

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS, INC.,
Petitioner,

IPR2026-00210
Patent No. 11,930,362

**PETITION FOR *INTER PARTES* REVIEW
UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104**

TABLE OF CONTENTS

Table of Contents	2
Petitioner’s Exhibit List	4
I. Introduction.....	7
II. Grounds for Standing.....	9
III. Note.....	10
IV. Effective priority date	10
V. Prosecution	10
VI. Level of ordinary skill in the art	10
VII. Background.....	11
VIII. Claim Construction.....	11
IX. Relief requested and reasons therefore	11
X. Identification of how the claims are unpatentable.....	11
A. Challenged Claims	11
XI. Statutory grounds.....	12
A. Statement of Material Facts.....	13
B. Ground 1	13
1. Abuan	13
2. Platt	15
3. Reasons to Combine Abuan with Platt	16
4. Morrison.....	17
5. Reasons to Combine Abuan, Platt, and Morrison.....	20
6. Independent Claim 1 and dependent claims	21
7. Independent Claim 75	77

C.	Ground 2	84
1.	Eisenberg	84
2.	Reasons to Combine	86
3.	Claim 15	86
D.	Ground 3	88
1.	Beilis	88
2.	Reasons to Combine	89
3.	Claim 21	90
XII.	Conclusion	91
XIII.	Mandatory notices	93
A.	Real party-in-interest	93
B.	Related matters	93
C.	Lead and back-up counsel and service information	93
XIV.	Claims Appendix	95
	Certificate of Word Count	102
	Certificate of Service	103

PETITIONER’S EXHIBIT LIST

Ex.1001	U.S. 11,930,362
Ex.1002	Prosecution History of U.S. 11,930,362
Ex.1003	Declaration of Bruce McNair under 37 C.F.R. § 1.68
Ex.1004	<i>Curriculum Vitae</i> of Bruce McNair
Ex.1005	U.S. 2011/0249077 to Abuan et al.
Ex.1006	U.S. 8,396,490 to Platt et al.
Ex.1007	U.S. 2014/0184723 to Morrison
Ex.1008	U.S. 5,689,641 to Ludwig et al.
Ex.1009	D. E. Knuth, Selected Papers on Computer Science (Cambridge Univ. Press), 1996
Ex.1010	B. W. Boehm, Software Engineering Economics (Prentice-Hall, Inc.), 1981
Ex.1011	U.S. 5,339,431
Ex.1012	U.S. 5,615,400
Ex.1013	U.S. 5,946,486
Ex.1014	U.S. 6,243,764
Ex.1015	U.S. 5,802,367
Ex.1016	U.S. 5,218,697
Ex.1017	U.S. 5,498,003
Ex.1018	U.S. 6,163,858
Ex.1019	U.S. 2010/0066807 to Eisenberg et al.
Ex.1020	Plaintiff’s Disclosure of Asserted Claims and Infringement Contentions, <i>Damaka, Inc. v. Cisco Systems, Inc.</i> , No. 2:25-cv-00593 (E.D. Tex. Nov. 13, 2025).
Ex.1021	MacRumors.com, “Official Skype App is Now Available in the App Store,” 2009 (retrieved from https://www.macrumors.com/2009/03/30/official-skype-app-is-

	now-available-in-the-app-store/)
Ex.1022	The Big Blog, “Skype now available for Android phones,” 2010 (retrieved from https://web.archive.org/web/20101006000452/http://blogs.skype.com/en/2010/10/android.html)
Ex.1023	Apple Newsroom, “Apple Announces iPhone 2.0 Software Beta,” 2008 (retrieved from https://www.apple.com/newsroom/2008/03/06Apple-Announces-iPhone-2-0-Software-Beta/)
Ex.1024	U.S. 2008/0201438
Ex.1025	U.S. 2012/0170722
Ex.1026	U.S. 2005/0265322
Ex.1027	U.S. 6,314,402
Ex.1028	U.S. 2005/0060183
Ex.1029	U.S. 2009/0193394
Ex.1030	U.S. 2008/0022327
Ex.1031	U.S. 2002/0199019
Ex.1032	U.S. 6,101,607
Ex.1033	U.S. 6,311,283
Ex.1034	U.S. 6,295,048
Ex.1035	U.S. 2009/0310023
Ex.1036	U.S. 2008/0127183
Ex.1037	iOS Reference Library, “Core Media Framework Reference,” 2010 (retrieved from https://web.archive.org/web/20100906024157/http://developer.apple.com/iphone/library/documentation/CoreMedia/Reference/CoreMediaFramework/index.html#apple_ref/doc/uid/TP40009756)
Ex.1038	MacStories, “App Is 2010’s Word Of The Year,” 2011 (retrieved from https://web.archive.org/web/20170401145352/https://www.macstories.net/news/app-is-2010s-word-of-the-year/)
Ex.1039	U.S. 2012/0019610
Ex.1040	U.S. 6,820,235

