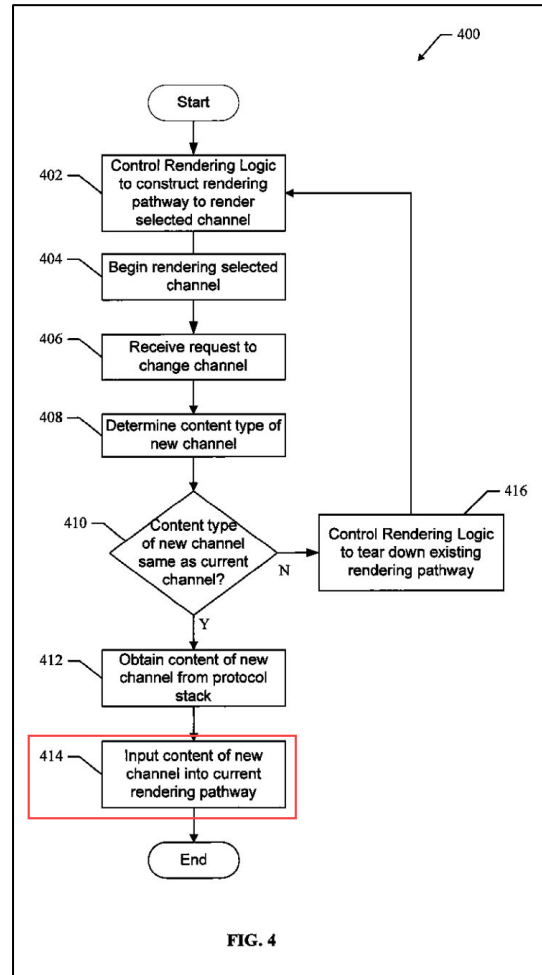
72. In my opinion, it also would have been obvious to include an audio decoder in Orr’s playback pipeline. Audio decoders to decode digital audio signals for media playback were well-known, including those often used with the H.264 video decoder like AAC. (EX1017, ¶[0055] (“The stream demultiplexer ... outputs ... the audio bit stream ABS to the audio decoder 612”); ¶¶[0091]-[0092] (“H.264

... may be similarly used ... MPEG-2 AAC is used as an encoding scheme for speech signals.”)). Orr already teaches that the rendering logic includes a decoder for rendering content, which includes audio. (EX1005, ¶¶[0018]-[0019].) A POSITA would be motivated to give effect to Orr’s teachings by using an audio decoder, because that would allow the audio to be decoded and rendered for playback to the user. A POSITA would have had a reasonable expectation of success because audio decoders were ubiquitous at the time of the ’703 patent and it was well-within the skill of an ordinary artisan to implement an audio decoder in a playback pipeline.

73. For the video decoder of the playback pipeline, Orr discloses that its “transform filter 216 is a pluggable filter, such as an H264 decoder,” a well-known video decoder. (EX1005, ¶[0024].)

74. For the limitation *based on the new piece of content being the same content type as the first piece of content, determining whether to preserve at least a portion of a playback pipeline used to play back the first piece of content*, Orr discloses this. When the content-type comparison at block 410 of FIG. 4 yields a positive result, Orr’s channel switch logic does not reinitialize the rendering logic but instead feeds the new content directly into the existing pathway: “If the channel switch logic 112 determines that the current rendering pathway 120 can process and render the content on the new channel, then the channel switch logic 112 does not re-initialize the rendering logic 116 but instead, feeds the content from the new

channel into the current rendering pathway.” (EX1005, ¶[0022].) FIG. 4 depicts this at block 414: “Input content of new channel into current rendering pathway.” (EX1005, FIG. 4, block 414.) The processing logic makes this determination based on content-type compatibility. (EX1005, ¶[0037]; claim 1.) Orr therefore describes determining whether the type of content of the new piece of content is the same as the type of content of the first piece of content (at step 410), and preserving at least a portion of the current playback pipeline.



75. For the limitation *preserving one or more of the source element, the demultiplexer, the audio decoder, or the video decoder used to obtain the first piece of content for the new piece of content*, Orr teaches that when the content types match, the existing source filter (*i.e. demultiplexer*) and transform filter (*i.e. audio and video decoders*) are preserved in place and used to process the new channel's content for display. (EX1005, ¶[0022]; ¶[0028]; ¶[0047].) These elements are thus “used to obtain the first piece of content for the new piece of content” as recited in this limitation.

2. Claim 2

- a) 2[pre] / 2[a]: *The method of claim 1, wherein the determining whether to preserve comprises: detecting codec information for the new piece of content, the codec information indicating types of decoders needed for the new piece of content; and*

76. In my opinion, Orr discloses this limitation.

77. Orr teaches *detecting the format and codec requirements of the new content as part of the content-type determination*. Orr describes its channel switch logic as determining “whether or not the functional elements 118 that currently exist on the rendering pathway 120 are able to process and render the content on the new channel.” (EX1005, ¶[0021].) A POSITA would understand that the format and codec requirements of content are included in metadata of that content. In order to “process and render content,” the rendering software must have the correct software

compatible with the content—*i.e.* the correct software for handling the content’s format and the correct codec for decoding the content based on how it was compressed.

78. Thus, determining whether the elements can *process and render* content requires ascertaining the codec and format requirements of the new channel’s content, since a determination must be made that the rendering pathway has the right software and codecs to process and render the content. Orr confirms this when it states that “there is no guarantee that the source 214 and transform 216 filters to *decode* the new channel will be the same as those being used for the current channel” (EX1005, ¶[0026]), confirming that the system evaluates what codec types the new content requires. Orr’s claim 4 also expressly recites “determining that the second content can be rendered using the first rendering pathway if the second content is of the same format as the first content.” This necessarily involves detecting the format and codec requirements of the new content. (EX1005, claim 4.)

b) *2[b]: determining whether the audio decoder and the video decoder in the playback pipeline are of the types indicated by the codec information.*

79. In my opinion, Orr discloses this limitation.

80. Orr teaches this determination as part of its format compatibility check. Having determined the format and codec requirements of the new content, Orr’s processing logic determines whether the existing functional elements (including the

transform filter and decoders) are capable of processing content in that format:

“channel switch logic 112 determines whether or not the functional elements 118 that currently exist on the rendering pathway 120 are able to process and render the content on the new channel.” (EX1005, ¶[0021].) As I discussed above in claim 1, Orr’s transform filter, which is part of the rendering pathway, uses “a pluggable filter” involving codecs. (EX1005, ¶[0024].) In order for the rendering pathway to be able to “process and render content,” the transform filter must have the compatible codec for the new content, including audio and video decoders. Thus, Orr teaches determining whether the rendering pathway has the audio and video decoders indicated by the codec information.

3. Claim 3: The method of claim 2, further comprising: preserving an entirety of the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being the types indicated by the codec information.

81. In my opinion, Orr discloses this limitation.

82. Orr teaches preserving the entirety of the existing rendering pathway when the content types match. As I discussed above in limitation 1[d], Orr discloses a *playback pipeline* in the form of a protocol stack (*i.e. source element*), a multimedia manager, and rendering logic (*i.e. demultiplexer and audio/video decoders*). Orr teaches that when the processing logic determines that the existing pathway can render the new content, “the channel switch logic 112 does not re-initialize the rendering logic 116 but instead, feeds the content from the new channel

into the current rendering pathway.” (EX1005, ¶[0022].) FIG. 4 reflects this at block 414: the entire rendering pathway (source filter, transform filter, and renderer) is preserved and used for the new content without teardown. (EX1005, ¶[0028]; ¶[0047]; FIG. 4, blocks 412–414.) Orr's claim 1 recites “inputting the second content into the first rendering pathway if it is determined that the second content can be rendered using the first rendering pathway,” confirming that the entire existing pathway is preserved and reused. (EX1005, claim 1.) By contrast, if the new content is “formatted differently from the content currently being rendered, the channel switch logic 202 notifies the control logic 212 to construct a new rendering pathway 220 to decode the content on the new channel.” (EX1005, ¶[0027].) Thus, for any new content, the protocol stack and multimedia manager of the playback pipeline are preserved, and if the new content can be decoded by the preexisting audio and video decoders, then the rendering logic is also preserved. Therefore, the entire playback pipeline is preserved.

4. Claim 4

- a) *4[pre]/4[a]: The method of claim 2, further comprising: releasing the audio decoder and the video decoder in the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being different than the types indicated by the codec information; and*

83. In my opinion, Orr discloses this limitation.

84. Orr teaches releasing the existing rendering pathway, which contain the audio and video decoders, when the content types differ. Specifically, Orr teaches that when “the content types are different,” Orr “signal[s] the rendering logic to tear down the existing rendering pathway.” (EX1005, ¶¶[0044]–¶¶[0045]; FIG. 4, block 416.) Orr’s claim 6 also recites “tearing down the first rendering pathway” when “the second content can not be rendered using the first rendering pathway,” which supports my opinion. (EX1005, claim 6.) “Tearing down” the rendering pathway releases the transform filter and decoder elements (including the audio and video decoders) that comprise it. For example, at FIG. 4, block 416, when the content types are determined not to match (block 410, “N[o]”), the processing logic controls the rendering logic interface to signal the rendering logic to tear down the existing rendering pathway. (EX1005, ¶¶[0044]–¶¶[0045]; FIG. 4, block 416.) The teardown releases all elements of the rendering pathway including the source filter, the transform filter (corresponding to the demultiplexer, audio decoder, and video decoder), and the renderer. (EX1005, ¶¶[0045]; FIG. 2.)

b) *4[b]: obtaining a new audio decoder and a new video decoder based on the codec information.*

85. In my opinion, Orr discloses this limitation.

86. Orr teaches that after tearing down the old pathway, it constructs a new rendering pathway with a new audio and video decoder that is suitable for rendering the new content based on its codec information. (EX1005, ¶¶[0045]–¶¶[0046]; FIG. 4,

block 416 (returning to block 402 to “construct rendering pathway to render selected channel”).) Specifically, the new rendering pathway can “render different content,” which a POSITA would understand would require obtaining a new audio and video codec suitable for rendering the content. These audio and video decoders are new because Orr teaches that the new content was “formatted differently,” thereby necessitating “a new rendering pathway 220 to *decode* the content on the new channel.” (EX1005, ¶[0027].) In order to decode the new content that has a different format, a new video and new audio codec would be needed—as the previous codecs had to be removed as they were incapable of decoding the new content.

