IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of:	James M. Barton, et al..		
U.S. Patent No.:	6,233,389	Attorney Docket No.:	39843-0037IP1
Issue Date:	May 15, 2001		
Appl. Serial No.:	09/126,071		
Filing Date:	July 30, 1998		
Title:	Multimedia Time Warping System		

DECLARATION OF JOHN MICHAEL STRAWN, PhD

TABLE OF CONTENTS

TABLE OF FIGURES

I, John Michael Strawn, Ph.D., of Larkspur, California, declare that:

I. QUALIFICATIONS AND BACKGROUND INFORMATION

1. I am currently an independent consultant working under the aegis of my corporation S Systems Inc. A copy of my curriculum vitae, which describes in further detail my qualifications, employment history, honors, patent, awards, professional associations, presentations, and publications is attached hereto.

2. My formal education includes a Bachelor's degree from Oberlin College in 1973. As a Fulbright scholar in Berlin, I attended lectures and seminars in German at the Free University and Technical University Berlin from 1973-1975. I earned a Ph.D. degree from Stanford in 1985, with my doctoral dissertation focusing on signal processing for analyzing digital audio. As part of that work, I implemented streaming audio recording and playback in real time on a mainframe computer using, for example, specially formatted hard disks that operated in a drive the size of a washing machine, long before the compact disc was invented.

3. With regard to the subject matter of this proceeding, I have extensive experience in streaming and related technology. I have studied analog and digital circuitry, analog and digital hardware, computer architecture, processor architecture, high-level language programming including object-oriented programming in languages such as C++ and Java, assembly language programming, digital signal processing, cybernetics, information theory,

compression (especially audio but also data, image, and video), television transmission formats, networking, user interface design, user interface implementation, and client/server interactions.

4. In addition, I have over 45 years involvement in software, digital media, digital signal processing, networking, and processor architecture. Working in those areas, I have been an employee, a manager of a team of other Ph.D.s, and an independent software consultant in signal processing specializing in high-level languages and assembly language. My specialties have included compression and decompression of media, streaming media, the Fourier transform, and the discrete cosine transform used in audio compression, JPEG, and MPEG video. Implementing buffering for streaming media has been the backbone of many of my consulting projects, such as for DTS or Verance.

5. Throughout my career, I have received a variety of awards including the Fulbright scholarship mentioned above and a grant from the IBM Thomas Watson Foundation to work in Europe and Japan. I was named Fellow of the Audio Engineering Society.

6. I have made extensive contributions to the practice of assembly language programming for real-time processing and digital signal processing. My work on the NeXT machine served as a tutorial for other programmers. I have

held seminars at industry gatherings teaching my programming methodology as well as compression to others practicing in the field.

7. In writing this Declaration, I have considered the following: my own knowledge and experience, including my work experience in the fields of audio and video streaming; my experience in teaching those subjects; and my experience in working with others involved in those fields. In addition, I have analyzed the following publications and materials, in addition to other materials I cite in my Declaration:

- U.S. Patent No. 6,233,389 (Exhibit SE1001), and its accompanying prosecution history (Exhibit SE1002);
- U.S. Patent No. 5,557,724 to Sampat et al. ("Sampat", Exhibit SE1004);
- U.S. Patent No. 5,710,895 to Gerber et al. ("Gerber", Exhibit SE1005);
- Sound Blaster Pro User Reference Manual (1991) ("SoundBlaster", Exhibit SE1006);
- Programmer's Guide, Microsoft Video for Windows Development Kit (February 1993) ("Video for Windows", Exhibit SE1007);
- U.S. Patent No. 5,546,103 to Rhodes et al. ("Rhodes", Exhibit SE1008).

- Concise Oxford Dictionary of Current English (1990) (Exhibit SE1009);
- Webster's New World Dictionary of Computer Terms (1988) (Exhibit SE1010);
- Claim Construction Order, TiVo Inc. v. Echostar Communications Corp., et al., 2:04-cv-00001 (8/18/2005) (Exhibit SE1011);
- Claim Construction Order, TiVo, Inc. v. AT&T Inc., et al., 2:09-cv-00259 (10/13/2011) (Exhibit SE1012);
- Claim Construction Order, TiVo, Inc. v. Verizon Comm'n, Inc. et al., 2:09-cv-00257 (3/12/2012) (Exhibit SE1013);
- Memorandum Opinion and Order, Motorola Mobility, Inc. et al. v. TiVo, Inc., 5:11-cv-00053 (12/06/2012) (Exhibit SE1014);
- Exhibit B, Preliminary Infringement Claim Chart for U.S. Pat. No. 6,233,389, Samsung Mobile Devices ("Infringement Contentions", Exhibit SE1015);
- Prosecution History of Ex Parte Reexamination of claims 1, 3-5, 15-18, 20-25, 32, 34-36, 46-49, and 51-55 of the '389 patent (Serial No. 90/007750) ("First Reexam", Exhibit SE1016);

- Prosecution History of Ex Parte Reexamination of claims 31 and 61 of the ’389 patent (Serial No. 90/009329) (“Second Reexam”, Exhibit SE1017);
- Bescos, Jesus et al., From Multimedia Stream Models to GUI Generation (1997) (“Bescos”, Exhibit SE1018);
- Screen capture of Amazon.com listing for Sound Blaster Pro User Reference Manual (accessed July 8, 2016) (Exhibit SE1019);
- Musser, John, A Multimedia Class Library for Windows, Dr. Dobb’s Journal (July 1993) (“Musser”, Exhibit SE1020); and
- Adams, Eric J., High Noon: Big Players Ready for Video Showdown, MacWEEK (Dec. 14, 1992) (“Adams”, Exhibit SE1021).

8. Each of these foregoing references (not including the legal documents or patents) were published in publications or libraries with which I am familiar, and which would have been available to and disseminated to members of the general technical community prior to July of 1998.

9. Although this Declaration refers to selected portions of the cited references for the sake of brevity, it should be understood that these are examples, and that one of ordinary skill in the art would have viewed the references cited herein in their entirety and in combination with other references cited herein or

cited within the references themselves. The references used in this Declaration, therefore, should be viewed as being incorporated herein in their entirety.

10. I am not, and never was, an employee of the Petitioner in this proceeding, Samsung Electronics Co., Ltd. and Samsung Electronics America, Inc. I have been engaged in the present matter to provide my independent analysis of the issues raised in the petition for *inter partes* review of the '389 patent. I received no compensation for this Declaration beyond my normal hourly compensation based on my time actually spent studying the matter, and I will not receive any added compensation based on the outcome of this *inter partes* review of the '389 patent.

II. LEGAL PRINCIPLES

A. Anticipation

11. I have been informed that a patent claim is invalid as anticipated under 35 U.S.C. § 102 if each and every element of a claim, as properly construed, is found either explicitly or inherently in a single prior art reference. Under the principles of inherency, if the prior art necessarily functions in accordance with, or includes the claimed limitations, it anticipates.

12. I have been informed that a claim is invalid under 35 U.S.C. § 102(a) if the claimed invention was known or used by others in the U.S., or was patented or published anywhere, before the applicant's invention. I further have been

informed that a claim is invalid under 35 U.S.C. § 102(b) if the invention was patented or published anywhere, or was in public use, on sale, or offered for sale in this country, more than one year prior to the filing date of the patent application. And a claim is invalid, as I have been informed, under 35 U.S.C. § 102(e), if an invention described by that claim was described in a U.S. patent granted on an application for a patent by another that was filed in the U.S. before the date of invention for such a claim.

B. Obviousness

13. I have been informed that a patent claim is invalid as "obvious" under 35 U.S.C. § 103 in light of one or more prior art references if it would have been obvious to a person of ordinary skill in the art at the time of the invention of the '389 patent ("POSITA"), taking into account (1) the scope and content of the prior art, (2) the differences between the prior art and the claims, (3) the level of ordinary skill in the art, and (4) any so called "secondary considerations" of non-obviousness, which include: (i) "long felt need" for the claimed invention, (ii) commercial success attributable to the claimed invention, (iii) unexpected results of the claimed invention, and (iv) "copying" of the claimed invention by others. For purposes of my analysis, and because I know of no indication from the patent owner or others to the contrary, I have applied a date of July 30, 1998, as the date

of invention in my analyses, although in many cases the same analysis would hold true even at a time earlier than July 30, 1998.

14. I have been informed that a claim can be obvious in light of a single prior art reference or multiple prior art references. To be obvious in light of a single prior art reference or multiple prior art references, there must be a reason to modify the single prior art reference, or combine two or more references, in order to achieve the claimed invention. This reason may come from a teaching, suggestion, or motivation to combine, or may come from the reference or references themselves, the knowledge or "common sense" of one skilled in the art, or from the nature of the problem to be solved, and may be explicit or implicit from the prior art as a whole. I have been informed that the combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results. I also understand it is improper to rely on hindsight in making the obviousness determination.

III. OVERVIEW OF CONCLUSIONS FORMED

15. This expert Declaration explains the conclusions that I have formed based on my analysis. To summarize those conclusions:

- Based upon my knowledge and experience and my review of the prior art publications listed above, I believe that claims 31 and 61 of the '389 patent are anticipated by Sampat (client-side).

- Based upon my knowledge and experience and my review of the prior art publications listed above, I believe that claims 31 and 61 of the ’389 patent are rendered obvious by Sampat (client-side) in view of Microsoft Video for Windows and SoundBlaster.
- Based upon my knowledge and experience and my review of the prior art publications listed above, I believe that claims 31 and 61 of the ’389 patent are anticipated by Sampat (server-side).
- Based upon my knowledge and experience and my review of the prior art publications listed above, I believe that claims 31 and 61 of the ’389 patent are rendered obvious by Sampat (server-side) in view of Microsoft Video for Windows, SoundBlaster, and Gerber.

IV. BACKGROUND KNOWLEDGE ONE OF SKILL IN THE ART WOULD HAVE HAD PRIOR TO THE PRIORITY DATE OF THE ’389 PATENT

16. The technology in the ’389 patent at issue generally relates to streaming of audio and video data. Prior to the filing date of the ’389 patent, there existed products, publications, and patents that implemented or described functionality claimed in the ’389 patent. Thus, the methodology of the ’389 patent was known in the prior art. Further, to the extent there was any problem to be solved in the ’389 patent, it had already been solved in prior art systems before the filing date of the ’389 patent.

A. Overview of the ’389 Patent

17. The ’389 patent’s disclosure “relates to the real time capture, storage, and display of television broadcast signals.” [SE1001 (the ’389 patent), 1:6-9.] Figure 1 of the ’389 patent provides a “high level view” of the ’389 patent’s system. [SE1001, 2:44-45.] I have reproduced Figure 1 below for clarity.

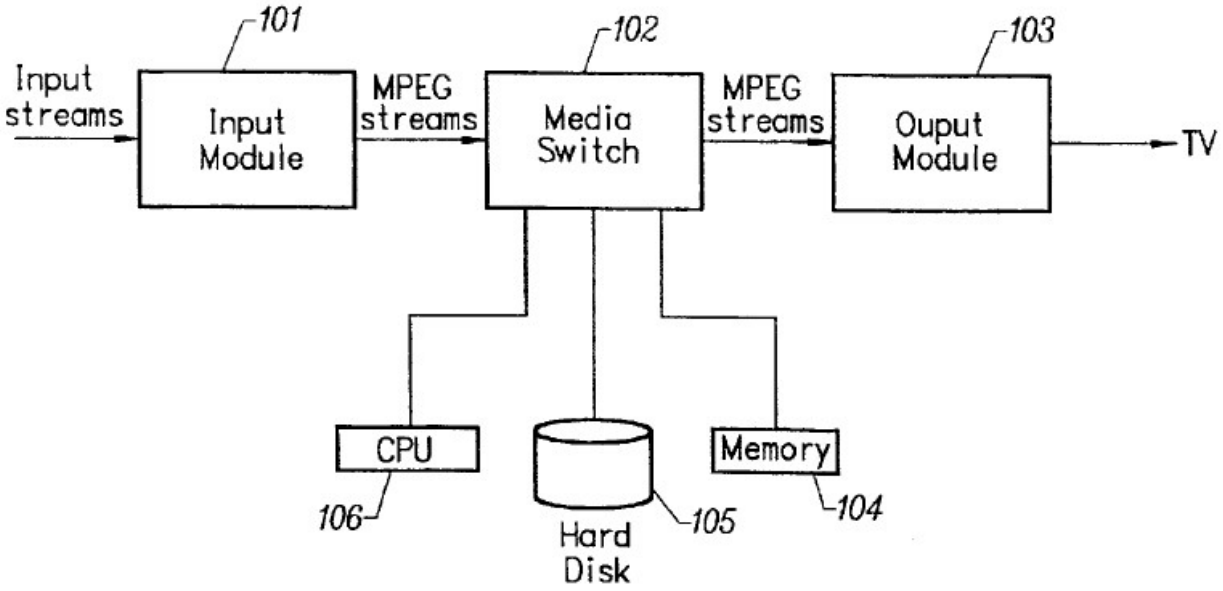

Figure 1. [SE1001, ’389 Patent FIG. 1.]

18. As shown in Figure 1, Input Module 101 receives an input stream (such as an analog television signal), converts the input stream into a digital MPEG format, and outputs a digital MPEG stream to Media Switch 102. [SE1001, 2:10-14, 3:30-65.] Downstream of Input Module 101 is Media Switch 102.

19. Media Switch 102 “parses the stream looking for MPEG distinguished events including the start of video, audio or private data segments.” [SE1001, 5:3-

6.] When video or audio segments are found, Media Switch 102 indexes the segments in memory 104 and stores the segments in storage device 105. [SE1001, 5:6 to 6:7.]

20. Downstream of Media Switch 102 is Output Module 103. Output Module 103 reads the stored digital segments from storage device 105, decodes the segments into an analog signal, and outputs the analog signal. [SE1001, 4:5-9.]

21. Within the high level framework discussed above, claims 31 and 61 are directed to operations that control movement of data through the '389 patent's system. The operations are performed by three conceptual components, illustrated in Figure 8 below as "Sources," "Transforms," and "Sinks."

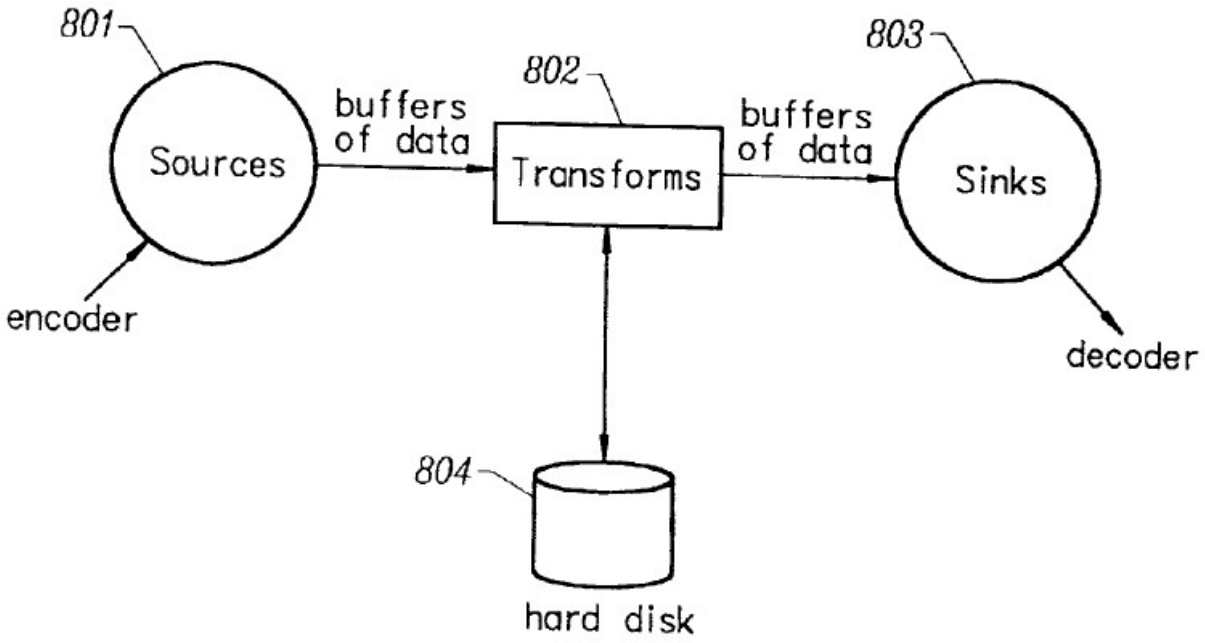

Figure 2. [SE1001, '389 Patent FIG. 8.]

22. Sources 801 accept digital data from an encoder and package the data in buffers acquired from transforms 802. Sources 801 then push the buffers to transform 802. [SE1001, 7:58-61.] Transforms 802 write the buffers to a file on the storage medium or hard disk 804. At a later time, transforms 802 pull out the buffers from hard disk 804 and sequence them with the stream - i.e., an operation the '389 patent describes as performing a temporal transform. [SE1001, 8:3-8.] Sinks 803 then take the buffers from transforms 802 and send, to a decoder, digital video/audio data from the buffers.

23. The '389 patent describes the use of object-oriented programming language (such as the C++ programming language) to implement the program logic illustrated in '389 patent Figure 8 above. As shown in '389 patent Figure 9 below, the '389 patent describes the use of a "source object" 901, a "transform object" 902, and a "sink object" 903, which correspond to sources 801, transforms 802, and sinks 803. [SE1001, 8:9-18, FIG. 9.] A "control object" 917 accepts user commands. [SE1001, 9:25-32.] I have reproduced Figure 9, below, and annotated it with colors and labels to show the source, sink, transform, and control objects, as well as the physical data source, storage device, and decoder.

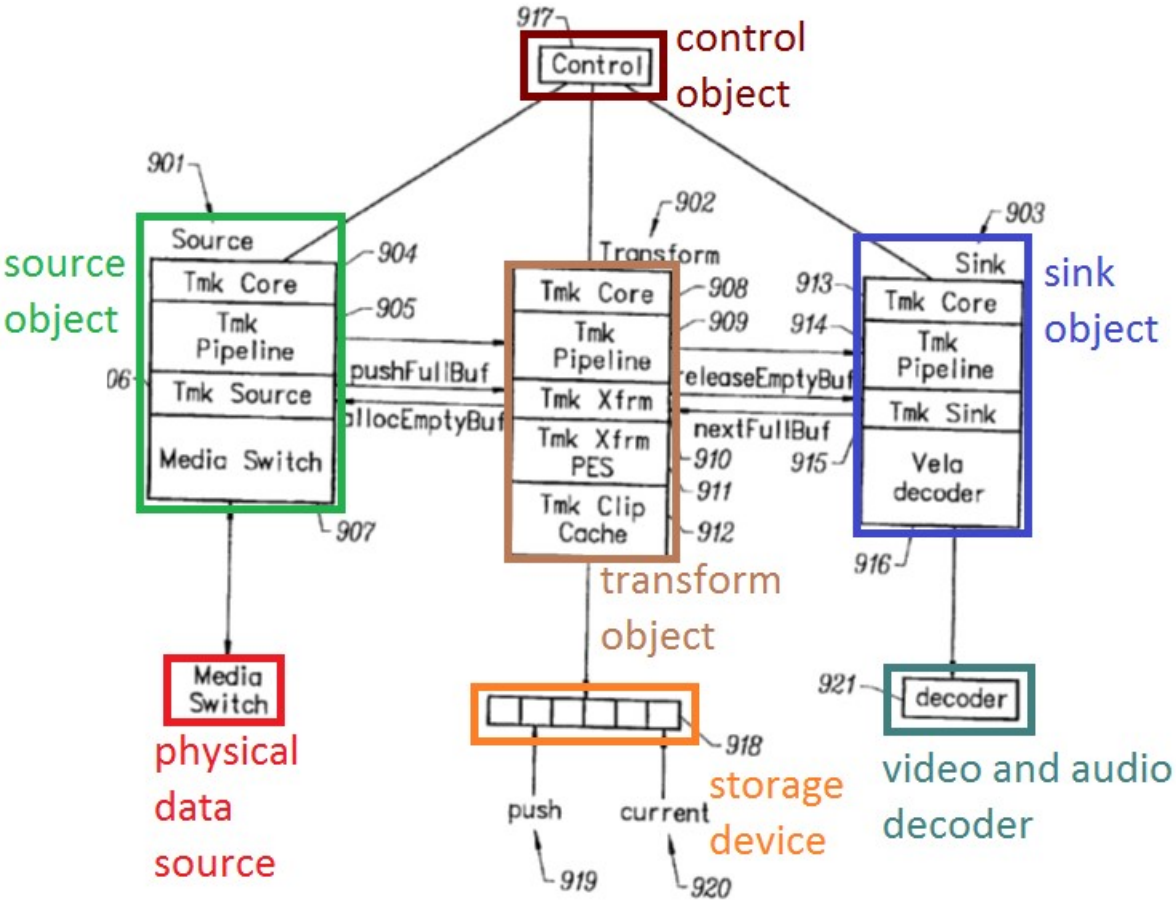

Figure 3. [SE1001, ’389 Patent FIG. 9 (annotated)]

24. The source, transform, and sink objects operate in conjunction with the components described above in ’389 patent Figure 1. For example, the source object “takes data out of a physical data source, such as the Media Switch.” [SE1001, 8:43-45.] The ’389 patent explains that the source object calls the transform object for a buffer to fill. [SE1001, 8:45-48.] The transform object provides the empty buffer to the source object and then takes the full buffer from the source object and stores it on hard disk or storage device 105 in Figure l.

[SE1001, 9:2-9.] The sink object calls the transform object for a full buffer and then sends the digital data to a decoder in Output Module 103 of Figure 1. [SE1001, 9:10-16.] It then releases the empty buffer to the transform object for use again by the source object. [SE1001, 8:55-59.]

25. Under this system, the source object waits for the transform object to provide an empty buffer. Similarly, the sink object also waits for the transform object to provide a full buffer. According to the '389 patent, "[t]his means that the pipeline is self-regulating; it has automatic flow control." [SE1001, 8:48-49.]

B. Background Prior Art - Sampat

26. A review of other relevant literature available at the time shows that the idea of a pipeline being "self-regulating" or exhibiting "automatic flow control" was well known in the technical community by 1998. For example, Sampat discloses a system for processing data streams that includes automatic flow control. Figure 1 of Sampat shows that system at a high level, with input devices 108, 110, and 112, a server 102, and clients 104.

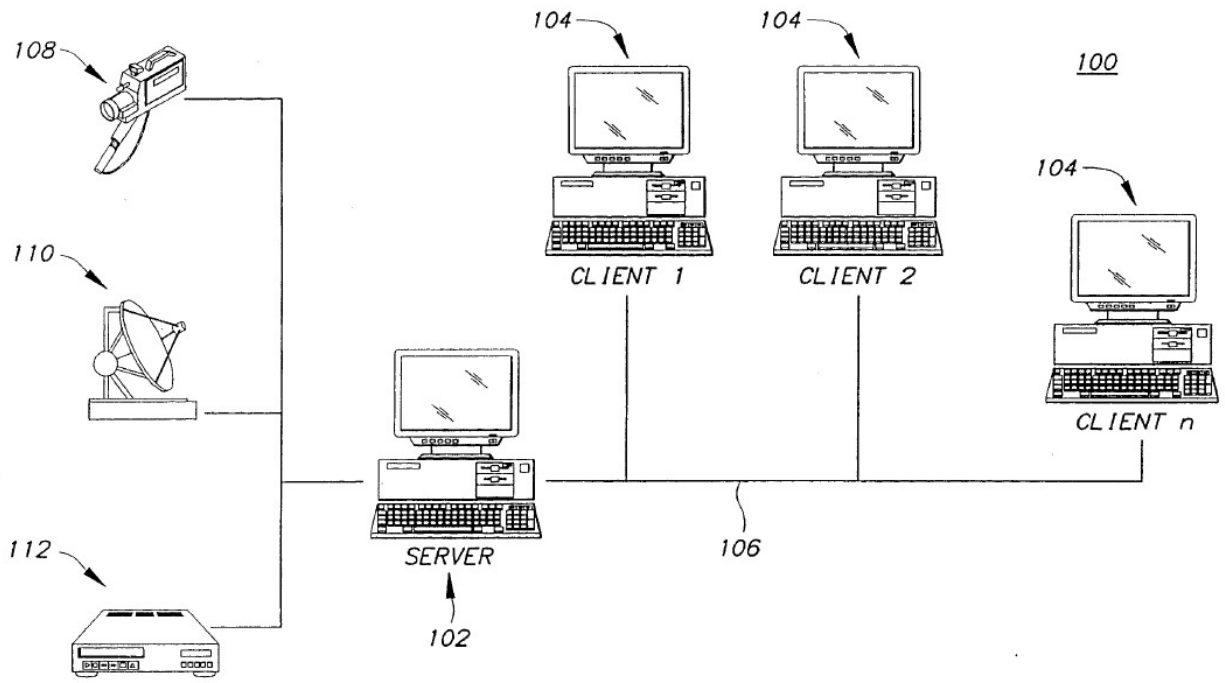

FIG. 1

Figure 4. [SE1004, Sampat FIG. 1.]

27. Sampat discloses source, sink, and transform objects in both its server 102 and its clients 104. For example, I have reproduced and annotated Figure 16 of Sampat (below) to show the source object colored green (source MSPs), the sink object colored blue (sink MSPs), the transform object colored lighter brown (media services manager)

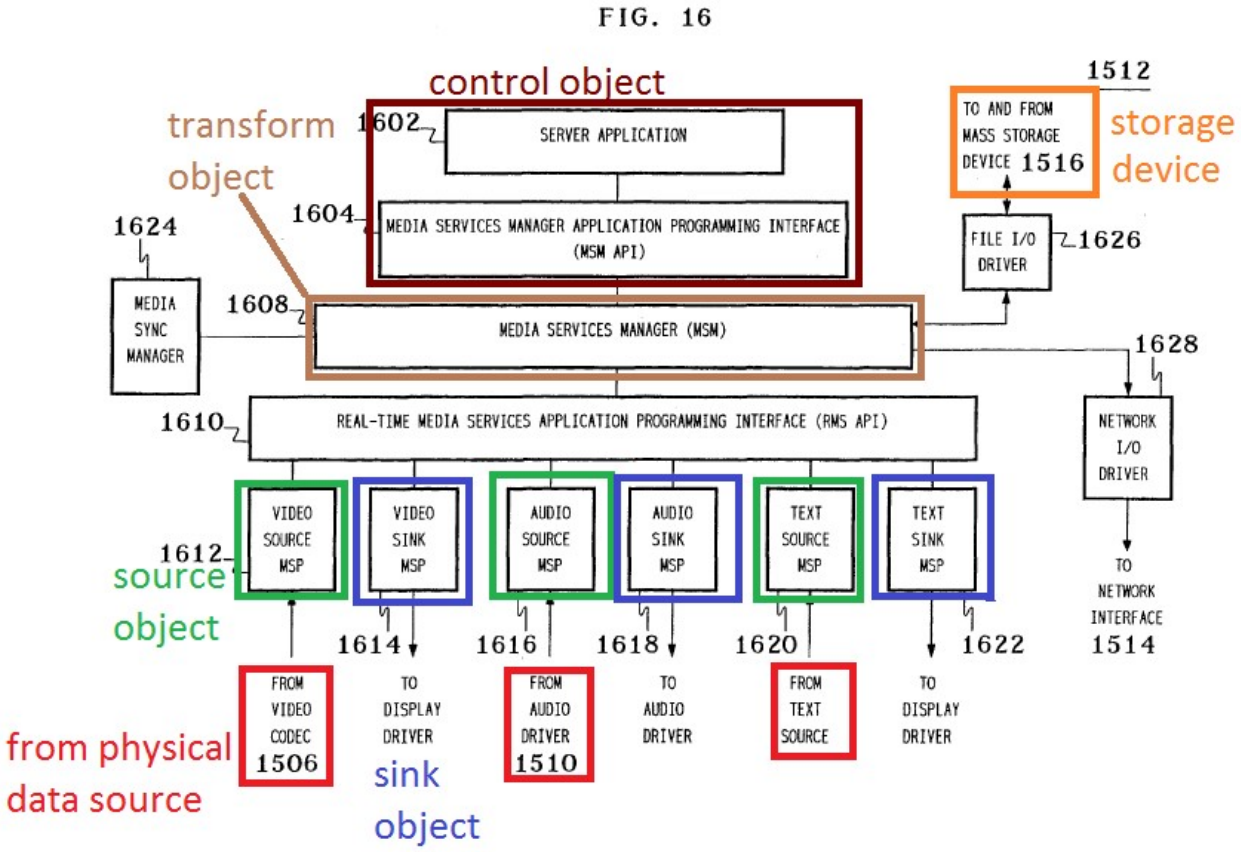

Figure 5. [SE1004, Sampat FIG. 16 (annotated)]

28. Sampat's transform object (the MSM 1608) performs the automatic flow control. For example, Sampat explains: "Media services manager (MSM) 1608 ***manages the flow of data*** through server software architecture 1512." [SE1004, 9:10-26 (emphasis added).] This includes managing the flow of data between the "source" and the "sink" objects, with the source and sink objects waiting on buffers being passed from the transform object. [SE1004, 9:10-26, 9:57-10:7, 10:26-29, 14:43-54, 15:41-17:28, 18:28-54.]

29. Sampat Figures 19 and 20, below, illustrate "the flow of data" through Sampat's server and client, showing that each includes a central "MSM" that automatically controls flow by passing "source buffers" and "sink buffers." [SE1004, 17:8-67.] I have reproduced Sampat Figures 19 and 20, below, and annotated them with colors and labels to show how each of Sampat's server and Sampat's client includes source, sink, and transform objects, with the transform object regulating the flow of data through buffer passing.

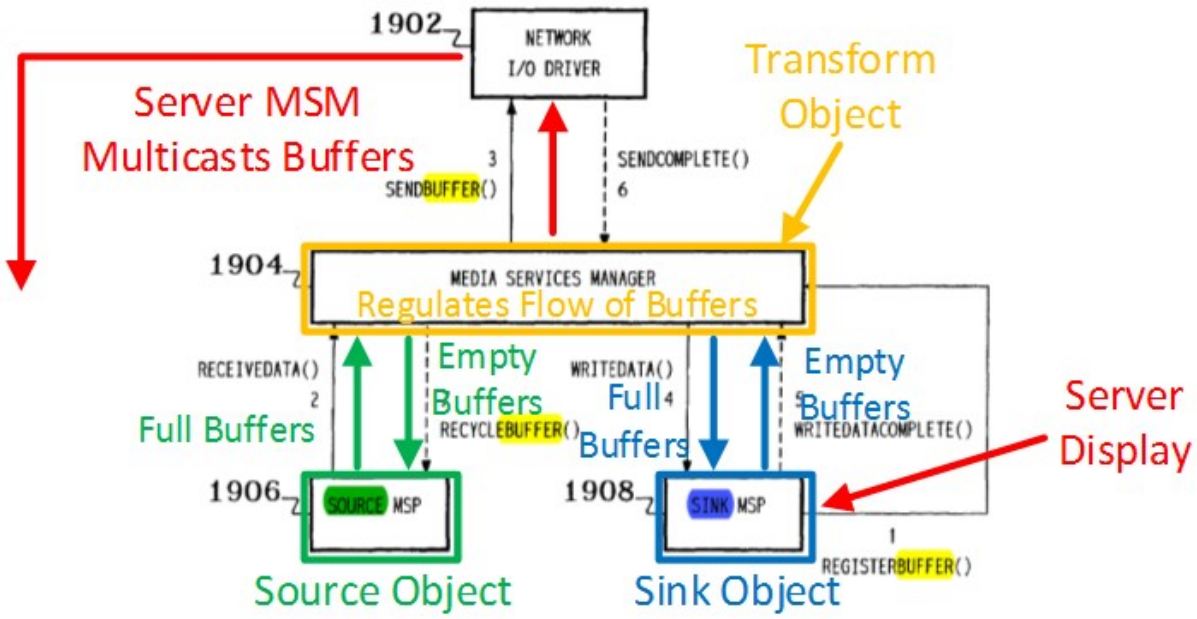

Figure 6. [SE1004, Sampat FIG. 19 (annotated)]

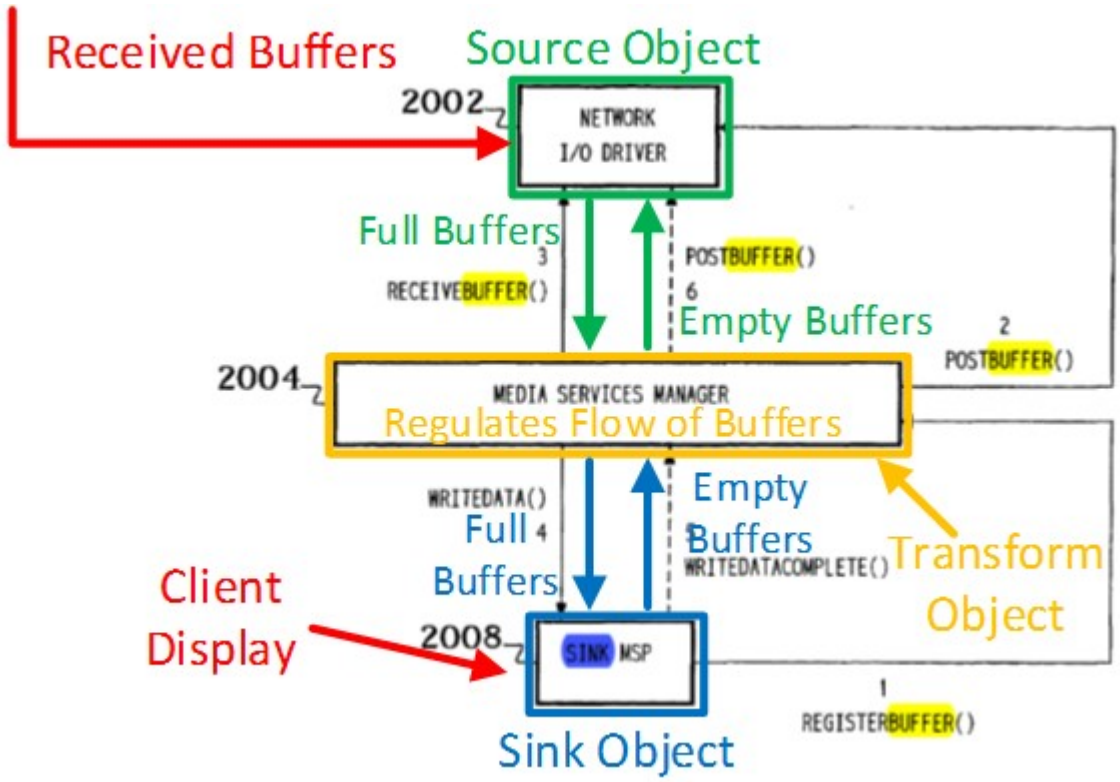

Figure 7. [SE1004, Sampat FIG. 20 (annotated)]

30. In addition to automatic flow control, other aspects of claims 31 and 61 were known by the time of the filing of the '389 patent. For example, Sampat discloses all elements of claims 31 and 61 of the '389 patent, including the claimed physical data source, source object, sink object, transform object, and control object. [SE1004, 2:10-16, 4:7-25, 7:59-8:22, 9:10-12, 9:37-53, 13:59-63, 14:33-54, 17:8-44, FIGS. 16-20.]