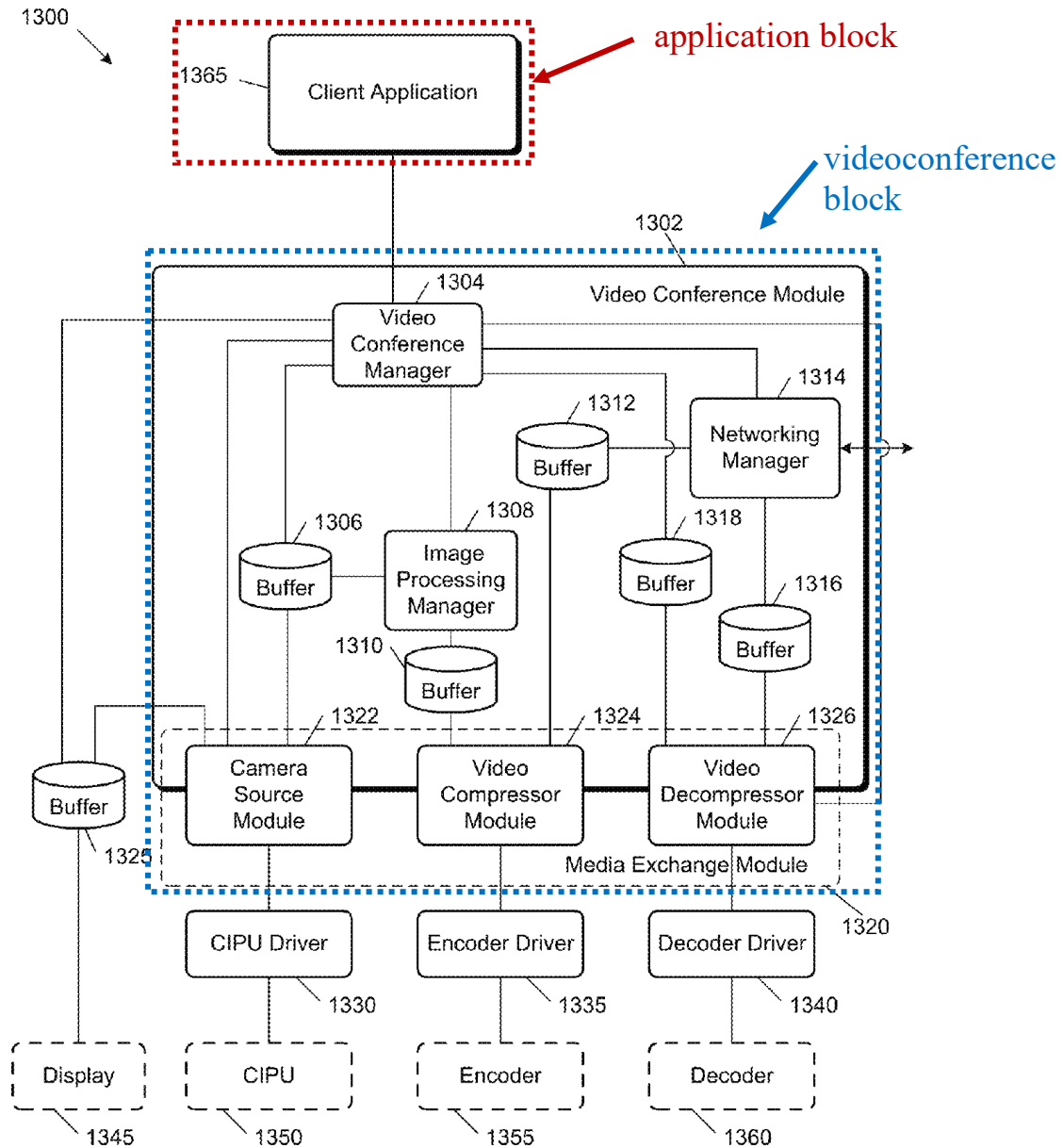
Ex.1041	U.S. 2013/0290982 to Beilis et al.
Ex.1042	IBM 1130 materials (Scientific Subroutine Package manual noted) (“IBM 1130”)
Ex.1043	U.S. 9,058,655

I. Introduction

Cisco Systems, Inc. (“Petitioner”) respectfully requests that the Board review and cancel as unpatentable under 35 U.S.C. §103 claims 1-28 and 75 (hereinafter, the “Challenged Claims”) of U.S. 11,930,362 (the “’362 patent,” Ex.1001).

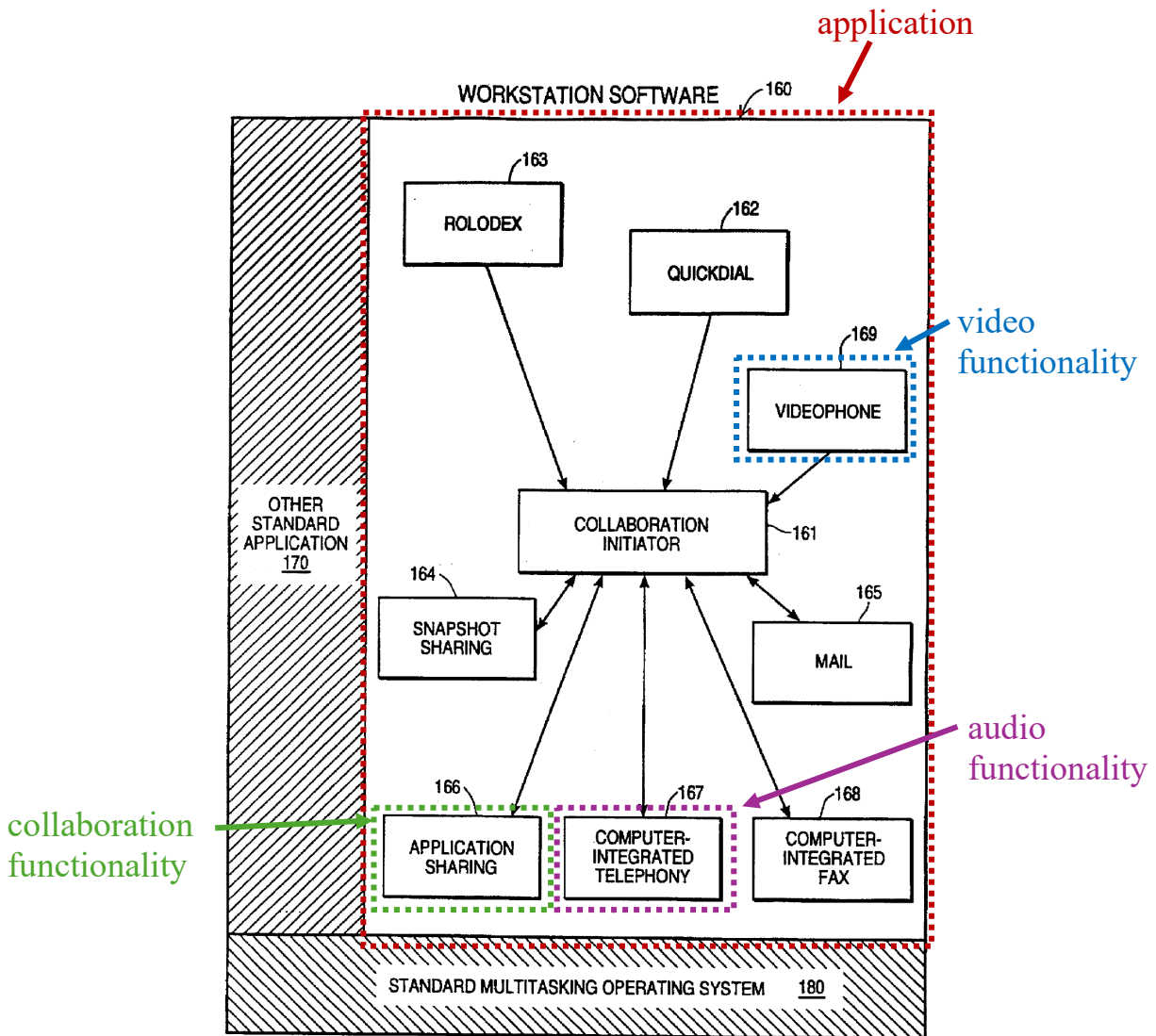
The ’362 patent purports to improve “functionality in software applications” by organizing that functionality into blocks. Ex.1001, Abstract, 4:29-38.