5. Claim 8

- a) *8[pre]/8[a]: The method of claim 1, further comprising: based on the first piece of content and the new piece of content not being the same content type, deconstructing the playback pipeline by releasing the source element, the demultiplexer, the audio decoder, and the video decoder; and*

87. In my opinion, Orr discloses or renders obvious this limitation.

88. Orr teaches full deconstruction of the rendering logic when content types differ. At FIG. 4, block 416, when the content types are determined not to match (block 410, “N[o]”), the processing logic controls the rendering logic interface to signal the rendering logic to tear down the existing rendering pathway. (EX1005, ¶[0044]–¶[0045]; FIG. 4, block 416.) Orr’s claim 6 recites “tearing down the first rendering pathway” in this scenario. (EX1005, claim 6.) The teardown

releases all elements of the rendering pathway including the transform filter (corresponding to the demultiplexer, audio decoder, and video decoder), and the renderer. (EX1005, ¶[0045]; FIG. 2.)

89. Orr also discloses that the protocol stack is released. As I discussed above for limitation 1[d], Orr describes that the protocol stack can use RTP to manage downloading of the content. However, Orr also describes that content can be provided “as files in (MP4) format,” and a POSITA would further understand that other formats and streaming protocols could be used. (EX1005, ¶[0017].) Orr also states that the “multimedia manager 204 communicates with the protocol stack 206 to receive real time content and *also* functions to multiplex in *streams from other transports if needed.*” (*Id.*, ¶[0025].) Thus, Orr teaches that “if needed,” *i.e.* if the streaming format or protocol is different from that supported by the current protocol stack, a new “transport” or protocol stack would be created as it would be needed. Accordingly, in the event the new piece of content is not the same content type because it also uses a completely different protocol or format, it would be necessary to release and create a new protocol stack that is compatible with the new protocol or format.

90. A POSITA would also have considered it obvious to release and create a new protocol stack in the event the new content uses a different protocol or format. A POSITA would understand that different protocols are supported using different

APIs. Therefore, if a different protocol or format is encountered, the multimedia manager would need to use a different API than the current protocol stack. In creating a new protocol stack by calling the new API, a POSITA would be motivated to release the current protocol stack, as it would save computing resources. A POSITA would also have a reasonable expectation of success, as releasing the protocol stack is easily implemented using standard programming techniques and well-within the skill of an ordinary artisan.

b) *8[b]: creating a new playback pipeline by acquiring other resources that are applicable to a content type of the new piece of content.*

91. In my opinion, Orr discloses this limitation.

92. Following the pathway teardown at block 416, FIG. 4 of Orr returns to block 402 (“Control Rendering Logic to construct rendering pathway to render selected channel”), at which point new resources appropriate to the new content’s format are acquired to construct a fresh rendering pathway. (EX1005, ¶[0040]; ¶[0045-46]; FIG. 4.) This includes acquiring a new demultiplexer, audio codec, and video codec—the latter two I discuss above for limitation 4[b]. In addition, Orr’s claim 6 recites “constructing a second rendering pathway to render the second content” which inherently includes acquiring new elements applicable to the new content type. (EX1005, claim 6.)

93. Further, as I discussed above for limitation 8[a], a POSITA would be motivated to release and create a new protocol stack.

6. **Claim 9: The method of claim 1, wherein the receiving the selection, transitioning to the new piece of content, determining whether the new piece of content is of a same content type as the first piece of content, and determining whether to preserve is performed by a player of the digital receiver.**

94. In my opinion, Orr discloses this limitation.

95. I begin with a discussion of what the '703 patent describes as the “player.” In the '703 patent, the “player” is the component that constructs the pipeline, manages it, and decides whether to preserve or reconstruct it upon a content switch: “The player 202 is configured to manage the pipeline 204 [and] the player 20[2] [*sic*] also decides whether to maintain at least a portion of the pipeline 204 or construct a new pipeline when a switch in content occurs.” (EX1001, 4:66-5:10.) The player thus serves as the top-level control component that receives content selections, initiates transitions, and makes pipeline preservation decisions.

96. Orr discloses such a *player*. Orr’s channel switch logic 202 (which may be embedded within the multimedia manager 204) (EX1005, ¶[0023]; FIG. 2.) performs each of the operations recited in this limitation. It receives channel selection requests from the user interface (EX1005, ¶[0025]; ¶[0042]); determines the content type of the new channel and whether it is compatible with the existing rendering pathway (EX1005, ¶[0021]; ¶[0026]; ¶[0043-44]); and controls the

rendering logic to either preserve the existing pathway or construct a new one.

(EX1005, ¶¶[0022]; ¶¶[0037]; ¶¶[0044-47].) Orr's claim 7 further confirms this mapping, reciting an "apparatus for channel switching" comprising "processing logic configured to determine whether the real time content can be rendered using an existing rendering pathway" and "a rendering interface configured to input the real time content into the existing rendering pathway." (EX1005, claim 7.)

7. Claim 11

a) *11[pre]: A system comprising:*

97. In my opinion, for the reasons I discussed above for limitation 1[pre], Orr discloses this limitation.

b) *11[a]: one or more hardware processors; and*

98. In my opinion, Orr discloses this limitation. Orr's device contains processing logic which is described as comprising "at least one of a CPU, processor, gate array, hardware logic, memory elements, virtual machine, and/or hardware executing software." (EX1005, ¶¶[0031].) Orr's apparatus claims similarly recite "processing logic configured to determine whether the real time content can be rendered using an existing rendering pathway." (EX1005, claim 7.)

c) *11[b]: a storage device storing instructions that configure the one or more hardware processors to perform operations comprising:*

99. In my opinion, Orr discloses this limitation. Orr teaches that its channel switching system "comprises a computer program product having one or more

program instructions (‘instructions’) or sets of ‘codes’ stored or embodied on a machine-readable medium,” and that “[w]hen the codes are executed by at least one processor ... their execution causes the processor to provide the functions of the channel switching system described herein.” (EX1005, ¶[0038].) Orr further describes specific example storage media including “a floppy disk, CDROM, memory card, FLASH memory device, RAM, ROM, or any other type of memory device or machine-readable medium.” (*Id.*) Orr’s claim 19 recites a corresponding computer program product comprising “a machine-readable medium encoded with codes executable to: receive a channel selection indicator ...; determine whether the real time content can be rendered using an existing rendering pathway; and input the real time content into the existing rendering pathway.” (EX1005, claim 19.)

- d) *11[c]: receiving a selection of a new piece of content for playback during playback of a first piece of content;*

100. In my opinion, for the reasons I discussed above for limitation 1[a], Orr discloses this limitation.

- e) *11[d]: in response to the receiving of the selection, transitioning to the new piece of content;*

101. In my opinion, for the reasons I discussed above for limitation 1[b], Orr discloses this limitation.

- f) *11[e]: determining whether the new piece of content is of a same content type as the first piece of content; and*

102. In my opinion, for the reasons I discussed above for limitation 1[c], Orr discloses this limitation.

- g) *11[f]: based on the new piece of content being the same content type as the first piece of content, determining whether to preserve at least a portion of a playback pipeline used to play back the first piece of content, the playback pipeline comprising a source element to manage the downloading of content data for the new piece of content, a demultiplexer, an audio decoder, and a video decoder, the preserving of at least the portion of the playback pipeline comprising preserving one or more of the source element, the demultiplexer, the audio decoder, or the video decoder used to obtain the first piece of content for the new piece of content.*

103. In my opinion, for the reasons I discussed above for limitation 1[d], Orr discloses or renders obvious this limitation.

- 8. **Claim 12: The system of claim 11, wherein the determining whether to preserve comprises: detecting codec information for the new piece of content, the codec information indicating types of decoders needed for the new piece of content; and determining whether the audio decoder and the video decoder in the playback pipeline are of the types indicated by the codec information.**

104. In my opinion, for the reasons I discussed above for claim 2, Orr discloses this limitation.

- 9. **Claim 13: The system of claim 12, wherein the operations further comprise: preserving an entirety of the playback pipeline in response to the audio decoder and the video**

decoder in the playback pipeline being the types indicated by the codec information.

105. In my opinion, for the reasons I discussed above for claim 3, Orr discloses this limitation.

10. Claim 14: The system of claim 12, wherein the operations further comprise: releasing the audio decoder and the video decoder in the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being different than the types indicated by the codec information; and obtaining a new audio decoder and a new video decoder based on the codec information.

106. In my opinion, for the reasons I discussed above for claim 4, Orr discloses this limitation.

11. Claim 20

a) *20[pre]: A machine-storage medium storing instructions that when executed by one or more hardware processors of a machine, cause the machine to perform operations comprising:*

107. In my opinion, for the reasons I discussed above for limitations 11[a] and 11[b], Orr discloses this limitation.

b) *20[a]: receiving a selection of a new piece of content for playback during playback of a first piece of content;*

108. In my opinion, for the reasons I discussed above for limitation 1[a], Orr discloses this limitation.

- c) *20[b]: in response to the receiving of the selection, transitioning to the new piece of content;*

109. In my opinion, for the reasons I discussed above for limitation 1[b], Orr discloses this limitation.

- d) *20[c]: determining whether the new piece of content is of a same content type as the first piece of content; and*

110. In my opinion, for the reasons I discussed above for limitation 1[c], Orr discloses this limitation.

- e) *20[d]: based on the new piece of content being the same content type as the first piece of content, determining whether to preserve at least a portion of a playback pipeline used to play back the first piece of content, the playback pipeline comprising a source element to manage the downloading of content data for the new piece of content, a demultiplexer, an audio decoder, and a video decoder, the preserving of at least the portion of the playback pipeline comprising preserving one or more of the source element, the demultiplexer, the audio decoder, or the video decoder used to obtain the first piece of content for the new piece of content.*

111. In my opinion, for the reasons I discussed above for limitation 1[d], Orr discloses this limitation.

VII. Ground 2: Orr in View of Hsu Renders Obvious Claims 6-7, 10, 16-17, and 19

A. Motivation to Combine

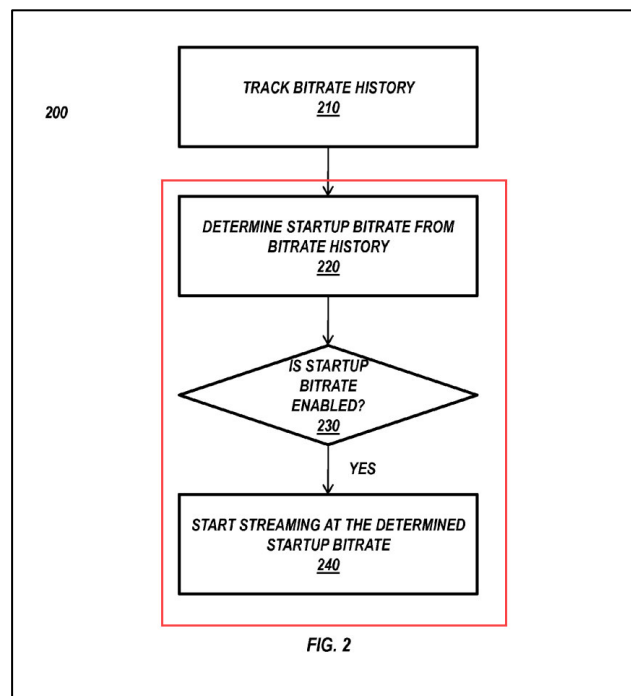
112. In my opinion, a POSITA would have been motivated to modify Orr in view of Hsu to incorporate teachings of bitrate initialization and would have had a reasonable expectation of success.