31. In particular, Sampat discloses that its preferred embodiment includes certain well-known components, including: "Intel® SmartVideo® Recorder (ISVR)," "a SoundBlaster Pro from Creative Labs," and "Microsoft Video for

Windows." [SE1004, 4:56-57, 8:25-27, 14:10-12.] These were all well-known to a POSITA and were described in contemporaneous publications. For example, Gerber describes the Intel Smart Video Recorder in further detail. [SE1005, 5:10-16, 5:20-60, FIG. 3.] The Sound Blaster Pro User Reference Manual ("SoundBlaster") describes details of the same SoundBlaster device preferred by Sampat. [SE1006, 10, 20, 146, 152-153, 158.] The Programmer's Guide, Microsoft Video for Windows Development Kit ("Microsoft Video for Windows") describes details of Video for Windows. [SE1007, 8, 157-211.] Accordingly, to the extent that Sampat does not *explicitly* state that a certain feature was present, a POSITA would have understood that such features were present based on what Sampat does disclose. Additionally, a POSITA would have been further prompted to reference the disclosures in SoundBlaster, Microsoft Video for Windows, and Gerber to supplement any gaps in his or her understanding. This will be explained further below.

C. Other Background Prior Art

32. In addition to Sampat, by the filing of the '389 patent, other systems included source, sink, and transform objects with the transform object controlling flow. For example, Bescos describes "source" and "sink" objects, as well as a monomedia stream as the transform object that "perform its flow control tasks." [SE1018 (Bescos), 4, *see also* 8 ("Implements flow control handling").] A

multimedia stream in Bescos, such as video and audio, included several such monomedia streams. Bescos describes how the multimedia stream acted as a control to pass user messages such as play and pause to individual monomedia streams. I have reproduced and annotated Figure 2 of Bescos to show this conventional source, sink, and transform arrangement.

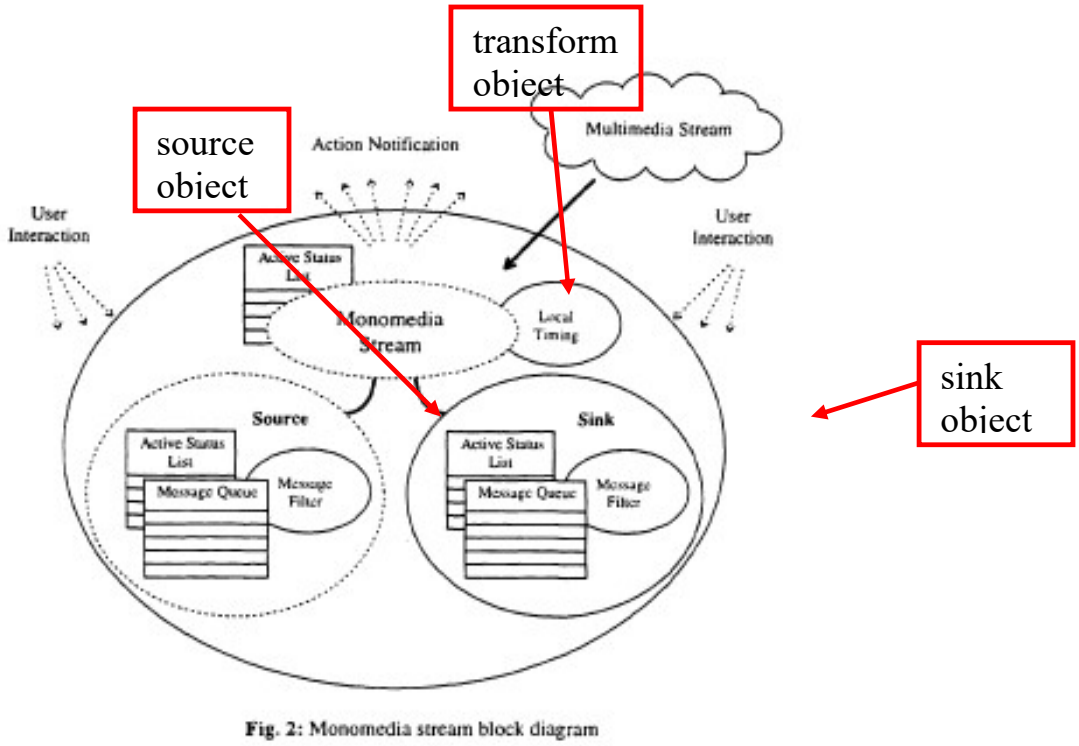

Fig. 2: Monomedia stream block diagram

Figure 8. [SE1018 (Bescos), 4 (annotated)]

D. Person of Ordinary Skill in the Art

33. Based on the foregoing and upon my experience in this area, a person of ordinary skill in the art in this field at the relevant time frame ("POSITA") would have had a combination of experience and education in software design. This typically would consist of a minimum of a bachelor of science degree in

computer science, software engineering, or a related field plus 2-5 years of work, graduate study, and/or research experience in the field of software design and the relevant field of signal processing. Additional education in a relevant field, such as computer science, software engineering, or a related field, or additional industry experience may compensate for a deficit in one of the other aspects of the requirements stated above.

34. Based on my experiences, I have a good understanding of the capabilities of a POSITA. Indeed, I have taught, participated in organizations, and worked closely with many such persons over the course of my career, including during the mid-1990's and certainly before the filing date of the '389 patent.

V. INTERPRETATIONS OF THE '389 PATENT CLAIMS AT ISSUE

35. I understand that, for purposes of my analysis in this *inter partes* review proceeding, the terms appearing in the patent claims should be interpreted according to their broadest reasonable construction in light of the specification of the patent in which it appears. In that regard, I understand that the best indicator of claim meaning is its usage in the context of the patent specification as understood by a POSITA. I further understand that the words of the claims should be given their plain meaning unless that meaning is inconsistent with the patent specification or the patent's history of examination before the Patent Office. I also understand that the words of the claims should be interpreted as they would have

been interpreted by a POSITA at the time of the invention was made (not today). Because I do not know at what date the invention as claimed was made, I have used the filing date of U.S. Patent No. 6,233,389 as the point in time for claim interpretation purposes. That date was July 30, 1998. I have been asked to provide my interpretation of the terms and phrases of the '389 patent set forth below.

36. I also understand that a number of claim terms of the '389 patent have been construed by district courts and that these constructions are consistent with what was proposed by Patent Owner. [*See* SE1011-SE1014.] I have reviewed these previous claim constructions made by the district courts. Here, the broadest reasonable interpretations of these claim terms are at least as broad as the district courts' constructions. A summary of these constructions appears in the following table.

Claim Term	**BRI Construction**
The Preamble	Not limiting. [SE1014, 82; SE1001, 14:53-54, 17:3-4.]
"accepts broadcast data"	"accepts data that was transmitted" [SE1014, 109-111, 3:30-61.]
"automatically flow controlled"	"self-regulated" [SE1014, 101-106; SE1001, 8:19-65.]

Claim Term	BRI Construction
"buffer"	"memory where data can be temporarily stored for transfer" [SE1014, 9; SE1001, 2:16-17, 4:55-6:15.]
"control object"	"a collection of data and operations that receives commands from a user that control the flow of broadcast data." [SE1014, 91-98; SE1001, 9:23-47.]
"control the flow of the broadcast data through the system"	"control the flow of the broadcast data within the system" [SE1014, 109-111; SE1001, 8:19-65.]
"Object"	"a collection of data and operations." [SE1014, 91-98; SE1001, 8:39-65, 9:23-47.]
"parses video and audio data from said broadcast data"	"analyzes video and audio data from the broadcast data" [SE1014, 82-88; SE1001, 5:3-9, 6:37-46.]
"parses"	"analyzes" [SE1014, 82-88; SE1001, 5:3-9, 6:37-46.]
"physical data source"	"hardware and software that parses video and audio data from said broadcast data"

Claim Term	BRI Construction
	[SE1014, 109-111; SE1001, 8:43-45.]
"sink object"	"a collection of data and operations that (1) obtains data stream buffers [memory where data can be temporarily stored for transfer] from a transform object and (2) outputs the streams to a video and audio decoder." [SE1014, 91-98; SE1001, 8:52-65.]
"source object"	"a collection of data and operations that (1) extracts video and audio data from a physical data source, (2) obtains a buffer [memory where data can be temporarily stored for transfer] from a transform object, (3) converts video data into data streams, and (4) fills the buffer [memory where data can be temporarily stored for transfer] with the streams." [SE1014, 91-98; SE1001, 8:39-51.]
"transform object"	"a collection of data and operations that transforms the form of data upon which it operates." [SE1014, 101-106; SE1001, 7:48-8:65.]

Claim Term	BRI Construction
"wherein said control object sends flow command events to said source, transform, and sink objects"	"the control object sends information relating to a change of condition in the flow of the broadcast data to the source, transform and sink objects." [SE1013, 16-17; SE1001, 9:22-47.]
"wherein said sink object is automatically flow controlled by said transform object"	"wherein said sink object is self-regulated by said transform object" [SE1014, 101-106; SE1001, 8:52-65.]
"wherein said source object extracts video and audio data from said physical data source"	"wherein the source object obtains video and audio data from said physical data source" [SE1014, 98-101; SE1001, 8:43-45.]
"wherein said source object is automatically flow controlled by said transform object"	"wherein said source object is self-regulated by said transform object" [SE1014, 101-106; SE1001, 8:39-51.]

37. Based upon my knowledge and experience in this field, I believe that a POSITA would have understood that these interpretations are consistent with the plain meaning of these terms under the broadest reasonable interpretation ("BRI") and are consistent with the '389 patent specification.

38. I also understand claim interpretation in district courts can be different from, and possibly narrower than, claim interpretation under the BRI standard. I have performed no analysis as to whether the above constructions are correct under the standard in district court, and consequently, offer no opinion on that subject. Rather, I have concluded that the BRI of the relevant terms of claims 31 and 61 of the '389 patent are at least as broad as the above constructions. Accordingly, I have applied the above constructions in my analysis of claims 31 and 61. Any claim terms not specifically construed above were also construed under their broadest reasonable interpretation in light of the specification.

VI. ANALYSIS OF SAMPAT (CLIENT SIDE) (CLAIMS 31 AND 61)

39. Sampat describes a computer system "for multicasting audio, video, and/or text data streams." [SE1004, Abstract.] Sampat's computer system includes a server 102 and multiple clients 104 shown in Sampat Figure 1, reproduced below [SE1004, 4:2-6.] "Server 102 is capable of capturing analog audio and video signals." [SE1004, 4:7-8.] "Server 102 digitizes the received analog audio and video signals to generate digital audio and video data streams"

and "selectively relates the digital audio, video, and text data streams together to form specified channels." [SE1004, 4:23-26.] Server 102 multicasts "one or more channels from server 102 to one or more clients 104." [SE1004, 4:38-48.] "When a client 104 selects a channel, the client may receive and process the network data packets corresponding to the data streams of the selected channel." [SE1004, 4:43-46.] Each of the server 102 and the client 104 has the capability of locally playing and/or recording the data streams. [SE1004, 9:10-57; 14:33-15:3.] In performing this functionality, each of Sampat's server 102 and Sampat's client 104 meet all features of claims 31 and 61. For the sake of clarity, I will separately address Sampat's server 102 and Sampat's client 104.

40. First, I will address Sampat when viewed from Sampat's client 104. I have reproduced Sampat's Figure 1, below, and annotated it with colors and labels to schematically show that, for Sampat's client side, the source, sink, transform, and control objects are in the client 104, while the input devices and physical data source are upstream of the client 104. Because Sampat teaches a unified system of server and client, the fact that the input devices and physical data source are upstream of the client 104 does not change my opinion.

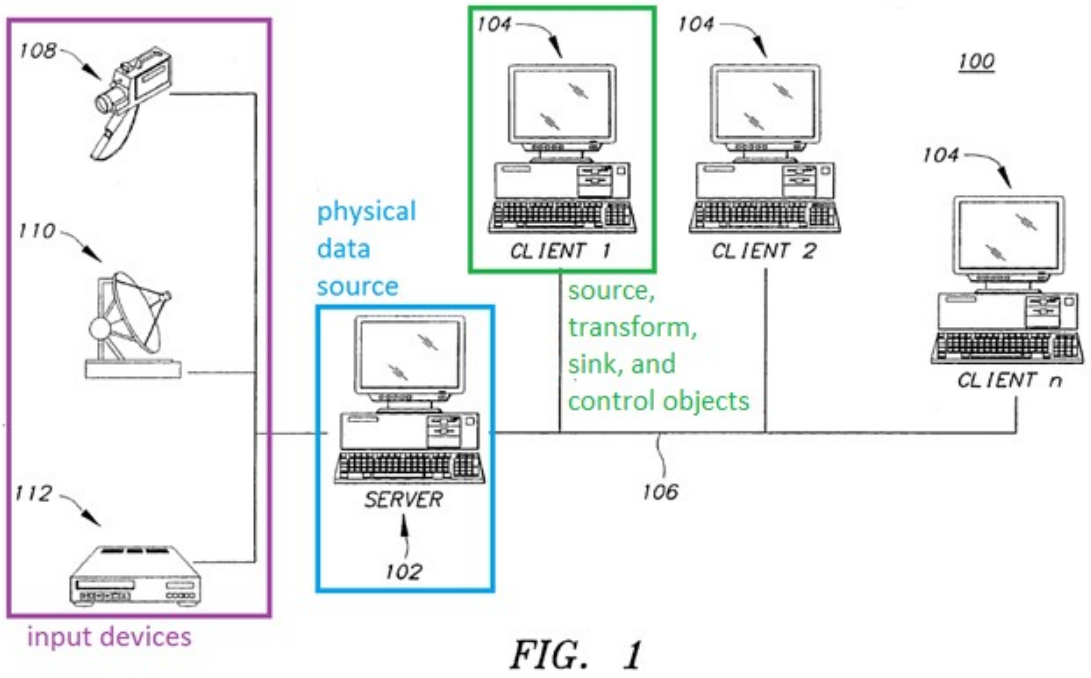

FIG. 1

Figure 9. [SE1004, Sampat FIG. 1 (annotated for Sampat's Client Side).]

41. I have reproduced Sampat's Figures 18 and 20, below, and annotated them with colors and labels to show the source, transform, sink, and control objects within the software architecture of client 104 shown above in Sampat Figure 1.

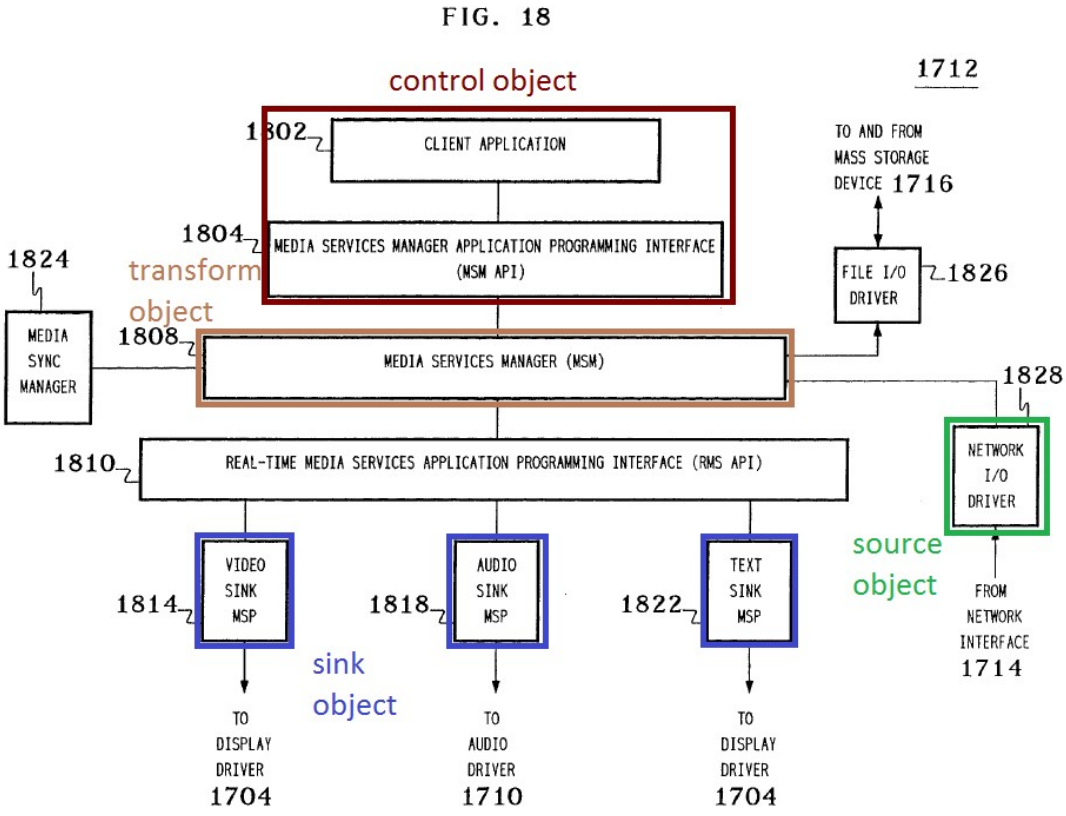

Figure 10. [SE1004, Sampat FIG. 18 (annotated)]

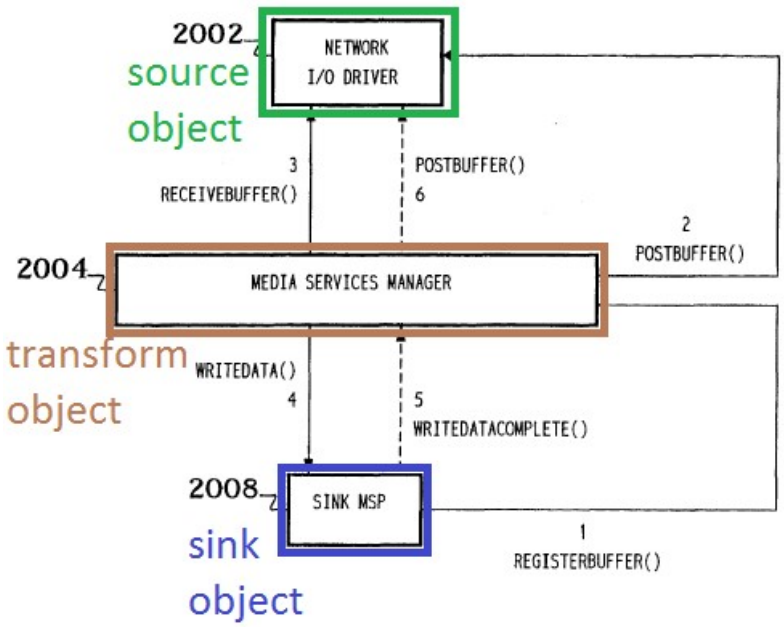

Figure 11. [SE1004, Sampat FIG. 20 (annotated)]

42. In the paragraphs that follow, I will discuss how, from the perspective of Sampat's client 104, Sampat discloses and renders obvious all features of claims 31 and 61.

A. Preambles of Claims 31 and 61

43. The preamble of claim 31 recites a "process for the simultaneous storage and play back of multimedia data" and the preamble of claim 61 recites an "apparatus for the simultaneous storage and play back of multimedia data."

44. I have been told that the Patent Owner has "agreed that the preambles of Claims 31 and 61 of the '389 Patent are not limiting." [SE1014, 82.] Accordingly, the preambles of claims 31 and 61 are not limiting.

45. Even if the preambles were limiting, my opinion would not change, because Sampat discloses a process for the simultaneous storage and play back of multimedia data. [SE1004, 13:59-63 ("recording of data from the network to mass storage device 1716 with or without concurrent playing of the multicast data"); *see also* 2:10-16.]. Sampat also discloses an apparatus for the simultaneous storage and play back of multimedia data. [SE1004, 8:18-22, 13:59-63, 2:10-16, Fig. 1]

B. Physical Data Source Features of Claims 31 and 61

46. Claim 31 of the '389 patent recites providing a physical data source and claim 61 recites a physical data source. In addition, claims 31 and 61 of the '389 patent each recite "wherein said physical data source accepts broadcast data

from an input device, parses video and audio data from said broadcast data, and temporarily stores said video and audio data." As discussed below, Sampat discloses these features.

47. As indicated in Section V above, the BRI of "physical data source" includes "hardware and software that parses video and audio data from said broadcast data." [SE1014, 109-111; SE1001, 8:43-45.] The BRI of "parses video and audio data from said broadcast data" includes "analyzes video and audio data from the broadcast data." [SE1014, 82-88; SE1001, 5:3-9, 6:37-46.]

48. As viewed from the client-side, Sampat discloses a physical data source in the form of server 102, which includes hardware and software. [SE1004, 4:7-25.] I have reproduced Sampat's Figure 1, below, and annotated it with colors and labels to show the physical data source in blue.

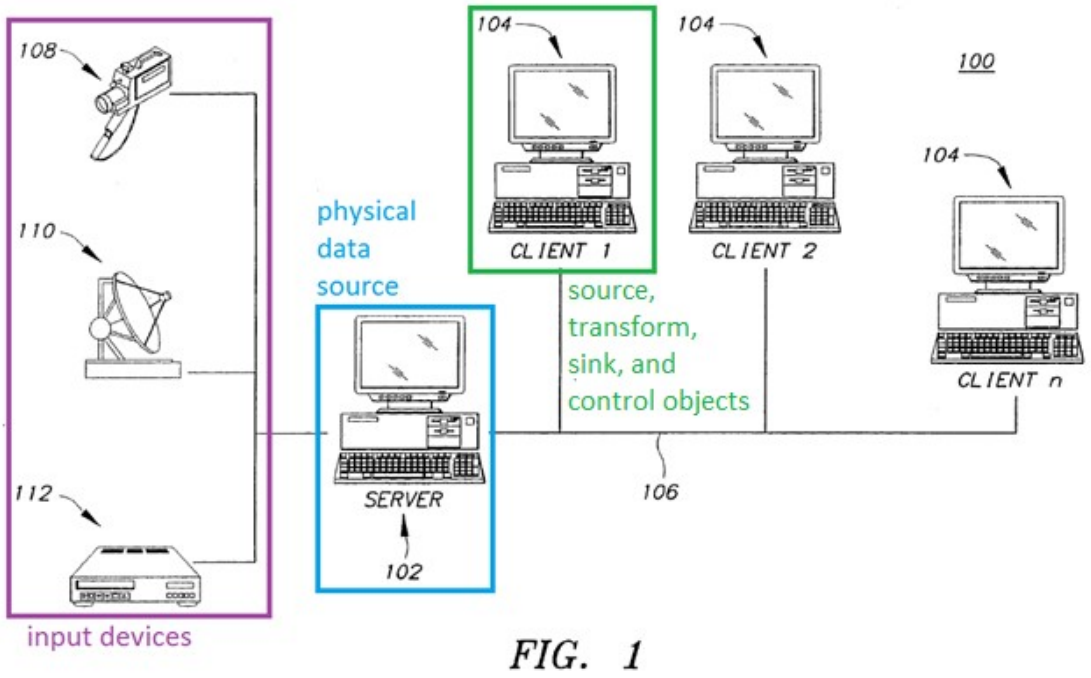

FIG. 1

Figure 12. [SE1004, Sampat FIG. 1 (annotated)]

49. Sampat discloses server 102 (the physical data source) accepting broadcast data from one or more input devices (*e.g.,* camera 108, antenna 110, and VCR 112). [SE1004, 4:7-14 ("Server 102 is capable of capturing analog audio and video signals from three different sources: (1) signals generated locally by camera 108, (2) signals received by antenna 110 from a remote source, and (3) recorded signals from VCR 112"), 38:42-54 (the system can be used with "broadcasting").] "For example, server 102 may receive via antenna 110 a first television program" and "may receive a second television program … from VCR 112." [SE1004, 4:15-23.] A television program received by an antenna includes broadcast data.

50. Sampat further discloses server 102 parsing video and audio data from the broadcast data. [SE1004, 7:55-63 ("In particular, tuner 1502 of server subsystem 102 receives, demodulates, and splits one or more analog television feed signals into their constituent analog audio and video signals."); 4:24-43 (describing the server 102 digitizing and fragmenting the received signals).] I have reproduced Sampat's Figure 15, below, and annotated it with colors and labels to show a block diagram of server 102, with its tuner 1502 parsing the broadcast data (purple) into video and audio data (brown).

FIG. 15

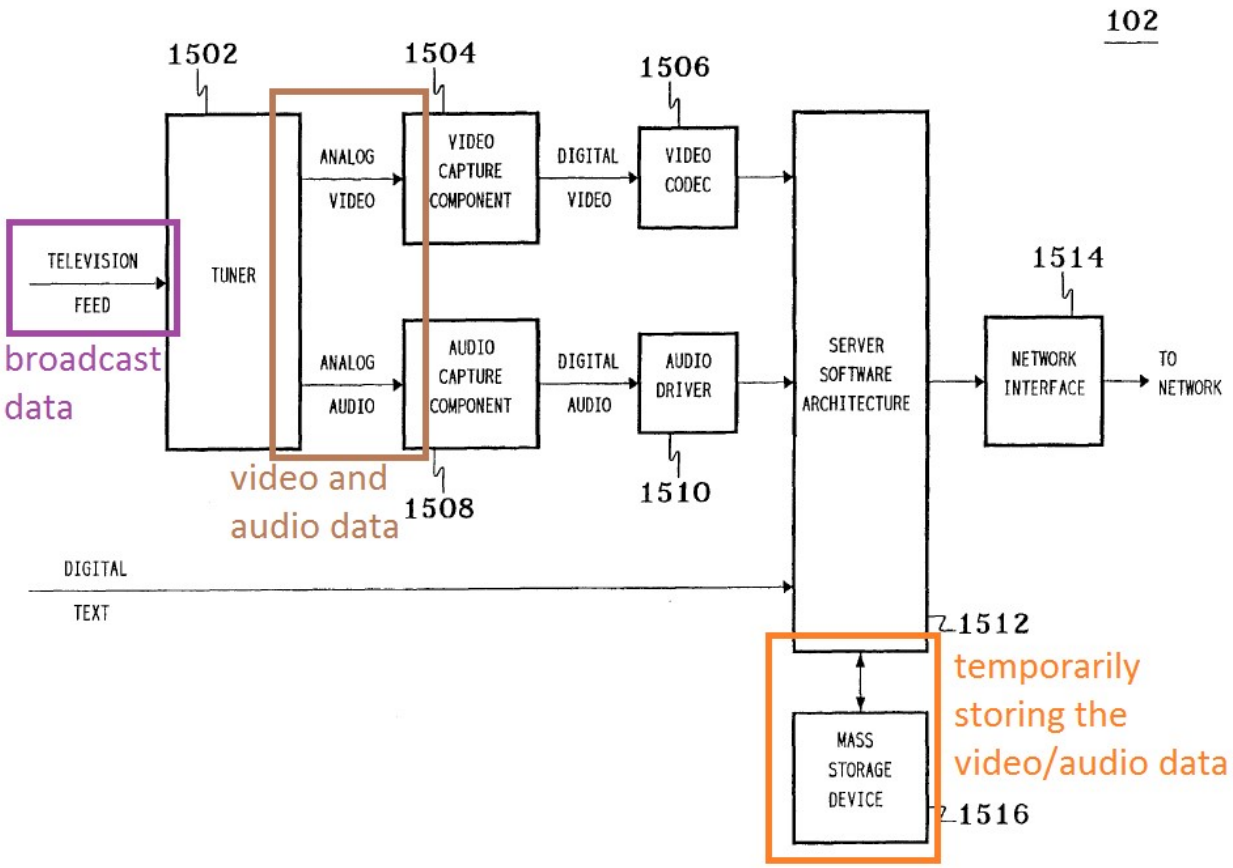

Figure 13. [SE1004, Sampat FIG. 15 (annotated)]

51. For these reasons, Sampat's server 102 includes hardware and software that parses (*e.g.*, analyzes) video and audio data from broadcast data. Thus, under the BRI of physical data source identified above, Sampat's server 102 is a physical data source and Sampat's disclosure of providing the server 102 is disclosure of providing a physical data source. As explained above, Sampat's server 102 (the physical data source) accepts broadcast data from an input device and parses video and audio data from said broadcast data.

52. Sampat's server 102 also temporarily stores the video and audio data. [SE1004, 9:10-26 (describing temporary storage for later multicasting or processing).] This temporary storage is also illustrated in Sampat Figure 15, shown above. Server software architecture 1512 of server 102 is shown in Figure 15 temporarily storing and retrieving the video and audio data to mass storage device 1516 (shown in orange). [SE1004, FIG. 15, 8:17-22, 9:10-26.]

53. I also understand that Patent Owner has alleged that certain components of a Samsung DVR meet this claim limitation. [SE1015, 39-48.] For example, Patent Owner discusses input devices, stating: "A Samsung DVR has an *input device* that may include cable input, *tuners*, an analog to digital converter, and/or, on information and belief, other components such as the Broadcom BCM3255 or other similar chip that may perform QAM demodulation, including chips made by other third parties." [SE1015, 39 (emphasis added).]

54. Under this alternative understanding, Sampat still discloses the claimed physical data source. For example, the server 102 would function as the physical data source without the tuner 1502, and the tuner 1502 would be considered the input device as in Patent Owner's infringement contentions. I have created an annotated view of Sampat Figure 15, below, to illustrate this alternative view.

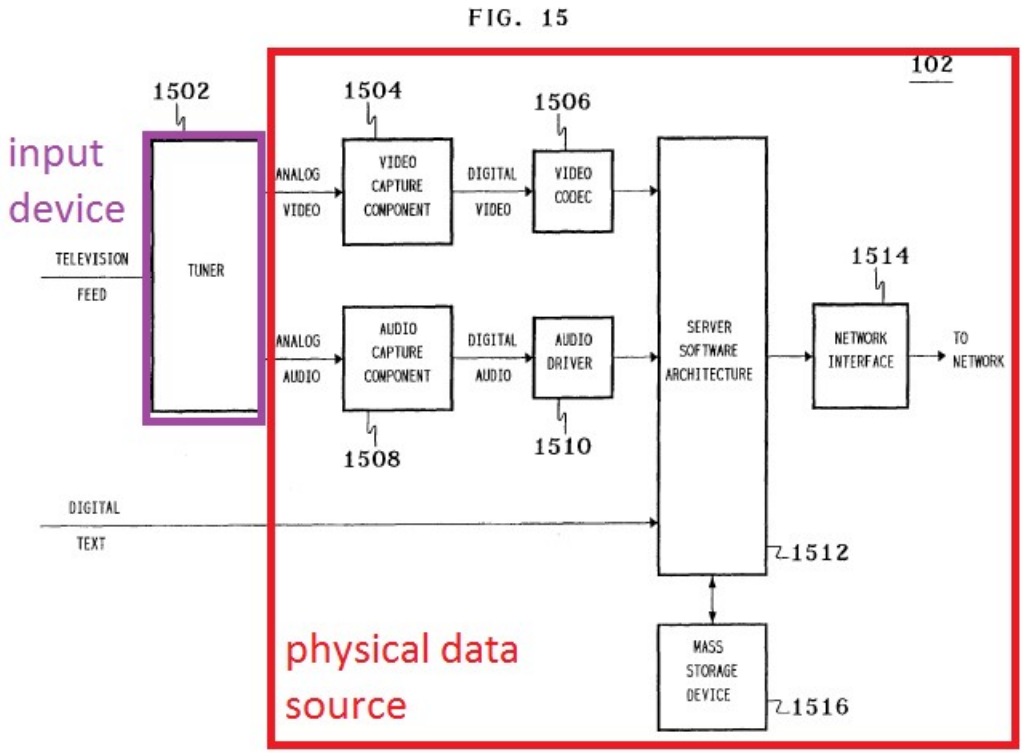

Figure 14. [SE1004, Sampat FIG. 15 (annotated)]

55. In this alternative interpretation, Sampat's disclosure of tuner 1502 is consistent with Patent Owner's allegation that the input device is met by a tuner. [SE1004, 7:59-61, 8:22-25; SE1015, 39.]

56. As I noted previously, the BRI of "parses" includes "*analyzes* video and audio data from said broadcast data." [SE1014, 82-88.] In this alternative interpretation, the server 102 is the physical data source which analyzes video and audio data from the broadcast data in multiple ways. Sampat explains, "[v]ideo capture component 1504 captures and converts the analog video signals into digital video data streams" and "audio capture component 1508 captures and converts the analog audio signals into digital audio data streams," which a POSITA would have understood includes analysis of video and audio data when converting into digital video. [SE1004, 7:62-65.] Sampat also explains that "[v]ideo codec 1506 compresses the digital video data streams" and "[a]udio driver 1510 places the audio data into buffers," which a POSITA would also understand includes analysis of video and audio data under this interpretation. [SE1004, 8:7-10.] A POSITA would have also understood that the server software architecture 1512 also analyzes the video and audio data. [SE1004, 8:12-17, *see also* 8:50-13:33.] Therefore, Sampat discloses the claimed physical data source whether the tuner is considered to be part of the physical data source, or alternatively, part of the input device.