Organizing computer software into blocks of code was not new. The figure below (from 2010) shows an application block using videoconference functionality provided by a separate block. Ex.1005, FIG. 13.



Ex.1005, Fig. 13 (annotated).

This “block” organization had been used for years. The figure below, from 1993, shows a similarly modular application with discrete blocks providing various functionalities. Ex.1008, FIG. 20.



Ex.1008, Fig. 20 (annotated).

Petitioner asks the Board to institute trial and find the claims unpatentable.

II. Grounds for Standing

Petitioner certifies that the '362 patent is eligible for IPR. Petitioner is not barred or estopped from requesting this IPR.

III. Note

Petitioner cites to exhibits' original page numbers. **Emphasis** in quoted material has been added. Claim terms are presented in *italics*.

IV. Effective priority date

The '362 patent claims the benefit of a provisional application filed 7/16/2013, which Petitioner's prior art predates. Accordingly, Petitioner has not undertaken a priority date analysis. Petitioner does not waive any right or opportunity to dispute the priority date of the '362 patent wherever the issue is relevant.

V. Prosecution

The issued claims were never subject to a prior art rejection. See Ex.1003, 78-79 & 61-72 (cancelling an initial claim rejected for statutory double-patenting). The examiner's allowance statement emphasized limitations [1.1]-[1.5] (and similar parts of other independent claims). Ex.1002, 14-16.

VI. Level of ordinary skill in the art

A Person of Ordinary Skill in The Art ("POSITA") in July 2013 would have had a working knowledge of the software programming art that is pertinent to the '362 Patent, including the use of software modules and application programming interfaces (APIs). A POSITA would have had a bachelor's degree in computer science, computer engineering, or an equivalent, and three years of professional

experience relating to computer programming. Lack of professional experience can be remedied by additional education, and vice versa. Ex.1003, ¶18.

VII. Background

Mr. McNair describes background information that would have been known to a POSITA. Ex.1003, ¶¶25-38.

VIII. Claim Construction

Petitioner proposes that no terms require express construction for this proceeding. Ex.1003, ¶¶39-40.

IX. Relief requested and reasons therefore

Petitioner asks that the Board institute a trial for *inter partes* review and cancel the Challenged Claims in view of the analysis below.

X. Identification of how the claims are unpatentable

A. Challenged Claims

Petitioner challenges claims 1-28 and 75 under 35 U.S.C 103.

Patent Owner asserts claims 1-3, 5-10, 14-16, 18-25, 27-28, and 75 in co-pending litigation. *See* Ex.1020, 2. Cancelling the Challenged Claims through this proceeding will resolve the parties' co-pending litigation and obviate a trial there regarding the '362 patent, saving time and costs for all parties.

XI. Statutory grounds

Grounds	Claims	§103 References
#1	1-14, 16-20, 22-28, 75	Abuan, Platt, and Morrison
#2	15	Abuan, Platt, Morrison, and Eisenberg
#3	21	Abuan, Platt, Morrison, and Beilis

US20110249077 (Ex.1005, “**Abuan**”) published 10/13/2011.

US8396490 (Ex.1006, “**Platt**”) issued 3/12/2013.

US2014/0184723 (Ex.1007, “**Morrison**”) was filed 12/13/2012 and published 7/3/2014.

US20100066807 (Ex.1019, “**Eisenberg**”) published 3/18/2010.

US20130290982 (Ex.1041, “**Beilis**”) was filed 4/30/2012 and published 10/31/2013.

Abuan, Platt, Morrison, Eisenberg, and Beilis are prior art under 35 U.S.C. § 102(a)(2).

Petitioner relies on the combined teachings of the references and not on a physical incorporation of elements. *See In re Mouttet*, 686 F.3d 1322, 1332 (Fed. Cir. 2012).

Petitioner also cites additional evidence of the background knowledge of a POSITA. *See Yeda Research v. Mylan Pharm. Inc.*, 906 F.3d 1031, 1041-1042 (Fed. Cir. 2018); 37 C.F.R. § 42.104(b).

A. Statement of Material Facts

1. A POSITA would have been familiar with voice-over-IP, which was a technique for exchange audio communications in real time. Ex.1026, [0085]; Ex.1027, 6:39-47; Ex.1028, [0026].

2. A POSITA would have been familiar with instant messaging, which was a technique for exchanging text-based messages in real time. Ex.1024, [0003]; Ex.1025, [0005].