113. In my proposed combination, Orr's protocol stack continues to manage the retrieval of content data from the content source. (EX1005, ¶[0033].) Hsu teaches initializing a new content download session at a historically derived bitrate rather than the system default. (EX1007, 6:1-5.) A POSITA would implement Hsu's bitrate initialization teaching in the protocol stack of Orr's pipeline, configuring it to begin downloading new content at the previously achieved bitrate. This requires no architectural modification to Orr and produces the entirely predictable result of faster, higher-quality content transitions. Moreover, the modification is well-within the skills of an ordinary artisan, as bitrate initialization was well-known and straightforward to implement. Neither reference teaches away from the combination. A POSITA would therefore have a reasonable expectation of success.

114. Both Orr and Hsu address the same fundamental user experience problem: the latency and quality degradation experienced by a user when switching between pieces of content on a streaming device. Orr addresses the pipeline reconstruction component of this problem: eliminating the delay caused by tearing

down and rebuilding the rendering pathway. (EX1005, ¶[0004]–¶[0005]; ¶[0015].)

As shown in FIG. 2 below, Hsu addresses the bitrate initialization component: eliminating the quality degradation caused by defaulting to the lowest bitrate at the start of each new session. (EX1007, 5:33-39; FIG. 2.) This would therefore motivate a POSITA to further improve the user experience consistent with Orr’s goals.



115. Orr and Hsu are therefore complementary solutions: Orr operates at the pipeline and rendering layer, while Hsu operates at the download and bitrate layer. A POSITA designing a comprehensive content-switch optimization system would naturally consider and implement both.

116. A POSITA combining Orr and Hsu would implement Hsu’s bitrate initialization teaching in the source element of Orr’s pipeline. For example, by configuring the source element to initiate the new content download at the bitrate

derived from Hsu's historical tracking mechanism, rather than at the system default lowest bitrate. This integration is architecturally natural and technically straightforward, as the source element is the component that manages download sessions and bitrate selection in Orr's pipeline.

117. In addition, both references are directed to the same class of devices. Orr is directed to devices receiving and rendering real-time content streams on wireless networks and Internet-connected consumer devices. (EX1005, ¶[0014-16].) Hsu is directed to adaptive bitrate streaming media players on client devices explicitly including "set-top box, Internet-enabled television" and other consumer electronics. (EX1007, 4:50-58.) A POSITA working on content-switch optimization for any such device would routinely consult both references.

B. Limitation-By-Limitation Analysis

- 1. Claim 6: The method of any one of claim 1, wherein the transitioning to the new piece of content comprises initiating playback of the new piece of content using a bitrate used for playback of the first piece of content to maintain the bitrate for playback of the new piece of content.**

118. In my opinion, Hsu discloses this limitation and therefore in combination with Orr renders obvious this limitation.

119. Hsu address bitrate maintenance across content transitions. Hsu discloses that, upon initiating playback of new content, the player uses a bitrate derived from the history of prior sessions, specifically the bitrate used during prior

playback, as the startup bitrate for the new content: “The technology can use the most recent bitrate as the startup bitrate.” (EX1007, 6:1-5.) Hsu’s claim 1 recites “determining a startup bitrate based on the bitrate history, the startup bitrate comprising an initial bitrate for streaming the media content ... wherein the streaming begins at the determined startup bitrate.” The bitrate “used for playback of the first piece of content” in claim 6 of the ’703 patent corresponds directly to the “most recent bitrate” tracked by Hsu’s player, which is used to initialize the new streaming session.

2. Claim 7

- a) *7[pre]/7[a]: The method of claim 6, further comprising: during playback of the new piece of content using the bitrate, detecting whether the new piece of content can maintain playback at the bitrate; and*

120. In my opinion, Hsu discloses this limitation and therefore in combination with Orr renders obvious this limitation.

121. Hsu teaches continuous monitoring of whether the current bitrate can be maintained during playback. Hsu’s adaptive bitrate player “determin[es] bitrate values used at a plurality of different times for providing the streaming media content” during playback, tracking whether the current bitrate is sustainable given available network bandwidth. (EX1007, claim 1). This continuous monitoring is the

mechanism by which the adaptive bitrate system detects that the maintained bitrate is or is not achievable for the new content.

- b)** *7[c]: in response to detecting that the new piece of content cannot continue playback at the bitrate, lowering the bitrate.*

122. In my opinion, Hsu discloses this limitation and therefore in combination with Orr renders obvious this limitation.

123. Hsu teaches downward bitrate adjustment when the current bitrate cannot be sustained. Hsu describes a capping mechanism that accounts for users moving between higher and lower bandwidth environments, and the standard adaptive bitrate mechanism taught by Hsu lowers the bitrate when download throughput is insufficient to sustain the current level: the player “adjusts the quality of a video stream accordingly” when bandwidth weakens (EX1007, 3:62-4:5) and describes the system's response to lower-bandwidth environments as selecting a lower bitrate. (EX1007, 6:16-26.)

- 3. Claim 10: The method of claim 1, wherein the source element initiates the playback of the new piece of content using a same bitrate as the first piece of content and adjusts the bitrate in response to detecting that the new piece of content cannot continue playback at the bitrate.**

124. I note that this claim combines the requirements of claims 6 and 7 and attributes them specifically to the source element. Thus, for the same reasons I discussed above for claims 6 and 7, Hsu discloses this limitation, and it would have

been obvious to incorporate these teachings into Orr's protocol stack (*i.e. source element*).

125. A POSITA integrating Hsu into Orr would be motivated to implement both of these teachings in the protocol stack of Orr's pipeline, the component responsible for obtaining content data from the content source and presenting it for downstream processing. A POSITA would have readily incorporate these teaching into Orr to ensure that a new piece of content played smoothly without quality degradation. The protocol stack is a natural and logical integration point for Hsu's bitrate teachings within Orr's architecture. It is the component that controls how content data is retrieved and the component at which an initial bitrate parameter would be set and at which bitrate adjustments would be made in response to network conditions. A POSITA applying Orr's pipeline reuse architecture to an adaptive HTTP streaming context would understand that the protocol stack in such an implementation manages the adaptive bitrate download session, and would apply Hsu's startup bitrate and adjustment teachings to that component as a matter of ordinary skill.

4. **Claim 16: The system of any one of claim 11, wherein the transitioning to the new piece of content comprises initiating playback of the new piece of content using a bitrate used for**

playback of the first piece of content to maintain the bitrate for playback of the new piece of content.

126. In my opinion, for the reasons I discussed above for claim 6, Orr in view of Hsu renders obvious this limitation.

- 5. Claim 17: The system of claim 16, wherein the operations further comprise: during playback of the new piece of content using the bitrate, detecting whether the new piece of content can maintain playback at the bitrate; and in response to detecting that the new piece of content cannot continue playback at the bitrate, lowering the bitrate.**

127. In my opinion, for the reasons I discussed above for claim 7, Orr in view of Hsu renders obvious this limitation.

- 6. Claim 18: The system of claim 11, wherein the operations further comprise: based on the first piece of content and the new piece of content not being the same content type, deconstructing the playback pipeline by releasing the source element, the demultiplexer, the audio decoder, and the video decoder; and creating a new playback pipeline by acquiring other resources that are applicable to a content type of the new piece of content.**

128. In my opinion, for the reasons I discussed above for claim 8, Orr in view of Hsu renders obvious this limitation.

- 7. Claim 19: The system of claim 11, wherein the source element initiates the playback of the new piece of content using a same bitrate as the first piece of content and adjusts the bitrate in response to detecting that the new piece of content cannot continue playback at the bitrate.**

129. In my opinion, for the reasons I discussed above for claim 10, Orr in view of Hsu renders obvious this limitation.

VIII. Ground 3: Orr in View of Lim Renders Obvious Claims 5 and 15

A. Motivation to Combine

130. In my opinion, a POSITA would have been motivated to modify Orr in view of Lim to incorporate Lim's teachings of prefetching of metadata into Orr, specifically Orr's multimedia manager, and would have had a reasonable expectation of success in doing so.

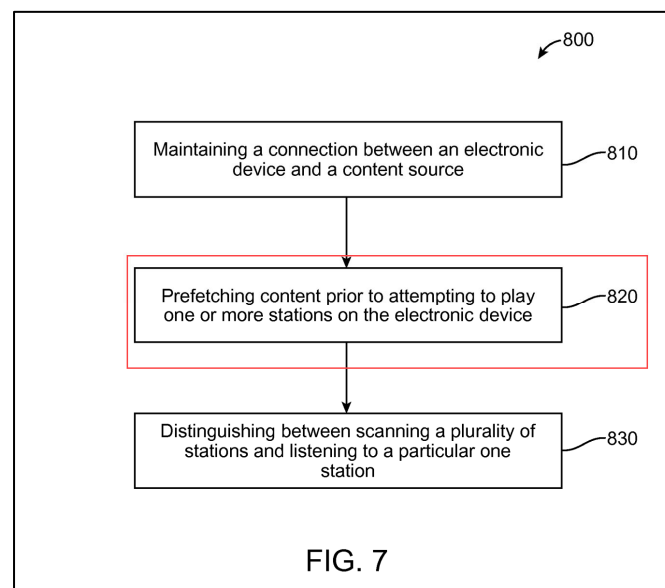
131. A POSITA would be motivated by the fact that Lim's prefetched metadata could supply the content-type information that Orr's format compatibility determination requires. Rather than obtaining this information at the time of the channel switch, a POSITA would have integrated Lim's prefetched metadata to perform Orr's compatibility check instantaneously before receiving the user's selection. This is the straightforward application of prefetching, a well-known technique for reducing access latency, to the specific bottleneck of on-demand content-type retrieval in Orr's channel-switch system.