C. Source Object Features of Claims 31 and 61

57. Claim 31 of the '389 patent recites providing a source object and claim 61 recites a source object. In addition, claims 31 and 61 of the '389 patent

each recite "wherein said source object extracts video and audio data from said physical data source" and "wherein said source object obtains a buffer from said transform object, said source object converts video data into data streams and fills said buffer with said streams." As discussed below, Sampat discloses these features.

58. As indicated in Section V above, the BRI for "source object" includes "a collection of data and operations that (1) extracts video and audio data from a physical data source, (2) obtains a buffer [memory where data can be temporarily stored for transfer] from a transform object, (3) converts video data into data streams, and (4) fills the buffer [memory where data can be temporarily stored, for transfer] with the streams." [SE1014, 9, 91-98; SE1001, 8:39-51.]

59. As viewed from the client-side, Sampat describes the client 104 having a source object in the form of the network I/O driver (referred to as network I/O driver 1828 in FIG. 18, reproduced above; network I/O driver 2002 in FIG. 20, reproduced above; and network I/O driver 2100 in FIG. 21). [SE1004, 14:38-43, 17:39-44, 18:9-16.] Network I/O driver is a collection of data and operations. [SE1004, 18:7-19:24; 19:24-28:62.] Network I/O driver also performs each of the four functions defined as being performed by a source object. These four functions are each addressed in turn below.

i. *First Function - Extracts Video and Audio Data from a Physical Data Source*

60. The network I/O driver (the source object) extracts video and audio data from the server 102 (the physical data source), which it obtains via the network interface 1714. [SE1004, 14:38-43.] As indicated in Section V above, the BRI of "extracts video and audio data from said physical data source" includes "obtains video and audio data from said physical data source." [SE1014, 98-101.] The formats of the audio and video data extracted from the server by the network I/O driver are shown in Sampat Fig. 25 and Fig. 26, respectively. Because Sampat's network I/O driver obtains video and audio data from the server 102, Sampat's network I/O driver extracts video and audio data from the physical data source, as contemplated by the '389 patent. [SE1004, 14:38-43.]

ii. *Second Function - Obtains a Buffer from a Transform Object*

61. Sampat describes the source object (network I/O driver) obtaining a buffer from a media services manager (MSM), stating: "MSM 2004 re-posts the buffer to network I/O driver 2002 to be reused." [SE1004, 17:64-67, *see also* 17:50-59.] As explained above at Section IV.B and below at Section VI.D, Sampat's MSM is a transform object. Thus, in Sampat, the source object (network I/O driver) obtains a buffer from the transform object (MSM). I have reproduced

Sampat's Figure 20, below, and annotated it with colors and labels to show how Sampat's network I/O driver (source object) obtains a buffer from Sampat's MSM (transform object).

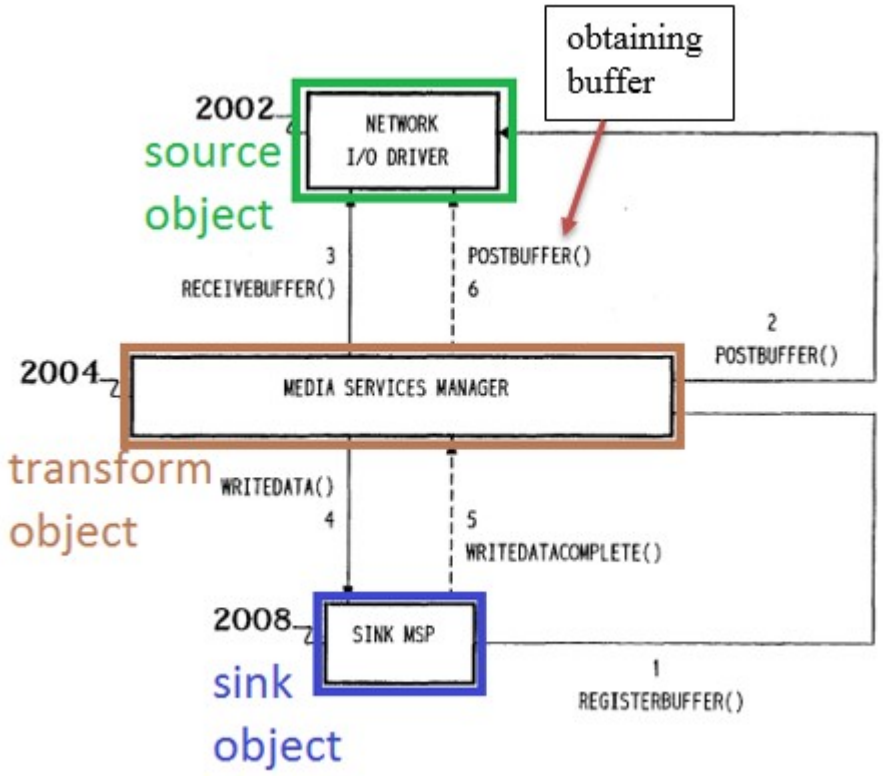

Figure 15. [SE1004, Sampat FIG. 20 (annotated), see also 17:50-59, 17:64-67.]

iii. *Third Function - Converts Video Data into Data Streams*

62. Sampat discloses that the source object (network I/O driver) converts video data packets into data streams: "In a client, network I/O driver 2100 receives related, time-stamped data ***packets*** from the network and transmits data ***streams*** corresponding to those data packets to the client media services manager for display and/or recording of the multicast channel data." [SE1004, 18:20-25

(emphasis added), *see also* 18:59-63 (describing the network I/O driver performing "fragmentation and re-assembly (i.e., de-fragmentation) of large data messages").] From this description, a POSITA would have understood that Sampat's network I/O driver converts video data into data streams. More particularly, the steps in converting network packets into video data streams are outlined in Sampat Figure 24 in general. [*See* SE1004, 28:66-29:39.] The format of the data received over the network at the client network interface is called a level 5 packet and shown in Sampat Fig. 24 and Sampat Fig. 30. [SE1004, 31:9-34.] The format of a video data packet in the data stream output by the network I/O driver is called level 1 and shown in Sampat Fig. 24 and Sampat Fig. 26. [SE1004, 29:51-62.] The procedure from converting from one format to the other encompasses several steps. [SE1004, 29:25-33 (overview of procedure), 30:1-31:43, Figure 24.] Because the network I/O driver outputs level 1 video data packets comprising a Microsoft Video for Windows header, the network I/O driver outputs a video data stream.

iv. *Fourth Function - Fills the Buffer with the Streams*

63. Sampat discloses that the source object (network I/O driver) fills the buffer with data streams, stating: "When data is received by network I/O driver 2002 from the network, network I/O driver 2002 ***fills a sink buffer*** and passes it to MSM 2004." [SE1004, 17:55-58 (emphasis added).] From this description, a

POSITA would have understood that Sampat's network I/O driver fills the buffer it obtained from the MSM with the data streams created as outlined above.

v. *Source Object - Conclusion*

64. For these reasons, a POSITA would have understood the network I/O driver in Sampat's client 104 to include a collection of data and operations that performs all four functions included in the BRI of the term source object. Thus, under the BRI of source object identified above, Sampat's network I/O driver includes a source object and Sampat's disclosure of providing the network I/O driver is disclosure of providing a source object. As explained above, Sampat's network I/O driver (the source object) extracts video and audio data from a physical data source, obtains a buffer from a transform object, converts video data into data streams, and fills the buffer with the streams.

D. Transform Object Features of Claims 31 and 61

65. Claim 31 of the '389 patent recites providing a transform object and claim 61 recites a transform object. In addition, claims 31 and 61 of the '389 patent each recite "wherein said transform object stores and retrieves data streams onto a storage device." As discussed below, Sampat discloses these features.

66. As indicated in Section V above, under BRI, a "transform object" is "a collection of data and operations that transforms the form of data upon which it operates." [SE1014, 101-106; SE1001, 7:48-8:65.] The '389 patent describes one

type of transform as a ***temporal*** transform. [SE1001, 7:64-8:8 (describing a transform as "temporal" when it "writes the buffer to a file 804 on the storage medium" and then pulls the buffer out "at a later time").]

67. A POSITA would have understood that Sampat describes a transform object that performs a temporal transform by describing the "media services manager" or "MSM" (MSM 1808 in Figure 18; MSM 2004 in Figure 20). [SE1004, 14:33-54, 17:39-67, FIGS. 18 and 20.] The MSM is a collection of data and operations. [SE1004, 14:30-16:58, 17:39-18:5, FIGS. 18, 20, 24.]

68. Sampat's MSM performs a temporal transform by writing data streams to the mass storage device 1716 and retrieving the data streams at a later time for playback. [SE1004, 15:7-9 ("MSM 1808 uses file I/O driver 1826 to store and retrieve data to and from mass storage device 1716."), 14:45-54, FIGS. 17-18.] Sampat describes the MSM temporarily storing and retrieving data flowing "through MSM 1808 ... [f]rom the network to mass storage device 1716 (for recording of multicast data for subsequent processing), and [f]rom mass storage device 1716 to a sink MSP (for playing of locally recorded multicast data)." [SE1004, 14:45-54.] In Figure 18, the MSM 1808 is shown transferring data streams "to and from mass storage device 1716" via the file I/O driver 1826. [SE1004, FIG. 18; *see also* FIG. 17.] I have reproduced Sampat's Figures 17 and

18, below, and annotated them with colors and labels to show how the MSM stores and retrieves data streams "to and from mass storage device 1716."

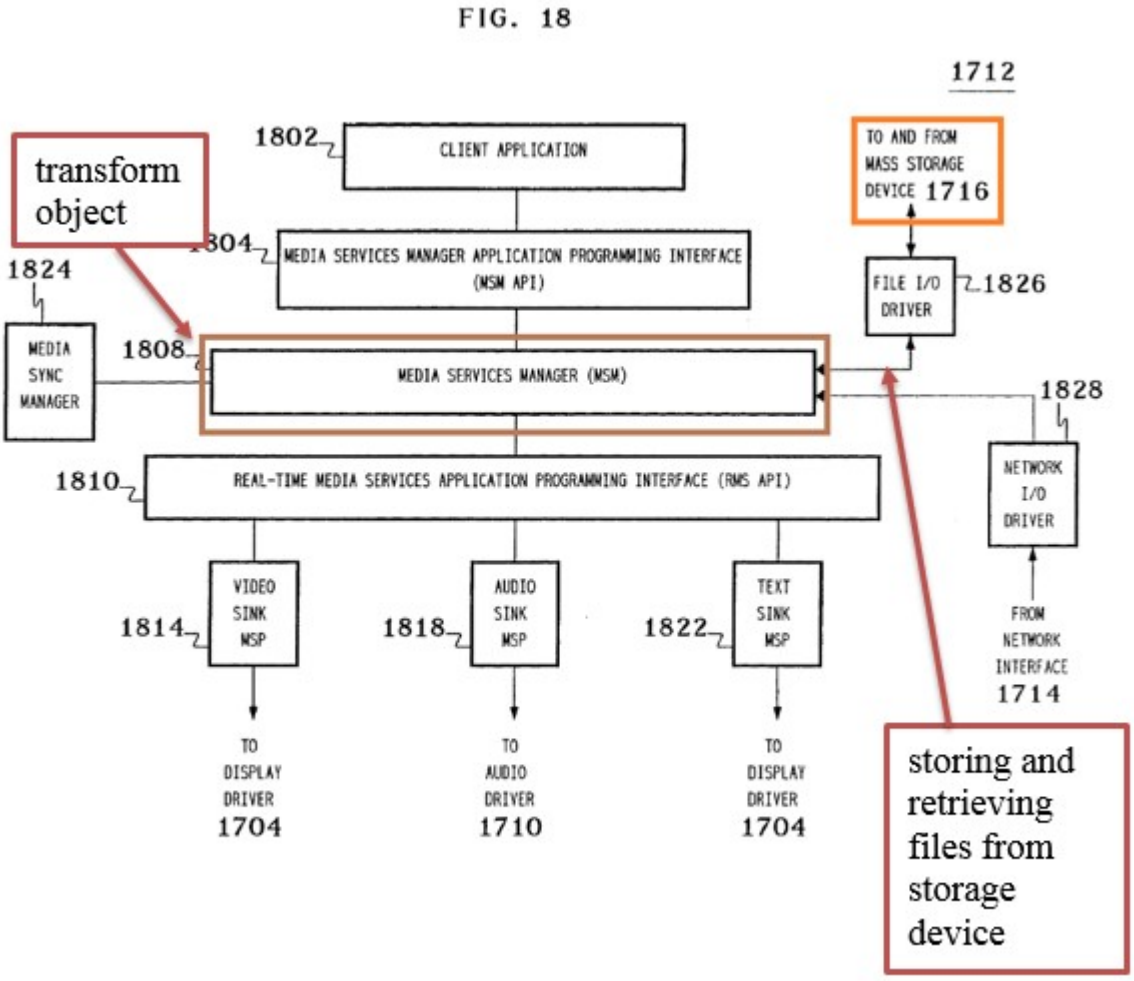

Figure 16. [SE1004, Sampat FIG. 18 (annotated)]

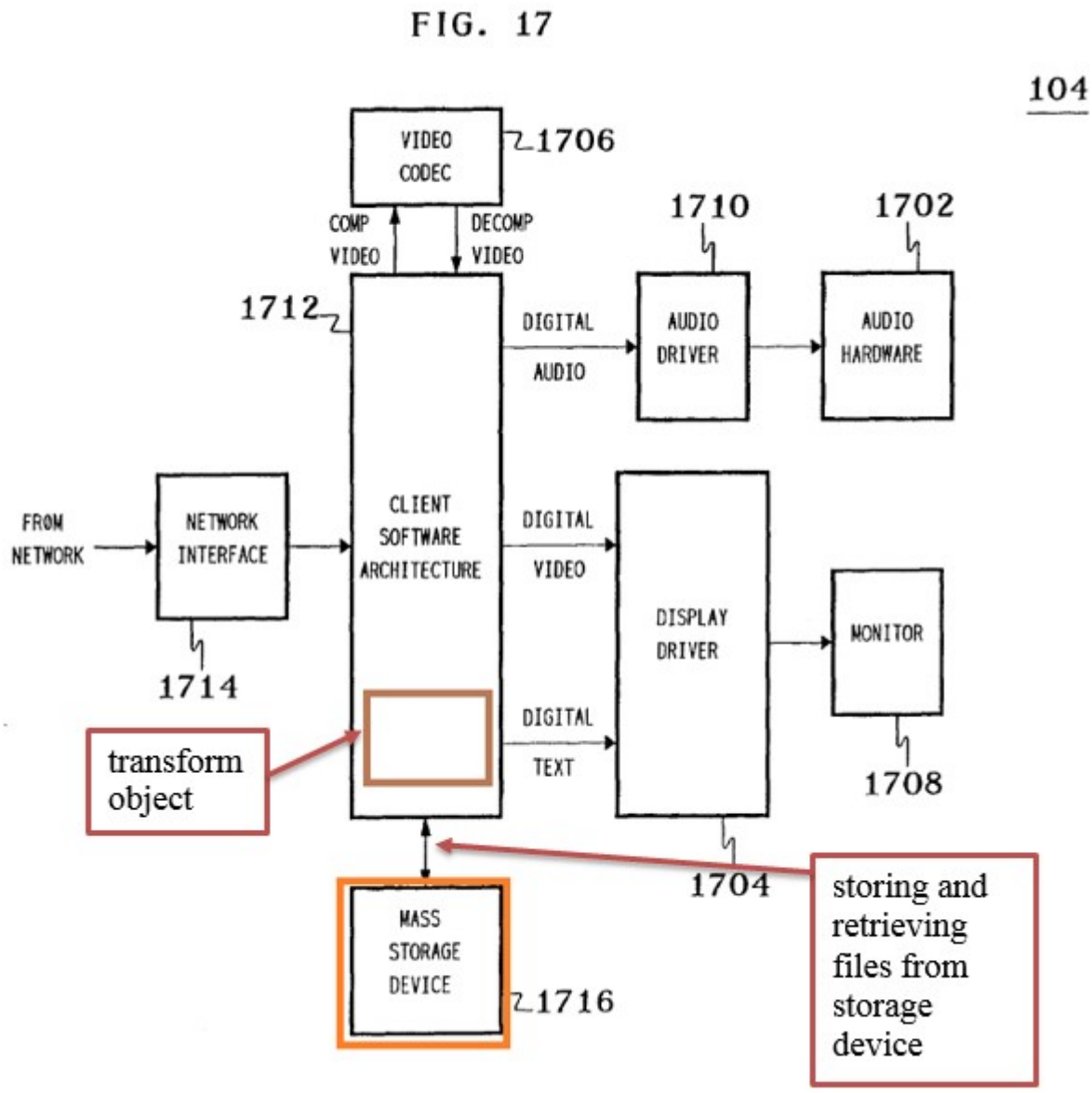

Figure 17. [SE1004, Sampat FIG. 17 (annotated)]

69. The temporal transform performed by Sampat's MSM is consistent with the '389 patent's temporal transform, which involves writing a buffer to a file on a storage medium and then pulling the buffer out at a later time. [*See* SE1001, 7:64-8:8.] Because Sampat's MSM writes data stream buffers as files on a storage

medium and then pulls the files out to fill buffers at a later time, Sampat discloses a transform object that transforms the form of data upon which it operates in a manner similar manner to the '389 patent. [*See* SE1004, 14:45-54; 15:7-9; FIGS. 17-18.]

70. For example, Sampat's MSM performs a temporal transform by writing data streams to the mass storage device 1716 and retrieving the data streams at a later time for playback. [SE1004, 14:45-54; 15:7-9; FIGS. 17-18.] The annotated version of Sampat's FIG. 18 above shows MSM 1808 transferring data streams "to and from mass storage device 1716." [SE1004, 14:45-54; 15:7-9; FIGS. 17-18.] By writing/retrieving data streams to/from a mass storage device, Sampat's transform object (MSM) stores and retrieves data streams onto a storage device. Sampat makes it clear that data streams can be played back later [SE1004, 13:59-63 ("recording of data from the network to mass storage device 1716 with or without concurrent playing of the multicast data"); 38:62-64 ("A client may receive and record non-real-time transmitted data for *future playback* at a real-time rate.")].

71. For these reasons, a POSITA would have understood the MSM in Sampat's client 104 to include a collection of data and operations that transforms the form of data upon which it operates. Thus, under the BRI of transform object identified above, Sampat's MSM includes a transform object and Sampat's

disclosure of providing the MSM is disclosure of providing a transform object. As explained above, Sampat's MSM (the transform object) stores and retrieves data streams onto a storage device.

E. Sink Object Features of Claims 31 and 61

72. Claim 31 of the '389 patent recites providing a sink object and claim 61 recites a sink object. In addition, claims 31 and 61 of the '389 patent each recite "wherein said sink object obtains data stream buffers from said transform object and outputs said streams to a video and audio decoder." As discussed below, Sampat discloses these features.

73. As indicated in Section V above, under BRI, "sink object" includes "a collection of data and operations that (1) obtains data stream buffers [memory where data can be temporarily stored for transfer] from a transform object and (2) outputs the streams to a video and audio decoder." [SE1014, 9, 91-98; SE1001, 8:52-65.]

74. Sampat describes a sink object in the form of three sink MSPs: video sink MSP 1814, text sink MSP 1822, and audio sink MSP 1818. [SE1004, 14:64-15:3.] These sink MSPs are a collection of data and operations. [SE1004 at 14:64-15:3, 17:39-67.] These sink MSPs also performs each of the two functions defined as being performed by a sink object. These two functions are each addressed in turn below.

i. *First Function - Obtains Data Stream Buffers from a Transform Object*

75. Sampat explains that the sink MSPs (sink object) obtain data stream buffers from the MSM (transform object), stating: "When data is received by network I/O driver 2002 from the network, network I/O driver 2002 fills a sink buffer and passes it to MSM 2004 (using the MSM API function ReceiveBuffer)" and that "MSM 2004 then writes the sink buffer data ***to the sink MSP*** that owns the buffer (using the WriteData function)." [SE1004, 17:55-60 (emphasis added).] I have illustrated this buffer passing between the MSM and the sink MSP using color annotations and labelling in the version of Figure 20 below.

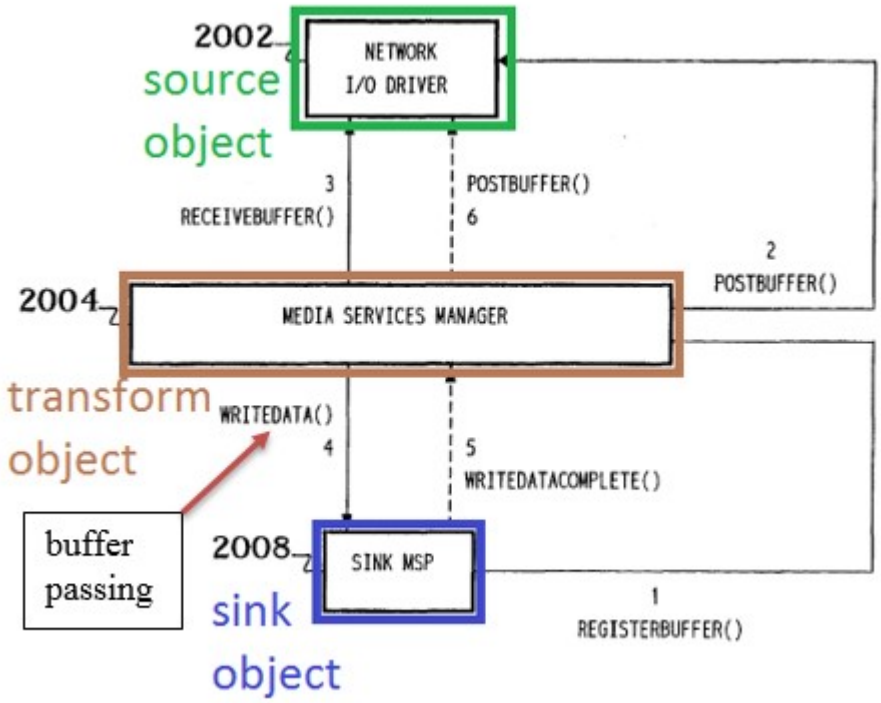

Figure 18. [SE1004, Sampat FIG. 20 (annotated)]

76. In fact, Sampat describes all three sink MSPs receiving a "data stream from MSM 1808." [SE1004, 14:64-15:3.] From these disclosures, a POSITA would have understood the sink MSPs as receiving the data stream buffers from the MSM.

ii. *Second Function - Outputs the Streams to a Video and Audio Decoder*

77. Sampat also teaches the sink MSPs outputting the data streams to a video and audio decoder. For example, Sampat explains regarding Fig. 18:

> Video sink MSP 1814 and text sink MSP 1822 receive a video data stream and a text data stream, respectively, from MSM 1808 and ***transmits the video and text data to display driver 1704*** of FIG. 17 for display on monitor 1708. Similarly, audio sink MSP 1818 receives an audio data stream from MSM 1808 and ***transmits the audio data to audio driver 1710 for play on audio hardware 1702***.

[SE1004, 14:64-15:3 (emphasis added).] The fact that two sink MSPs are involved is consistent with the BRI of sink object, in which an object is "a collection of data and operations" (see Section V).

78. A POSITA would have understood that Sampat's display driver 1704 illustrated in Fig. 17 and Fig. 18 includes a video decoder. Such drivers were known to include decoders to convert signals for reproducing via a display.

Sampat gives examples for the display driver 1704 as the "Microsoft Video for Windows." [SE1004, 14:6-12 ("Display driver 1704 may be any suitable driver for displaying video and text data and is preferably Microsoft Video for Windows.").] Microsoft Video for Windows was well-known to a POSITA prior to the 1998 filing date of the '389 patent, and was also known to include decoders. [*See, e.g.*, SE1007, 8 (describing Microsoft Video for Windows having "video compression and decompression drivers"), 157-211 (video decompression); SE1021, 2 (stating that "Video for Windows" display driver was well-known to include "three compressor-decompressor (codec) algorithms"); SE1008, 2:10-40 (describing "Video for Windows" receiving compressed video data and decoding/decompressing it).] POSITA understands that in this context, decoder and decompressor are synonymous.

79. A POSITA also would have understood that the system of Sampat includes an audio decoder. For example, Sampat explains that "clients 104 are preferably sound enabled with a SoundBlaster Pro from Creative Labs." [SE1004, 4:56-57.] Accordingly, Sampat's disclosure of the sink object outputting to audio driver 1710 illustrated in Fig. 17 and Fig. 18 and audio hardware 1702 would output to the SoundBlaster driver and hardware. [SE1004, 4:56-57, 15:1-15:3.] Such SoundBlaster Pro devices were well-known to a POSITA prior to the 1998

filing date of the ’389 patent, and were also known to include audio decoders. [*See, e.g.*, SE1006, 10, 146, 152-153.]

80. Under the BRI, Sampat’s decoding of audio and video via separate audio and video decoders is consistent with an embodiment disclosed in the ’389 patent. [SE1001, 4:23-29 (“separate decoders are used to convert MPEG elements back into audio or video analog components”).] According to Patent Owner’s infringement contentions, Patent Owner also asserts that this claim covers a device with separate audio and video decoders. [SE1015, 60 (accusing device with “two high definition … video decoders, two advanced-audio decoders”), see also 56-60.] Accordingly, under the BRI, Sampat teaches the sink object outputting data streams to a video and audio decoder in a manner similar to how the ’389 patent describes outputting data streams to separate video and audio decoders. [SE1001,4:23-29].

iii. *Sink Object - Conclusion*

81. For these reasons, a POSITA would have understood the sink MSPs in Sampat’s client 104 to include a collection of data and operations that perform each of two functions included in the BRI of the term sink object. Thus, under the BRI of sink object identified above, Sampat’s sink MSPs include a sink object and Sampat’s disclosure of providing the sink MSPs is a disclosure of providing a sink object. As explained above, Sampat’s sink MSPs (the sink object) obtain data

stream buffers from the MSM (the transform object) and output the streams to a video and audio decoder.

F. Automatic Flow Control Features of Claims 31 and 61

i. *Automatic Flow Control - Construction*

82. Claims 31 and 61 of the '389 patent each recite "wherein said source object is automatically flow controlled by said transform object" and "wherein said sink object is automatically flow controlled by said transform object." As discussed below, Sampat discloses these features.

83. As indicated in Section V above, under BRI, "automatically flow controlled" includes "self-regulated" and, consequently, this claim element includes "wherein said source object is ***self-regulated*** by said transform object" and "wherein said sink object is ***self-regulated*** by said transform object." [SE1014, 101, 102, 106 (emphasis added).] District courts have repeatedly reached this construction and agreed that the analysis in the second reexamination of the '389 patent "did not limit or redefine" this relatively broad construction of "automatically flow controlled." [SE1014, 106; SE1013, 12.] The most recent district court ruling is consistent with earlier rulings and explained: "TiVo merely reiterated during reexamination that ***rather than the source object and sink object being merely reactive to the data flow, they are flow controlled by the transform object***." [SE1014 at 106 (emphasis added).] Accordingly, under BRI, a POSITA

would have understood that the arguments made in the second reexamination of the ’389 patent do not require narrowing of this construction. Thus, the BRI of this claim term covers being “self-regulated” by the transform object.

84. In Sampat, the MSM (transform object) self-regulates flow [SE1004, 14:43-54.] Sampat explains: “***MSM 1808 manages the flow*** of data through the client software architecture.” [SE1004, 14:43-44 (emphasis added).] The following illustration compares FIG. 9 of the ’389 patent to FIG. 20 of Sampat to show the flow control in both systems.

FIG. 20

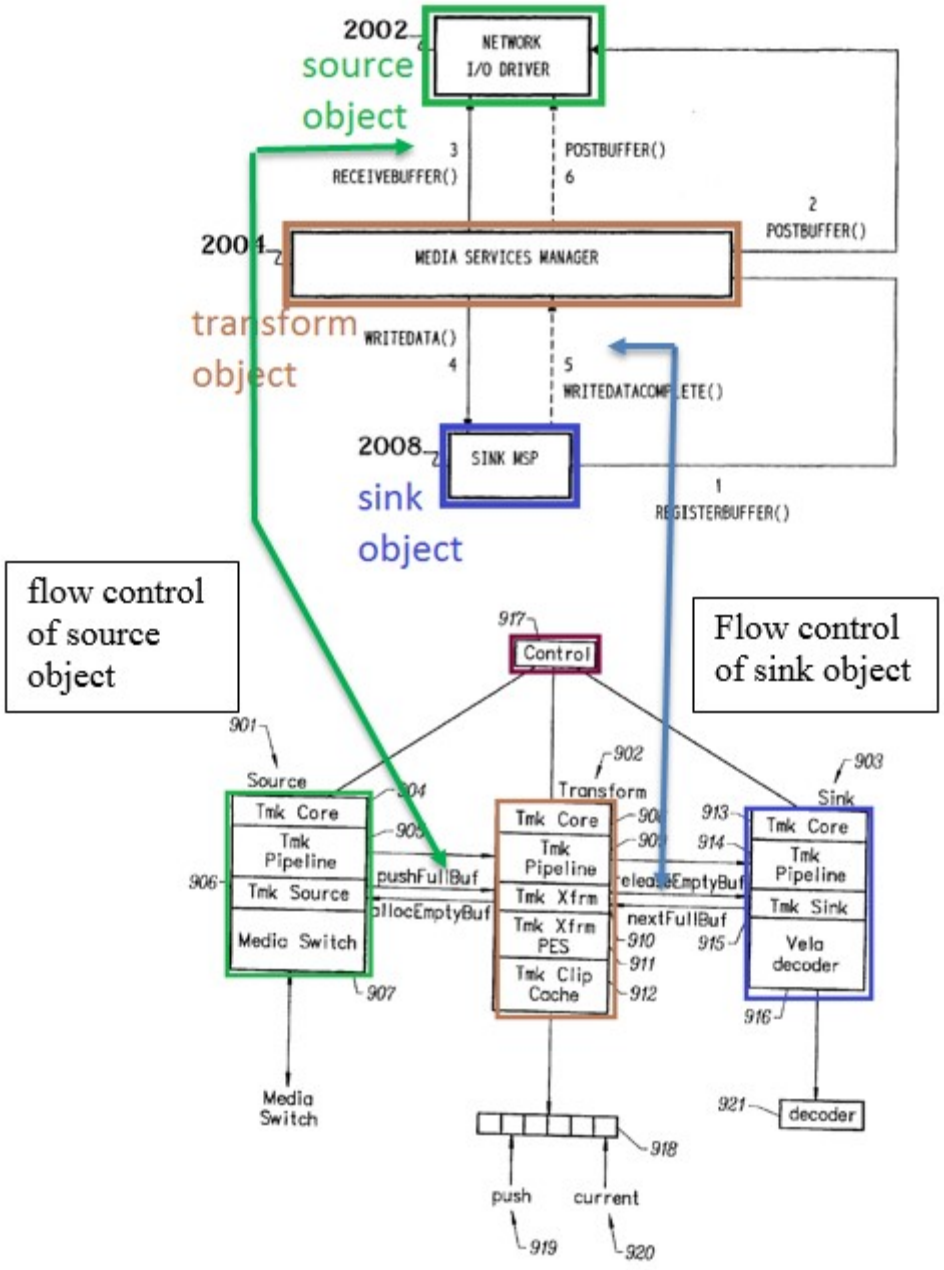

FIG. 9

Figure 19. Flow control in Sampat Fig. 20 and '389 patent Fig. 9

85. First, I will address automatic flow control of the source object. Then, I will address automatic flow control of the sink object.

ii. *Source Object - Automatic Flow Control*

86. As noted above, in Sampat, the MSM (transform object) self-regulates flow. This includes regulating flow from the network I/O driver (source object). [SE1004, 14:43-54.] Sampat explains: "***MSM 1808 manages the flow*** of data through the client software architecture" including "[f]rom the network to a sink media service provider (MSP)" and "[f]rom the network to a mass storage device." [SE1004, 14:43-54 (emphasis added).] The MSM regulates the flow of data from the source object by starting and stopping operations [SE1004, 15:37-57, 18:28-54, *see also* 17:50] and especially by regulating buffer passing during flow [SE1004, 17:39-18:5)].

87. Sampat shows and describes its transform object (MSM) controlling flow in the same way as described in the '389 patent, with the source object (network I/O driver) filling buffers only when received from the MSM. [SE1004 at 17:39-18:5); FIG. 20.] I have reproduced Sampat's Figure 20, below, and annotated it with colors and labels to show how Sampat's MSM (the transform object) regulates flow of data for Sampat's network I/O driver (the source object).