3. The use of APIs to define standard, programmatic ways for an application to access functionality of a software module was well-known and commonly used. Ex.1029, [0087]; Ex.1030, [0038]; Ex.1031, [0040]; Ex.1032, 5:32-48.

4. A POSITA would have been familiar with device drivers for output devices and would have recognized that they commonly included APIs for reporting on an output device's capabilities. Ex.1033, 6:30-32; Ex.1034, 11:8-26; Ex.1035, Abstract, [0004]; Ex.1036, [0033].

B. Ground 1

1. Abuan

Abuan generally relates to the field of communications software for mobile devices, and more particularly, to real-time video conferencing from mobile devices. Ex.1005, [0003]-[0004]. Abuan describes a modular approach to

facilitating such conferencing features, with functionality organized into components such as a video processing and encoding module, media exchange module, video processing module, video conferencing and processing module, and many others. Ex.1005, [0043], [0099], [0056]. The modules' functionalities are accessed via "a set of application program interfaces (APIs)." Ex.1005, [0044]. Ex.1003, ¶¶44-49.

Abuan is analogous art to the '362 patent. First, Abuan is in the same field of endeavor as the '362 patent, namely, the "manner in which functionality is accessed in certain environments, such as mobile device environments." Ex.1001, 1:66-67; *see* Ex.1005, [0001]-[0002] (describing ways that "functionality" is accessed on "portable devices, such as smartphones"). Abuan and the '362 patent also both generally relate to implementing functionality via software. *See* Ex.1001, Abstract ("providing additional functionality to existing software"); Ex.1005, [0006]-[0007], [0105] (describing how instant messaging or voice-over-IP applications "use the video conferencing functions of the video conference module"). Abuan and the '362 patent both describe at length features for facilitating videoconferencing. Ex.1001, Abstract, 4:43-51 (purpose of a function block is to provide functions including "audio, video,...conferencing, meetings, and/or other functions"); Ex.1005, [0099] (describing a "video conference module...for performing a variety of video conferencing functions"). Ex.1003,

¶50.

Second, Abuan is also reasonably pertinent to a problem allegedly addressed by the '362 patent. The '362 patent describes purported problems associated with using multiple applications on a mobile device, potentially requiring a user to “switch[] back and forth between the video window of [a] call and [a] superblock application.” Ex.1001, 4:22-26. The '362 patent’s proposed solution is to provide functionality—such as communication functions—to a superblock application. Ex.1001, 4:43-61. Abuan similarly describes how communication functions are provided to a client application, which may be a “stand-alone application” or may itself be further “integrated into another application.” Ex.1005, [0105]. Ex.1003,

¶51.

2. Platt

Platt generally relates to the transmission and receipt of “messages among mobile and other devices in a communication network.” Ex.1006, 1:20-22. Platt describes implementation details regarding the distribution of applications, such as software client applications, to the mobile devices of various users. More specifically, Platt explains that wireless devices, such as iPhones, “download[] a software client application (sometimes referred to as a ‘client app’) from a server, such as...a server maintained by an independent distributor of software applications, such as the Apple App Store.” Ex.1006, 7:45-49; *see also* Ex.1006,

11:32-33 (“The client application can also be downloaded and installed on other non-mobile devices.”). Ex.1003, ¶¶52-53.

Platt is analogous art to the ’362 patent because it is in the same field of endeavor as the ’362 patent, namely, the “manner in which functionality is accessed in certain environments, such as mobile device environments.” Ex.1001, 1:66-67; *see* Ex.1006, [0001]-[0002] (describing ways that “functionality” is accessed on “portable devices, such as smartphones”). Abuan and the ’362 patent also both generally relate to implementing functionality via software. *See* Ex.1001, Abstract (“providing additional functionality to existing software”); Ex.1006, 1:66-2:1 (“Systems, methods, and apparatus are provided to enable real-time communication with people, groups and devices”), 7:45-49 (describing a “mobile node 102 (e.g., an iPhone) or other wireless device [that] downloads a software client application (sometimes referred to as a “client app”) from a server”). Ex.1003, ¶54.

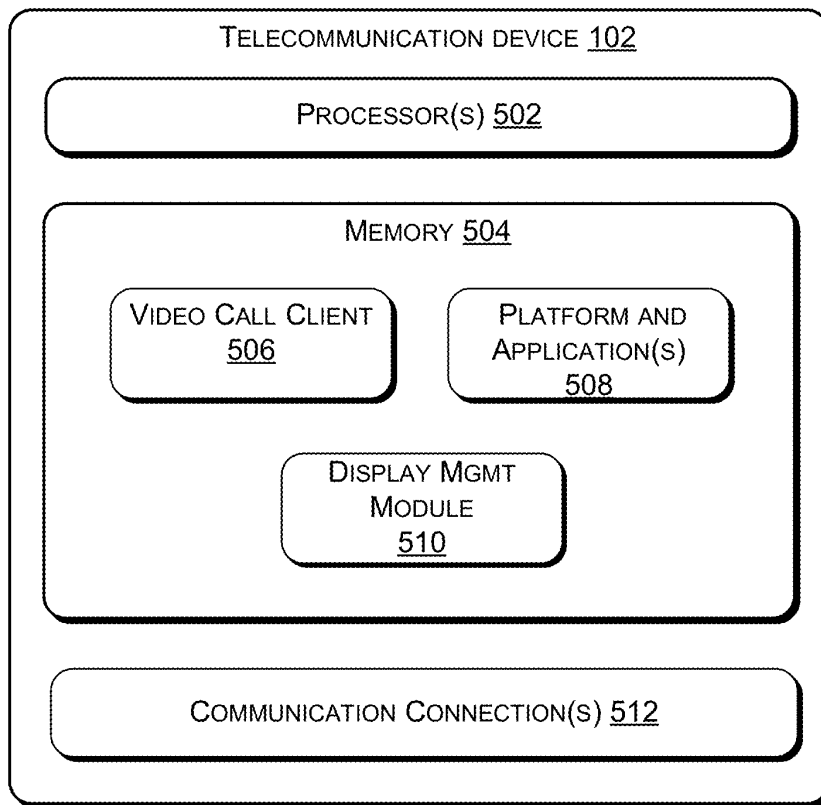
3. Reasons to Combine Abuan with Platt

A POSITA would have combined Abuan and Platt because Platt merely provides additional implementation details regarding the distribution of an application, such as Abuan’s video conference client, to a mobile device, and doing so would have been no more than the combination of prior art elements (Abuan’s video conference client with Platt’s app store) according to known methods to