132. Orr and Lim address sequential and complementary steps in the same content-switch optimization workflow. Orr's channel switch logic performs a format compatibility determination at the time of the channel switch (after the user has made a selection) to decide whether the existing rendering pathway can be reused to reduce latency. (EX1005, ¶[0021]; ¶[0043-44].) To make this determination, Orr's

processing logic must have access to information about the new content's format.

(EX1005, ¶[0026].)

133. Lim teaches prefetching this kind of content information (metadata including type, format, and codec-relevant data) during playback of current content, before the user makes a selection. (EX1008, *e.g.*, 3:51-67; 5:23-39; 15:37-46; FIG. 2; claim 1.) This reduces latency between a user's content selection and the commencement of playback, because "when the user changes the content source (*e.g.*, radio stations, video stations, etc.), the mobile device starts playing without reduced buffering time." (EX1008, 4:1-7.) This is shown below in Figure 7 of Lim:



134. Instead of waiting for Orr's channel switch logic to perform a format compatibility determination at the time of the channel switch (after the user has made a selection), a POSITA would have been motivated to integrate Lim's prefetching

of metadata *before* the user makes a selection to produce faster, higher-quality content transitions between content.

135. Both references are directed to reducing the latency experienced when a user switches between content channels on a playback device, and both are premised on the same insight: that preparing the system in advance of a user's selection reduces the delay after that selection is made. Orr prepares the rendering pathway by evaluating whether existing components can be reused; Lim prepares the content metadata so it is available the moment a selection occurs. A POSITA working on content-switch optimization would recognize these as naturally complementary techniques operating at different points in the same workflow.

136. In my opinion, a POSITA would have reasonably expected the combination to succeed. Lim provides detailed disclosures regarding implementation of prefetching, and it would have been well-within the skill of an ordinary artisan to implement such functionality into the software or hardware of Orr's multimedia manager. Such a modification would only require ordinary programming knowledge.

B. Limitation-By-Limitation Analysis

- 1. Claim 5: The method of claim 2, wherein the detecting the codec information for the new piece of content comprises examining prefetched metadata for the new piece of content.**

137. Claim 5 adds to the codec detection requirement of claim 2 the specific limitation that the codec information is obtained by examining metadata that was prefetched, such as retrieved and stored locally before the user selects the content.

138. As I discussed above for claim 2, Orr teaches detecting of codec and format information. Specifically, Orr's channel switch logic determines the content format of the new channel and assesses whether the existing pipeline elements can handle it.

139. Lim teaches prefetching metadata for candidate next-play content during playback of current content. Lim's claim 1 recites "prefetching, by the electronic device, content prior to attempting to play at least one station on the electronic device," and claim 2 expressly specifies that "prefetching content comprises prefetching preview content clips, cover art, and metadata for all on-deck content for the plurality of stations." (EX1008, claims 1-2.) Lim's specification confirms this teaching: Lim describes "the client 305 efficiently prefetch[ing] the preview clips, cover art, and metadata for all the on-deck songs of all the stations" using a cold start function while current content plays. (EX1008, 9:6-10.) Lim further describes the prefetch manager module 450 implementing a "Prefetch Song"

process that “downloads a song’s preview clip and cover art and stores them in the prefetch inventory 455 and image cache 470,” and a “CheckOnDeckPrefetch” process that triggers a prefetch of the next on-deck song's data while the current song still has sufficient buffer remaining, *i.e.*, while current content is still playing. (EX1008, 12:44-55.) Lim further describes the metadata prefetched for each song as including the “song URL and cover art URL,” and notes that the aggregate station binary downloaded at cold start contains “all of the stations’ first song metadata, preview clip, and cover-art.” (EX1008, 10:27-41; 12:37-42.)

140. In the combination with Orr, Orr’s format compatibility check uses metadata prefetched per Lim’s method. Lim’s prefetch manager module 450 fetches and stores in the prefetch inventory 455 the metadata for candidate next-play content during playback of current content. When the user makes a channel selection, Orr’s processing logic reads from the locally cached prefetch inventory rather than waiting to obtain format information from the incoming stream, enabling the compatibility determination to be made instantaneously upon selection.

2. Claim 15

141. In my opinion, for the reasons I discussed above for claim 15, Orr in view of Lim renders obvious this limitation.

IX. Ground 4: Hervey Renders Obvious Claims 1-4, 8-9, 11-14, 18, and 20

A. Limitation-By-Limitation Analysis

1. Claim 1

- a) *1[pre]: A method performed by a digital receiver, the method comprising:*

142. In my opinion, and to the extent limiting, Hervey discloses the preamble of claim 1.

143. Hervey describes GStreamer as a software framework for implementing playback pipelines on electronic devices that receive and process digital content streams, precisely the function of a *digital receiver*. Specifically, Hervey describes playbin as a “convenience pipeline” that manages uridecodebin, decodebin, and playsink to receive a URI, decode the content, and present it to a sink, respectively. This is the complete sequence of operations performed by a digital receiver. (EX1009, slide 4.)

Playback in GStreamer

- playbin (convenience pipeline)
 - uridecodebin
 - decodebin
 - playsink
- And some non-reusable code

144. A POSITA familiar with GStreamer would understand that the framework described in Hervey is a software framework that is deployed on devices in order to receive and render digital content, and would understand the methods described in Hervey as performed by such a device.

b) *l[a]: receiving a selection of a new piece of content for playback during playback of a first piece of content;*

145. In my opinion, Hervey discloses this limitation.

146. Hervey describes receiving a new stream selection during active playback of a first stream. For example, Hervey describes receiving a GST_EVENT_SELECT_STREAMS event during ongoing playback which “[c]ompare[s] available, active, pending and requested streams.” The reference to “active” streams confirms that the selection is received while first content is playing. (EX1009, slide 28.)

Switching Streams

- GST_EVENT_SELECT_STREAMS
 - Ex : “videol”, “audio2”
 - Compare available, active, pending and requested streams.
 - Need to switch audiol to audio2
 - Put audio2 in pending selection
 - Set idle probe on multiqueue output of audiol
 - Unlinks from decoder
 - Send GST_EVENT_RECONFIGURE to audio2 multiqueue source pad
 - Audio2 multiqueue source pad will use same logic as before

147. Hervey further describes the system receiving a new content input while current content plays: “When EOS reaches all multiqueue slot input, emit ‘about-to-finish,’ [u]sers have a chance to change the input of decodebin3.” (EX1009, slide 32.) A POSITA would understand that the “about-to-finish” flag indicates that the current playing content is about to finish.

Gapless playback in decodebin3

- Can we re-use more elements ?
- Just re-use elements for gapless playback
- When EOS reaches all multiqueue slot input, emit 'about-to-finish'
 - Users have a chance to change the input of decodebin3
 - Earlier than with legacy playbin (EOS on output of decodebin)
- GstParseBin reconfigurable
 - Handles the reconfiguration
 - And we just get new streams... as if it was a demuxer update

c) *1[b]: in response to the receiving of the selection, transitioning to the new piece of content;*

148. In my opinion, Hervey discloses this limitation.

149. Hervey teaches transitioning to the newly selected stream in direct response to the stream selection event. Upon receiving `GST_EVENT_SELECT_STREAMS`, `decodebin3` performs the stream comparison and pipeline reconfiguration steps and then begins delivering the new content through the pipeline: “Send `GST_EVENT_RECONFIGURE` to audio2 multiqueue source pad,” and “Audio2 multiqueue source pad will use same logic as before.” (EX1009, slide 28.)

Switching Streams

- `GST_EVENT_SELECT_STREAMS`
 - Ex : “video1”, “audio2”
 - Compare available, active, pending and requested streams.
 - Need to switch audio1 to audio2
 - Put audio2 in pending selection
 - Set idle probe on multiqueue output of audio1
 - Unlinks from decoder
 - Send `GST_EVENT_RECONFIGURE` to audio2 multiqueue source pad
 - Audio2 multiqueue source pad will use same logic as before

150. The gapless playback slide similarly describes switching to new content upon the “about-to-finish” signal. (EX1009, slide 32.)

Gapless playback in decodebin3

- Can we re-use more elements ?
- Just re-use elements for gapless playback
- When EOS reaches all multiqueue slot input, emit 'about-to-finish'
 - Users have a chance to change the input of decodebin3
 - Earlier than with legacy playbin (EOS on output of decodebin)
- `GstParseBin` reconfigurable
 - Handles the reconfiguration
 - And we just get new streams... as if it was a demuxer update

- d) *1[c]: determining whether the new piece of content is of a same content type as the first piece of content; and*

151. In my opinion, Hervey discloses or renders obvious this limitation.

152. Hervey teaches determining whether a new stream is compatible with the existing pipeline, which a POSITA would understand as determining whether the new content is of the same content type as the current content. The “Switching Streams” slide describes comparing “available, active, pending and requested streams” to determine what action to take. (EX1006, slide 28.)

Switching Streams

- GST_EVENT_SELECT_STREAMS
 - Ex : “video1”, “audio2”
 - Compare available, active, pending and requested streams.
 - Need to switch audio1 to audio2
 - Put audio2 in pending selection
 - Set idle probe on multiqueue output of audio1
 - Unlinks from decoder
 - Send GST_EVENT_RECONFIGURE to audio2 multiqueue source pad
 - Audio2 multiqueue source pad will use same logic as before

153. The “Re-using decoders” slide describes examining the CAPS of the new stream: “STREAM_START / CAPS on multiqueue source pad.” For example, this “[c]hecks if in requested/pending selection,” “[c]heck[s] if there is an output

available to be re-used,” and “[c]heck[s] if decoder can accept CAPS.” (EX1006, slide 29.)

Re-using decoders

- STREAM_START / CAPS on multiqueue source pad
 - Checks if in requested / pending selection
 - Check if there is an output available to be re-used
 - Check if decoder can accept CAPS
 - If so, just link
 - If not, unlink, insert new decoder, link

154. CAPS in GStreamer is the descriptor of stream format and content type. (EX1018, 2 (“Caps (capabilities) are lightweight refcounted objects describing media types.”) A POSITA would understand that checking whether a decoder can accept a stream’s CAPS determines whether the new content is of the same content type as the content currently being decoded. This is because a decoder of one particular type can only decode audio or video streams of that same type. For example, an H.264 decoder can only decode H.264 encoded video streams, or the AAC decoder can only decode ACC encoded audio streams. Thus, by checking “if decoder can accept CAPS,” this checks if the new CAPS contains content that can be decoded by the decoder compatible with the first content, *i.e. has the same content type*.