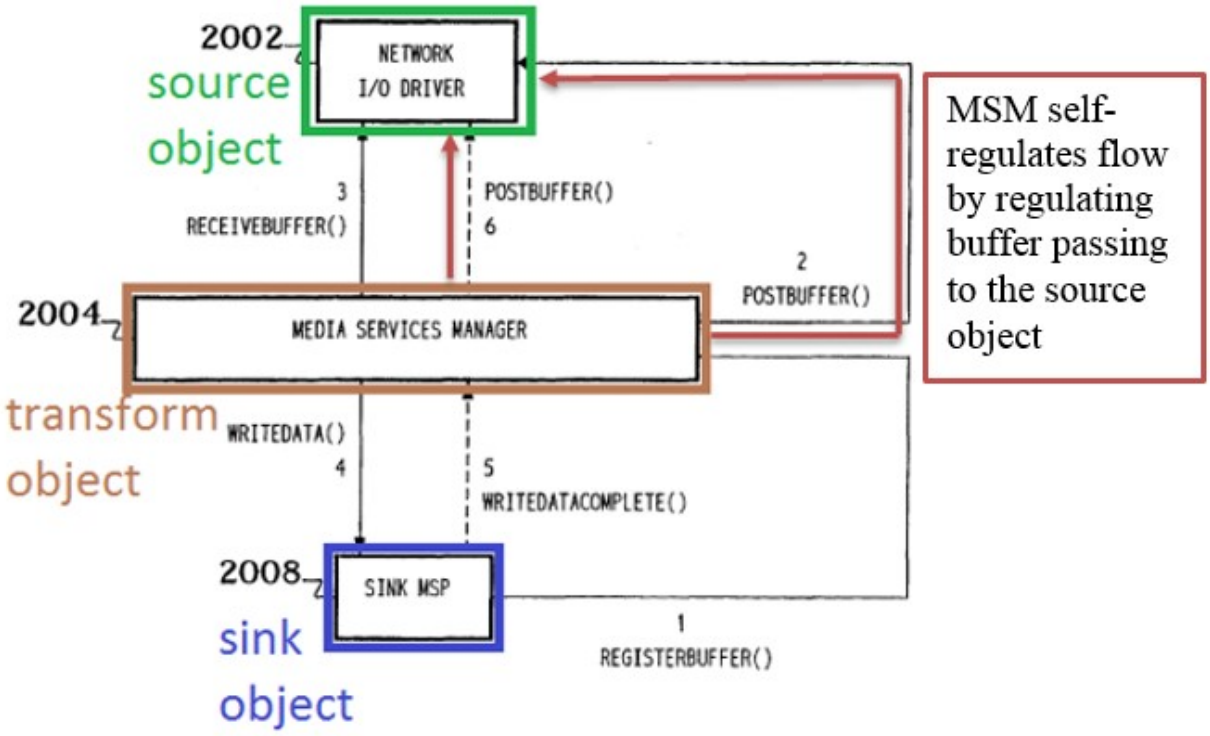

Figure 20. [SE1004, Sampat FIG. 20 (annotated)]

88. In step 2, Sampat explains: "When MSM 2004 initializes network I/O driver 2002, the MSM specifies the data streams to be received (i.e., which sink MSPs are open). MSM 2004 then posts all of the appropriate sink buffers to the network (using the MSM API function PostBuffer)." [SE1004, 17:50-54; *see also* 17:64-67 (Describing step 6 of the MSM's buffer control, illustrated above, as follows: "MSM 2004 re-posts the buffer to network I/O driver 2002 to be reused (using the PostBuffer function)").] This description in Sampat is consistent with the description of automatic flow control in the '389 patent, which states:

> The source object 901 takes data out of a physical data source, such as the Media Switch, and places it into a PES buffer. To obtain the

> buffer, the source object 901 asks the down stream object in his [*sic*] pipeline for a buffer (allocEmptyBuf). The source object 901 is blocked until there is sufficient memory. ***<u>This means</u> that the pipeline is self-regulating; it has automatic flow control***. When the source object 901 has filled up the buffer, it hands it back to the transform 902 through the pushFullBuf function.

[SE1001 at 8:45-51 (emphasis added); *see also* 8:19-9:16.]

89. Accordingly, because, like the '389 patent, Sampat's MSM passes empty buffers to and receives full buffers from the network I/O driver, a POSITA would have understood that Sampat's MSM (transform object) automatically controls the flow of Sampat's network I/O driver (source object). A POSITA would not have understood that the flow is merely reactive and independent of the MSM. For these reasons, Sampat's network I/O driver (source object) is self-regulated by Sampat's MSM (transform object). Thus, under the BRI of automatically flow controlled identified above, Sampat's network I/O driver (source object) is automatically flow controlled by Sampat's MSM (transform object).

iii. *<u>Sink Object - Automatic Flow Control</u>*

90. As noted above and indicated in Section V above, under BRI, "automatically flow controlled" includes "self-regulated" and, consequently, this claim element covers "wherein said sink object is ***self-regulated*** by said transform object." [SE1014, 101-106; SE1001, 8:52-65.]

91. In Sampat, the MSM (transform object) self-regulates flow, including flow to the sink MSPs (sink object). [SE1004, 14:43-54.] Sampat explains: "***MSM 1808 manages the flow*** of data through the client software architecture" including "[f]rom the network to a sink media service provider (MSP)" and "[f]rom mass storage device 1716 to a sink MSP." [SE1004, 14:43-54 (emphasis added).]

92. The MSM regulates the flow of data to the sink object both by starting and stopping operations [SE1004, 15:36-16:57, 18:28-54, see also 17:50)] and especially by regulating buffer passing during flow [SE1004, 17:39-18:5]. Sampat shows and describes its transform object (MSM) controlling flow in the same way as described in the '389 patent, with the sink object (sink MSP) emptying buffers only when received from the MSM. [SE1004 at 17:39-18:5; FIG. 20.] I have reproduced Sampat's Figure 20, below, and annotated it with colors and labels to show how Sampat's MSM (the transform object) regulates flow of data for Sampat's sink MSP (the sink object).

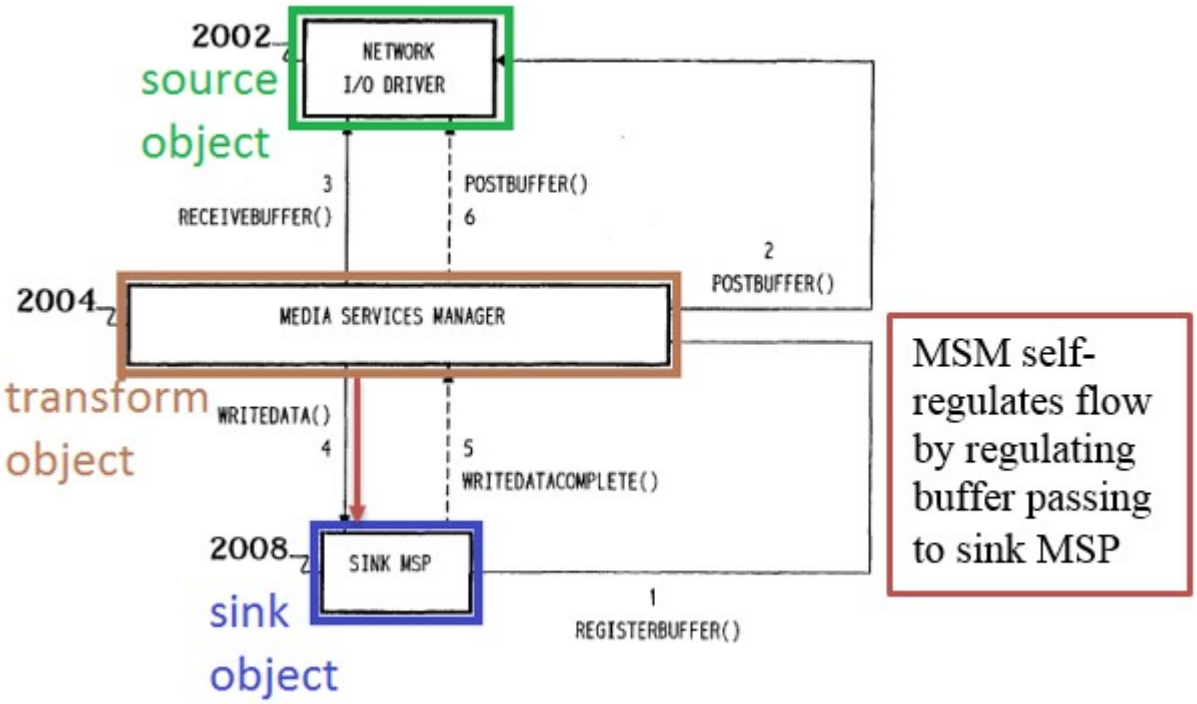

Figure 21. [SE1004, Sampat FIG. 20 (annotated)]

93. Further, Sampat describes the MSM writing the sink buffer data to the sink MSP in step 4 (shown in the figure above) before the sink MSP can play the sink buffer data in step 5. [SE1004, 17:59-63.] This description in Sampat is consistent with the description of automatic flow control in the '389 patent, which equates the sink object waiting for buffers from the transform object to being "self-regulating" and "automatic flow control." [SE1001, 8:52-65; *see also* 8:19-9:16.] As noted above, this description in Sampat is consistent with the description of automatic flow control in the '389 patent, which points out the "automatic flow-control benefit of this method" [SE1001, 8:60-61] and explains:

> The sink 903 is flow controlled as well. It calls nextFullBuf which tells the transform 902 that it is ready for the next filled buffer. This operation can block the sink 903 until a buffer is ready. When the sink 903 is finished with a buffer (i.e., it has consumed the data in the buffer) it calls releaseEmptyBuf. ReleaseEmptyBuf gives the buffer back to the transform 902. The transform 902 can then hand that buffer, for example, back to the source object 901 to fill up again.

[SE1001 at 8:45-51 *see also* 8:19-9:16.]

94. Accordingly, a POSITA would have understood that Sampat's MSM (transform object) automatically controls the flow of the sink MSP (sink object). A POSITA would not have understood that the flow is merely reactive and independent of the MSM. For these reasons, Sampat's sink MSPs (sink object) are self-regulated by Sampat's MSM (transform object). Thus, under the BRI of automatically flow controlled identified above, Sampat's sink MSPs (sink object) are automatically flow controlled by Sampat's MSM (transform object).

G. Decoder Features of Claims 31 and 61

95. Claims 31 and 61 of the '389 patent each recite "wherein said decoder converts said streams into display signals and sends said signals to a display." As discussed below, Sampat discloses these features.

96. Sampat explains "Video sink MSP 1814 and text sink MSP 1822 receive a video data stream and a text data stream, respectively, from MSM 1808 and ***transmits the video and text data to display driver 1704 of FIG. 17 for display***

on monitor 1708." [SE1004, 14:64-67 (emphasis added).] As explained above at Section VI.E.ii, Sampat's display driver 1704 is preferably Microsoft Video for Windows, which includes a decoder. [SE1004, 14:6-12; SE1007, 8, 157-211.] Figure 17 of Sampat also shows the display driver 1704 (which includes the decoder) converting streams into display signals and sending the signals to the display (monitor 1708). I have reproduced Sampat's Figure 17, below, and annotated it with colors and labels to show how Sampat's display driver 1704 includes a decoder that converts streams into display signals and sends the signals to the display.

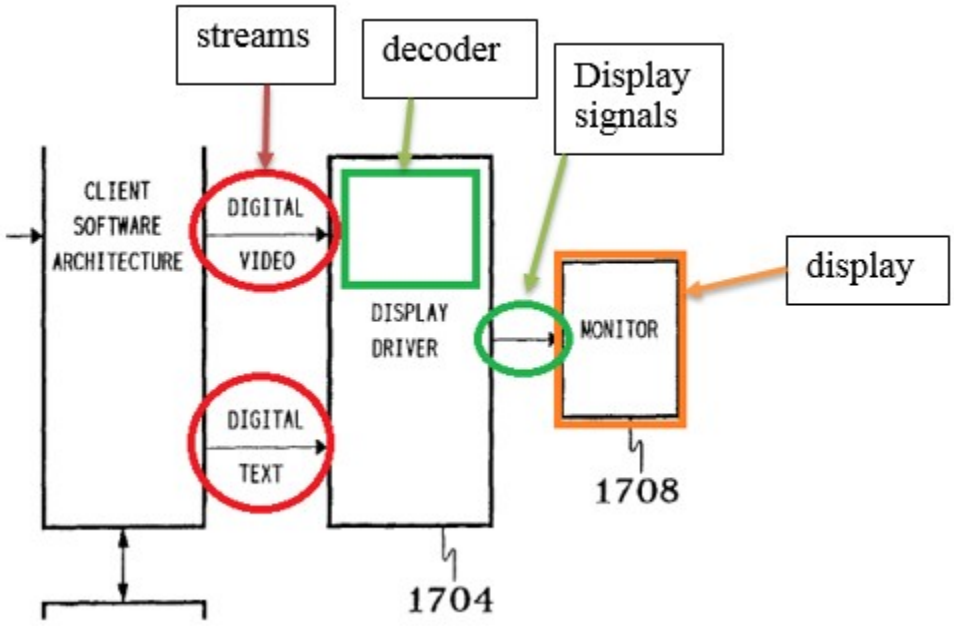

Figure 22. [SE1004, Sampat FIG. 17 (annotated)]

97. For these reasons, a POSITA would have understood that Sampat's display driver includes a decoder that converts the streams into display signals and sends the signals to a display.

H. Control Object Features of Claims 31 and 61

98. Claim 31 of the '389 patent recites providing a control object and claim 61 recites a control object. In addition, claims 31 and 61 of the '389 patent each recite "wherein said control object receives commands from a user, said commands control the flow of the broadcast data through the system," and "wherein said control object sends flow command events to said source, transform, and sink objects." As discussed below, Sampat discloses these features.

i. *Control Object – Receives Commands that Control the Flow of Broadcast Data*

99. As indicated in Section V above, under BRI, the term "control object" includes "a collection of data and operations that receives commands from a user that control the flow of broadcast data." [SE1014, 92, 98.] Sampat discloses a control object in the form of the client application 1802 and the media services manager (MSM) application programming interface (API) 1804. [SE1004, 14:21-54, 15:4-6, FIG. 18.] Client application 1802 and MSM API 1804 are a collection of data and operations. [SE1004, 14:21-15:16.] Sampat gives explicit function calls implemented in MSM API 1804 [SE1004, 15:3-7, *see also* 9:58-10:9]. I have

reproduced Sampat's Figure 18, below, and annotated it with colors and labels to show Sampat's control object colored burgundy.

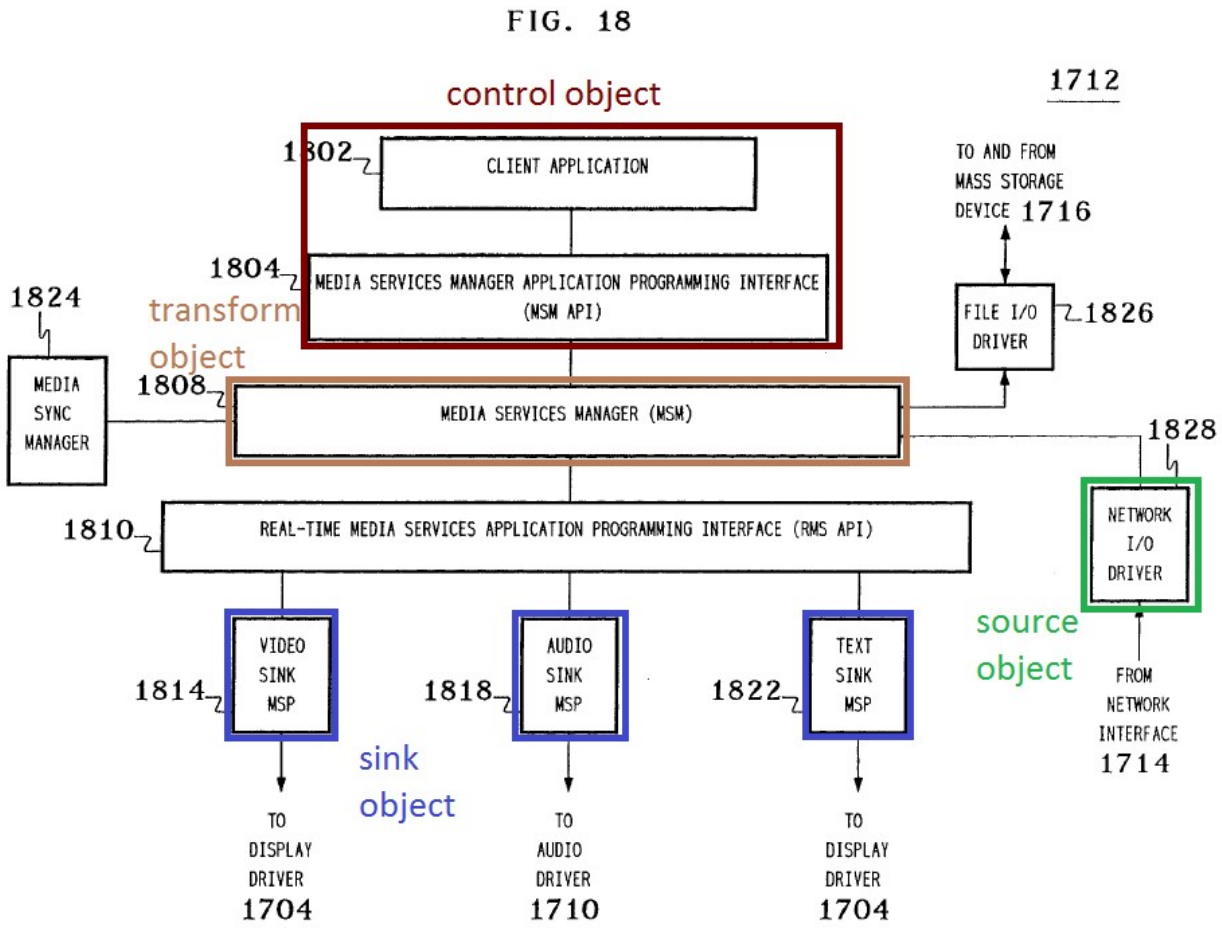

Figure 23. [SE1004, Sampat FIG. 18 (annotated)]

100. The control object's client application 1802 uses the user interface shown in Sampat Figures 2-14 to control the flow of broadcast data through the system. [SE1004, 2:17-19, 14:34-54, 39:51-54.] "Remote control window functions include changing channels; changing audio volume; and playing, recording, or rewinding the audio, video, or text components of the current

channel." [SE1004, 7:28-31.] "… the options pop-up window allows the user to temporarily pause play of the selected multicast channel … " [SE1004, 39:51-54]. Accordingly, a POSITA would have understood that Sampat's client application/MSM API controls the flow of broadcast data through the system by, for example, playing, pausing, recording, and rewinding. I have reproduced Sampat's Figure 13, below, and annotated it with colors and labels to show buttons for receiving user commands for pausing, recording, and rewinding broadcast data.

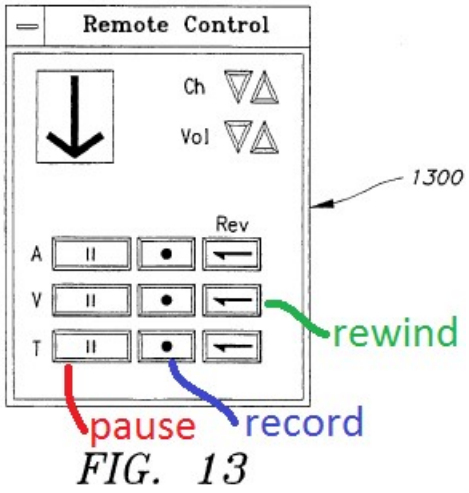

Figure 24. [SE1004, Sampat FIG. 13 (annotated)]

101. Sampat also describes user commands to perform the functions of "record, play, and rewind." [SE1004, 7:28-31; 39:55-62.] For these reasons, a POSITA would have understood the client application/MSM API in Sampat's client 104 to include a collection of data and operations that receives commands that control the flow of broadcast data through the system from the user. Thus, under the BRI of control object identified above, Sampat's client application/MSM

API includes a control object and Sampat's disclosure of providing the client application/MSM API is disclosure of providing a control object. As explained above, Sampat's client application/MSM API (the control object) receives commands from a user, and the commands control the flow of the broadcast data through the system.

ii. *Control Object – Sends Flow Command Events*

102. Sampat discloses the control object sending flow command events to the source, transform, and sink objects. As indicated in Section V above, under BRI, this claim element is interpreted broadly enough to include "the control object sends information relating to a change of condition in the flow of the broadcast data to the source, transform and sink objects." [SE1013, 16-17.]

103. Sampat explains that the control object (client application 1802 and MSM API 1804) sends flow command events to the transform object (MSM) and also to the source and sink objects via the transform object. [SE1004, 15:36-16:57.] The flow command events reach each of source, transform, and sink objects: "the MSM tells ***each appropriate MSP*** to start receiving and playing data (using the StartService function)." [SE1004, 15:66 (emphasis added)]. Sampat shows this architecture in Figure 18. I have reproduced Sampat's Figure 18, below, and annotated it with colors and labels to show Sampat's source, sink, transform, and control objects.

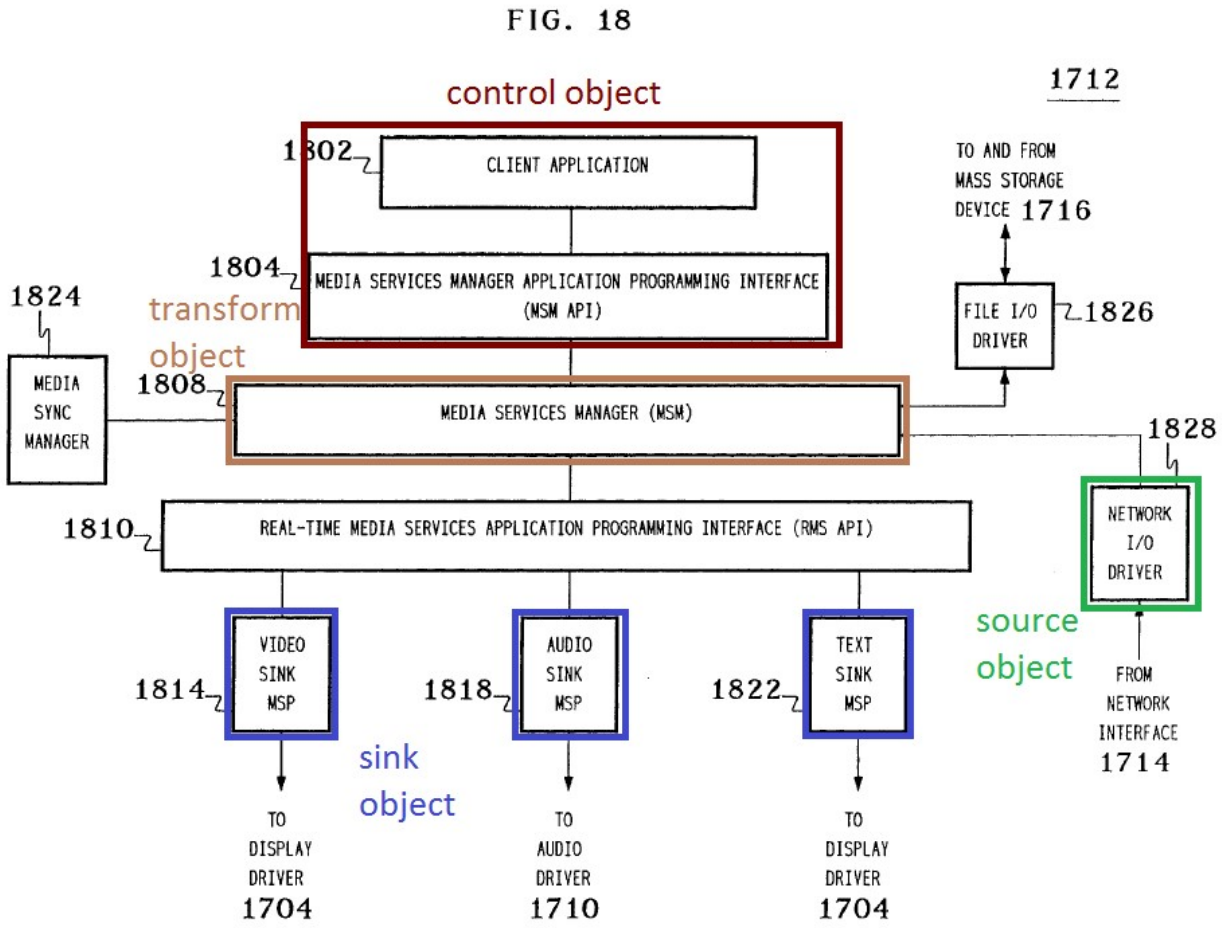

Figure 25. [SE1004, Sampat FIG. 18 (annotated)]

104. Sampat describes a number of examples of the control object sending information regarding a change of condition, such as starting, pausing, and shutting down. [SE1004, 15:36-16:57.] In one example of starting or unpausing after a pause, the user first "asks the client application to start processing specified data streams" and then the "client application passes the MSM a list of the data streams to be started." [SE1004, 15:66-16:8.] The MSM then relays information to the source and sink objects relating to the change of condition. [SE1004, 16:9-29.]

105. While information sent from the control object passes to the MSM and is relayed to the sink MSPs and network I/O driver, a POSITA would have understood that the BRI for this claim element does not require the flow command events to pass *directly* to each object, only that the information is sent to each object. [SE1013, 16-17.] For example, the '389 patent describes flow command events going first to one object and then being relayed to other objects. [SE1001, 8:21-24 ("To pause the pipeline, for example, an event called 'pause' is sent to the first object in the pipeline. The event is ***relayed on to the next object*** and so on down the pipeline.") (emphasis added).]

106. For these reasons, a POSITA would have understood that the client application/MSM API in Sampat's client 104 sends information relating to a change of condition in the flow of the broadcast data to the source, transform and sink objects. Thus, under the BRI, Sampat's client application/MSM API (the control object) sends flow command events to the source, transform, and sink objects.

VII. ANALYSIS OF SAMPAT (CLIENT SIDE) IN VIEW OF MICROSOFT VIDEO FOR WINDOWS AND SOUNDBLASTER (CLAIMS 31 AND 61)

107. As described above in Section VI, a POSITA would have understood that Sampat (client-side) discloses each and every element of claims 31 and 61. To the extent that Sampat is found not to disclose one or more claim elements, such

claim elements would have been obvious in view of the predictable combination of Sampat, Microsoft Video for Windows, and SoundBlaster. Below, I first address obviousness of a video decoder based on Microsoft Video for Windows and then address obviousness of an audio decoder based on SoundBlaster. For the other features of claims 31 and 61, my analysis in Section VI applies fully and a POSITA would have found all of the other features obvious over the disclosure in Sampat referenced in Section VI.

A. Video Decoder Features of Claims 31 and 61

108. To the extent that Sampat does not *explicitly* say that the disclosed display driver includes a video decoder, such decoders were commonly known in similar drivers. The Programmer's Guide, Microsoft Video for Windows Development Kit ("Microsoft Video for Windows") confirms that the Microsoft Video for Windows display driver includes a video decoder. [SE1007, 8 ("video compression and decompression drivers"), 157-211 (video decompression).] This is the same "Microsoft Video for Windows" display driver that Sampat describes as the preferred embodiment. [SE1004, 14:10-12 ("Display driver 1704 may be any suitable driver for displaying video and text data and is preferably Microsoft Video for Windows."); see also 29:50-55.] As Microsoft Video for Windows confirms, a POSITA would have understood that Sampat's disclosure of outputting

streams to a display driver, such as Microsoft Video for Windows, would output the streams to a video decoder.

109. Multiple reasons would have prompted a POSITA to implement the well-known prior art functionality of including a video decoder in the display driver (as suggested by Microsoft Video for Windows) in the system disclosed by Sampat.

110. First, a POSITA would have been prompted to implement a video decoder in the display driver of Sampat because Sampat uses encoded data (*e.g.*, MPEG) and video displays required encoded data to be decoded. For instance, Sampat describes the display driver 1704 as being used "for processing" prior to "display on monitor 1708." [SE1004, 13:55-57.] A POSITA would have recognized that any coded data would require decoding in order for the display to properly reproduce a video stream, and would have recognized that the display driver should include a decoder as was traditional in such systems.

111. Second, a POSITA would have been prompted to implement a video decoder in the display driver of Sampat because Sampat specifically identifies "Microsoft Video for Windows" as the preferred display driver. [SE1004, 14:10-12; see also 29:50-55.] Microsoft Video for Window included decoders. [SE1007, 8, 157-211.] This disclosure in Sampat that Microsoft Video for Windows is

preferred would have motivated a POSITA to use that very driver, which was conventional, well-known, and predictable.

112. Third, a POSITA would have been prompted to implement a video decoder in the display driver of Sampat because doing so would be merely the application of a known technique (*e.g.,* decoding video using a display driver) to a known system (*e.g.,* Sampat's system having a display driver) ready for improvement to yield predictable results. I have been told that, when a patent simply arranges old elements, with each performing the same function it had been known to perform, and yields no more than what one would expect from such an arrangement, the combination is obvious. Here, both Sampat and Microsoft Video for Windows disclose display drivers (in fact, the same display driver), and a POSITA would have recognized that applying the suggestions in Microsoft Video for Windows to Sampat's system would have led to predictable results without significantly altering or hindering the functions performed by the system of Sampat.

B. Audio Decoder Features of Claims 31 and 61

113. To the extent that Sampat does not *explicitly* say that the sink outputs to an audio decoder, such decoders were commonly known in similar drivers. For example, the Sound Blaster Pro User Reference Manual ("SoundBlaster") discloses decoding in the form of "decompression." [SE1006, 152.] In particular,

SoundBlaster discloses "ADPCM, hardware decompression" associated with various compression schemes. [SE1006, 152.] A POSITA would have understood that ADPCM decompression stands for "adaptive differential pulse-code modulation," which is used for coding and decoding an audio signal. [SE1006, 152.] SoundBlaster also discloses having a "Digital to Analog Converter," which a POSITA would have understood performs decoding of audio encoded with pulse code modulation. [SE1006, 10, 146, 152-153.] This SoundBlaster reference is the same "SoundBlaster" that Sampat describes as the preferred embodiment. [SE1004, 4:56-57; 15:1-3.] A POSITA would have understood that decompression is synonymous with decoding in this context.

114. Multiple reasons would have prompted a POSITA to implement the well-known prior art functionality of outputting to an audio decoder (as suggested by SoundBlaster) in the system disclosed by Sampat.

115. First, a POSITA would have been prompted to implement the well-known prior art functionality of outputting to an audio decoder (as suggested by SoundBlaster) in the system disclosed by Sampat because Sampat uses encoded data (e.g., MPEG) and audio hardware (e.g., an amplifier and speakers) requires encoded data to be decoded. A POSITA would have understood that Sampat's system would be unable to process encoded audio data without a decoder, and

SoundBlaster discloses a suitable and well-known option for providing such decoding. [SE1006, 10, 146, 152-153.]

116. Second, a POSITA would have been prompted to implement the well-known prior art functionality of outputting to an audio decoder (as suggested by SoundBlaster) in the system disclosed by Sampat because Sampat specifically identifies "SoundBlaster Pro from Creative Labs" as the preferred device for enabling sound. [SE1004, 4:56-57.] SoundBlaster included decoders. [SE1006, 10, 146, 152-153.] This disclosure in Sampat that SoundBlaster is preferred would have motivated a POSITA to use that very device and its decoder, which would have been conventional, well-known, and predictable.

117. Third, a POSITA would have been prompted to implement the well-known prior art functionality of outputting to an audio decoder (as suggested by SoundBlaster) in the system disclosed by Sampat because doing so would be merely the application of a known technique (*e.g.*, decoding audio using a decoder) to a known system (*e.g.*, Sampat's system having sound enabled) ready for improvement to yield predictable results. I have been told that, when a patent simply arranges old elements, with each performing the same function it had been known to perform, and yields no more than what one would expect from such an arrangement, the combination is obvious. Here, both Sampat and SoundBlaster disclose a device for audio processing (in fact, the same SoundBlaster device), and

a POSITA would have recognized that applying the suggestions in SoundBlaster to Sampat's system would have led to predictable results without significantly altering or hindering the functions performed by the system of Sampat.

C. Obviousness Conclusion for Claims 31 and 61

118. For the above reasons, a POSITA would have understood that Sampat in view of Microsoft Video for Windows and SoundBlaster renders obvious providing a sink object that obtains data stream buffers from the transform object and outputs the streams to a video and audio decoder.

119. As explained above in Section VI.G, Sampat describes the decoder (display driver 1704) converting steams into display signals and sending the signals to a display. Microsoft Video for Windows also discloses its decoder converting streams into display signals and sending those signals to a display monitor. [SE1007, 157-158 (converting compressed video data streams and sending to a "Video Display"), 158-159 ("a decompression driver with the ability to write to the video display").] Accordingly, Sampat, Microsoft Video for Windows, and SoundBlaster render obvious a decoder converting the streams into display signals and sending the signals to a display.

VIII. ANALYSIS OF SAMPAT (SERVER SIDE) (CLAIMS 31 AND 61)

120. In this section, I will address Sampat when viewed from Sampat's server 102. I have reproduced Sampat's Figure 1, below, and annotated it with

colors and labels to schematically show that, for Sampat's server side, the input devices are upstream of the server 102, and that the physical data source, as well as the source, sink, transform, and control objects, are in the server 102.

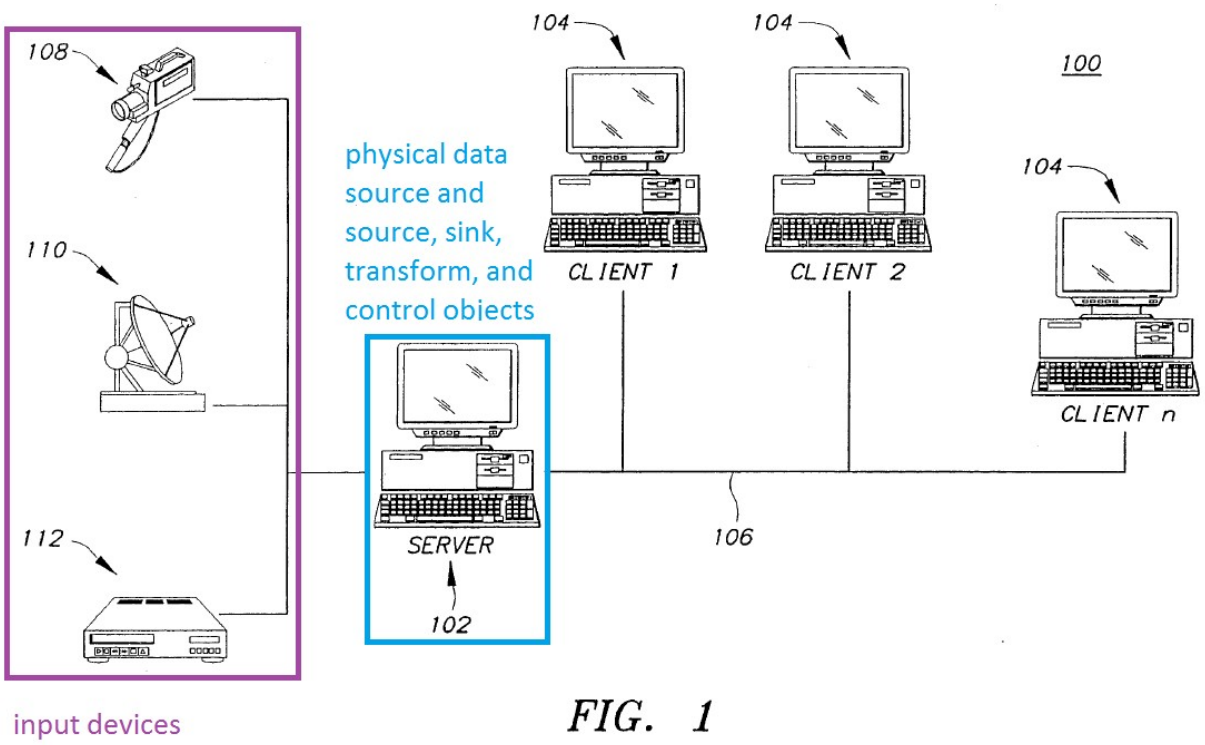

Figure 26. [SE1004, Sampat FIG. 1 (annotated)]

121. I also have reproduced Sampat's Figures 15 and 16, below, and annotated them with colors and labels to show the source, transform, sink, and control objects within the server 102.