yield predictable results (distribution of Abuan’s video conference client over an app store). Ex.1003, ¶61. More specifically, Abuan contemplates that its video conference client is “an application that may use the video conferencing functions” such as Skype. Ex.1005, [0105], [0171]. And it was well-known that such applications, including Skype itself, were distributed through Apple’s App Store. *See, e.g.*, Ex.1021; Ex.1003, ¶61. Platt provides additional confirmation of how wireless application distribution is achieved for applications such as Skype or Abuan’s application. Ex.1006, 7:45-49. Ex.1003, ¶61.

4. Morrison

Morrison relates to device operations during a video call, for telecommunications devices including “a smart phone, a tablet computer, a personal digital assistant (PDA), a personal computer (PC), a laptop computer, a media center, a work station, etc.” Ex.1007, [0020], [0014]. More specifically, Morrison “describes, in part, a telecommunication device that allows a user to multi-task during a video call by concurrently displaying a video call user interface and an activity user interface.” Ex.1007, [0014]. Morrison’s Figure 5, below, “illustrates an example telecommunication device configured to enable a user to engage in a telecommunication device activity while participating in a video call.” Ex.1007, [0044]. Ex.1003, ¶¶55-56.



Ex.1007, Fig. 5

Morrison provides implementation details regarding how an application would identify rendering information, explaining the operation of the “display management module 510” that “receive[s] user interface data from the video call client 506 and/or from the platform and application(s) 508.” Ex.1007, [0049]. The display management module renders user interface data, and controls sizing of the video call UI and activity UI. Ex.1007, [0049]; Ex.1003, ¶¶57-58. These rendering details, including resizing the video call UI and moving the video call UI, are reflected in Figure 1, below:

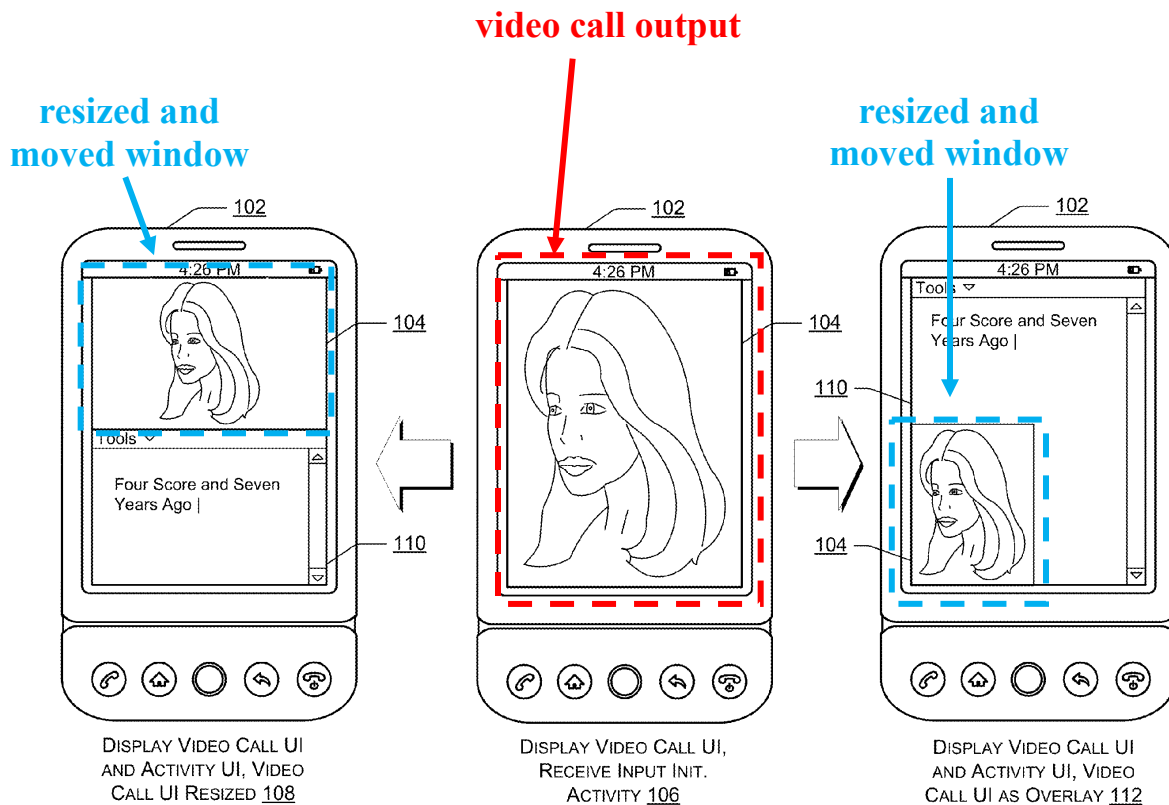


FIG. 1

Ex.1007, Fig. 1 (annotated).

Morrison explains, with respect to Figure 1, that “the telecommunication device 102 receives 106 input initiating a telecommunication device activity,” such as a user interaction with the screen or a button. Ex.1007, [0022]. “In response to receiving the input, the telecommunication device 102 may either display 108 an activity UI 110 adjacent to a reduced-size video call UI 104 or display 112 a reduced-size video call UI 104 as an overlay to the activity UI 110.” Ex.1007, [0023]. In both instances reflected in Figure 1 (e.g., 108 on the left and 112 on the

right), “involve reducing a display size of the video call UI 104.” Ex.1007, [0024], [0025]. Ex.1003, ¶¶58-59.