155. In my opinion, a POSITA would also consider it obvious to determine whether the content is of the same content type when “[c]hecking if decoder can accept CAPS.” As discussed, Hervey teaches linking the stream to the current rendering pipeline “if the decoder can accept” the media type of the new content. A POSITA would be motivated by the fact that a simple way of checking if the decoder can accept the media type of the new content is to compare the media type of the new content with the media type of the old content—because if the capabilities of the old and new content are the same, then the current decoder can “accept CAPS” of the new content. A POSITA would have had a reasonable expectation of success in implementing such a functionality as it would be a simple programming change.

- e) *1[d]: based on the new piece of content being the same content type as the first piece of content, determining whether to preserve at least a portion of a playback pipeline used to play back the first piece of content, the playback pipeline comprising a source element to manage the downloading of content data for the new piece of content, a demultiplexer, an audio decoder, and a video decoder, the preserving of at least the portion of the playback pipeline comprising preserving one or more of the source element, the demultiplexer, the audio decoder, or the video decoder used to obtain the first piece of content for the new piece of content.*

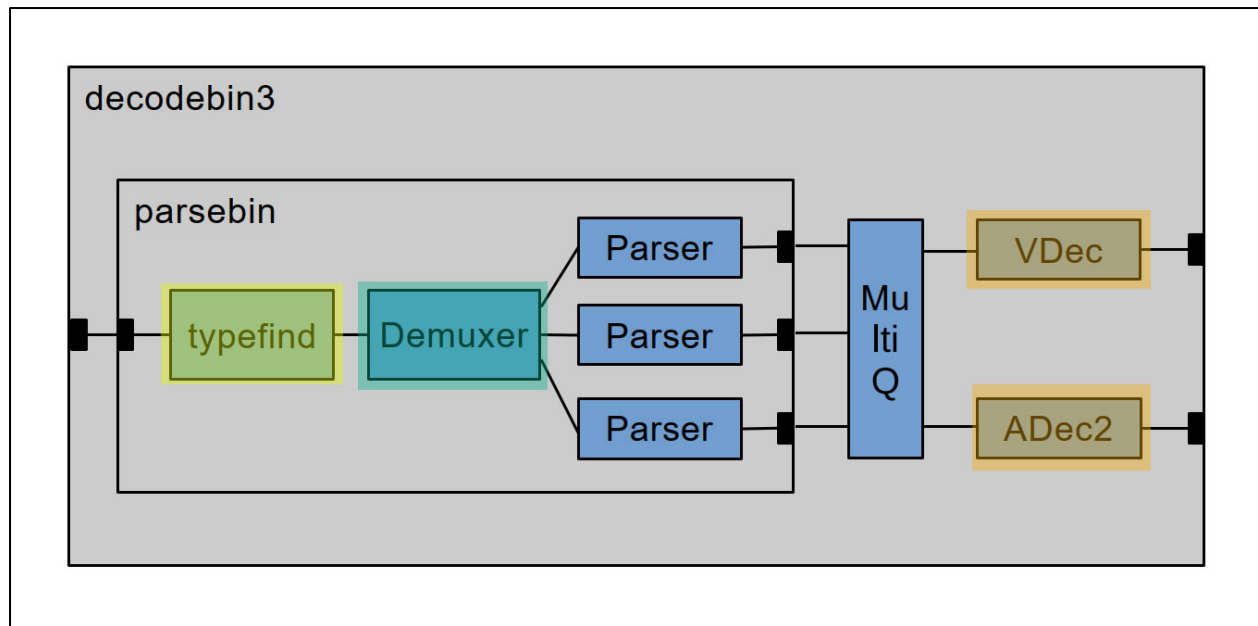
156. In my opinion, Hervey discloses or renders obvious this limitation.

157. For the limitation “***playback pipeline comprising ... video decoder,***” Hevey’s decodebin3 architecture explicitly discloses a ***playback pipeline***

comprising the claimed *source element, demultiplexer, audio decoder, and video decoder*. I discuss each component below.

158. For the “[s]ource element to manage the downloading of content data for the new piece of content,” Hervey teaches that decodebin3 has a “single input sink pad,” *i.e.* a single element for receiving media input. (EX1006, slide 25.) Hervey further teaches that decodebin3 contains the functionalities of “uridecodebin,” which was part of a previous version of decodebin. (EX1006, slide 33.) Hervey teaches that “uridecodebin” performs “buffering/queueing” of “network streams,” which is *managing the downloading of content data for the new piece of content*.

159. It is also my opinion that a POSITA would consider it obvious to combine Hervey’s teachings of the “uridecodebin” with the “typefind” component of decodebin3 as shown in slide 23. A POSITA would be motivated to do so because “uridecodebin” allows for the buffering/queueing of network streams, and such functionality would be highly desirable if not entirely necessary for decodebin3 to parse and render streams. Without such functionality, decodebin3 would not be able to support the input of streams. A POSITA would have had a reasonable expectation of success, because “uridecodebin” was already present functionality in at least earlier versions of decodebin, and therefore implementing in the new version was known and within the skill of an ordinary artisan.



160. For the “demultiplexer,” Hervey discloses that decodebin3’s *playback pipeline* has a “Demuxer,” which is a demultiplexer. (EX1006, slide 24.) The demuxer outputs a “GstStreamCollection,” which is a “collection” of streams, indicating that it demultiplexes the audio and video signals. (EX1006, slide 20.)

161. For the “audio decoder,” Hervey discloses that decodebin3’s *playback pipeline* has an “ADec2,” which a POSITA would understand refers to an *audio decoder*. (EX1006, slide 24.)

162. “A video decoder,” Hervey discloses that decodebin3’s *playback pipeline* has a “VDec,” which a POSITA would understand refers to a *video decoder*. (EX1006, slide 24.)

163. For the limitation “*based on the new piece of content being the same content type as the first piece of content, determining whether to preserve at least*

a portion of a playback pipeline,” Hervey’s “Re-using decoders” slide states:

“Check if decoder can accept CAPS. If so, just link. If not, unlink, insert new decoder, link.” (EX1006, slide 29.)

Re-using decoders

- STREAM_START / CAPS on multiqueue source pad
 - Checks if in requested / pending selection
 - Check if there is an output available to be re-used
 - Check if decoder can accept CAPS
 - If so, just link
 - If not, unlink, insert new decoder, link

164. This is a determination of whether to preserve the existing decoder based on whether the new content type is the same as the old content type. If the decoder can accept the CAPS of the new content, *i.e.* the CAPS of the new content matches the CAPS of the old content, the decoder is preserved (“just link”); if not, it is replaced (“unlink, insert new decoder”).

165. For the limitation *“preserving one or more of the source element, the demultiplexer, the audio decoder, or the video decoder used to obtain the first piece of content for the new piece of content,”* Hervey’s “Demuxer handling” slide teaches that the decoders, the demuxer and source elements may be preserved: “Video stream remains the same, but other streams change.” “No need to

remove/add pad for the video stream.” “No breakage in data stream.” (EX1006, slide 30.)

Demuxer handling

- Decodebin3 backwards compatible with current behaviour
- Demuxers (ex: tsdemux) can be smarter
 - Ex: Video stream remains the same, but other streams change
 - No need to remove/add pad for the video stream
 - No breakage in data stream
- Any element can add/remove streams at any time
 - Ex: [CC] parsing in video parsers

2. Claim 2

- a) *2[pre] / 2[a]: The method of claim 1, wherein the determining whether to preserve comprises: detecting codec information for the new piece of content, the codec information indicating types of decoders needed for the new piece of content; and*

166. In my opinion, Hervey discloses this limitation.

167. Hervey teaches detecting codec information for new content. In GStreamer, a stream’s CAPS describes the media type of that stream. (EX1018, 2 (“Caps (capabilities) are lightweight refcounted objects describing media types.”)). Hervey describes examining the CAPS of new streams at the multiqueue output: “STREAM_START / CAPS on multiqueue source pad.” “Check if in

requested/pending selection.” “Check if there is an output available to be re-used.”

“Check if decoder can accept CAPS.” (EX1006, slide 29.)

Re-using decoders

- **STREAM_START / CAPS on multiqueue source pad**
 - Checks if in requested / pending selection
 - Check if there is an output available to be re-used
 - Check if decoder can accept CAPS
 - If so, just link
 - If not, unlink, insert new decoder, link

168. Because a decoder of one particular media type can only decode audio or video streams of that same type (for example, H.264, AAC), a POSITA would understand that examining a stream’s CAPS to determine decoder compatibility detects codec information in the stream’s CAPS to determine the decoder needed to decode the stream.

- b)** *2[b]: determining whether the audio decoder and the video decoder in the playback pipeline are of the types indicated by the codec information.*

169. In my opinion, Hervey discloses this limitation.

170. The “Re-using decoders” slide describes checking whether the existing decoder can accept the CAPS of the new stream: “Check if decoder can accept CAPS. If so, just link.” (EX1006, slide 29.) This check, comparing the existing decoder’s capabilities against the CAPS (codec requirements) of the new stream, is

the determination recited in this limitation. If the decoder's capabilities match the CAPS, the decoder is "of the types indicated by the codec information." A POSITA would recognize this as a disclosure of this limitation.

Re-using decoders

- STREAM_START / CAPS on multiqueue source pad
 - Checks if in requested / pending selection
 - Check if there is an output available to be re-used
 - Check if decoder can accept CAPS
 - If so, just link
 - If not, unlink, insert new decoder, link

3. Claim 3: The method of claim 2, further comprising: preserving an entirety of the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being the types indicated by the codec information.

171. In my opinion, Hervey discloses this limitation.

172. Hervey expressly teaches full pipeline preservation when the CAPS check succeeds. When "decoder can accept CAPS," the result is "just link." This means that the existing decoder is preserved in place and connected directly to the new stream. (EX1006, slide 29.) Hervey further teaches that, when the video and audio streams are format-compatible, the demuxer and source elements are also preserved: "Video stream remains the same, but other streams change. No need to remove/add pad for the video stream. No breakage in data stream." (EX1006, slide 30.)

Demuxer handling

- Decodebin3 backwards compatible with current behaviour
- Demuxers (ex: tsdemux) can be smarter
 - Ex: Video stream remains the same, but other streams change
 - No need to remove/add pad for the video stream
 - No breakage in data stream
- Any element can add/remove streams at any time
 - Ex: [CC] parsing in video parsers

173. Together, these passages teach full pipeline preservation when the codec types match, including preserving the source element, demuxer, and all decoders.