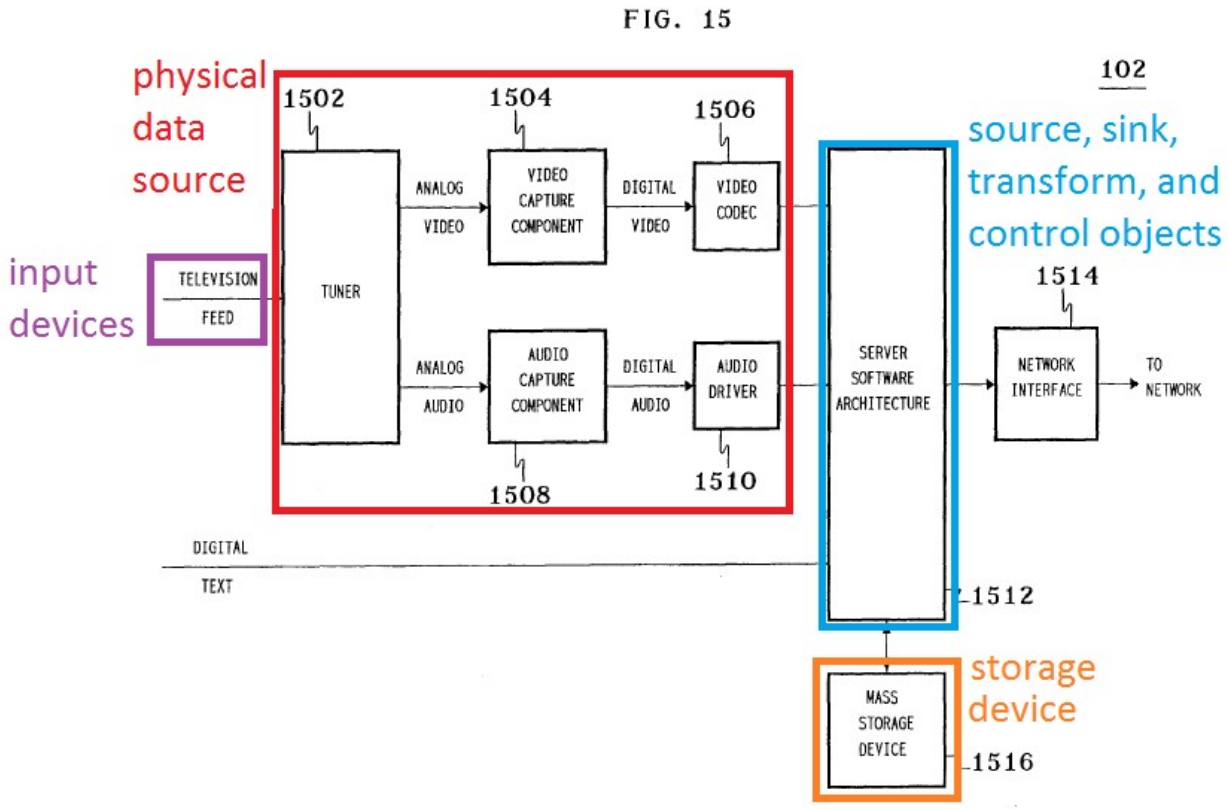

Figure 27. [SE1004, Sampat FIG. 15 (annotated)]

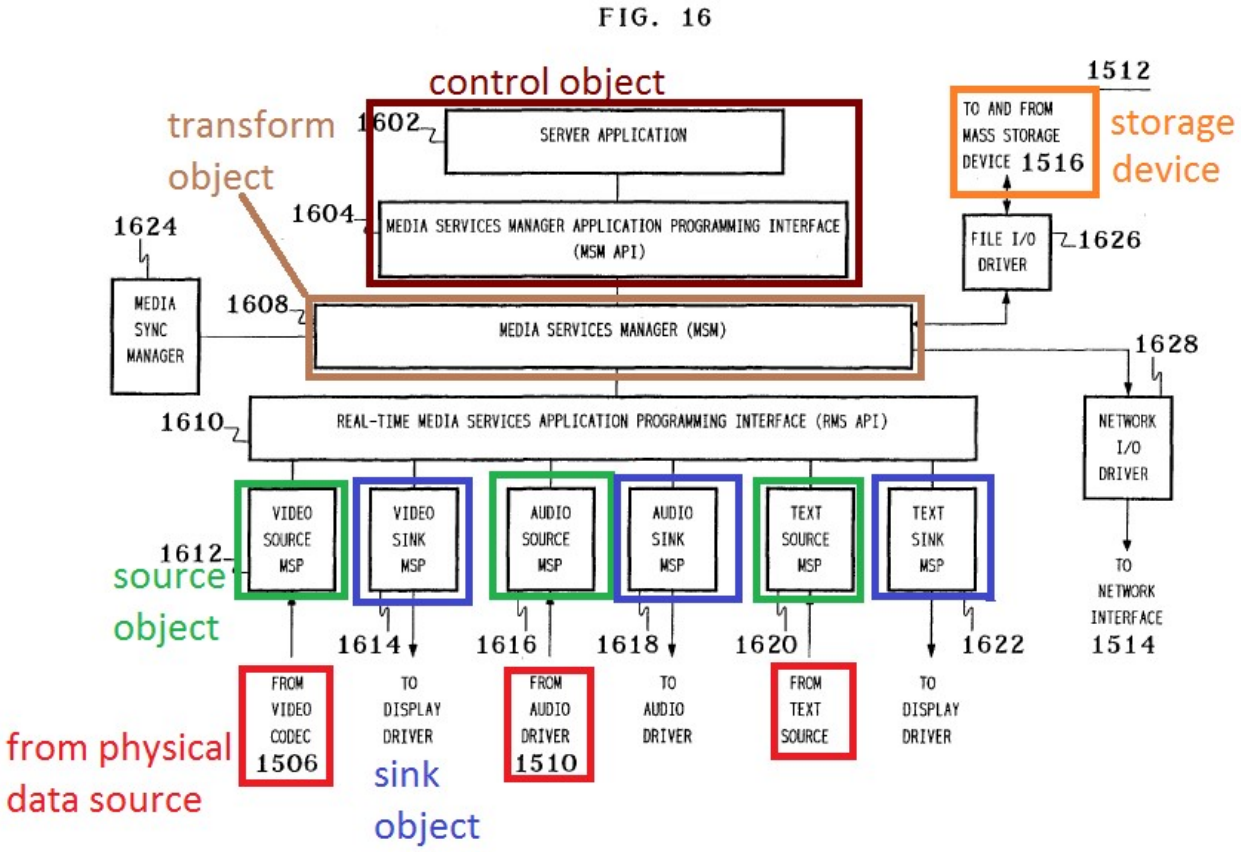

Figure 28. [SE1004, Sampat FIG. 16 (annotated)]

122. In the paragraphs that follow, I will discuss how, from the perspective of Sampat's server 102, Sampat discloses and renders obvious all features of claims 31 and 61.

A. Preambles of Claims 31 and 61

123. The preamble of claim 31 recites a "process for the simultaneous storage and play back of multimedia data" and the preamble of claim 61 recites an "apparatus for the simultaneous storage and play back of multimedia data."

124. I have been told that the Patent Owner has "agreed that the preambles of Claims 31 and 61 of the '389 Patent are not limiting." [SE1014, 82.] Accordingly, the preambles of claims 31 and 61 are not limiting.

125. Even if the preambles were limiting, my opinion would not change, because Sampat discloses a process for the simultaneous storage and play back of multimedia data. [SE1004, 8:18-22, 13:59-63, 2:10-16.] Sampat also discloses an apparatus for the simultaneous storage and play back of multimedia data. [SE1004, 8:18-22, 13:59-63, 2:10-16.]

B. Physical Data Source Features of Claims 31 and 61

126. Claim 31 of the '389 patent recites providing a physical data source and claim 61 recites a physical data source. In addition, claims 31 and 61 of the '389 patent each recite "wherein said physical data source accepts broadcast data from an input device, parses video and audio data from said broadcast data, and temporarily stores said video and audio data." As discussed below, Sampat discloses these features.

127. As indicated in Section V above, the BRI of "physical data source" includes "hardware and software that parses video and audio data from said broadcast data." [SE1014, 109-111; SE1001, 8:43-45.] The BRI of "parses video and audio data from said broadcast data" includes "analyzes video and audio data from the broadcast data." [SE1014, 82-88; SE1001, 5:3-9, 6:37-46.]

128. As viewed from the server-side, Sampat discloses a physical data source (hardware and software) in the form of the tuner 1502 in conjunction with the video capture component 1504, video codec 1506, audio capture component 1508, and audio driver 1510. [SE1004, 7:59-8:11.] I have reproduced Sampat's Figure 15, below, and annotated it with colors and labels to show the physical data source in red.

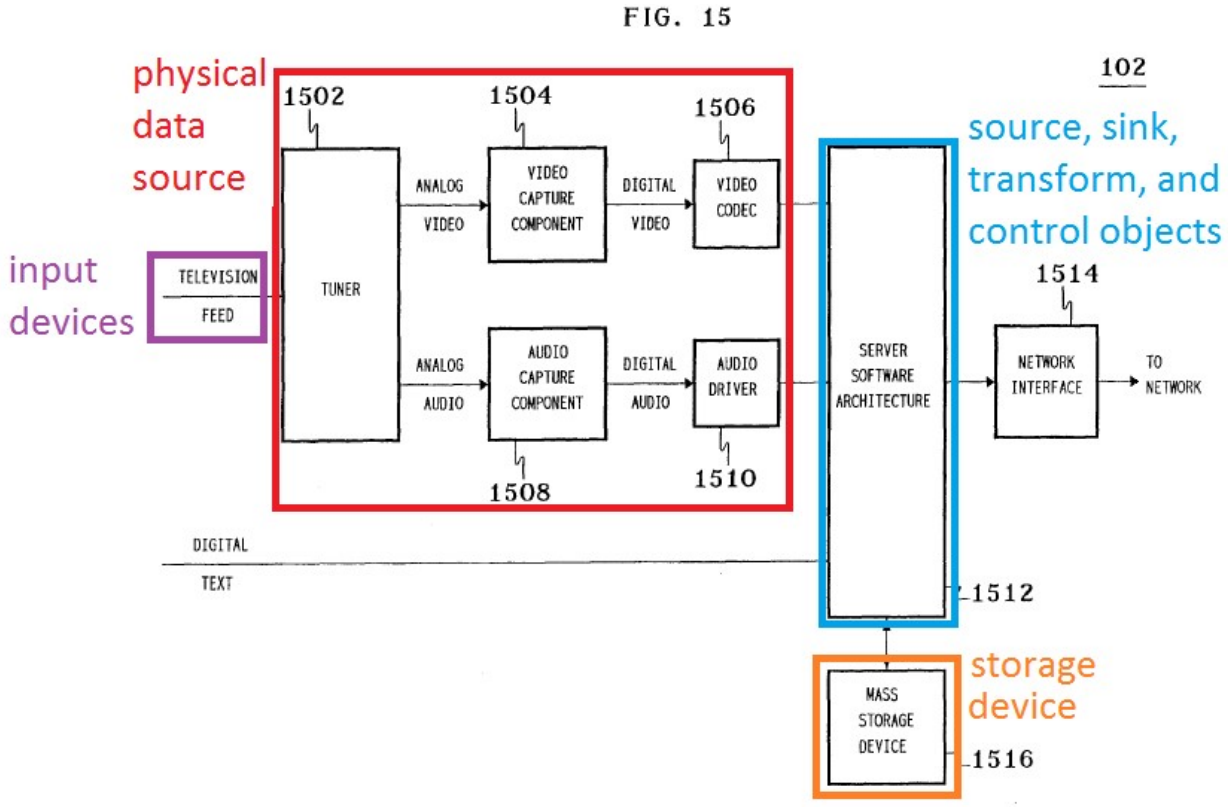

Figure 29. [SE1004, Sampat FIG. 15 (annotated)]

129. Sampat discloses physical data source (the tuner 1502, etc.) accepting broadcast data from one or more input devices (*e.g.*, camera 108, antenna 110, and VCR 112). [SE1004, 4:7-14 ("Server 102 is capable of capturing analog audio and

video signals from three different sources: (1) signals generated locally by camera 108, (2) signals received by antenna 110 from a remote source, and (3) recorded signals from VCR 112"), 38:42-54 (the system can be used with "broadcasting"); see Sampat Fig. 1, reproduced above.] "For example, server 102 may receive via antenna 110 a first television program" and "may receive a second television program … from VCR 112." [SE1004, 4:15-23.] A television program received by an antenna includes broadcast data.

130. Sampat further discloses the physical data source parsing video and audio data from the broadcast data. [SE1004, 7:55-63; 4:24-43.] "In particular, tuner 1502 of server subsystem 102 receives, demodulates, and splits one or more analog television feed signals into their constituent analog audio and video signals." [SE1004, 7:55-63.] I have reproduced Sampat's Figure 15, below, and annotated it with colors and labels to show a block diagram of the tuner 1502 parsing the broadcast data (purple) into video and audio data (brown).

FIG. 15

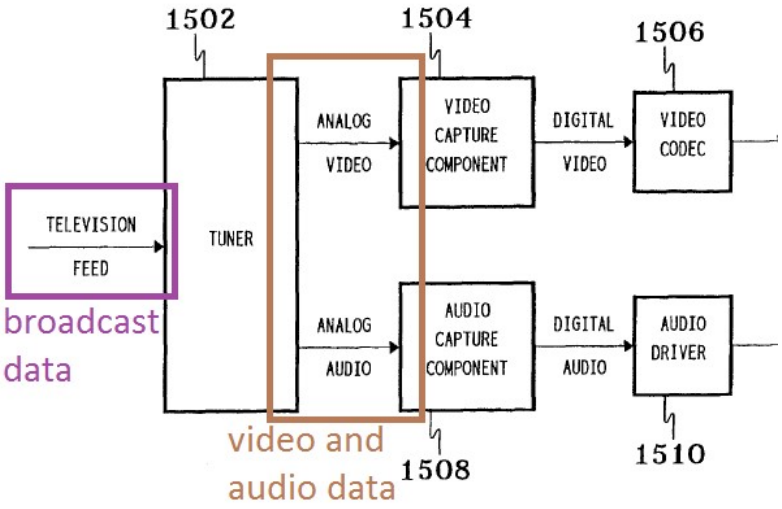

Figure 30. [SE1004, Sampat FIG. 15 (annotated)]

131. For these reasons, Sampat's tuner 1502 in conjunction with the video capture component 1504, video codec 1506, audio capture component 1508, and audio driver 1510 includes hardware and software that parses (*e.g.,* analyzes) video and audio data from broadcast data. Thus, under the BRI of physical data source identified above, Sampat's tuner 1502 in conjunction with the video capture component 1504, video codec 1506, audio capture component 1508, and audio driver 1510 is a physical data source; and Sampat's disclosure of providing the tuner 1502 in conjunction with the video capture component 1504, video codec 1506, audio capture component 1508, and audio driver 1510 is disclosure of providing a physical data source. As explained above, Sampat's tuner 1502 (the

physical data source) accepts broadcast data from an input device and parses video and audio data from said broadcast data.

132. Sampat further discloses the physical data source (Sampat's tuner 1502 in conjunction with the video capture component 1504, video codec 1506, audio capture component 1508, and audio driver 1510) temporarily storing the video and audio data. For example, Sampat discloses that the part of the physical data source that includes the video capture component 1504 and codec 1506 is hardware such as "an Intel® SmartVideo® Recorder (ISVR)" and that the part of the physical data source that includes the audio capture component 1508 is "a Creative Labs SoundBlaster Pro." [SE1004, 8:25-31.] A POSITA would have recognized that the ISVR and the Creative Labs SoundBlaster Pro would temporarily store the video and audio data as that data is being processed. Such ISVR and SoundBlaster Pro devices were well-known to a POSITA prior to the 1998 filing date of the '389 patent, and were also known to include temporary storage of video and audio data. [*See, e.g.*, SE1005, 5:10-16, 5:20-60; FIG. 3.; SE1006, 20, 92, 98, 146, 153, 158.]

133. Also, a POSITA would have understood Sampat's disclosure of "capturing" the video and audio as meaning that the components *store* the video and audio data. [SE1004, 7:62-65 ("Video *capture* component 1504 *captures* and converts the analog video signals into digital video data streams…audio *capture*

component 1508 *captures* and converts the analog audio signals into digital audio data streams") (emphasis added), 31:65-32:3 ("video *capture* component 1504 and audio *capture* component 1508 of server 102 of FIG. 15 may *capture* data at different rates") (emphasis added).] A POSITA would have understood that the term "capture" as used in this context conventionally meant that such data was stored. [SE1009, 2 ("cause (data) to be stored in a computer"); SE1010, 2 ("The recording of data…into a computer.").] Accordingly, Sampat discloses that the tuner 1502 in conjunction with the video capture component 1504, video codec 1506, audio capture component 1508, and audio driver 1510 (the physical data source) accepts broadcast data from an input device, parses video and audio data from the broadcast data, and temporarily stores the video and audio data.

134. As I noted above at Section VI.B, I understand that Patent Owner has alleged that certain components of a Samsung DVR meet this claim limitation. [SE1015, 39-48.] Under this alternative understanding, Sampat still discloses the claimed physical data source when viewed from the server-side. For example, the combination of the video capture component 1504, video codec 1506, audio capture component 1508, and audio driver 1510, would function as the physical data source without the tuner 1502, and the tuner 1502 would be considered the input device as in Patent Owner's infringement contentions. I have created an annotated view of Sampat Figure 15, below, to illustrate this alternative view.

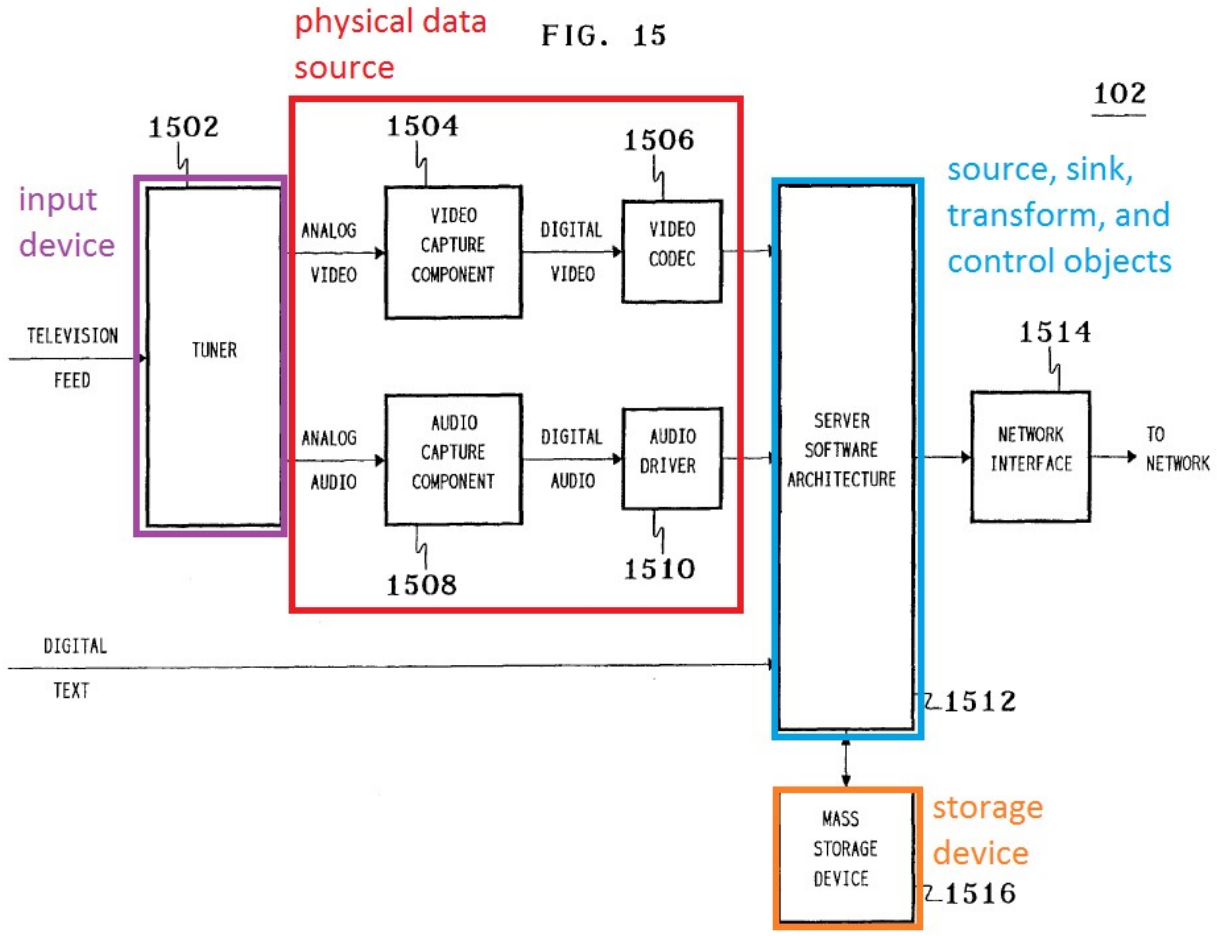

Figure 31. [SE1004, Sampat FIG. 15 (annotated)]

135. In this alternative interpretation, Sampat's disclosure of tuner 1502 is consistent with Patent Owner's allegation that the input device is met by a tuner. [SE1004, 7:59-61, 8:22-25; SE1015, 39.]

136. As I noted previously, the BRI of "parses" includes "*analyzes* video and audio data from said broadcast data." [SE1014, 82-88.] In this alternative interpretation, the physical data source of Sampat illustrated above analyzes video and audio data from the broadcast data in multiple ways. Sampat explains,

"[v]ideo capture component 1504 captures and converts the analog video signals into digital video data streams" and "audio capture component 1508 captures and converts the analog audio signals into digital audio data streams," which a POSITA would have understood includes analysis of video and audio data when converting into digital video. [SE1004, 7:62-65.] Sampat also explains that "[v]ideo codec 1506 compresses the digital video data streams" and "[a]udio driver 1510 places the audio data into buffers," which a POSITA would also understand includes analysis of video and audio data under this interpretation. [SE1004, 8:7-10.] Therefore, Sampat discloses the claimed physical data source whether the tuner is considered to be part of the physical data source, or alternatively, part of the input device.

C. Source Object Features of Claims 31 and 61

137. Claim 31 of the '389 patent recites providing a source object and claim 61 recites a source object. In addition, claims 31 and 61 of the '389 patent each recite "wherein said source object extracts video and audio data from said physical data source" and "wherein said source object obtains a buffer from said transform object, said source object converts video data into data streams and fills said buffer with said streams." As discussed below, Sampat discloses these features.

138. As indicated in Section V above, the BRI for "source object" includes "a collection of data and operations that (1) extracts video and audio data from a physical data source, (2) obtains a buffer [memory where data can be temporarily stored for transfer] from a transform object, (3) converts video data into data streams, and (4) fills the buffer [memory where data can be temporarily stored for transfer] with the streams." [SE1014, 9, 91-98; SE1001, 8:39-51.]

139. As viewed from the server-side, Sampat describes a source object in the form of one or more source MSPs, explaining: "A source MSP is a media service provider that assists in the receipt of a data stream from an external source or local mass storage device." [SE1004, 9:37-40.] The source MSPs are a collection of data and operations. [SE1004, 9:7-56.] The source MSPs also perform each of the four functions defined as being performed by a source object. These four functions are each addressed in turn below.

i. *First Function - Extracts Video and Audio Data from a Physical Data Source*

140. The source MSPs extract audio and video from the physical data source (*e.g.*, from the video codec 1616 and audio driver 1510). [SE1004, 9:46-51 ("Video source MSP 1612 receives a video data stream from video codec 1506 of FIG. 15 and transmits the video data to MSM 1608. Similarly, audio source MSP 1616 and text source MSP 1620 receive audio and text data streams from audio

driver 1510 and the text source, respectively, and transmit the audio and text data to MSM 1608.").] I have reproduced Sampat's Figure 16, below, and annotated it with colors and labels to show how Sampat's source MSPs (source object, colored green) extract video and audio data from the physical data source (colored red).

FIG. 16

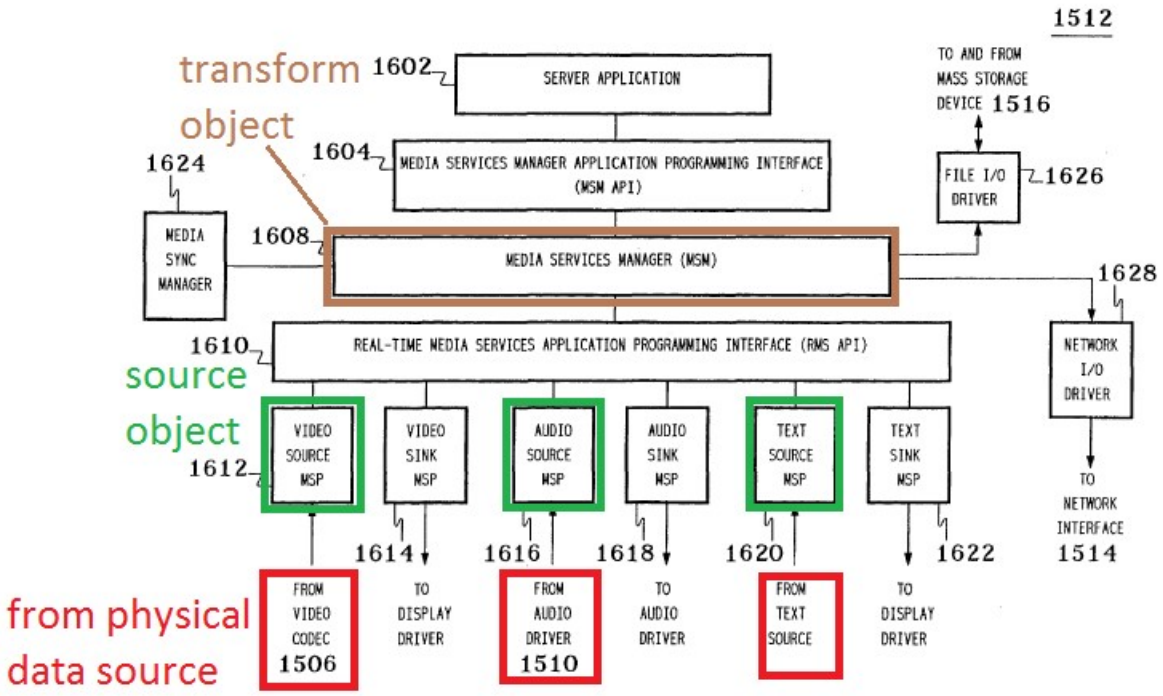

Figure 32. [SE1004, Sampat FIG. 16 (annotated)]

ii. *Second Function - Obtains a Buffer from a Transform Object*

141. Sampat describes the source object (source MSPs) obtaining a buffer from the transform object (MSM), stating: "MSM recycles the buffer to the appropriate source MSP." [SE1004, 12:64-13:3]. Sampat states further: "MSM 1904 returns the source buffer to source MSP 1906 for reuse (using the RMS API

function RecycleBuffer)" [SE1004, 17:36-38]. As explained above at Section IV.B and below at Section VIII.D, Sampat's MSM is a transform object. Thus, in Sampat, the source object (source MSPs) obtains a buffer from the transform object (MSM). I have reproduced Sampat's Figure 19, below, and annotated it with colors and labels to show how Sampat's source MSP (source object) obtains a buffer from Sampat's MSM (transform object).

FIG. 19

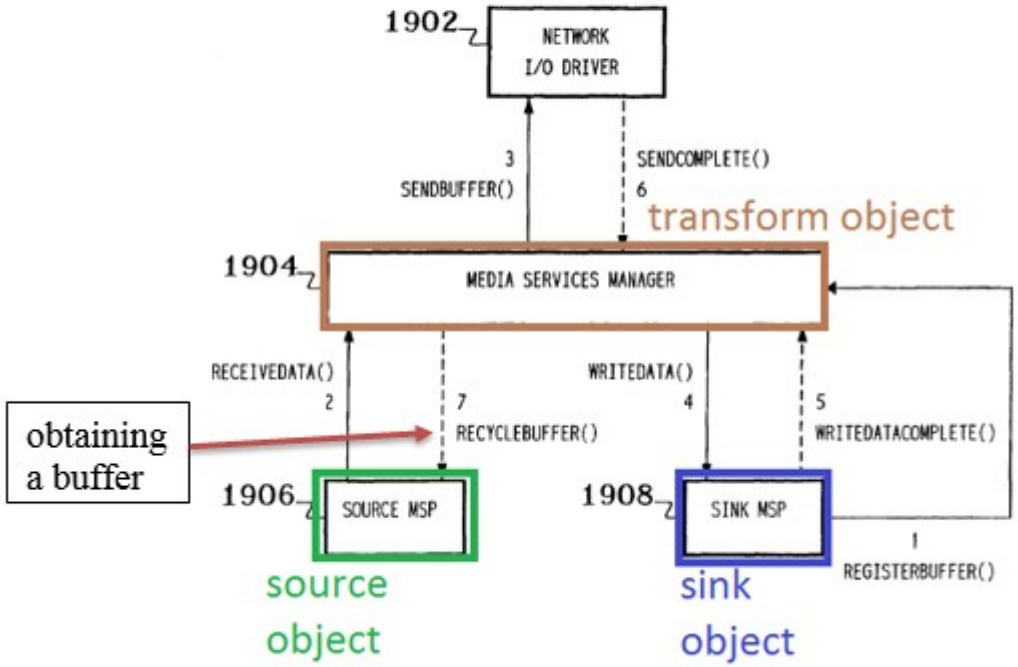

Figure 33. [SE1004, Sampat FIG. 19 (annotated)]

iii. *Third Function - Converts Video Data into Data Streams*

142. Sampat discloses that the server converts video data into data streams. [SE1004, 4:23-24 ("Server 102 digitizes the received analog audio and video signals to generate digital audio and video data streams.")].

143. More specifically, Sampat discloses that the source object (source MSP) converts video data into time-stamped data streams, "as the source MSPs stamp their data with the appropriate capture time." [SE1004, 11:46-55.]

> In the server, when a source MSP (1612, 1616, or 1620 of FIG. 16) receives new data, the MSP asks MSM 1608 for a new time-stamp from media sync manager 1624, which the MSP adds to the data header before sending the data to MSM 1608 for transmission to the network and/or storage to mass storage device 1516.

[SE1004, 32:6-18.]

Sampat emphasizes the importance of time stamping the data stream:

> Upon capturing new data, the MSP asks the MSM for an appropriate time stamp value for the MSP's new data packet (using the NewSyncStamp function). All MSP data packets are preferably time stamped even if they are not being synchronized with other data from other MSPs. [¶] The MSP delivers the time-stamped data packet to the MSM (using the ReceiveData callback function).

[SE1004, 12:46-53.]

> Media sync manager 1624 provides time stamps for the component data streams.

[SE1004, 11:46-47.]

From this description, a POSITA would have understood that Sampat's source MSP converts video data into data streams.

iv. *Fourth Function - Fills the Buffer with the Streams*

144. Sampat discloses that the source object (source MSPs) fills the buffer with data streams, stating: "The source MSP is then free to refill the buffer." [SE1004, 13:2-4, 17:16-19 ("fills them with data").] I have reproduced Sampat's Figure 19, below, and annotated it with colors and labels to show how Sampat's source MSP (source object) obtains a buffer from the MSM and sends the filled buffer back to the MSM.