Morrison is analogous art to the ’362 patent because it is in the same field of endeavor as the ’362 patent, namely, the “manner in which functionality is accessed in certain environments, such as mobile device environments.” Ex.1001, 1:66-67; *see* Ex.1007, [0020], [0044]. Morrison and the ’362 patent also both generally relate to implementing functionality via software. *See* Ex.1001, Abstract (“providing additional functionality to existing software”); Ex.1007, [0044]-[0046] (“video call client 506, platform and application(s) 508, and display management module 510 [that] comprise... any... sort of executable instructions.”). Morrison and the ’362 patent both describe at length features for facilitating videoconferencing. Ex.1001, Abstract, 4:43-51 (function blocks provide “audio, video,...conferencing, meetings, and/or other functions”); Ex.1007, [0047] (describing a “video call client...capable of receiving and rendering video and audio streams”). Ex.1003, ¶60.

5. Reasons to Combine Abuan, Platt, and Morrison

A POSITA would have combined Abuan, Platt, and Morrison because Morrison provides additional implementation details regarding how an application, such as Abuan’s video conference client, would operate to identify hardware information and render video content, and doing so would have been no more than

the use of known techniques (Morrison's modules for rendering video calls) to improve similar devices (Abuan's video conference client modules) in the same way. Ex.1003, ¶62. Abuan's video conference client includes modules "a video conference module 1302, a media exchange module 1320, a buffer 1325, a captured image processing unit (CIPU) driver 1330, an encoder driver 1335, and a decoder driver 1340." Ex.1005, [0170]. Morrison further describes similar modules, including a "display management module 510" for a video-call client. Ex.1006, [0049]. Ex.1003, ¶62.

6. Independent Claim 1 and dependent claims

[1.0] A method for enabling an end to end connection for a first user and a second user to participate in a real-time communication session over the internet, the method comprising:

Abuan provides "a method for managing a video conference between a first device and a second device." Ex.1005, Abstract. Abuan's devices transmit "captured images to one or more devices **during a real-time communication session** between the [devices'] users." Ex.1005, [0003]; *see also* Ex.1005, [0033]-[0034]. Ex.1003, ¶¶63-64.

Figure 7, below, illustrates commencing a video conference between two mobile devices. In the figure, "one device (i.e., the device 705) requests a video conference and another device (i.e., the device 720) responds to such request." Ex.1005, [0108]; Ex.1003, ¶65.

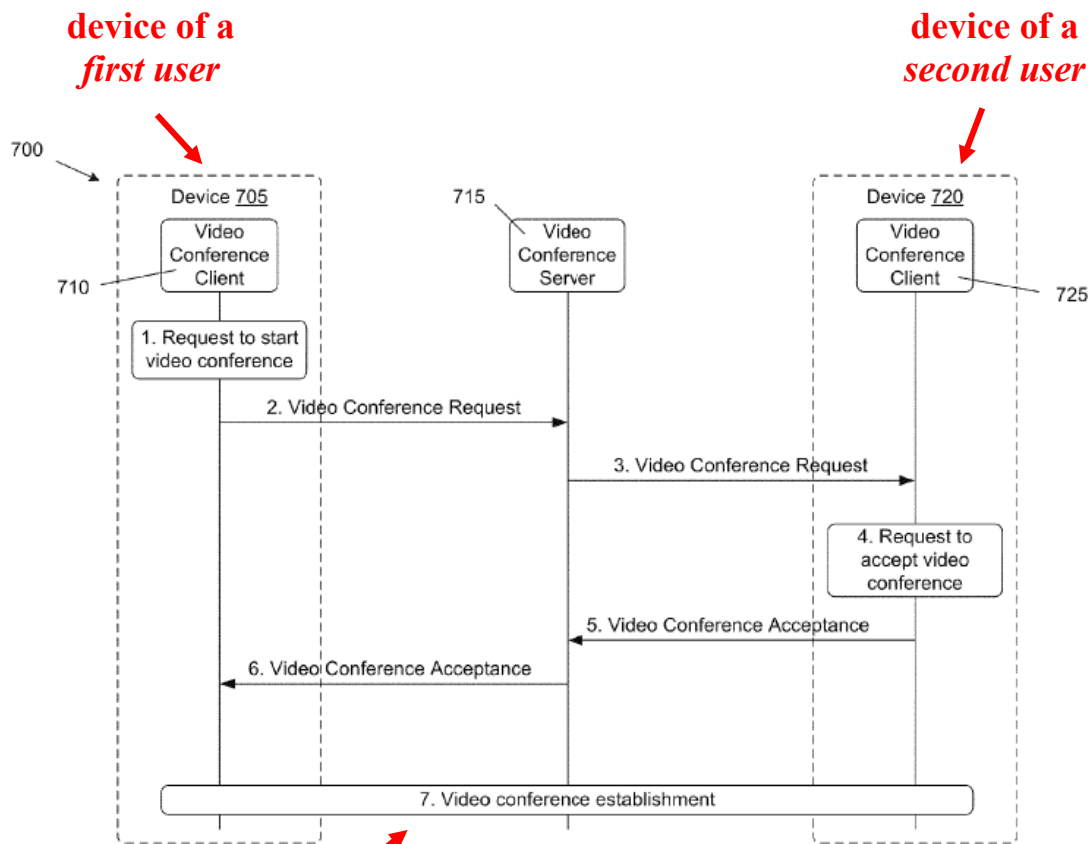


Figure 7

**Real-time communication session
(video conference)**

Ex.1005, Fig. 7 (annotated).

Abuan enabl[es] an end to end connection by establishing a video conference via “Video Conference Request” and “Video Conference Acceptance” messages. The video conference that is established in step 7 of Figure 7, above, is between a first user’s device 705 and a second user’s device 720. “A peer-to-peer video conference connection of some embodiments allows the mobile devices in the video conference to directly communicate with each other (instead of communicating through a central server, for example).” Ex.1005, [0151]; *see also*

Ex.1005, [0114] (conference establishment includes “negotiating a connection” between devices). Ex.1003, ¶66.