4. Claim 4

- a) 4[pre]/4[a]: *The method of claim 2, further comprising: releasing the audio decoder and the video decoder in the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being different than the types indicated by the codec information; and*

174. In my opinion, Hervey discloses this limitation.

175. Hervey teaches releasing incompatible decoders. When the CAPS check fails, *e.g.*, when the existing decoder cannot accept the new stream's CAPS, Hervey teaches: "unlink, insert new decoder, link." (EX1006, slide 29.)

Re-using decoders

- STREAM_START / CAPS on multiqueue source pad
 - Checks if in requested / pending selection
 - Check if there is an output available to be re-used
 - Check if decoder can accept CAPS
 - If so, just link
 - If not, unlink, insert new decoder, link

176. The “unlink” step is the release of the existing decoder: it is disconnected from the pipeline. A POSITA would understand “unlink” in this context as equivalent to “releasing” the decoder, as the effect is that the decoder is removed from the active pipeline and its resources are freed.

b) *4[b]: obtaining a new audio decoder and a new video decoder based on the codec information.*

177. In my opinion, Hervey discloses this limitation.

178. Hervey teaches obtaining new decoders when the existing decoders are incompatible. Following the “unlink” step, Hervey teaches “insert new decoder.” A new decoder is instantiated and inserted into the pipeline. (EX1006, slide 29.)

Re-using decoders

- STREAM_START / CAPS on multiqueue source pad
 - Checks if in requested / pending selection
 - Check if there is an output available to be re-used
 - Check if decoder can accept CAPS
 - If so, just link
 - If not, unlink, insert new decoder, link

179. The new decoder is selected based on the CAPS of the new stream: the GstStreamCollection mechanism and GStreamer's auto-plugging logic identify the appropriate decoder type based on the stream's codec descriptor. (See, e.g., EX1006, slides 20-21, 31.) A POSITA would understand that selecting and instantiating a new decoder based on the stream's CAPS is equivalent to “obtaining a new audio decoder and a new video decoder based on the codec information” as recited in this limitation.

5. Claim 8

- a) *8[pre]/8[a]: The method of claim 1, further comprising: based on the first piece of content and the new piece of content not being the same content type, deconstructing the playback pipeline by releasing the source element, the demultiplexer, the audio decoder, and the video decoder; and*

180. In my opinion, Orr discloses or renders obvious this limitation.

181. Hervey teaches full pipeline deconstruction when content types are incompatible through decodebin3's stream switching and decoder management

logic. The “Re-using decoders” slide teaches that when the existing decoder cannot accept the new stream’s CAPS (meaning the content types are incompatible), decodebin3 executes “unlink, insert new decoder, link.”

Re-using decoders

- STREAM_START / CAPS on multiqueue source pad
 - Checks if in requested / pending selection
 - Check if there is an output available to be re-used
 - Check if decoder can accept CAPS
 - If so, just link
 - If not, unlink, insert new decoder, link

182. The “unlink” step releases the existing decoder from the pipeline. For cases of fundamental content-type incompatibility extending beyond the decoders to the demuxer and source elements, Hervey’s “Demuxer handling” slide teaches that when streams change at the demuxer level (meaning the new content requires a different demuxer configuration) decodebin3 handles that reconfiguration by releasing and replacing the affected elements: “Any element can add/remove streams at any time.” (EX1006, slide 30.)

Demuxer handling

- Decodebin3 backwards compatible with current behaviour
- Demuxers (ex: tsdemux) can be smarter
 - Ex: Video stream remains the same, but other streams change
 - No need to remove/add pad for the video stream
 - No breakage in data stream
- Any element can add/remove streams at any time
 - Ex: [CC] parsing in video parsers

183. Hervey further teaches through the GstParseBin mechanism that when input changes require complete reconfiguration, GstParseBin handles the full pipeline reconfiguration, releasing existing elements and acquiring new ones: “GstParseBin reconfigurable; Handles the reconfiguration; And we just get new streams.” (EX1006, slide 32.)

Gapless playback in decodebin3

- Can we re-use more elements ?
- Just re-use elements for gapless playback
- When EOS reaches all multiqueue slot input, emit 'about-to-finish'
 - Users have a chance to change the input of decodebin3
 - Earlier than with legacy playbin (EOS on output of decodebin)
- GstParseBin reconfigurable
 - Handles the reconfiguration
 - And we just get new streams... as if it was a demuxer update

184. In my opinion, A POSITA would also have considered it obvious to release and create a new “sink pad” for media input in the event the new content uses a different protocol or format. A POSITA would understand that different protocols are supported using different APIs. Therefore, if a different protocol or format is encountered, a different API than the current protocol stack would be needed. In creating a new protocol stack by calling the new API, a POSITA would be motivated to release the current sink pad, as it would save computing resources. A POSITA would also have a reasonable expectation of success, as releasing the sink pad is easily implemented using standard programming techniques and well-within the skill of an ordinary artisan..

b) *8[b]: creating a new playback pipeline by acquiring other resources that are applicable to a content type of the new piece of content.*

185. In my opinion, Hervey discloses this limitation.

186. Hervey teaches creating a new pipeline with elements appropriate to the new content type as the natural continuation of the full reconfiguration described in claim 8[a]. Hervey’s GstParseBin reconfiguration mechanism handles this directly: “GstParseBin reconfigurable. Handles the reconfiguration. And we just get new streams.” (EX1006, slide 32.)

Gapless playback in decodebin3

- Can we re-use more elements ?
- Just re-use elements for gapless playback
- When EOS reaches all multiqueue slot input, emit 'about-to-finish'
 - Users have a chance to change the input of decodebin3
 - Earlier than with legacy playbin (EOS on output of decodebin)
- GstParseBin reconfigurable
 - Handles the reconfiguration
 - And we just get new streams... as if it was a demuxer update

187. The phrase “we just get new streams” reflects that GstParseBin, upon reconfiguration, automatically identifies and instantiates the elements needed to process the new content, acquiring resources applicable to the new content type. Hervey further teaches through the GstParseBin description that it “recursively figures out ‘decodable’ elements needed. Demuxers, depayloaders, parsers... but not decoders” (EX1009, slide 25) for any given input stream, and that the output of GstParseBin feeds the multiqueue at which new decoders are selected and instantiated based on the stream’s CAPS. (EX1009, slide 26.)

GstParseBin

- One single input sink pad
- Recursively figures out “decodable” elements needed
 - Demuxers, depayloaders, parsers ... but not decoders
- No queueing
- Creates/Posts GstStreamCollection/GstStream if the element didn't create it
 - All pipelines can get new API support \o/
- Longer term : Reconfigurable
 - On input changes, re-use elements if possible, else switch

decodebin3

- Output of GstParseBin(s) fed to multiqueue
 - Parsed Elementary Streams
 - Multiqueue “slots” (sink + src pad) are typed (audio, video,...)
 - If available slot of proper type, re-use, else create new one
- Selection is done post-multiqueue
 - Only thing that (might) need to be plugged is a decoder
 - By default only expose/decode one stream of each type
 - But you can also expose everything...
 - Or just what you selected via GST_EVENT_SELECT_STREAMS

188. Together these teachings describe a system that, upon encountering a new content type, automatically determines and acquires the source, demuxer, and decoder elements applicable to that content type. A POSITA would understand

decodebin3's auto-plugging and reconfiguration architecture as disclosing this limitation.

189. Further, as I discussed above for limitation 8[a], a POSITA would be motivated to release and create a new sinkpad.

- 6. Claim 9: The method of claim 1, wherein the receiving the selection, transitioning to the new piece of content, determining whether the new piece of content is of a same content type as the first piece of content, and determining whether to preserve is performed by a player of the digital receiver.**

190. In my opinion, Hervey renders obvious this limitation.

191. Hervey's playbin/decodebin3 stack collectively performs all of the operations recited in claim 9 and corresponds to the "player" as described in the '703 patent. Playbin is the top-level player component that manages stream listing and selection and handles content switching: "Uses uridecodebin. Output of uridecodebin to inputselector(s) (to switch streams)," and "[a]llows stream listing/selection." (EX1009, slide 14.)

PlayBin

- Uses uridecodebin
 - Output of uridecodebin to inputselector(s) (to switch streams)
 - Into playsink
- Allows stream listing/selection
- Extra-uri (for subtitle in separate URI)
 - Creates a new parallel uridecodebin
- Gapless support (about-to-finish / next-uri)
 - Prerolls another uridecodebin and switches

192. Playbin receives content selections and manages the transition to new content through its gapless and stream-switching mechanisms. (EX1009, slide 14; “Gapless support (about-to-finish / next-uri). Prerolls another uridecodebin and switches”). The content-type determination and pipeline preservation decision are performed by decodebin3, which operates within and under the management of playbin: “Shifted most of the logic to decodebin3 (stream selection, gapless, multi-input support).” (EX1009, slide 33.) This passage also describes that stream selection and gapless switching logic (including the preservation determination) is part of the integrated player stack managed by playbin.

193. A POSITA would understand the playbin/decodebin3 stack as a unified player component in which playbin manages the top-level pipeline and delegates the stream compatibility and preservation determination to decodebin3. To the extent

any arguable distinction exists between Hervey's distributed playbin/ decodebin3 architecture and the '703 patent's "player" component, a POSITA would understand implementing these functions within an integrated player stack as an obvious design choice that would require no inventive step.

7. Claim 11

a) *11[pre]: A system comprising:*

194. In my opinion, for the reasons I discussed above for limitation 1[pre], Hervey discloses this limitation.

b) *11[a]: one or more hardware processors; and*

195. In my opinion, Hervey discloses this limitation.

196. Hervey expressly references the CPU as a hardware resource consumed by the media playback pipeline. The "Summary" slide identifies reducing "CPU" usage as a primary goal of decodebin3: "Avoid as much unneeded processing as possible (CPU, MEM, I/O, ...)." (EX1009, slide 34.)

Summary

- Avoid as much unneeded processing as possible (CPU, MEM, I/O, ...)
- Re-use as many elements as possible
- Still WIP

197. The “Decodebin2: Stream Switching” slide similarly identifies “Useless CPU/MEM/IO usage” as a problem with the prior decodebin2 architecture that decodebin3 is designed to address. (EX1009, slide 7.)

Decodebin2: Stream Switching

We might switch streams later on ...

We don't want the other stream to be drained completely

Else we will have a gap when switching
multiqueue throttles unlinked streams

But ... we're decoding everything :(

Useless CPU/MEM/IO usage

198. These references to CPU confirm that the playback pipeline described in Hervey executes on a CPU, a hardware processor, and that the operations performed by decodebin3’s stream switching, codec detection, and pipeline reuse logic consume CPU resources. In my opinion, a POSITA would understand these explicit references to CPU usage as disclosing that Hervey's system operates on one or more hardware processors within the meaning of this claim.