FIG. 19

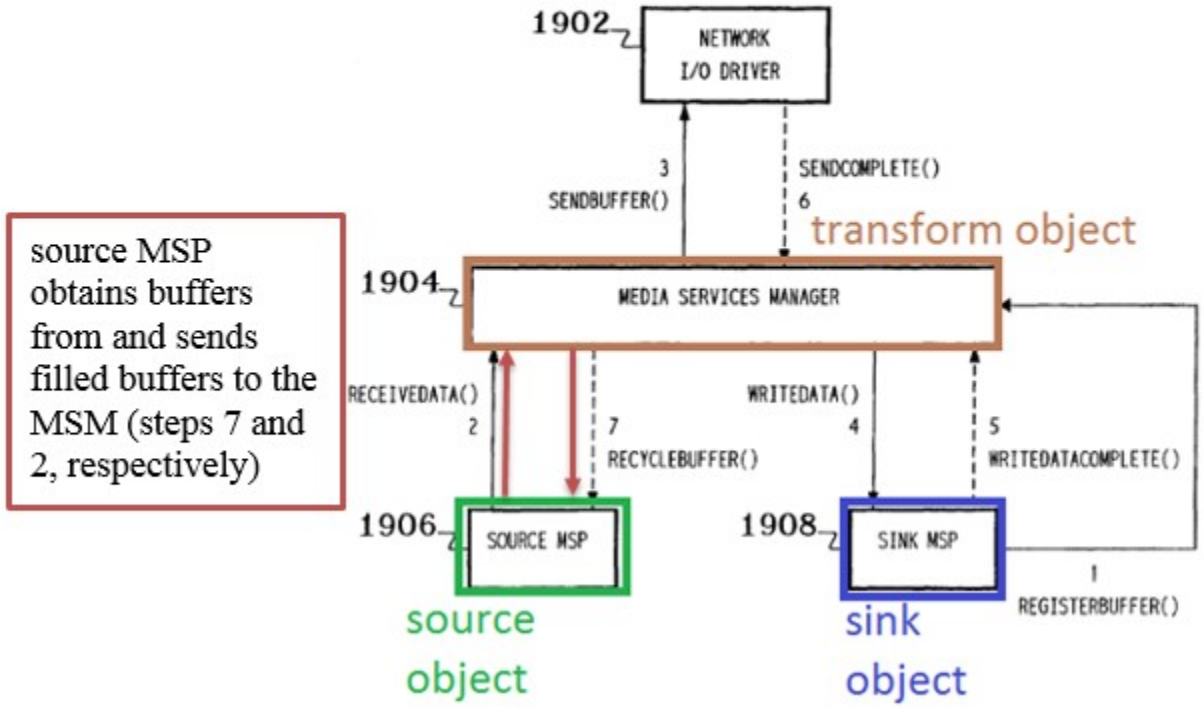

Figure 34. [SE1004, Sampat FIG. 19 (annotated)]

v. *Source Object - Conclusion*

145. For these reasons, a POSITA would have understood the source MSP in Sampat's server 102 to include a collection of data and operations that performs all four functions included in the BRI of the term source object. Thus, under the BRI of source object identified above, Sampat's source MSP includes a source object and Sampat's disclosure of providing the source MSP is disclosure of providing a source object. As explained above, Sampat's source MSP (the source object) extracts video and audio data from a physical data source, obtains a buffer from a transform object, converts video data into data streams, and fills the buffer with the streams.

D. Transform Object Features of Claims 31 and 61

146. Claim 31 of the '389 patent recites providing a transform object and claim 61 recites a transform object. In addition, claims 31 and 61 of the '389 patent each recite "wherein said transform object stores and retrieves data streams onto a storage device." As discussed below, Sampat discloses these features.

147. As indicated in Section V above, under BRI, a "transform object" is "a collection of data and operations that transforms the form of data upon which it operates." [SE1014, 101-106; SE1001, 7:48-8:65.] The '389 patent describes one type of transform as a ***temporal*** transform. [SE1001, 7:64-8:8 (describing a

transform as "temporal" when it "writes the buffer to a file 804 on the storage medium" and then pulls the buffer out "at a later time").]

148. A POSITA would have understood that Sampat describes a transform object that performs a temporal transform by describing the "media services manager" or "MSM" for short (called MSM 1608 in Figure 16 and MSM 1904 in Figure 19). [SE1004, 9:10-12, 17:8-38, FIGS. 16 and 19.] The MSM is a collection of data and operations. [SE1004, 8:50-9:45, FIGS. 16 and 19.]

149. Sampat describes the MSM transforming the form of data upon which it operates by performing a temporal transform, which the MSM performs by writing data to the mass storage device 1516 and retrieving the data at a later time for playback. [SE1004, 10:10-11 ("MSM 1608 uses file I/O driver 1626 to store and retrieve data to and from mass storage device 1516."); 8:17-21; FIGS. 15-16.] Sampat describes the MSM temporarily storing and retrieving data flowing:

> through MSM1608 … [f]rom a source MSP to mass storage device 1516 (for storage of data received from an external source for subsequent processing), [f]rom mass storage device 1516 to the network (for multicasting of locally recorded data), and [f]rom mass storage device 1516 to a local sink MSP (for monitoring the processing of locally recorded data).

[SE1004, 9:10-26.]

150. The MSM 1608 is shown transferring data streams "to and from mass storage device 1516" in FIG. 16 via the file I/O driver 1626. [SE1004, FIG. 16, *see also* FIG. 15.] I have reproduced Sampat's Figure 16, below, and annotated it with colors and labels to show how the MSM stores and retrieves data streams "to and from mass storage device 1516."

FIG. 16

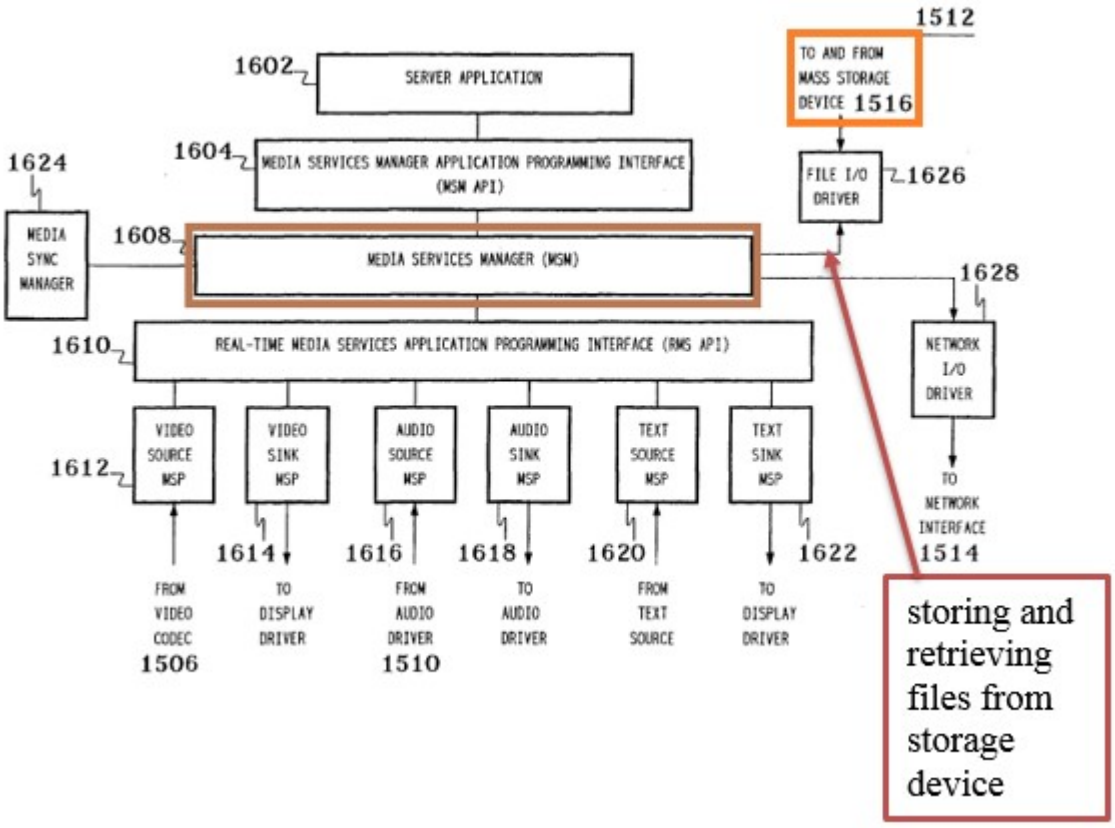

Figure 35. [SE1004, Sampat FIG. 16 (annotated)]

151. The temporal transform performed by Sampat's MSM is consistent with the '389 patent's temporal transform, which involves writing a buffer to a file on a storage medium and then pulling the buffer out at a later time. [*See* SE1001,

7:64-8:8 (describing a transform as "temporal" when it "writes the buffer to a file 804 on the storage medium" and then pulls the buffer out "at a later time").] Because Sampat's MSM writes data streams as files on a storage medium and then pulls the data streams out at a later time, Sampat discloses a transform object that transforms the form of data upon which it operates in a manner similar to the '389 patent. [*See* SE1004, 8:17-21; 9:10-26; 10:10-11; FIGS. 15-16.]

152. For example, Sampat's MSM performs a temporal transform by writing data streams to the mass storage device 1516 and retrieving the data streams at a later time for playback. [SE1004, 8:17-21; 9:10-26; 10:10-11; FIGS. 15-16.] The annotated version of Sampat's FIG. 16 above shows MSM 1808 transferring data streams "to and from mass storage device 1516." [SE1004, 8:17-21; 9:10-26; 10:10-11; FIGS. 15-16.] By writing/retrieving data streams to/from a mass storage device, Sampat's transform object (MSM) stores and retrieves data streams onto a storage device.

153. For these reasons, a POSITA would have understood the MSM in Sampat's server 102 to include a collection of data and operations that transforms the form of data upon which it operates. Thus, under the BRI of transform object identified above, Sampat's MSM includes a transform object and Sampat's disclosure of providing the MSM is disclosure of providing a transform object. As

explained above, Sampat's MSM (the transform object) stores and retrieves data streams onto a storage device.

E. Sink Object Features of Claims 31 and 61

154. Claim 31 of the '389 patent recites providing a sink object and claim 61 recites a sink object. In addition, claims 31 and 61 of the '389 patent each recite "wherein said sink object obtains data stream buffers from said transform object and outputs said streams to a video and audio decoder." As discussed below, Sampat discloses these features.

155. As indicated in Section V above, under BRI, "sink object" includes "a collection of data and operations that (1) obtains data stream buffers [memory where data can be temporarily stored for transfer] from a transform object and (2) outputs the streams to a video and audio decoder." [SE1014, 9, 91-98; SE1001, 8:52-65.]

156. Sampat describes a sink object in the form of three sink MSPs: the video sink MSP 1614, text sink MSP 1622, and audio sink MSP 1618. [SE1004, 9:50-53.] These sink MSPs are a collection of data and operations. [SE1004, 9:39-53.]

157. In Sampat, the server 102 processes data to be (a) multicast to clients 104 and/or (b) played locally on the monitor of the server 102. [SE1004, 3:66-4:6, 9:10-26, FIG. 1.] The purpose of the sink MSP is for locally playing or recording a

data stream. [SE1004, 9:39-40 ("A sink MSP is a media service provider that assists in the local playing or recording of a data stream.").] The sink MSPs in the server 102 are used for playing video and audio in substantially the same manner as described with respect to the client 104, for example:

> ***Server software architecture 1512*** also preferably has video, audio, and text sink MSPs 1614, 1618, and 1622 to provide ***local monitoring*** capabilities. The processing of sink MSPs is described in further detail later in this specification ***in conjunction with FIG. 18 and the discussion of the client software architecture***.

[SE1004, 9:50-57, emphasis added.]

158. "Preferably, ***each server has all the functionality of a client*** to provide monitoring capabilities." [SE1004, 38:34-36 (emphasis added).] Accordingly, the disclosure in Sampat with respect to monitoring (playing video and audio) on the client 104 (*see* Section VI.E) also applies to the server 102.

159. With this functionality in the server 102, the server sink MSPs perform each of the two functions defined as being performed by a sink object. These two functions are each addressed in turn below.

i. *First Function - Obtains Data Stream Buffers from a Transform Object*

160. Sampat explains that the sink MSPs (sink object) obtain data stream buffers from the MSM (transform object), stating: "MSM 1904 will copy the

source buffer data into the next available sink buffer, and write the sink buffer to be played by sink MSP 1908." [SE1004, 17:24-27, *see also* 14:64-15:3.] I have illustrated this buffer passing between the MSM and the sink MSP using color annotations and labelling in the version of Figure 19 below.

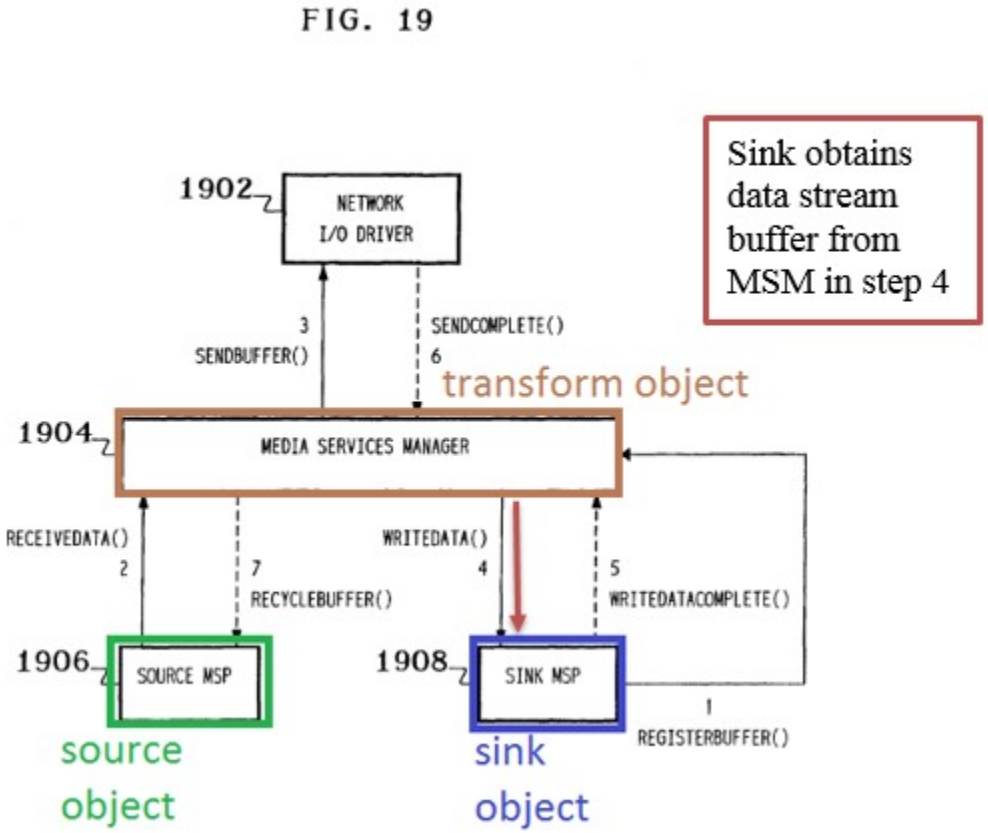

Figure 36. [SE1004, Sampat FIG. 19 (annotated)]

161. In fact, Sampat describes "data … flow from the MSM 1904 to a sink MSP 1908" in that "MSM 1904 will copy the source buffer data into the next available sink buffer, and write the sink buffer to be played by sink MSP 1908." [SE1004, 16:60-17:27.] Especially "[i]f local monitoring is selected, then the MSM sends a copy of the new data to the appropriate server sink MSP (using the

WriteData function)." [SE1004, 12:61-63.] The WriteData function, shown in Sampat Fig. 19 above, "[n]otifies a sink MSP that MSM has data for the sink MSP to play." [SE1004, 11:20-21.] From these disclosures, a POSITA would have understood the sink MSPs as receiving the data stream buffers from the MSM.

ii. *Second Function - Outputs the Streams to a Video and Audio Decoder*

162. Sampat also teaches the sink MSPs outputting the data streams to a video and audio decoder. For example, Figure 16 of Sampat shows the video sink MSP and audio sink MSP outputting streams to display and audio drivers. I have reproduced Sampat's Figure 16, below, and annotated it with colors and labels to show how the video sink MSP and audio sink MSP output streams to display and audio drivers.

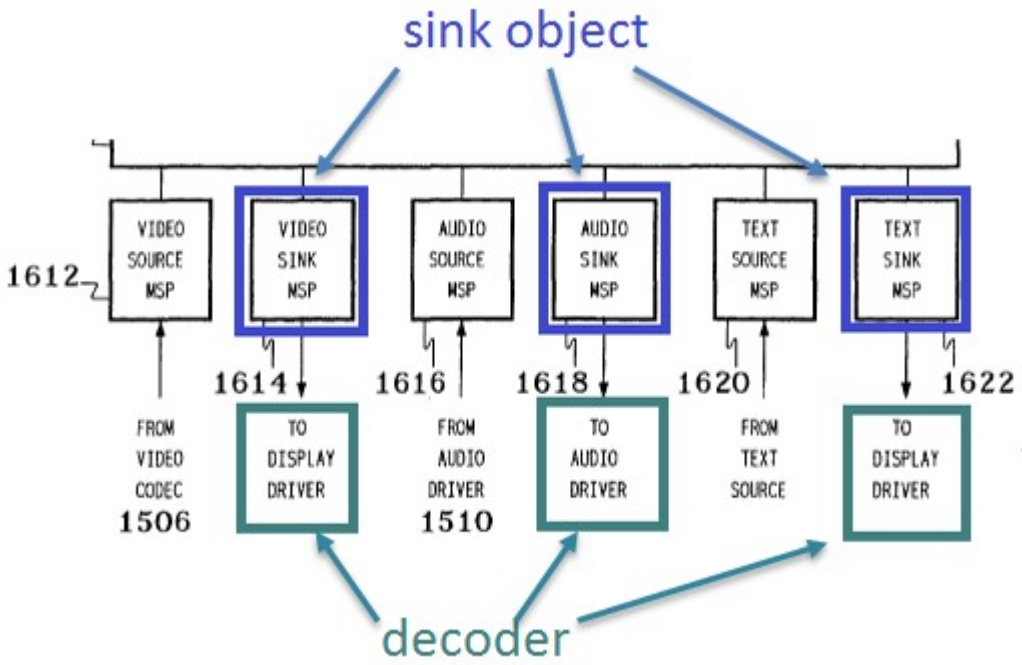

Figure 37. [SE1004, Sampat FIG. 16 (annotated)]

163. Sampat explains that the sink MSPs of the server function similarly to those of the client, for which Sampat explains:

> Video sink MSP 1814 and text sink MSP 1822 receive a video data stream and a text data stream, respectively, from MSM 1808 and ***transmits the video and text data to display driver 1704*** of FIG. 17 for display on monitor 1708. Similarly, audio sink MSP 1818 receives an audio data stream from MSM 1808 and ***transmits the audio data to audio driver 1710 for play on audio hardware 1702***.

[SE1004, 14:64-15:3 (emphasis added).] The fact that two sink MSPs are involved is consistent with the BRI of sink object, in which an object is "a collection of data and operations" (see Section V).

164. As explained above in Section VI.E.ii, a POSITA would have understood that the system of Sampat includes audio and video decoders and that the sink object would output to the audio and video decoders when monitoring on the client 104. [SE1004, 14:6-12, 4:56-59, 14:64-15:3; SE1007, 8, 157-211; SE1021, 2; SE1006, 10, 146, 152-153; SE1001, 4:23-29, 6:63-65. FIG. 7.] Because "each server has all the functionality of a client to provide monitoring capabilities" [SE1004, 38:34-36], a POSITA would have understood that the system of Sampat includes audio and video decoders and that the sink object would output to the audio and video decoders when monitoring on the server 102 for the same reasons. [SE1004, 14:6-12, 4:56-59, 14:64-15:3; SE1007, 8, 157-211; SE1021, 2; SE1006, 10, 146, 152-153; SE1001, 4:23-29, 6:63-65.; FIG. 7.]

iii. *Sink Object - Conclusion*

165. For these reasons, a POSITA would have understood the sink MSPs in Sampat's server 102 to include a collection of data and operations that perform each of two functions included in the BRI of the term sink object. Thus, under the BRI of sink object identified above, Sampat's sink MSPs include a sink object and Sampat's disclosure of providing the sink MSPs is disclosure of providing a sink

object. As explained above, Sampat's sink MSPs (the sink object) obtain data stream buffers from the MSM (the transform object) and output the streams to a video and audio decoder.

F. Automatic Flow Control Features of Claims 31 and 61

i. *Automatic Flow Control - Construction*

166. Claims 31 and 61 of the '389 patent each recite "wherein said source object is automatically flow controlled by said transform object" and "wherein said sink object is automatically flow controlled by said transform object." As discussed below, Sampat discloses these features.

167. As indicated in Section V above, under BRI, "automatically flow controlled" includes "self-regulated" and, consequently, this claim element includes "wherein said source object is ***self-regulated*** by said transform object." [SE1014, 101, 106 (emphasis added).] District courts have repeatedly reached this construction and agreed that the analysis in the second reexamination of the '389 patent "did not limit or redefine" this relatively broad construction of "automatically flow controlled." [SE1014, 106; SE1013, 12.] The most recent district court ruling is consistent with earlier rulings and explained: "TiVo merely reiterated during reexamination that ***rather than the source object and sink object being merely reactive to the data flow, they are flow controlled by the transform object***." [SE1014 , 106 (emphasis added).] Accordingly, under BRI, a POSITA

would have understood that the arguments made in the second reexamination of the ’389 patent do not require narrowing of this construction. Thus, the BRI of this claim term covers being “self-regulated” by the transform object.

168. In Sampat, the MSM (transform object) self-regulates flow, including flow from the source MSPs (source object). [SE1004, 9:10-26.] Sampat explains: “Media services manager (MSM) 1608 ***manages the flow of data*** through server software architecture 1512” including “[f]rom a source MSP to a local sink MSP” and “[f]rom a source MSP to mass storage device 1516.” [SE1004, 9:10-26.] The following illustration compares FIG. 9 of the ’389 patent to FIG. 19 of Sampat to show the flow control in both systems.

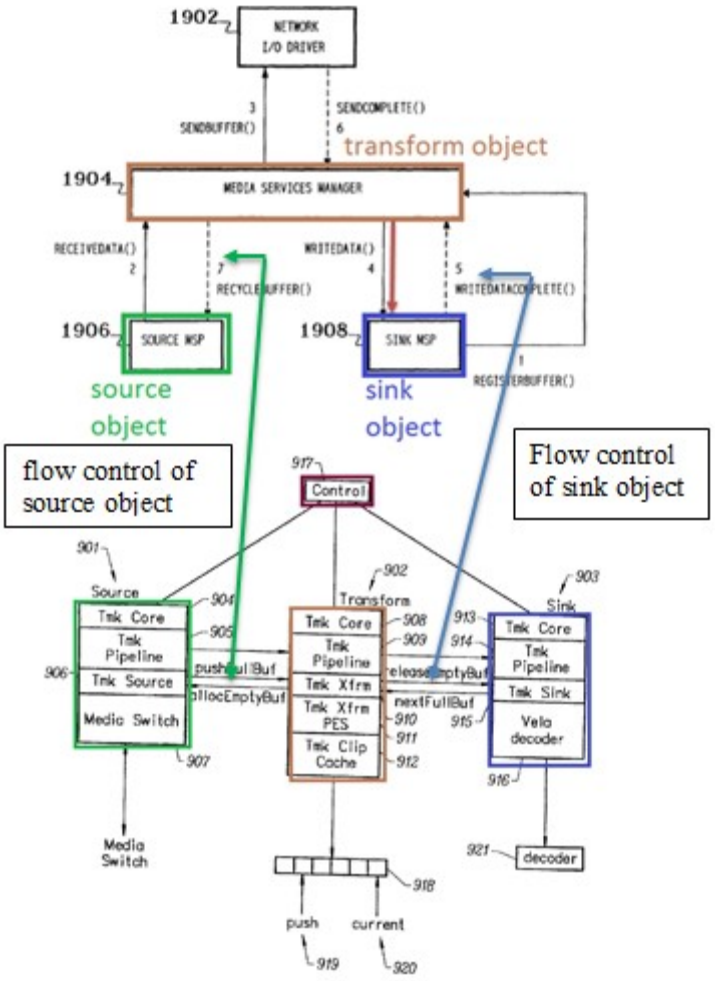

FIG. 9

Figure 38. Flow control in Sampat Fig. 19 and '389 patent Fig. 9.

169. First, I will address automatic flow control of the source object. Then, I will address automatic flow control of the sink object.

ii. *Source Object - Automatic Flow Control*

170. The MSM regulates the flow of data from the source object both by starting and stopping operations [SE1004, 9:57-10:7, 11:65-12:2, *see also* 10:26-

29] and especially by regulating buffer passing during flow [SE1004, 16:60-17:39]. Sampat shows and describes its transform object (MSM) controlling flow in the same way as described in the '389 patent, with the source object (source MSP) filling buffers only when received from the MSM. [SE1004, 16:60-17:7; 17:35-38; FIG. 19.] I have reproduced Sampat's Figure 19, below, and annotated it with colors and labels to show how Sampat's MSM (the transform object) regulates flow of data for Sampat's source MSP (the source object).

FIG. 19

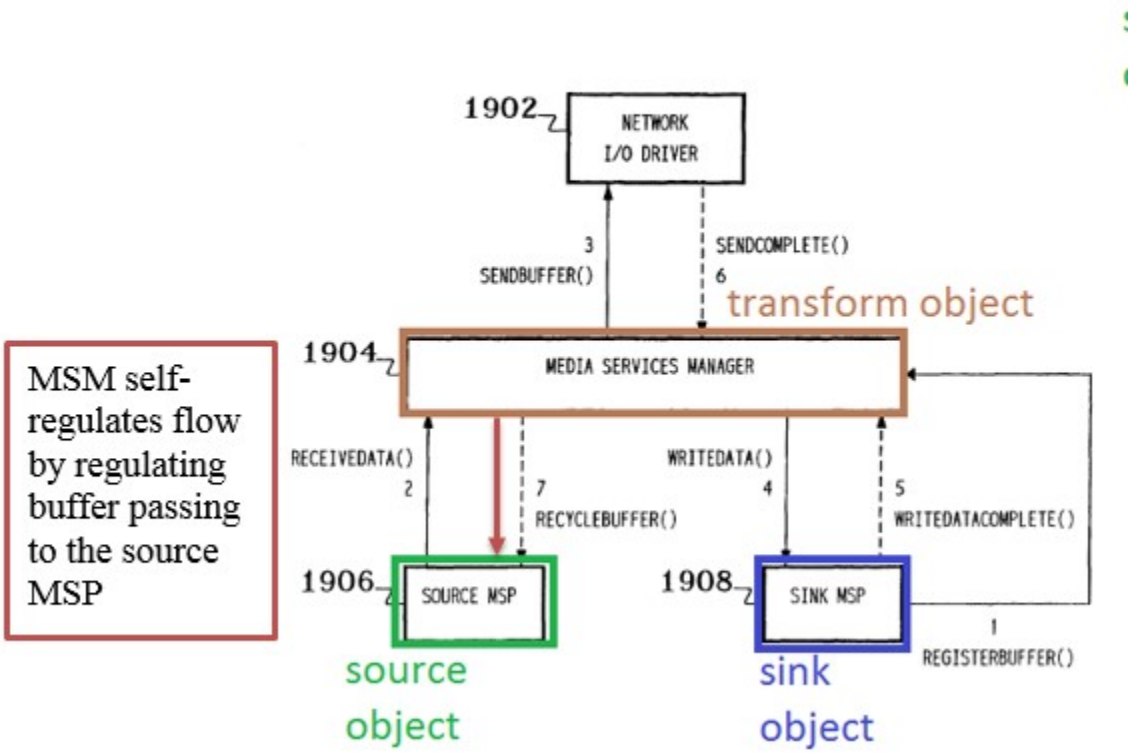

Figure 39. [SE1004, Sampat FIG. 19 (annotated)]

171. In step 7, Sampat describes recycling the buffer from the MSM to the source MSP. [SE1004, 17:35-38.]

172. In step 2, Sampat explains: "2. Source MSP 1906 allocates source buffers, fills them with data (on some regular interval for real-time data), and tells MSM 1904 when there is new data for MSM 1904 to receive (using the RMS API function ReceiveData)." [SE1004, 17:16-19.] Then, "5. After sink MSP 1908 plays a sink buffer, sink MSP 1908 informs MSM 1904 that the sink buffer can be reused (using the RMS API function WriteDataComplete). … 7. After network I/O driver 1902 and sink MSP 1908 have released the source buffer back to MSM 1904, MSM 1904 returns the source buffer to source MSP 1906 for reuse (using the RMS API function RecycleBuffer)." [SE1004, 17:5-38.] Steps 2, 5, and 7 are illustrated in Sampat Fig. 19, reproduced above.

173. This description in Sampat is consistent with the description of automatic flow control in the '389 patent, which states:

> The source object 901 takes data out of a physical data source, such as the Media Switch, and places it into a PES buffer. To obtain the buffer, the source object 901 asks the down stream object in his [*sic*] pipeline for a buffer (allocEmptyBuf). The source object 901 is blocked until there is sufficient memory. ***This means that the pipeline is self-regulating; it has automatic flow control***. When the source object 901 has filled up the buffer, it hands it back to the transform 902 through the pushFullBuf function.

[SE1001, 8:45-51, emphasis added; *see also* 8:19-9:16.]

174. Accordingly, because, like the '389 patent, Sampat's MSM passes empty buffers to and receives full buffers from the source MSP, a POSITA would have understood that Sampat's MSM (transform object) automatically controls the flow of Sampat's source MSP (source object). A POSITA would not have understood that the flow is merely reactive and independent of the MSM. For these reasons, Sampat's source MSP (source object) is self-regulated by Sampat's MSM (transform object). Thus, under the BRI of automatically flow controlled identified above, Sampat's source MSP (source object) is automatically flow controlled by Sampat's MSM (transform object).

iii. *Sink Object - Automatic Flow Control*

175. As indicated in Section V above, under BRI, "automatically flow controlled" includes "self-regulated" and, consequently, this claim element covers "wherein said sink object is ***self-regulated*** by said transform object." [SE1014, 101-106; SE1001, 8:52-65.]

176. Sampat explains: "Media services manager (MSM) 1608 ***manages the flow of data*** through server software architecture 1512" including "[f]rom a source MSP to a local sink MSP" and "[f]rom mass storage device 1516 to a local sink MSP." [SE1004, 9:10-26 (emphasis added).]

177. The MSM regulates the flow of data to the sink object both by starting and stopping operations [SE1004, 9:57-10:43, 11:65-12:2] and also by regulating

buffers passing during flow [SE1004, 16:60-17:39]. Sampat shows and describes its transform object (MSM) controlling flow in the same way as described in the '389 patent, with the sink object (sink MSP) playing buffers only when received from the MSM. [SE1004, 16:60-17:39; FIG. 19.] I have reproduced Sampat's Figure 19, below, and annotated it with colors and labels to show how Sampat's MSM (the transform object) regulates flow of data for Sampat's sink MSP (the sink object).

FIG. 19

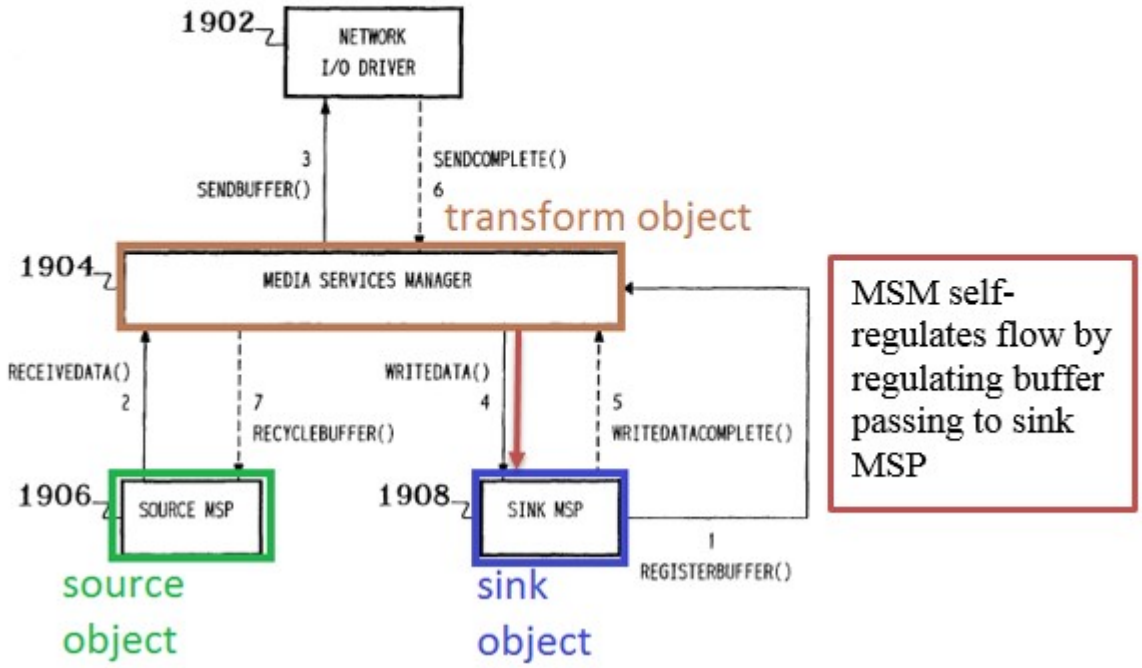

Figure 40. [SE1004, Sampat FIG. 19 (annotated)]

178. Further, Sampat describes the MSM writing the sink buffer data to the sink MSP in step 4 before the sink MSP can play the sink buffer data in step 5.

[SE1004, 17:24-30.] This description in Sampat is consistent with the description of automatic flow control in the '389 patent, which equates waiting for buffers from the transform object to being "self-regulating" and "automatic flow control." [SE1001, 8:52-65; *see also* 8:19-9:16.] As noted above, this description in Sampat is consistent with the description of automatic flow control in the '389 patent, which points out the "automatic flow-control benefit of this method" [SE1001, 8:60-61] and explains:

> The sink 903 is flow controlled as well. It calls nextFullBuf which tells the transform 902 that it is ready for the next filled buffer. This operation can block the sink 903 until a buffer is ready. When the sink 903 is finished with a buffer (i.e., it has consumed the data in the buffer) it calls releaseEmptyBuf. ReleaseEmptyBuf gives the buffer back to the transform 902. The transform 902 can then hand that buffer, for example, back to the source object 901 to fill up again.

[SE1001 at 8:45-51 *see also* 8:19-9:16.]

179. Accordingly, a POSITA would have understood that Sampat's MSM (transform object) automatically controls the flow of the sink MSP (sink object). A POSITA would not have understood that the flow is merely reactive and independent of the MSM. For these reasons, Sampat's sink MSPs (sink object) are self-regulated by Sampat's MSM (transform object). Thus, under the BRI of automatically flow controlled identified above, Sampat's sink MSPs (sink object) are automatically flow controlled by Sampat's MSM (transform object).