Abuan explains that the video conference client for each respective user communicates using a network connection such as the Internet. Ex.1005, [0109] (“Each of the video conference clients 710 and 725 of some embodiments communicates with the video conference server 715 over a network (e.g.,...the Internet etc.)”). From the point of view of each user device, data for a video conference is sent to and received from the other user’s device over a network such as the Internet. Ex.1005, [0106] (“[T]he network interface 650 is a communication interface that allows the video conference module 625 and the video conference client 645 to send data and receive data over a network,” such as sending “data (e.g., images captured by cameras of the dual camera mobile device) to another device on the Internet”)); Ex.1003, ¶67.

In sum, and as detailed further in the analysis of each element of claim 1 below, Abuan discloses or renders obvious *enabling an end to end connection* (establishing a video conference connecting two user devices) *for a first user* (e.g., user of device 705) *and a second user* (e.g., user of device 720) *to participate in a real-time communication session* (establishing a real-time communication session such as a video conference) *over the internet* (communicating via the Internet). Ex.1003, ¶68.

[1.1] creating for storage on a server, a superblock application configured for use by a computing device of the first user and a superblock application configured for use by a computing device of the second user,

Abuan and Platt render obvious this limitation.

“*superblock application*”: Abuan describes a video conferencing and processing module 1300 that includes a client application 1365 and various modules and components, including a video conference module 1302, that provide services to the client application. Ex.1005, [0170], [0183]. The client application is a “video conference client...that uses the video conferencing functions of the video conference module 1302.” Ex.1005, [0170]-[0171]. Abuan’s “applications are implemented as software processes that are specified as a set of instructions recorded on a computer readable storage medium,” such as the memory of Abuan’s mobile device. Ex.1005, [0233]. The modules are comprised of software instructions, such as video conferencing instructions 1986, that are stored in memory. Ex.1005, [0233], [0254], Fig. 19; *see also* [0254] (explaining how an “API-implementing component” may be a “module”); Ex.1039, [0032] (“programmable elements, often referred to as modules” “may include a program, a subroutine, a portion of a program, or a software component”); Ex.1003, ¶69-70. Thus, the video conferencing and processing module 1300 (including the client application 1365 and the various modules) corresponds to the claimed “*superblock application*.” Ex.1003, ¶70.

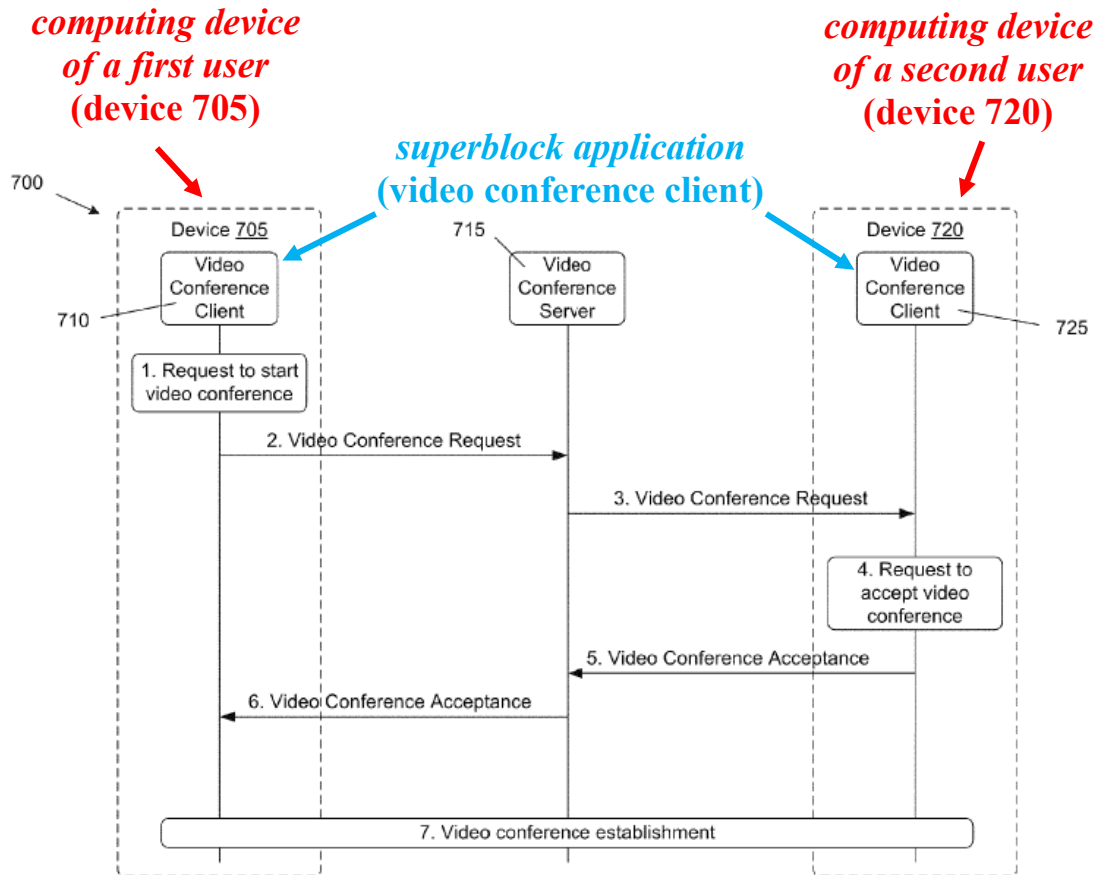


Figure 7

Ex.1005, Fig. 7 (annotated).

Each of video conference clients 710 and 725 is the same as the client application 1365, because Abuan explains that “video conference clients 710 and 725 are the same as the video conference client 645,” which in turn is the same as “the client application 1365.” Ex.1005, [0108], [0170]; Ex.1003, ¶72. Thus, the video conference client 710 is part of the “*superblock application*,” as explained above. Ex.1003, ¶72.