- c) *11[b]: a storage device storing instructions that configure the one or more hardware processors to perform operations comprising:*

199. In my opinion, Hervey discloses or renders obvious this limitation.

200. Hervey expressly references memory (MEM) as a hardware resource consumed and managed by the decodebin pipeline: “Avoid as much unneeded processing as possible (CPU, MEM, I/O, ...).” (EX1009, slide 34.)

Summary

- Avoid as much unneeded processing as possible (CPU, MEM, I/O, ...)
- Re-use as many elements as possible
- Still WIP

201. The “Decodebin2: Stream Switching” slide (EX1009, slide 7) and the “Chain pitfalls” slide identifies “Increased memory usage (multiqueue)” and “Increased CPU usage (duplicated elements)” as specific resource problems that decodebin3's design addresses by reducing the number of instantiated pipeline elements. (EX1009, slide 11.)

Chain pitfalls

- New “pending” DecodeGroup
 - Increased memory usage (multiqueue)
 - Increased CPU usage (duplicated elements)
- Input and output of decodebin is no longer fully linked
 - Ex : seek event ending nowhere :(
- Want to just add/remove a stream ?
 - Still need to re-create a new bag of source pads
 - Breaks playback (switch video decoder in GOP)
 - Note: wasn't possible in 0.10 due to SEGMENT limitations

202. These references confirm that decodebin3’s pipeline elements are software components that occupy memory and execute on the CPU. A POSITA would understand that software components of this kind that consume CPU and memory resources are necessarily stored as instructions on a storage device that, when executed by the CPU, configure it to perform the pipeline operations described in Hervey. In addition, storing CPU instructions in memory is expressly disclosed in Orr, as discussed in Ground 1. This establishes that well-known fact CPU instructions are stored in memory and loaded into the CPU registers for execution.

d) *11[c]: receiving a selection of a new piece of content for playback during playback of a first piece of content;*

203. In my opinion, for the reasons I discussed above for limitation 1[a], Hervey discloses this limitation.

- e) *11[d]: in response to the receiving of the selection, transitioning to the new piece of content;*

204. In my opinion, for the reasons I discussed above for limitation 1[b], Hervey discloses this limitation.

- f) *11[e]: determining whether the new piece of content is of a same content type as the first piece of content; and*

205. In my opinion, for the reasons I discussed above for limitation 1[c], Hervey discloses or renders obvious this limitation.

- g) *11[f]: based on the new piece of content being the same content type as the first piece of content, determining whether to preserve at least a portion of a playback pipeline used to play back the first piece of content, the playback pipeline comprising a source element to manage the downloading of content data for the new piece of content, a demultiplexer, an audio decoder, and a video decoder, the preserving of at least the portion of the playback pipeline comprising preserving one or more of the source element, the demultiplexer, the audio decoder, or the video decoder used to obtain the first piece of content for the new piece of content.*

206. In my opinion, for the reasons I discussed above for limitation 1[d], Hervey discloses or renders obvious this limitation.

- 8. **Claim 12: The system of claim 11, wherein the determining whether to preserve comprises: detecting codec information for the new piece of content, the codec information indicating types of decoders needed for the new piece of content; and determining whether the audio decoder and the video**

decoder in the playback pipeline are of the types indicated by the codec information.

207. In my opinion, for the reasons I discussed above for claim 2, Hervey discloses this limitation.

- 9. Claim 13: The system of claim 12, wherein the operations further comprise: preserving an entirety of the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being the types indicated by the codec information.**

208. In my opinion, for the reasons I discussed above for claim 3, Hervey discloses this limitation.

- 10. Claim 14: The system of claim 12, wherein the operations further comprise: releasing the audio decoder and the video decoder in the playback pipeline in response to the audio decoder and the video decoder in the playback pipeline being different than the types indicated by the codec information; and obtaining a new audio decoder and a new video decoder based on the codec information.**

209. In my opinion, for the reasons I discussed above for claim 4, Hervey discloses this limitation.

11. Claim 20

- a)** *20[pre]: A machine-storage medium storing instructions that when executed by one or more hardware processors of a machine, cause the machine to perform operations comprising:*

210. In my opinion, for the reasons I discussed above for limitations 11[a] and 11[b], Hervey discloses this limitation.

- b)** *20[a]: receiving a selection of a new piece of content for playback during playback of a first piece of content;*

211. In my opinion, for the reasons I discussed above for limitation 1[a], Hervey discloses this limitation.

- c)** *20[b]: in response to the receiving of the selection, transitioning to the new piece of content;*

212. In my opinion, for the reasons I discussed above for limitation 1[b], Hervey discloses this limitation.

- d)** *20[c]: determining whether the new piece of content is of a same content type as the first piece of content; and*

213. In my opinion, for the reasons I discussed above for limitation 1[c], Hervey discloses or renders obvious this limitation.

- e)** *20[d]: based on the new piece of content being the same content type as the first piece of content, determining whether to preserve at least a portion of a playback pipeline used to play back the first piece of content, the playback pipeline comprising a source element to manage the downloading of content data for the new piece of content, a demultiplexer, an audio decoder, and a video decoder, the preserving of at least the portion of the playback pipeline comprising preserving one or more of the source element, the demultiplexer, the audio decoder, or the video decoder used to obtain the first piece of content for the new piece of content.*

214. In my opinion, for the reasons I discussed above for limitation 1[d], Hervey discloses or renders obvious this limitation.

X. Ground 5: Orr in View of Hsu Renders Obvious Claims 6-7, 10, 16-17, and 19

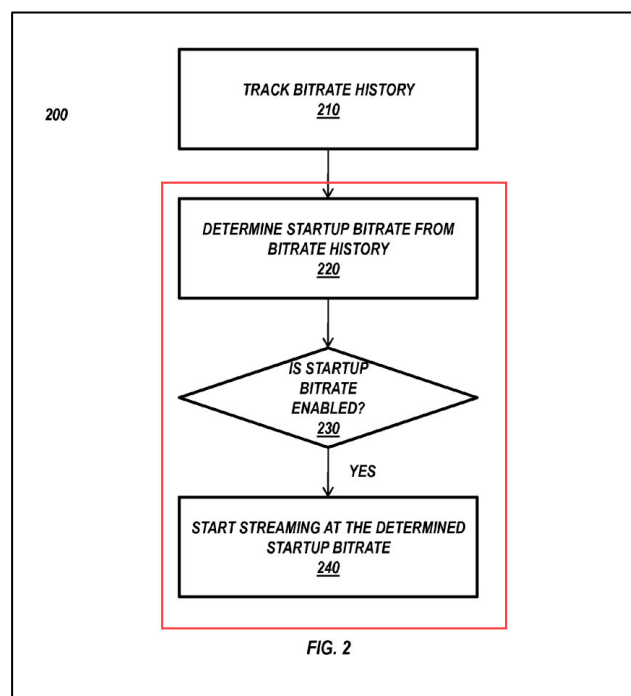
A. Motivation to Combine

215. In my opinion, a POSITA would have been motivated to modify Hervey in view of Hsu to incorporate teachings of bitrate initialization and would have had a reasonable expectation of success.

216. In my proposed combination, Hervey's source element (*i.e.* sink pad) continues to manage the retrieval of content data from the content source. (EX1005, ¶[0033].) Hsu teaches initializing a new content download session at a historically derived bitrate rather than the system default. (EX1007, 6:1-5.) A POSITA would implement Hsu's bitrate initialization teaching in the sink pad of Hervey's pipeline, configuring uridecodebin to begin downloading new content at the previously achieved bitrate. This requires no architectural modification to Hervey and produces the entirely predictable result of faster, higher-quality content transitions. Moreover, the modification is well-within the skills of an ordinary artisan, as bitrate initialization was well-known and straightforward to implement. Neither reference teaches away from the combination. A POSITA would therefore have a reasonable expectation of success.

217. Both Hervey and Hsu address the same fundamental user experience problem: the latency and quality degradation experienced by a user when switching between pieces of content on a streaming device. Hervey describes reusing pipeline

elements to avoid the overhead of teardown and reconstruction when switching streams, which reduced buffering and latency to provide reduced gaps and smoother transitions between content. *See, e.g.*, EX1009 at slide 7 (“We might switch streams [but w]e don’t want the other stream to be drained completely. Else we will have a gap when switching.”); slide 11 (just adding/removing a stream “[b]reaks playback.”); and slide 23 (the new decodebin3 “[r]e-use[s] as many elements as possible,” and “[r]educe[]s buffering.”) As shown in FIG. 2 below, Hsu addresses the bitrate initialization component: eliminating the quality degradation caused by defaulting to the lowest bitrate at the start of each new session. (EX1007, 5:33-39; FIG. 2.) This would therefore motivate a POSITA to further improve the user experience consistent with Orr’s goals.



218. Hervey and Hsu are therefore complementary solutions: Hervey operates at the pipeline and rendering layer, while Hsu operates at the download and bitrate layer. A POSITA designing a comprehensive content-switch optimization system would naturally consider and implement both.

219. A POSITA combining Hervey and Hsu would implement Hsu's bitrate initialization teaching in the source element of Hervey's pipeline. For example, by configuring the source element to initiate the new content download at the bitrate derived from Hsu's historical tracking mechanism, rather than at the system default lowest bitrate. This integration is architecturally natural and technically straightforward, as the source element is the component that manages download sessions and bitrate selection in Hervey's pipeline.

220. In addition, both references are directed to adaptive HTTP streaming contexts. Hervey expressly addresses HLS and DASH pipelines. (EX1009, slide 22.) Meanwhile, Hsu is directed to HTTP-based adaptive bitrate streaming. (EX1007, 3:62-4:27.) A POSITA working in adaptive HTTP streaming would consult both references as directly applicable to the same technical context, and would recognize the bitrate initialization teaching of Hsu as a natural complement to the pipeline reuse teaching of Hervey.

B. Limitation-By-Limitation Analysis

- 1. Claim 6: The method of any one of claim 1, wherein the transitioning to the new piece of content comprises initiating**

**playback of the new piece of content using a bitrate used for
playback of the first piece of content to maintain the bitrate
for playback of the new piece of content.**

221. In my opinion, Hsu discloses this limitation and therefore in combination with Hervey renders obvious this limitation.