G. Decoder Features of Claims 31 and 61

180. Claims 31 and 61 of the ’389 patent each recite “wherein said decoder converts said streams into display signals and sends said signals to a display.” As discussed below, Sampat discloses these features.

181. Sampat describes the server 102 having a display [*see, e.g.,* FIG. 1] and describes sending signals to the display for monitoring [SE1004, 9:5-6, 9:50-57]. “Preferably, each server has all the functionality of a client to provide monitoring capabilities.” [SE1004, 38:34-36.] As explained above at claim Section VI.G and VIII.E.ii Sampat teaches the decoder converting streams into display signals and sending those signals to a display monitor. [SE1004, 14:6-12; SE1007, 8, 157-211; FIG. 17.]

182. For these reasons, a POSITA would have understood that Sampat’s display driver includes a decoder that converts the streams into display signals and sends the signals to a display.

H. Control Object Features of Claims 31 and 61

183. Claim 31 of the ’389 patent recites providing a control object and claim 61 recites a control object. In addition, claims 31 and 61 of the ’389 patent each recite “wherein said control object receives commands from a user, said commands control the flow of the broadcast data through the system,” and

"wherein said control object sends flow command events to said source, transform, and sink objects." As discussed below, Sampat discloses these features.

i. *Control Object – Receives Commands that Control the Flow of Broadcast Data*

184. As indicated in Section V above, under BRI, the term "control object" includes "a collection of data and operations that receives commands from a user that control the flow of broadcast data." [SE1014, 92, 98.] Sampat discloses a control object in the form of the server application 1602 and the media services manager (MSM) application programming interface (API) 1604. [SE1004, 8:63-9:6 ("server application 1602 is used to select… whether to play any of the selected data streams locally to monitor the multicasting services."), FIG. 16.] Server application 1602 and MSM API 1604 are a collection of data and operations. [SE1004, 8:63-9:6.] I have reproduced Sampat's Figure 16, below, and annotated it with colors and labels to show Sampat's control object colored burgundy.

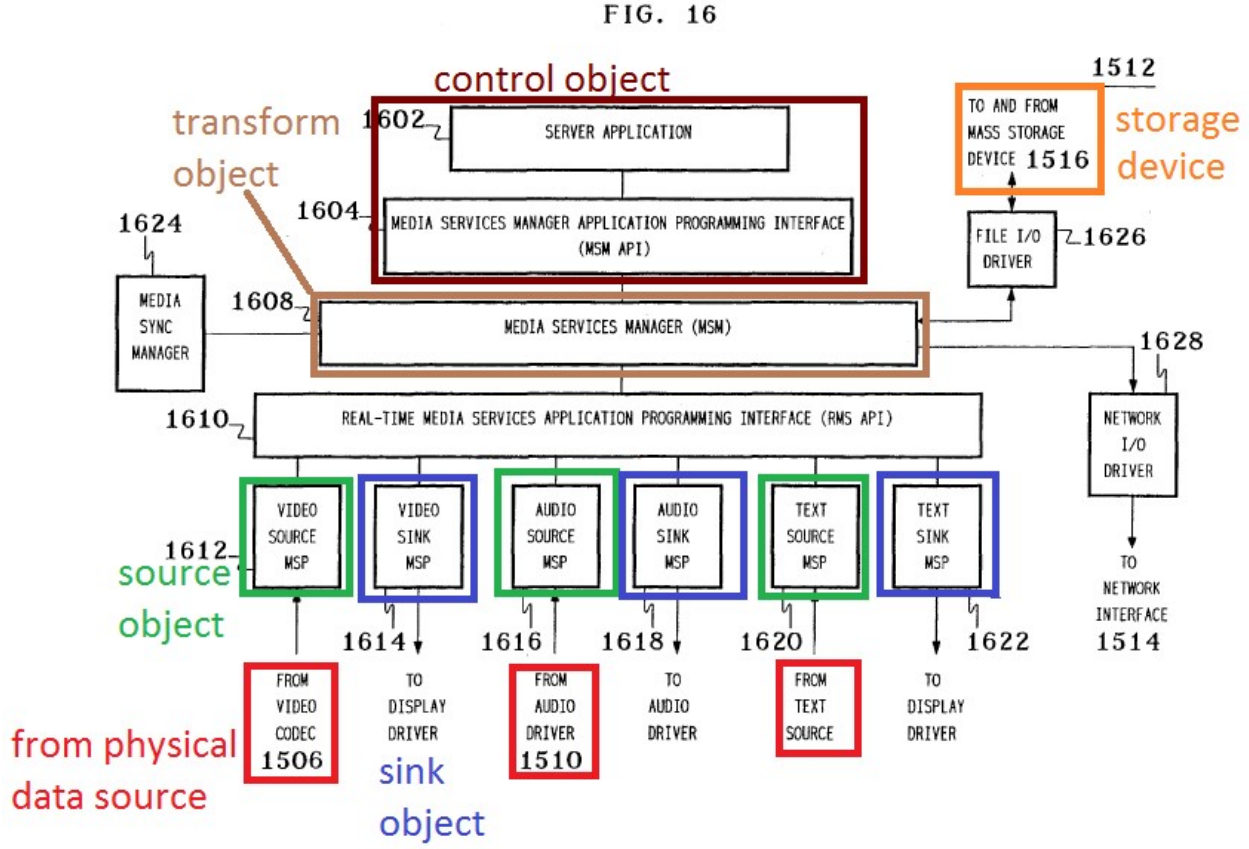

Figure 41. [SE1004, Sampat FIG. 16 (annotated)]

185. Sampat explains that the combination of server application 1602 and MSM API 1604 controls the flow of broadcast data, stating: "Media services manager (MSM) 1608 ***manages the flow*** of data through server software architecture 1512 ***as specified by server application 1602***." [SE1004, 9:10-26 (emphasis added).] "Preferably, ***each server has all the functionality of a client*** to provide monitoring capabilities." [SE1004, 38:34-36 (emphasis added).] Accordingly, the disclosure in Sampat with respect to receiving user input on the client 104 (*see* Section VI.H) also applies to the server 102. The control object

(client application 1802 in the client, and server application 1602 in the server) uses the user interface shown in Sampat FIGS. 2-14 to control the flow of broadcast data through the system. [SE1004, 2:17-19; 14:34-54; 39:51-54 ("… the options pop-up window allows the user to temporarily pause play of the selected multicast channel … "); 7:28-31 ("Remote control window functions include changing channels; changing audio volume; and playing, recording, or rewinding the audio, video, or text components of the current channel.").] Accordingly, a POSITA would have understood that Sampat's server application/MSM API controls the flow of broadcast data through the system by, for example, playing, pausing, recording, and rewinding.

186. The control object (server application 1602) uses the user interface shown in Sampat FIGS. 2-14 to control the flow of broadcast data through the system. [SE1004, 2:17-19, 14:34-54, 38:34-36; 39:51-54.] "Remote control window functions include changing channels; changing audio volume; and playing, recording, or rewinding the audio, video, or text components of the current channel." [SE1004, 7:28-31.] Accordingly, a POSITA would have understood that Sampat's server application/MSM API controls the flow of broadcast data through the system by, for example, playing, pausing, recording, and rewinding. I have reproduced Sampat's Figure 13, below, and annotated it

with colors and labels to show buttons for receiving user commands for pausing, recording, and rewinding broadcast data.

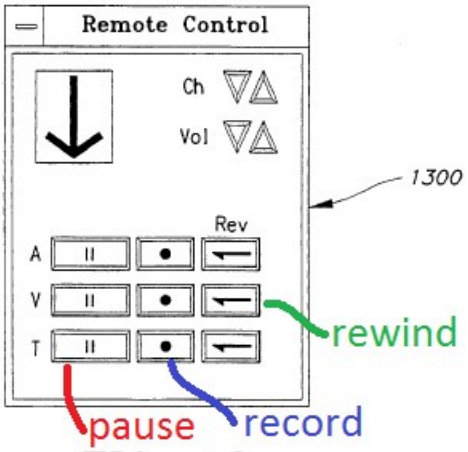

FIG. 13

Figure 42. [SE1004, Sampat FIG. 13 (annotated)]

187. Sampat also describes user commands to perform the functions of "record, play, and rewind." [SE1004, 7:28-31; 39:55-62.] For these reasons, a POSITA would have understood the server application/MSM API in Sampat's server 102 to include a collection of data and operations that receives commands that control the flow of broadcast data through the system from the user. Thus, under the BRI of control object identified above, Sampat's server application/MSM API includes a control object and Sampat's disclosure of providing the server application/MSM API is disclosure of providing a control object. As explained above, Sampat's server application/MSM API (the control object) receives commands from a user, and the commands control the flow of the broadcast data through the system.

ii. *Control Object – Sends Flow Command Events*

188. Sampat discloses the control object sending flow command events to the source, transform, and sink objects. As indicated in Section V above, under BRI, this claim element is interpreted broadly enough to include "the control object sends information relating to a change of condition in the flow of the broadcast data to the source, transform and sink objects." [SE1013, 16-17.]

189. Sampat explains that the control object (server application 1602 and MSM API 1604) sends flow command events to the transform object (the MSM) and also to the source and sink objects via the transform object. [SE1004, 8:63-9:33, 9:57-10:7.] The flow command events reach each of source, transform, and sink objects because the server application 1602 has access to the MSM API function MSM_StartServices() which "[s]tarts (or unpauses) any or ***all of the MSPs*** that were initialized" and MSM_StopServices() which "[s]tops (or pauses) any or ***all of the MSPs*** that were initialized." [SE1004, 9:66-102, emphasis added]. Sampat shows this architecture in FIG. 16. I have reproduced Sampat's Figure 16, below, and annotated it with colors and labels to show Sampat's source, sink, transform, and control objects.

FIG. 16

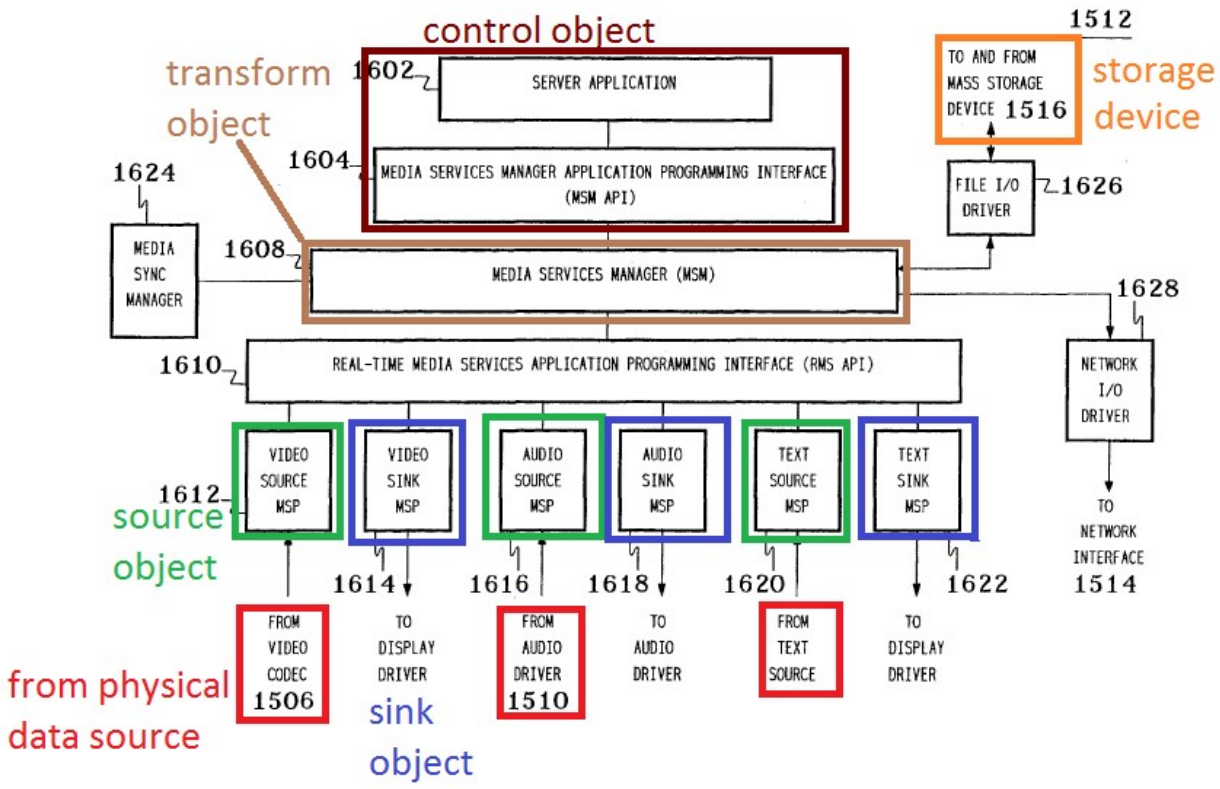

Figure 43. [SE1004, Sampat FIG. 16 (annotated)]

190. Sampat describes a number of examples of the control object sending information regarding a change of condition, such as starting, pausing, and shutting down. [SE1004, 9:57-10:7.] In one example of starting or unpausing after a pause, a function call by the server application 1602 "[s]tarts (or unpauses) any or all of the MSPs that were initialized." [SE1004, 9:66-67.]

191. While information sent from the control object passes to the MSM and is relayed to the sink MSPs and network I/O driver, a POSITA would have understood that the BRI for this claim element does not require the flow command events to pass *directly* to each object, only that the information is sent to each

object. [SE1013, 16-17.] For example, the '389 patent describes flow command events going first to one object and then being relayed to other objects. [SE1001, 8:21-24 ("To pause the pipeline, for example, an event called 'pause' is sent to the first object in the pipeline. The event is ***relayed on to the next object*** and so on down the pipeline.") (emphasis added).]

192. For these reasons, a POSITA would have understood that the server application/MSM API in Sampat's server 102 sends information relating to a change of condition in the flow of the broadcast data to the source, transform and sink objects. Thus, under the BRI, Sampat's server application/MSM API (the control object) sends flow command events to the source, transform, and sink objects.

IX. ANALYSIS OF SAMPAT (SERVER SIDE) IN VIEW OF MICROSOFT VIDEO FOR WINDOWS, SOUNDBLASTER, AND GERBER (CLAIMS 31 AND 61)

193. As described above in Section VIII, a POSITA would have understood that Sampat (server-side) discloses each and every element of claims 31 and 61. To the extent that Sampat is found not to disclose one or more claim elements, such claim elements would have been obvious in view of the predictable combination of Sampat, Microsoft Video for Windows, SoundBlaster, and Gerber. Below, I first address obviousness of a physical data source temporarily storing video and audio data based on Gerber and SoundBlaster, then address obviousness

of a video decoder based on Microsoft Video for Windows, and finally address obviousness of an audio decoder based on SoundBlaster. For the other features of claims 31 and 61, my analysis in Section VIII applies fully and a POSITA would have found all of the other features obvious over the disclosure in Sampat referenced in Section VIII.

A. Physical Data Source Temporarily Stores Video and Audio Data of Claims 31 and 61

194. To the extent that Sampat does not *explicitly* disclose that the physical data source temporarily stores the video and audio data, such functionality was commonly known in similar prior art capture components.

195. For storing audio data, for example, SoundBlaster discloses storing to memory by describing "Direct Memory Access" (or "DMA") for use when recording. [SE1006, 20, 146, 153, 158.] SoundBlaster discloses temporarily storing audio data when being processed in an audio capture component. [SE1006, 98 ("voice is recorded directly to the diskette or hard disk using a double-buffering technique"), 92 ("When you record a sound with the Sound Blaster Pro, it changes analog signals into digital signals.").]

196. Multiple reasons would have prompted to a POSITA to implement the well-known prior art functionality of temporarily storing audio data in an audio capture component (as suggested by SoundBlaster) in the system disclosed by Sampat.

197. First, a POSITA would have been prompted to implement the well-known prior art functionality of temporarily storing audio data in an audio capture component (as suggested by SoundBlaster) in the system disclosed by Sampat because temporarily storing data was a traditional function performed when capturing audio data. A POSITA would have understood that Sampat's system "captures and converts the analog audio signals into digital audio data streams" and SoundBlaster disclosed a suitable and well-known option for capturing and converting analog audio signals. [SE1004, 7:63-65; SE1006, 10, 20, 92, 98, 146, 153, 158.]

198. Second, a POSITA would have been prompted to implement the well-known prior art functionality of temporarily storing audio data in an audio capture component (as suggested by SoundBlaster) in the system disclosed by Sampat because Sampat specifies that "[a]udio capture component 1508 may be any suitable device for capturing and digitizing analog audio signals and ***is preferably a Creative Labs SoundBlaster Pro***." [SE1004, 4:56-57 (emphasis added).] SoundBlaster captures and digitizes audio signals, thus temporarily storing the signals. [SE1006, 10, 20, 92, 98, 146, 153, 158.] This disclosure in Sampat that SoundBlaster is preferred would have motivated a POSITA to use that very device as the audio capture component, which would have been conventional, well-known, and predictable.

199. Third, a POSITA would have been prompted to implement the well-known prior art functionality of temporarily storing audio data in an audio capture component (as suggested by SoundBlaster) in the system disclosed by Sampat because doing so would be merely the application of a known technique (*e.g.*, temporarily storing data via an audio capture device like SoundBlaster) to a known system (*e.g.*, Sampat's system having sound enabled) ready for improvement to yield predictable results. I have been told that, when a patent simply arranges old elements, with each performing the same function it had been known to perform, and yields no more than what one would expect from such an arrangement, the combination is obvious. Here, both Sampat and SoundBlaster disclose a device for audio processing (in fact, the same SoundBlaster device), and a POSITA would have recognized that applying the suggestions in SoundBlaster to Sampat's system would have led to predictable results without significantly altering or hindering the functions performed by the system of Sampat.

200. To the extent that Sampat does not *explicitly* disclose that the physical data source temporarily stores the video data, such functionality was also commonly known in similar prior art capture components. For example, Gerber describes the "INTEL® SMART VIDEO RECORDER™," which is the same "Intel® SmartVideo® Recorder (ISVR)" described in Sampat. [SE1005, 5:10-16, 5:25-28; SE1004, 4:59-60, 8:24-28.] Gerber explains that the Smart Video

Recorder includes "dynamic random access memory 41 and video random access memory ('VRAM') 38" as well as a "digitizer 36." [SE1005, 5:20-28.] "In response to a control signal by the CC driver 34, the digitizer 36 ***captures*** the video data from the video input device 18 and ***stores*** the captured video data in a first storage element 39a" of the VRAM. [SE1005, 5:28-33 (emphasis added), *see also* 5:33-60 (further describing processing of the video data, including compression and storage), 7:3-5, 7:66-8:4, FIG. 3, FIG. 5.] I have reproduced Gerber Figure 3, below, and annotated with red to show storage element 39a inside VRAM memory.

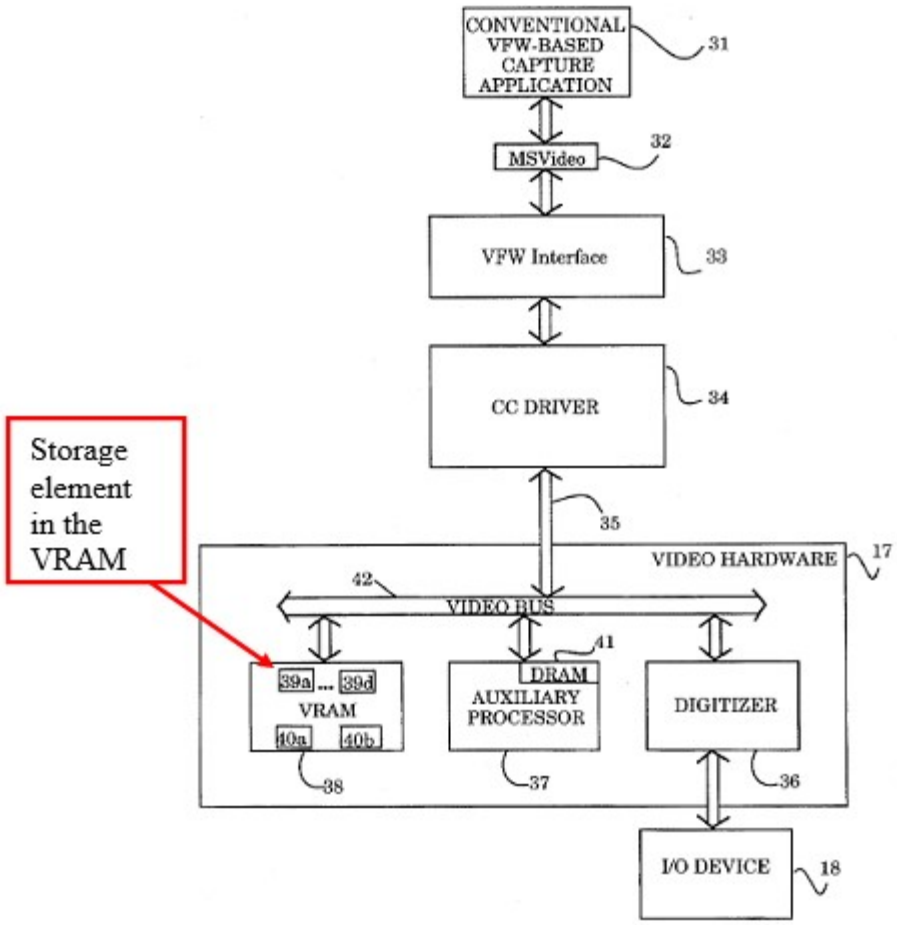

Figure 3

Figure 44. [SE1005, Gerber FIG. 3 (annotated)]

201. Multiple reasons would have prompted to a POSITA to implement the well-known prior art functionality of temporarily storing video data in a video capture component (as suggested by Gerber's disclosure of the Intel Smart Video Recorder) in the system disclosed by Sampat.

202. First, a POSITA would have been prompted to implement the well-known prior art functionality of temporarily storing video data in a video capture component (as suggested by Gerber) in the system disclosed by Sampat because

temporarily storing data was a traditional function performed when capturing video data. A POSITA would have understood that the video capture component in Sampat's system "captures and converts the analog video signals into digital video data streams" and Gerber disclosed a suitable and well-known option for capturing and converting analog video signals. [SE1004, 7:62-63; SE1005, 5:10-16, 5:20-60, FIG. 3.]

203. Second, a POSITA would have been prompted to implement the well-known prior art functionality of temporarily storing video data in a video capture component (as suggested by Gerber) in the system disclosed by Sampat because Sampat specifies that the "[v]ideo capture component 1504 and codec 1506 may be any suitable hardware/software device or devices for capturing and compressing video and are preferably components of an Intel® SmartVideo® Recorder (ISVR)." [SE1004, 8:25-27.] The Smart Video Recorder captures and compresses video—temporarily storing the signals. [SE1005, 5:10-16, 5:20-60, FIG. 3.] This disclosure in Sampat that the Smart Video Recorder is preferred would have motivated a POSITA to use that very device as its video capture component and codec, which would have been conventional, well-known, and predictable.

204. Third, a POSITA would have been prompted to implement the well-known prior art functionality of temporarily storing video data in a video capture component (as suggested by Gerber) in the system disclosed by Sampat because

doing so would be merely the application of a known technique (*e.g.,* temporarily storing data via a video capture device like the Intel Smart Video Recorder described in Gerber) to a known system (*e.g.,* Sampat's system having video enabled, preferably by the Intel Smart Video Recorder) ready for improvement to yield predictable results. I have been told that, when a patent simply arranges old elements, with each performing the same function it had been known to perform, and yields no more than what one would expect from such an arrangement, the combination is obvious. Here, both Sampat and Gerber disclose a device for video processing (in fact, the same Intel Smart Video Recorder device), and a POSITA would have recognized that applying the suggestions in Gerber to Sampat's system would have led to predictable results without significantly altering or hindering the functions performed by the system of Sampat.

B. Video Decoder Features of Claims 31 and 61

205. To the extent that Sampat does not *explicitly* say that the disclosed display driver includes a video decoder, such decoders were commonly known in similar drivers. The Programmer's Guide, Microsoft Video for Windows Development Kit ("Microsoft Video for Windows") confirms that the Microsoft Video for Windows display driver includes a video decoder. [SE1007, 8 ("video compression and decompression drivers"), 157-211 (video decompression).] This is the same "Microsoft Video for Windows" display driver that Sampat describes

as the preferred embodiment. [SE1004, 14:10-12 ("Display driver 1704 may be any suitable driver for displaying video and text data and is preferably Microsoft Video for Windows.").] As Microsoft Video for Windows confirms, a POSITA would have understood that Sampat's disclosure of outputting streams to a display driver, such as Microsoft Video for Windows, would output the streams to a video decoder.

206. Multiple reasons would have prompted a POSITA to implement the well-known prior art functionality of including a video decoder in the display driver (as suggested by Microsoft Video for Windows) in the system disclosed by Sampat.

207. First, a POSITA would have been prompted to implement a video decoder in the display driver of Sampat because Sampat uses encoded data (*e.g.*, MPEG) and video displays required encoded data to be decoded. For instance, Sampat describes the display driver 1704 as being used "for processing" prior to "display on monitor 1708." [SE1004, 13:55-57.] A POSITA would have recognized that any coded data would require decoding in order for the display to properly reproduce a video stream, and would have recognized that the display driver should include a decoder as was traditional in such systems.

208. Second, a POSITA would have been prompted to implement a video decoder in the display driver of Sampat because Sampat specifically identifies

"Microsoft Video for Windows" as the preferred display driver. [SE1004, 14:10-12; see also 29:50-55.] Microsoft Video for Window included decoders. [SE1007, 8, 157-211.] This disclosure in Sampat that Microsoft Video for Windows is preferred would have motivated a POSITA to use that very driver, which was conventional, well-known, and predictable.

209. Third, a POSITA would have been prompted to implement a video decoder in the display driver of Sampat because doing so would be merely the application of a known technique (*e.g.,* decoding video using a display driver) to a known system (*e.g.,* Sampat's system having a display driver) ready for improvement to yield predictable results. I have been told that, when a patent simply arranges old elements, with each performing the same function it had been known to perform, and yields no more than what one would expect from such an arrangement, the combination is obvious. Here, both Sampat and Microsoft Video for Windows disclose display drivers (in fact, the same display driver), and a POSITA would have recognized that applying the suggestions in Microsoft Video for Windows to Sampat's system would have led to predictable results without significantly altering or hindering the functions performed by the system of Sampat.

C. Audio Decoder Features of Claims 31 and 61

210. To the extent that Sampat does not *explicitly* say that the sink object outputs to an audio decoder, such decoders were commonly known in similar drivers. For example, the Sound Blaster Pro User Reference Manual ("SoundBlaster") discloses decoding in the form of "decompression." [SE1006, 152.] In particular, SoundBlaster discloses "ADPCM, hardware decompression" associated with various compression schemes. [SE1006, 152.] A POSITA would have understood that ADPCM decompression stands for "adaptive differential pulse-code modulation," which is used for coding and decoding an audio signal. [SE1006, 152.] SoundBlaster also discloses having a "Digital to Analog Converter," which a POSITA would have understood performs decoding of digital audio encoded with pulse code modulation. [SE1006, 10, 146, 152-153.] This SoundBlaster reference is the same "SoundBlaster" that Sampat describes as the preferred embodiment. [SE1004, 4:56-57; 15:1-3.]

211. Multiple reasons would have prompted a POSITA to implement the well-known prior art functionality of outputting to an audio decoder (as suggested by SoundBlaster) in the system disclosed by Sampat.

212. First, a POSITA would have been prompted to implement the well-known prior art functionality of outputting to an audio decoder (as suggested by SoundBlaster) in the system disclosed by Sampat because Sampat uses encoded data (*e.g.*, MPEG, digital audio) and audio hardware (*e.g.*, an amplifier and

speakers) requires encoded data to be decoded. A POSITA would have understood that Sampat's system would be unable to process encoded audio data without a decoder, and SoundBlaster discloses a suitable and well-known option for providing such decoding. [SE1006, 10, 146, 152-153.]

213. Second, a POSITA would have been prompted to implement the well-known prior art functionality of outputting to an audio decoder (as suggested by SoundBlaster) in the system disclosed by Sampat because Sampat specifically identifies "SoundBlaster Pro from Creative Labs" as the preferred device for enabling sound. [SE1004, 4:56-57.] SoundBlaster included decoders. [SE1006, 10, 146, 152-153.] This disclosure in Sampat that SoundBlaster is preferred would have motivated a POSITA to use that very device and its decoder, which would have been conventional, well-known, and predictable.

214. Third, a POSITA would have been prompted to implement the well-known prior art functionality of outputting to an audio decoder (as suggested by SoundBlaster) in the system disclosed by Sampat because doing so would be merely the application of a known technique (*e.g.*, decoding audio using a decoder) to a known system (*e.g.,* Sampat's system having sound enabled) ready for improvement to yield predictable results. I have been told that, when a patent simply arranges old elements, with each performing the same function it had been known to perform, and yields no more than what one would expect from such an

arrangement, the combination is obvious. Here, both Sampat and SoundBlaster disclose a device for audio processing (in fact, the same SoundBlaster device), and a POSITA would have recognized that applying the suggestions in SoundBlaster to Sampat's system would have led to predictable results without significantly altering or hindering the functions performed by the system of Sampat.

D. Obviousness Conclusion for Claims 31 and 61

215. For the above reasons, a POSITA would have understood that Sampat in view of Microsoft Video for Windows and SoundBlaster renders obvious providing a sink object that obtains data stream buffers from the transform object and outputs the streams to a video and audio decoder.

216. As explained above in Section VIII.G, Sampat describes the decoder (display driver 1704) converting steams into display signals and sending the signals to a display. Microsoft Video for Windows also discloses its decoder converting streams into display signals and sending those signals to a display monitor. [SE1007, 157-158 (converting compressed video data streams and sending to a "Video Display"), 158-159 ("a decompression driver with the ability to write to the video display").] Accordingly, Microsoft Sampat, Video for Windows, and SoundBlaster render obvious a decoder converting the streams into display signals and sending the signals to a display.

217. In addition, for the above reasons, a POSITA would have understood that Sampat in view of Microsoft Video for Windows, SoundBlaster, and Gerber renders obvious providing a source object that temporarily stores video and audio data.

X. SECONDARY CONSIDERATIONS

218. I do not believe "secondary considerations" evidence impacts the obviousness conclusions set forth in this declaration for claims 31 and 61. I have reviewed the file wrapper for the second reexamination proceeding, including the alleged "Secondary Considerations" discussed starting at page 1033. [SE1017, 1033-1036, 1076-1085, 1281-1303.] The secondary considerations identified do not appear to be related to claims 31 and 61 of the '389 patent. For example, Patent Owner and its declarant James Barton point out sales figures and recognition of the TiVo DVR device, but do not show that such sales and recognition were a result of the features claimed in claims 31 and 61. [SE1017, 1079-1081.] Such sales and recognition could have been driven by features present only in other claims of the '389 patent, features present only in claims of other TiVo patents, or features not patented at all.

219. For example, Mr. Barton points out one industry award crediting TiVo's success as being related to the user interface—a feature absent from claims

31 and 61. [SE1017, 1080, 1100.] That same award also credits TiVo because "it has Linux inside"—another unclaimed feature. [SE1017, 1100.] Another commentator credits TiVo's *marketing* with creating the demand for DVRs. [SE1017, 1096-1097.] That commentator refers to "heavy salesmanship" and explains that: "Since its inception in 1997, TiVo has spent $178 million on sales and marketing, more than twice the $79 million it spent on R&D during the same period" and that "TiVo has spent $767 to win each of its 464,000 subscribers." [SE1017, 1096.] Accordingly, the success and praise of such DVRs appears to be driven by unclaimed factors.

220. Mr. Barton also presented sales figures showing TiVo's DVR sales relative to a competitor's DVR sales. [SE1017, 1084, 1118.] Mr. Barton stated that the DVR products by both companies read on claims 31 and 61, and yet "the number of TiVo subscribers has dropped significantly every quarter" and the competitor "EchoStar has gained DVR subscribers every quarter, and it has gone from fewer DVR subscribers than TiVo to twice as many DVR subscribers as TiVo." [SE1017, 1083-1084.] While I am not aware of what has driven this drastic difference in sales, it must have been factors other than the features of claims 31 and 61 if both products included those features. Accordingly, I do not see the evidence of DVR sales, other praise, or other secondary considerations being linked to the features claimed in claims 31 and 61. In fact, none of the cited

evidence in Mr. Barton's first declaration credits the features actually claimed in claims 31 and 61, let alone credits any claimed features or combinations that were not already known in the prior art.

221. In Mr. Barton's second declaration, he alleges that investment, sales, and industry praise were related to the transform object—however, he provides no evidence to support this conclusion. [SE1017, 1300-1303.] For example, Mr. Barton concludes that certain products embody claims 31 and 61, but as evidence he cites only to advertisements showing a picture of the product with a brief recitation of some specifications. [SE1017, 1301-1302, 1304-1312.] Mr. Barton also includes statements regarding motivation for investment, again without evidence.

222. Mr. Barton ultimately concludes that the "transform object architecture and automatic flow control was critical to the provision of the smooth control of live TV which drove the sales of the TiVo DVR products" and also drove the investments. [SE1017, 1302-1303.] However, TiVo did not invent the transform object or automatic flow control—both were already known as evidenced by Sampat. [*See, supra,* Section VI.D, VI.F, VIII.D, and VIII.F.] As I explained above in Sections VI and VIII, Sampat teaches every element of claims 31 and 61, anticipating those claims. However, even in Sections VII and IX where I discuss the obvious combinations, none of the secondary references are relied on

for the transform object or for automatic flow control—they are taught entirely by Sampat. Accordingly, even if the alleged secondary considerations related to a transform object as Mr. Barton asserts, such a transform object was already known in Sampat prior to the '389 patent.

223. Similarly, Mr. Barton alleges and emphasizes that Patent Owner's DVRs were the first *commercial* products that could pause and rewind live television broadcasts. [SE1017, 1301.] Even if that were true for *commercial* products, Sampat teaches both pausing and rewinding of such broadcasts. [SE1004, FIG. 13 (showing pause and rewind), 7:27-32 ("rewinding" like "a standard television remote control"), 6:20-29, 9:61-10:7, 16:29-44 (describing pausing).] Therefore, even if these features drove sales and praise, they were known. Moreover, pausing and rewinding are not required by claims 31 and 61and are therefore additional unclaimed features driving the secondary considerations.

224. Additionally, the combination of Sampat with one or all of Microsoft Video for Windows, SoundBlaster, and Gerber is a simple combination of adding software and/or a device that Sampat specifically tells the reader to add. Therefore, even if any secondary considerations related to the obviousness of these combinations, the simplicity of these combinations would have been recognized by a POSITA, and the express statements in Sampat to use Microsoft Video for

Windows, the SoundBlaster device, and the Intel Smart Video Recorder device overcomes the evidence of nonobviousness previously provided by TiVo.

XI. ADDITIONAL REMARKS

225. I currently hold the opinions set expressed in this Declaration. But my analysis may continue, and I may acquire additional information and/or attain supplemental insights that may result in added or even modified observations.

226. I hereby declare that all statements made of my own knowledge are true and that all statements made on information and belief are believed to be true. I further declare 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 the Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patents issued thereon.

Dated: August 2, 2016 By: John Michael Strawn

John Michael Strawn, Ph.D.,
of Larkspur, California

Curriculum Vitae
John M. Strawn, Ph.D.
(contact information on last page)

Professional Profile

Several decades of involvement in software, digital audio, digital music, digital signal processing, and processor architecture. Successful independent software consultant in high-level languages and assembly language. Seasoned testifying expert with experience in patent and class action litigation, skilled at explaining complex ideas to attorneys and juries. Stanford Ph.D. Former Fulbright Scholar. Prolific author. Experienced manager with long-range research and development experience. Facile with foreign languages and working with people from outside the USA.

Professional Experience

From: 1992 **S Systems, Inc.**
To: Present Larkspur, CA
Position: *Owner*
Duties: Full-time independent consultant:

- **Programming** hand-crafted audio and music software for signal processing, written in C, C++, JAVA, and especially assembly language for digital signal processing chips. Consulting on processor architecture and networking.
- **Testifying Expert witness** in patent litigation relating to software, computers, signal processing.

From: 1987 **Yamaha Music Technologies USA**
To: 1991 Larkspur, CA
Position: *1989-1991: President; 1987-1989: Vice President*
Duties: Helped establish and manage a nine-person Ph.D.-level research group, including site search, architectural design, construction, move-in, and hiring. Conducted original research on electronic musical instruments, software, micromachining, networking, and recent technological developments. Extensive experience designing scientific, engineering, and musical object-oriented applications, especially C++ (UNIX). Research on Yamaha's Vocaloid started in this group. Patents listed below.

From: 1986 **S Systems**
To: 1988 Larkspur, CA
Position: *Full-time Consultant*
Duties: This was my first stint as a consultant. See Consulting Assignments, below.

From: 1985 **Lucasfilm/Droid Works**
To: 1986 San Rafael, CA
Position: *Programmer*
Duties: Full-time programming experience as an employee, designing signal-processing modules and writing (96-bit VLIW) microcode for the ASP/SoundDroid developed by James A. Moorer. Experience in audio and video post-production. Extensive work in C (Unix). Another six months full-time experience writing tightly packed assembly code for the TI TMS32010 signal processor, especially for a two-channel hard-disk audio record playback unit that played without bugs on the exhibit floor of the National Association of Broadcasters convention, 1986.

From: 1976 **Stanford University**
To: 1985 Stanford, CA
Position: *Doctoral Student*
Duties: Nine years programming experience developing code in high-level languages (Algol, Fortran, SAIL) and PDP-10 assembly language for musical and audio signal processing applications during doctoral thesis work. Includes original published research in spline fitting and pattern recognition, a 30,000-line two- and three-dimensional graphical editor for waveforms and spectra, implementation (with John Gordon) of the short-time Fourier transform, device drivers, and libraries for graphic user interfaces. Also part-time consulting work:

- SRI International (FORTRAN for mechanical engineering).
- Mattel Electronics (music in consumer electronic toys).
- IntelliGenetics (ALGOL-like code for biotechnology).
- Digital Keyboards (product specification and complete manuals for GDS and Synergy Synthesizers).

From: 1972 **Revox**
To: 1972 Long Island, New York
Position: *Summer intern*
Duties: Solder cables, write German- and Dutch-English translations, manufacture PC boards, assemble hardware.

Education and Training

Year	College/University	Degree
1968-1973	Oberlin	B. Mus, double degree in organ performance and music theory. Exchange semester, University of Hamburg, Germany, 1971, course work in German literature and psychology. Experience with analog synthesizers and digital music synthesis, BASIC, FORTRAN, MUSIC V on an IBM 360.
1973-1975	Technical University, Berlin	Fulbright Scholar. Graduate-level coursework in music theory/history, audio engineering, electronics, information theory, cybernetics, Japanese; all coursework in German. Extensive recording studio and live concert sound reinforcement experience. PDP-11 and PDP-8 assembly and machine language. Travel throughout Europe.
1975-1976	IBM Thomas Watson Foundation	Grant to study electronic music, Tokyo, Japan, 1976. Live performances on piano and Roland System 700 analog synthesizer. Also travel through Turkey, Iran, Afghanistan, Pakistan, India, Thailand, and Hong Kong.
1985	Stanford	Ph.D., CCRMA. Advisor: John Chowning. Graduate course work in music, computer and processor architecture, high-level and assembly-language programming, digital audio, digital signal processing, acoustics, psychoacoustics, and digital hardware. Dissertation on analysis of music instruments with the short-time Fourier transform. Software development experience listed elsewhere in this resume.

Expertise

- Implement/optimize signal processing algorithms: Fourier transform (FFT), discrete cosine transform (DCT), DTMF, speech synthesis.
- Port/optimize audio compression algorithms: AC-3, MP-3, AAC.
- Implement audio algorithms: reverberator, pitch shifter, sample rate converter, compressor, filter, flanger, 3-d audio (Dolby surround), dither.
- Implement music synthesis (additive, physical modeling, wavetable, FM).
- Create bug-free software from academic signal processing research.
- Work in floating- and fixed-point math.
- Assembler, object-oriented, C, C++.
- Extensive experience optimizing code in assembler
- PC, Mac, Unix.
- DSP architectures: Motorola 56000, 56300, and 56800 families; TI TMS320C10 and TMS320C54 family; Code Composer Studio; Analog Devices 21xx family and TigerSharc;

VLIW; custom processors; I learn new architectures quickly.
- Embedded processors: Hitachi SH-DSP, SH3-DSP, SH-4, and SH-5; ARM7/ARM9; configurable processors (Tensilica).
- Processor architecture.
- Debugging hardware prototypes.
- Audio networks, such as AES/EBU (IEC 60958), IEEE-1394/FireWire, AV/C, 61883, mLan, and others.
- File downloading.
- Practical audio experience in live sound and in studios.
- Testifying expert witness (including expert reports, deposition).
- Software analysis for litigation.
- Functionally bilingual in German; able to read French, Dutch; some Japanese

Expert Witness and Litigation Support Experience

Summary: patent litigation, class action litigation, ITC actions, and USPTO declarations relating to software (in a wide variety of areas), cell phones, audio, music, speech, processor architecture, compression, multimedia, digital cameras, telephony, video games, user interfaces, and file downloading, among others. Testimony at trial three times including one ITC; 15 depositions; prior art analysis, infringement analysis, expert reports, tutorials, Markman hearings.

Date: 2015 - Present **Denko, Lauff, Austin TX**

Case: In The Matter Of Certain Audio Processing Hardware And Software And Products Containing Same, ITC 337-TA-949, engaged on behalf of Intervenor Waves (Israel) and Respondent Dell.

Project: Source code analysis, expert report, deposition, witness statement for infringement involving noise reduction, adaptive filtering, echo cancellation.

Date: 2014 - Present **Orrick, San Francisco, Washington, DC, and Los Angeles**

Case: Blue Spike, LLC v. Texas Instruments Incorporated, TXED-6-12-cv-00499, engaged on behalf of lead defendant Audible Magic.

Project: Two expert reports, one on infringement, both involving extensive source code analysis. Declaration. Deposition.

Date: 2014 - 2015 **Wiley Rein, Washington, DC**

Case: Six petitions for Inter Partes Review by Verizon involving US7,319,866; US7,257,395; and US7,295,864.

Project: Declaration, 21 exhibits, regarding cell phone ring tones.

Date: 2013 - present **Greenberg Traurig, New Jersey and Denver**
Case: Petition for Inter Partes Review by Samsung, PTAB-IPR2014-00044.
Project: Declaration, three claim charts, deposition.

Date: 2011 - 2014 **Kirkland & Ellis, Chicago; Irwin IP, Chicago; Fliesler Meyer, San Francisco**
Case: Adobe v. Wowza, CV 11-02243-CW, California Northern District.
Project: Five patents related to real-time video and audio streaming. Deposition related to Markman. Expert report and deposition on infringement including Java source code analysis.

Date: 2013-2014 **THAT Corporation; McDermott Will & Emery, Boston**
Action: US Patent Application 11/445,670, BTSC Encoder.
Project: Declaration to USPTO regarding non-obviousness for audio in television. Patent prosecution had lasted 8 years. US 8,908,872 issued 3 months after my declaration was submitted with amendment.

Date: 2013 –14 **Novak Druce, Wilmington DE; Washington DC; Houston**
Case: SmartPhone v. ZTE, 6:12cv350, Eastern District of Texas, Tyler.
Project: Five patents involving cell phones. Expert reports related to validity and infringement, including Android source code analysis. Deposition.

Date: 2012 - 13 **Morgan, Lewis & Bockius, Washington, DC**
Case: SmartPhone v. LG, 6:10-cv-74, Eastern District of Texas, Tyler.
Project: Two patents related to cell phones. Expert report and deposition related to validity.

Date: 2012 **Quarles + Brady**
Case: SmartSound v. Avid, 3:12-cv-223, Wisconsin Western District.
Project: Infringement for two patents involving automated composition of sound tracks for video.

Date: 2012 **Jones Day**
Case: LSI v. Vizio, 8:10-cv-01602, California Central District.
Project: Validity and infringement for four patents involving digital memory and MPEG audio. Case settled before Markman.

Date: 2008-9, 2011-present **Alston Bird, Atlanta**
Case: Move Inc. v. Real Estate Alliance Ltd et al., 2:07-cv-02185, Central District of California, Los Angeles.
Project: Expert reports on infringement concerning real estate sales website. Source code analysis. Deposition.

Date: 2011 **Quinn Emanuel, New York**
Case: Motorola v. Apple (Certain Wireless Communication Devices, Portable Music and Data Processing Devices, Computers and Components Thereof, ITC 337-TA-745).
Project: Three expert reports and two witness statements relating to infringement by Apple Accused Products (iPhones), technical prong of domestic industry, and validity, focusing on cell phone GPS. Android and iPhone source code analysis. Deposition. Testimony at trial, 8-15 December 2011.

Date: 2010-2011 **Finnegan, Henderson, Washington, DC**
Case: HTC v. Apple (In the Matter of Certain Portable Electronic Devices and Related Software, ITC 337-TA-721).
Project: Expert reports relating to technical prong of domestic industry for 24 HTC Windows Mobile cell phones, including source code analysis relating to user interface, memory, and caller ID. Consulting expert relating to iPhone, iPad, and iPod touch concerning validity and power management.

Date: 2010-2012 **Robins, Kaplan, Miller, & Ciresi, Minneapolis**
Case: Fair Isaac v. Actimize and NICE, 1:2009cv00688, Delaware.
Project: Infringement and source code analysis relating to financial transaction verification.

Date: 2010 **Orrick, Washington, D.C.**
Case: Affinity Labs v. Alpine Electronics, JVC Kenwood, et al., 08-171-RC, Eastern District Texas.
Project: Involved car audio, marine audio, and home theatre products that connect to iPod/iPhone. Rebuttal expert report on infringement. Deposition.

Date: 2009-10 **Wolf Haldenstein, New York**
Case: In re Apple & ATTM Antitrust Litigation, Northern District of California, San Jose, C 07-5152.
Project: Analyze iPhone source code for antitrust plaintiffs. Expert report and various declarations, in particular regarding class certification. Deposition.

Date: 2008-10 **Paul Hastings, Palo Alto, CA**
Case: Konami Digital Entertainment v. Harmonix Music Systems, Texas Eastern District 6:08-cv-00286
Project: Analyze Rock Band video game source code (Playstation 2, PS3, Wii, XBox). Expert reports on infringement and validity. Two-day deposition.

Date: 2009-10 **Jones Day, Palo Alto, CA**
Case: SanDisk v. LSI, California Northern District, 3:09-cv-02737
Project: Attend tutorial and Markman hearing regarding MP3 patent litigation.

Date: 2009 **Weil Gotschal, Redwood Shores, CA**
Case: Samsung v. Kodak (In the Matter of Certain Digital Cameras, ITC 337-TA-671).
Project: Consulting expert. Analyze Samsung cell phone source code relating to digital cameras. This involved baseband chips from Qualcomm, Philips, Agere, Texas Instruments; register-level code for camera image sensors from Samsung, Sony, Micron, Omnivision; Windows Mobile 5 and 6 digital camera device drivers; Qualcomm BREW 2 and BREW 3 cell phone operating systems; patents involving digital cameras, Bayer subsampling, and pixel interpolation; and standard digital optical concepts such as RGB, YUV, YCbCr, EXIF, and JPEG.

Date: 2009 **Finnegan, Henderson, Washington, DC**
Case: Voice Domain Technologies LLC v. Philips Electronics North America Inc. et al., Western District of Oklahoma, 5:08−cv−00701.
Project: Declarations for Markman hearing on hand-held consumer devices.

Date: 2009 **THAT Corporation; McDermott Will & Emery, Boston**
Action: US Patent Application 09/638,245, BTSC Encoder.
Project: Declaration to USPTO regarding non-obviousness for audio in television.

Date: 2007-8 **Fish and Richardson, Atlanta**
Case: Nice Systems Inc. and Nice Systems Ltd. v. Witness Systems Inc. Civil Action No. 06-311-JJF, Delaware District.
Project: Expert reports on validity and infringement concerning telephone call centers (telephony, software, hardware architecture, digital recording.) Deposition, jury trial testimony.

Date: 2005-7 **Fish and Richardson, San Diego, CA**
Case: Lucent Technologies Inc. v. Gateway, Inc., et al., defendants, and Microsoft Corporation, Intervener. Case No. 02-CV-2060 B (CAB) consolidated with 03-CV-0699 B (CAB) and 03-CV-1108 B (CAB).
Project: Expert involving audio compression and MP3. Deposition and two days testimony at three-week jury trial, cross-examined by Kirkland and Ellis. Prepared 7 expert reports (total 497 pages) on infringement and validity including 20 claim charts and 15 other substantive attachments. Analyzed over 4000 pages of C/C++ source code; analysis of assembly and machine code. Worked directly with German documents.
Status: Judge Brewster overturned jury decision and ruled in favor of defense. Judge Brewster's decision upheld on appeal, http://www.cafc.uscourts.gov/opinions/07-1546.pdf

Date: 2007 **Morrison and Foerster, Los Angeles**
Case: 3:06-cv-7736 CA Northern District, Seer Systems v. Yamaha.
Project: Prior art research for music synthesis.

Date:	2006-7	**Mayer Brown Rowe & Maw, Houston, TX**
	Case:	2:06-cv-156, Digital Technology Licensing (DTL) v. Cingular Wireless.
	Project:	Claim charts, technology background for microphone in cell phones.
Date:	2007	**Meyer & Associates Co. LPA, Columbus, Ohio**
	Case:	Health Science Products LLC and Kairos & Associates, Inc., v. Sage Software SB, Inc., 1:2005cv03329, Georgia Northern District.
	Project:	For class action litigation, analyze database software before and after release of ACT 2005.
Date:	2005-6	**Black Lowe & Graham, Seattle**
	Case:	Digeo, Inc. v. Audible, Inc., Case No. C05-00464-JLR, Seattle
	Project:	Expert reports regarding Markman, validity and infringement involving Internet file downloading. Analyze C/C++ source code. Deposed for Markman hearing.
Date:	2006	**Ropes and Gray, Palo Alto**
	Case:	MediaTek, ASUSTek & ASUS v. Sanyo.
	Project:	Prepare claim charts on 24-hour notice. Assist in preparation of tutorial.
Date:	2006	**Wilmer Hale (New York)**
	Case:	Information Technology Innovation, LLC v. Motorola, Inc. et al., Northern District of Illinois 04-C-7121.
	Project:	Provide and supervise an expert witness colleague who prepared an expert report on infringement.
Date:	2004-5	**Weil, Gotshal & Manges, New York**
	Case:	Antor Media Corporation v. Apple Computer, Inc., Microsoft Corporation, RealNetworks, Inc., Civil Action No. 2:03CV320
	Project:	Prior art regarding file downloading.
Date:	2005	**Trop, Pruner & Hu, Austin, TX**
	Project:	Prior art involving signal processors.
Date:	2002-3	**Robins, Kaplan, Miller, & Ciresi, Minneapolis**
	Case:	Intergraph v. Dell et al., EDTX, 2-02cv-312
	Project:	Prior art for hardware architecture, virtual memory and cache memory.
Date:	1997-8	**Cesari and McKenna, Boston**
	Case:	Lucent vs. Young Chang/Kurzweil.
	Project:	Prior art for music synthesis, digital hardware, software, architecture.
Date:	1994	**Small, Larkin, Los Angeles**
	Case:	L.C. Concepts v. Digital Theatre Systems (DTS)
	Project:	Prior art for cinema sound equipment in USA and Germany.

Consulting Assignments

From: 2011 **Client: iZotope**
To: 2011 Boston
Duties: Port iZotope's pitch correction effect from C++ source code to Avid TDM environment in Motorola 56000 family assembly language.

From: 2009 **Client: Congruity**
To: 2009 Palo Alto
Duties: For this music industry startup, create audio effects in Freescale DSPM56364 assembly language. Write and debug code without access to hardware, working only with software tools. Initial delivery of code ran bug-free in target hardware.

From: 2008 **Client: DTS Digital Cinema (now Datasat Digital Entertainment)**
To: 2008 Location: Agoura Hills, CA
Duties: For DTS Digital Cinema's XD20 eight-track cinema media player (this is the box that sits in the movie theater projection booths for playback of multi-channel audio and video), port DTS Coherent Acoustics decode (two versions, one 8-channel, one stereo), DTS Digital Cinema 8-channel decode, and DTS Neo6 5.1 decode from DTS Digital Cinema's existing XD10 cinema media player. This required me to isolate the four algorithms by stripping away unneeded code; integrate the four algorithms into Motorola 56721 dual-core processor; and write new wrapper code in assembly language. Responsible for approximately 25,000 lines of assembly language source.

From: 2007 **Client: Berkeley Design Technology, Inc. (**http://www.bdti.com/**)**
To: 2008 Location: Oakland, CA
Duties: Contribute to research and writing of the following articles on processor architecture at BDTI's website Inside DSP (http://www.insidedsp.com/):

- TI Offers OMAP3 Application Processors to the Mass Market
- Avnera releases ASSPs for wireless audio applications
- XMOS Introduces Low-cost Multi-core Chip Family with Programmable I/O
- VeriSilicon's New Silicon IP Solution for HD Audio
- Behind the scenes: Dolby's acquisition of Coding Technologies
- Tips and Tricks for Debugging Audio

Other BDTI assignments are listed below.

From: 1995 **Client: Yamaha**
To: 2007 Location: Hamamatsu, Japan
Duties: Chair, AES standards working group SC-02-12 on digital audio networking via IEEE-1394 (Firewire), with the support of Yamaha. Involved a trip to AES conventions twice a year, including one in Europe. Past member, IEC

TC100 TA4, Digital System Interfaces. Various public appearances worldwide and various company site visits to discuss multimedia networking, audio over 1394 and Yamaha's mLAN.

From: 2005 **Client: Sonic Network,** http://www.sonicimplants.com/
To: 2006 Location: Somerville, MA
Duties: For this well-known provider of wavetables, synthesis software, and cell phone ring tones (among others), provide and supervise subcontractors for these projects:

- Design and implementation of filters for sample rate conversion;
- Design and implementation of filters following the DLS-2 specification (used in cell phones for ring tones);
- Port synthesizer code to Tensilica HiFi2 audio engine.

From: 2004 **Client: Bias,** http://www.bias-inc.com/
To: 2006 Location: Petaluma, CA
Duties: For this well-known provider of audio software, provide and supervise a subcontractor to port a complicated digital signal processing algorithm into the DigiDesign TDM Environment, in Motorola 56K assembly language.

From: 2005 **Client: Audio Research Labs, http://** www.audioresearchlabs.com/
To: 2005 Location: Scotch Plains, NJ
Duties: For ARL founder Schuyler Quackenbush provide and supervise a subcontractor to design and implement a digital filter algorithm in Motorola 56K assembly language.

From: 2004 **Client: Verance,** http://www.verance.com/
To: 2005 Location: San Diego, CA
Duties: Working closely with Verance R&D staff, implement the Verance Content Management System/Audio-Visual (VCMS/AV) watermarking technology for motion picture sound (now known as Cinavia) in Motorola 56300 assembly language in the TC Electronics M6000 environment. This program is used by major film studios starting early 2005 to watermark nearly every DVD released. Travel at client's request to TC Electronics headquarters in Denmark to facilitate integration. Provide and supervise a subcontractor to assist with filter design, filter implementation, and other tasks. More than 30,000+ lines of 56K assembler source, several hundred pages of documentation, a dozen CD-ROMs of debugging data and lab notebooks.

From: 2002 **Client: Universal Audio (**http://www.uaudio.com/**)**
To: 2004 Location: Santa Cruz, CA
Duties: For this well-known manufacturer of audio plugins, port two audio processing algorithms (Pultec filter, LN1176 stereo compressor) from C/C++ to Motorola 563xx assembly language in the DigiDesign ProTools TDM environment, including numerical approximation and streamlining the original C/C++ implementation. Publicly released 2004. Contribute extensively also to port of an extremely complicated high-end reverberator, and to another equalizer.

From: 2003 **Client: Stretch (**http://www.stretchinc.com/**)**
To: 2004 Location: Mountain View, CA
Duties: For this software configurable processor startup, study how to port MPEG-2 AAC and MP-3 decode reference C++ code to 16- and 32-bit integerized C. Do the same for MP-3 encode based on publicly available source. Learn their software configurable architecture well enough to write optimizations.

From: 2003 **Client: RIC International Precision Translation Services**
To: 2003 Location: Cambridge, MA
Duties: For this major translation house**,** proofread German-English translations involving, among other things, audio compression (including German-language doctoral dissertations).

From: 2003 **Client: Analog Devices**
To: 2003 Location: Santa Clara, CA (Audio Rendering Technology Center)
Duties: Port music synthesis algorithms in assembly language for the ARM7/TDMI processor, following ARM's C calling conventions. This project ran under very tight time constraints, cost only 2/3 of the projected budget, and resulted in code that runs much faster than the original implementation.

From: 2002 **Client: Dorrough Electronics (**http://www.dorrough.com**)**
To: 2003 Location: Chatsworth, CA
Duties: Implement in C and Analog Devices Sharc 21161 assembly language a novel scheme based on their patented technology to improve the perceived loudness of audio signals sent over broadcast. Provide a subcontractor who made significant contributions to filter design.

From: 2002 **Client: Analog Devices**
To: 2002 Location: Wilmington, MA (Ray Stata Technology Center)
Duties: After an on-site visit to learn more about the technology and meet the team, I made recommendations on changes to architecture for a new version of an idiosyncratic signal processing chip. I also provided code examples for the new architecture.

From: 2001 **Client: Tensilica**
To: 2002 Location: Santa Clara, CA
Duties: For this configurable processor IP core company, implement a highly optimized version of the modified discrete cosine transform (MDCT) for audio compression. Extensive investigation of theory and variants of the MDCT. Also port MPEG-2 low-complexity AAC decode and MP3 encode from Thomson reference C++ code to 16-bit integerized C. Prepare various optimizations closer to the hardware than C++ usually allows.

From: 1999 **Client: Berkeley Design Technology, Inc. (**http://www.bdti.com/**)**
To: 2001 Location: Oakland, CA
Duties:

- For BDTI's Buyer's Guide to DSP Processors, 2001 Edition, contribute major portions of the text analyzing processor architectures including the Analog Devices TigerSharc, and contribute also to the analyses of Motorola 56300, 56800, and 56800E processors; verification and in some cases re-writing assembly-language implementations of BDTI's benchmarks;
- Prepare written analyses of Hitachi SH-DSP, SH3-DSP, SH-4, and SH-5 processor architectures. This again included verification and in some cases re-writing assembly-language implementations of BDTI's benchmarks;
- Implement assembly-language routines related to multimedia compression in ARM7/ARM9 processor assembly language;
- Develop and present a four-hour presentation on audio compression, given first at Embedded Processor Forum, June, 2000; contribute to a four-hour presentation on digital audio and music given by Dana Massie at the same Embedded Processor Forum; revised and presented both talks at Microprocessor Forum, October 2000; both talks revised again with emphasis on streaming audio and presented at Embedded Processor Forum, June, 2001.

From: 1995-6 **Client: Audio Precision (**http://www.audioprecision.com**)**
And 1998-9 Location: Portland, Oregon
Duties: Audio Precision (Portland, Oregon). For their System 2 audio measurement device, developed double-precision Fourier transform (FFT) in assembly language for Motorola 56002 processor, including (Microsoft) C code to study where to maintain double-precision. Also, extensive code for AES/EBU and square wave measurement test suite, including jitter and eye pattern (assembling bit map for graphics display in 56002 data memory space). 28K+ lines of assembly language source. 1998-1999: Revise Audio Precision System 2 code for new 96 kHz Cascade hardware (Motorola 56303).

From: 1997 **Client: Euphonics (later part of 3COM)**
To: 1999 Location: Boulder, CO
Duties: Implement Dolby AC-3 decoder (used in Dolby Digital cinema sound) in

16-bit integer assembly language on new Analog Devices 16-bit integer AD1818 processor (PCI SoundComm). 20K+ lines of assembler source. Passed first round of Dolby testing on first try. Integrate with Euphonics' Real-Time Kernel operating system.

From: 1996 **Client: Digital Technics (DTI)**
To: 1997 Location: Baltimore, MD.
Duties: Implementation of CCITT R2 encoder/decoder (similar to DTMF) in Motorola 56002 assembly language, based on Goertzel algorithm. 13K+ lines assembler. Deployed in the field in Asia and South America.

From: 1996 **Client: VM Labs**
To: 1996 Location: Los Altos, CA
Duties: For this multimedia chip startup, provide detailed critique of their proprietary DSP chip architecture.

From: 1993 **Client: Oculix**
To: 1995 Location: Switzerland
Duties: Motorola DSP 56000 assembly language for numerical and FFT analysis of real-time data gathered by laser from the human eye for medical applications. 150K source.

From: 1993 **Client: Centigram Communications Corporation.**
To: 1994 Location: Silicon Valley CA (apparently now part of SS8 Networks)
Duties: Port of speech synthesis code from TI TMS320E17 assembly language to Motorola DSP 56002 assembly language on Motorola PC Media card; port to Analog Devices ADSP 2115 assembly language on Echo Personal Sound System.

From: 1993 **Client: Atari**
To: 1994 Location: Sunnyvale, CA
Duties: implement physical modeling music synthesis techniques on custom RISC/DSP chip inside Jaguar game console. Recommend improvements to new custom DSP architecture.

From: 1993 **Client: Euphonics**
To: 1993 Location: Boulder, CO
Duties: For this software music synthesizer company, write C routines to emulate certain hardware elements in the target architecture. This allowed the company to study aspects of caching parameter updates, for optimizing real-time performance.

From: 1993 **Internal Project**
To: 1993 Location: Bay Area, CA
Duties: For a research project involving DSP architecture, write a series of Java classes to emulate the typical components of a DSP chip.

From:	1987	**Client: Shure**
To:	1988	Location: Evanston (now Niles), IL
	Duties:	Working from the written specification for a proprietary algorithm, develop C and TI TMS 32010 assembly language for a multi-channel consumer audio product prototype.

From:	1987	**Client: NeXT, Inc.**
To:	1988	Location: Silicon Valley, CA
	Duties:	NeXT Inc. Developed, debugged, and documented more than 50 routines in the Motorola DSP 56000 assembly language vector library (with Julius O. Smith; source code printout is 2" thick.). While working off-site for over a year before NeXT was publicly released, maintain secrecy about the fact that NeXT would include a 56000 processor.

From:	1986 or 1987	**Client: Sonic Solutions**
To:	1986 or 1987	Location: San Francisco CA
	Duties:	As one of the first consultants hired by Sonic Solutions (located in their first office in San Francisco), port their C-language noise-reduction code from one flavor of Unix to another.

Other experience:

- Studies of micromachining and nanotechnology.
- Experience with the Star Semiconductor SPROC chip, the IBM MWAVE chip and operating system, OS-9, and Spectron's SPOX operating system.

Patents

Patent Number	Date Issued	Title
5,569,871	October 29, 1996	Musical tone generating apparatus employing microresonator array (co-inventor; micromachining) As Vice-President and President of Yamaha Music Technologies Inc., I supervised the patent applications by my employees that resulted in US patents 5,245,130, 5,288,938, 5,386,568, 5,422,956, 5,536,902, and 5,541,358.

Teaching appointments

From:	2003	**University of Colorado at Denver, College of Arts & Media**
To:	2008	Denver, CO
	Position:	*Lecturer, College of Arts & Media*
	Duties:	Teach special topics course on audio data compression to upper-level undergraduate and graduate students.

Major Publications

- "Approximation and Syntactic Analysis of Amplitude and Frequency Functions for Digital Sound Synthesis." *Computer Music Journal* 4(3):3-22, 1980.

- *Modeling Musical Transitions.* Ph.D. Thesis, Stanford University, 1985. 243 pp.

- (with C. Roads). *Foundations of Computer Music.* MIT Press, 1985. 600 pp.

- *Digital Audio Engineering: An Anthology.* Madison, WI: A-R Editions, 1985. 144 pp.

- *Digital Audio Signal Processing: An Anthology.* Madison: A-R Editions, 1985. 283 pp.

- "Orchestral Instruments: Analysis of Performed Transitions." *Journal of the Audio Engineering Society* 34(11):867-80, 1986.

- "Editing Time-varying Spectra." *Journal of the Audio Engineering Society* 35(5):337-51, 1987.

- "Analysis and Synthesis of Musical Transitions Using the Discrete Short-time Fourier Transform." *Journal of the Audio Engineering Society* 35(1/2):3-14, 1987.

- "Implementing Table Lookup Oscillators for Music with the Motorola DSP56000 Family." Presented at the 85th Convention of the AES, 1988. Preprint No. 2716.

- "Digital Audio Representation and Processing." *Multimedia Systems*, edited by John F. Koegel. ACM and Addison-Wesley, 1993.

- "Technological Change: The challenge to the audio and music industries" (written version of AES keynote address). *Journal of the Audio Engineering Society,* March 1997.

- (with James Grunke, Ben Novak, Bruce Pennycook, Zack Settel, Phil Wiser, and Wieslaw Woszczyk). "AES White Paper: Networking Audio and Music using Internet2 and Next Generation Internet Capabilities." *Journal of the Audio Engineering Society* 47(4):300-310, April 1999. Presented (with Betsy Cohen and AES President Marina Bosi) to White House National Economic Council, December 1998. http://www.aes.org/technical/i2.html.

- (with Yamaha's Mike Overlin). "Playing with Fire," *Electronic Musician*, May 2003, pp. 31-38 (http://emusician.com/ar/emusic_playing_fire/index.htm, on audio networking over 1394).

Professional Associations and Achievements

- Fellow (1996), Audio Engineering Society.
- Convention Co-chair, 2008 AES Convention, San Francisco.
- Convention Chair, 2006 AES Convention, San Francisco.

- Convention Chair, 2004 AES Convention, San Francisco. Recipient of an Anderton Award, Pro Sound News, December 2004, p. 30.
- Keynote Speaker, November 1996 Audio Engineering Society Convention.
- Elected member of the AES Board of Governors, 1992-1994; again 2005-2007.
- Chair, Audio Engineering Society Convention Policy Committee, 2006-2008.
- Technical Papers chair, 1992 AES Convention, San Francisco (first AES San Francisco Convention). Technical Papers co-chair, 2002 AES convention, Los Angeles.
- Conference Chair, 1987 Audio Engineering Society (AES) International Conference on Music and Digital Technology (Los Angeles).
- Former member of review board, *Journal of the Audio Engineering Society.*
- Assistant Editor, *Computer Music Journal*, (MIT Press), 1978-1982.
- Co-founder (1980), International Computer Music Association.
- Founder and Series Editor (1984-1996), *The Computer Music and Digital Audio Series*.
- Honorary Member (since 1998), The MIDI Association (MMA).
- Technical presentations and session chair at various conferences such as Audio Engineering Society, Acoustical Society of America, International Computer Music Conference, DSP World.
- Conference paper reviewer for many International Computer Music Conferences (ICMC).
- Member, Acoustical Society of America. Senior Member, IEEE.

Further qualifications

Functionally bilingual in German. Reading ability in French, Dutch. Some experience with Spanish, Italian, Japanese, Latin. Separate list of foreign language experience available on request. Extensive experience traveling abroad and communicating with foreigners.

Other activities

I currently enjoy spending time with my family, hiking, and weightlifting. In earlier years I have especially enjoyed travel, aikido, operating a Maerklin Z-gauge model railroad, performing a wide variety of folk and classical music, and attending musical events. Member of Toy Train Operating Society of America.

References

Full vita and references from industry, academia, and lawfirms available on request.

Contact Information

S Systems, Inc., 15 Willow Avenue, Larkspur, CA 94939, USA
Tel +1 (415) 927 8856
http://www.s-systems-inc.com Email jstrawn@s-systems-inc.com