222. Hsu address bitrate maintenance across content transitions. Hsu discloses that, upon initiating playback of new content, the player uses a bitrate derived from the history of prior sessions, specifically the bitrate used during prior playback, as the startup bitrate for the new content: “The technology can use the most recent bitrate as the startup bitrate.” (EX1007, 6:1-5.) Hsu’s claim 1 recites “determining a startup bitrate based on the bitrate history, the startup bitrate comprising an initial bitrate for streaming the media content ... wherein the streaming begins at the determined startup bitrate.” The bitrate “used for playback of the first piece of content” in claim 6 of the ’703 patent corresponds directly to the “most recent bitrate” tracked by Hsu’s player, which is used to initialize the new streaming session.

2. Claim 7

- a)** *7[pre]/7[a]: The method of claim 6, further comprising: during playback of the new piece of content using the bitrate, detecting whether the new piece of content can maintain playback at the bitrate; and*

223. In my opinion, Hsu discloses this limitation and therefore in combination with Hervey renders obvious this limitation.

224. Hsu teaches continuous monitoring of whether the current bitrate can be maintained during playback. Hsu's adaptive bitrate player "determin[es] bitrate values used at a plurality of different times for providing the streaming media content" during playback, tracking whether the current bitrate is sustainable given available network bandwidth. (EX1007, claim 1). This continuous monitoring is the mechanism by which the adaptive bitrate system detects that the maintained bitrate is or is not achievable for the new content.

- b)** *7[c]: in response to detecting that the new piece of content cannot continue playback at the bitrate, lowering the bitrate.*

225. In my opinion, Hsu discloses this limitation and therefore in combination with Hervey renders obvious this limitation.

226. Hsu teaches downward bitrate adjustment when the current bitrate cannot be sustained. Hsu describes a capping mechanism that accounts for users moving between higher and lower bandwidth environments, and the standard adaptive bitrate mechanism taught by Hsu lowers the bitrate when download throughput is insufficient to sustain the current level: the player "adjusts the quality of a video stream accordingly" when bandwidth weakens (EX1007, 3:62-4:5) and describes the system's response to lower-bandwidth environments as selecting a lower bitrate. (EX1007, 6:16-26.)

- 3. Claim 10: The method of claim 1, wherein the source element initiates the playback of the new piece of content using a same**

bitrate as the first piece of content and adjusts the bitrate in response to detecting that the new piece of content cannot continue playback at the bitrate.

227. I note that this claim combines the requirements of claims 6 and 7 and attributes them specifically to the source element. Thus, for the same reasons I discussed above for claims 6 and 7, Hsu discloses this limitation, and it would have been obvious to incorporate these teachings into Hervey's source element.

228. A POSITA integrating Hsu into Hervey would be motivated to implement both of these teachings in Hervey's source element, the component responsible for obtaining content data from the content source and presenting it for downstream processing. A POSITA would have readily incorporate these teaching into Hervey to ensure that a new piece of content played smoothly without quality degradation. The protocol stack is a natural and logical integration point for Hsu's bitrate teachings within Hervey's architecture. It is the component that controls how content data is retrieved and the component at which an initial bitrate parameter would be set and at which bitrate adjustments would be made in response to network conditions. A POSITA applying Hervey's pipeline reuse architecture to an adaptive HTTP streaming context would understand that the protocol stack in such an implementation manages the adaptive bitrate download session, and would apply Hsu's startup bitrate and adjustment teachings to that component as a matter of ordinary skill.

4. **Claim 16: The system of any one of claim 11, wherein the transitioning to the new piece of content comprises initiating playback of the new piece of content using a bitrate used for playback of the first piece of content to maintain the bitrate for playback of the new piece of content.**

229. In my opinion, for the reasons I discussed above for claim 6, Hervey in view of Hsu renders obvious this limitation.

5. **Claim 17: The system of claim 16, wherein the operations further comprise: during playback of the new piece of content using the bitrate, detecting whether the new piece of content can maintain playback at the bitrate; and in response to detecting that the new piece of content cannot continue playback at the bitrate, lowering the bitrate.**

230. In my opinion, for the reasons I discussed above for claim 7, Hervey in view of Hsu renders obvious this limitation.

6. **Claim 18: The system of claim 11, wherein the operations further comprise: based on the first piece of content and the new piece of content not being the same content type, deconstructing the playback pipeline by releasing the source element, the demultiplexer, the audio decoder, and the video decoder; and creating a new playback pipeline by acquiring other resources that are applicable to a content type of the new piece of content.**

231. In my opinion, for the reasons I discussed above for claim 8, Hervey in view of Hsu renders obvious this limitation.

7. **Claim 19: The system of claim 11, wherein the source element initiates the playback of the new piece of content using a same bitrate as the first piece of content and adjusts**

the bitrate in response to detecting that the new piece of content cannot continue playback at the bitrate.

232. In my opinion, for the reasons I discussed above for claim 10, Hervey in view of Hsu renders obvious this limitation.

XI. Ground 3: Hervey in View of Lim Renders Obvious Claims 5 and 15

A. Motivation to Combine

233. A POSITA would have been motivated to integrate Lim's prefetching of metadata into Hervey, and would have had a reasonable expectation of success in doing so.

234. A POSITA would be motivated by the fact that Lim's prefetched metadata could supply the stream and codec information that might not be available when Hervey switches to new HLS or DASH content. Rather than obtaining this information at the time of switching to HLS or DASH content, a POSITA would have integrated Lim's prefetched metadata to obtain the stream and codec information before switching. This is the straightforward application of prefetching, a well-known technique for reducing access latency, to the specific bottleneck of on-demand content-type retrieval in Hervey's stream-switch system. Hervey and Lim are both motivated by the same user-facing problem: the delay experienced when a user switches from one piece of content to another on a playback device. Hervey describes reusing pipeline elements to avoid the overhead of teardown and reconstruction when switching streams, which reduces buffering and latency to

provide reduced gaps and smoother transitions between content. *See, e.g.*, EX1009 at slide 7 (“We might switch streams [but w]e don’t want the other stream to be drained completely. Else we will have a gap when switching.”); slide 11 (just adding/removing a stream “[b]reaks playback.”); and slide 23 (the new decodebin3 “[r]e-use[s] as many elements as possible,” and “[r]educe[]s buffering.”) Lim is similarly directed to reducing the delay between a user’s content selection and the commencement of playback: “when the user changes the content source (e.g., radio stations, video stations, etc.), the mobile device starts playing without reduced buffering time.” (EX1008, 4:1-7.) Both references thus share the same design objective—minimizing perceived latency at content switch time—and a POSITA working on that problem would naturally consider both as relevant to the same design challenge, and be motivated by Lim’s teachings to further Hervey’s goals.

235. Hervey identifies a limitation in its stream selection architecture that Lim’s prefetching teaching directly addresses. The “Pitfalls of Stream Selection” slide (EX1009, slide 15) identifies the potential absence of pads (*e.g.*, HLS/DASH) as a problem. That is, when switching to new HLS or DASH content, stream and codec information may not yet be available from source pads at the time of the switch, because the new stream has not yet been opened. Here, the prefetching the metadata for candidate next-play content, as taught by Lim, provides direct value. If the codec and stream information for the new content were already locally available

from a prefetch operation performed while prior content was playing, Hervey's stream selection logic could make its compatibility determination immediately without waiting for source pads to be created. (*See, e.g.*, EX1009, slide 27 ("Startup example: ... Do a pre-emptive decision of which streams to use.") A POSITA reading both references together would immediately recognize that Lim's prefetching architecture fills an information gap that Hervey itself identifies.

236. Hervey and Lim are analogous art as they are both directed to processing video streams.

237. A POSITA would have reasonably expected the combination to succeed. Lim provides detailed disclosures regarding implementation of prefetching, and it would have been well-within the skill of an ordinary artisan to implement such functionality into the software or hardware of Hervey's decodebin3. Such a modification would only require ordinary programming knowledge.

B. Limitation-By-Limitation Analysis

1. Claim 5: The method of claim 2, wherein the detecting the codec information for the new piece of content comprises examining prefetched metadata for the new piece of content.

238. Claim 5 adds to the codec detection requirement of claim 2 the specific limitation that the codec information is obtained by examining metadata that was prefetched, such as retrieved and stored locally before the user selects the content.

239. As I discussed above for claim 2, Hervey teaches detecting of codec and format information. Specifically, Hervey teaches examining the CAPS of the new stream to determine decoder compatibility. (EX1009, slide 29.) Hervey teaches obtaining this CAPS information at the time the stream data arrives at the multiqueue output, rather than from prefetched metadata.

240. Lim teaches prefetching metadata for candidate next-play content during playback of current content. Lim's claim 1 recites "prefetching, by the electronic device, content prior to attempting to play at least one station on the electronic device," and claim 2 expressly specifies that "prefetching content comprises prefetching preview content clips, cover art, and metadata for all on-deck content for the plurality of stations." (EX1008, claims 1-2.) Lim's specification confirms this teaching: Lim describes "the client 305 efficiently prefetch[ing] the preview clips, cover art, and metadata for all the on-deck songs of all the stations" using a cold start function while current content plays. (EX1008, 9:6-10.) Lim further describes the prefetch manager module 450 implementing a "Prefetch Song" process that "downloads a song's preview clip and cover art and stores them in the prefetch inventory 455 and image cache 470," and a "CheckOnDeckPrefetch" process that triggers a prefetch of the next on-deck song's data while the current song still has sufficient buffer remaining — *i.e.*, while current content is still playing. (EX1008, 12:44-55.) Lim further describes the metadata prefetched for each song as

including the “song URL and cover art URL,” and notes that the aggregate station binary downloaded at cold start contains “all of the stations’ first song metadata, preview clip, and cover-art.” (EX1008, 10:27-41; 12:37-42.)

241. A POSITA combining Hervey and Lim would implement Hervey’s codec detection step of claim 2 using metadata prefetched per Lim. Rather than waiting for stream data to arrive at the multiqueue output after the user has made a selection, the system would read the codec information from the locally cached metadata fetched by Lim’s prefetch manager during playback of the prior content.

242. This is a straightforward integration. Lim’s prefetched metadata provides the input, and Hervey’s CAPS check provides the processing and produces the predictable result of allowing the pipeline reuse determination to be made immediately upon receiving the user’s selection.

2. Claim 15

243. In my opinion, for the reasons I discussed above for claim 15, Hervey in view of Lim renders obvious this limitation.

XII. Conclusion

244. In signing this declaration, I recognize that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the

IPR of U.S. Patent No. 12,231,703

Decl. of Dr. Immanuel Freedman

United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

* * *

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on the information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Respectfully submitted,

Date: April 3rd, 2026


Immanuel Freedman, Ph.D.