Platt describes distribution an application to devices such as iPhones (e.g., *a computing device of the first user and...a computing device of the second user*).

Ex.1003, ¶73. Platt explains that wireless devices download a software client application (*superblock application*) from a *server* maintained by a software application distributor, such as the Apple App Store. Ex.1006, 7:44-65; Ex.1003, ¶73.

Such teachings from Platt are complementary to Abuan, which contemplates details regarding a stand-alone video conferencing application such as Skype. Ex.1005, [0105]. Platt, in turn, describes how to market and distribute applications, such as Abuan's stand-alone video conferencing application, to users of smartphones. Ex.1006; Ex.1038 (highlighting that 2010 was "the year of apps," highlighting both the Apple App Store and Android Marketplace); Ex.1003, ¶74. Accordingly, it would have been obvious to a POSITA for the video conferencing clients detailed by Abuan (*superblock application*) to be distributed using a *server* maintained or operated by a provider of an App Store as taught by Platt. Ex.1003, ¶¶74-75.

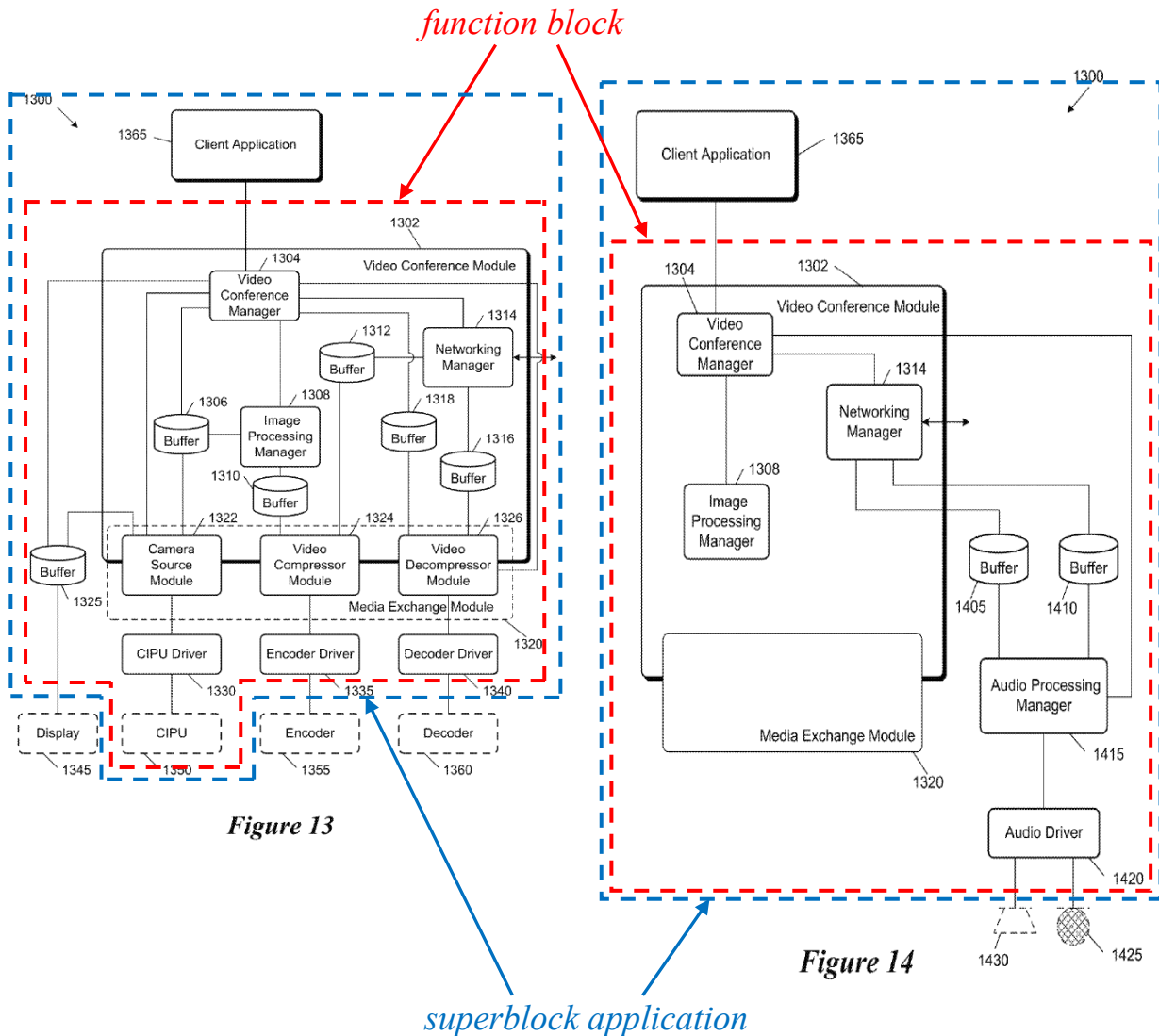
Thus, Abuan in view of Platt renders obvious this limitation.

[1.2] wherein each of the superblock applications has a function block compiled therein, wherein each respective superblock application communicates with its respective function block using application programming interface (API) calls;

As explained above at [1.1], Abuan's video conferencing and processing module 1300 (**including** the client application 1365 and the various modules) corresponds to the claimed "*superblock application*." Ex.1003, ¶¶76-77.

“function block”: Abuan’s various modules and components implementing functions provided to the client application 1365 (e.g., the video conference module 1302) correspond to the claimed *“function block.”* See, e.g., Ex.1005, [0056], [0099], [0170], [0183]. In other words, the video conferencing and processing module 1300, but excluding client application 1365, corresponds to the claimed *“function block.”* Ex.1003, ¶78.

The *superblock application* and *function block* are annotated in Abuan’s Figures 13 and 14 below. Consistent with the ’362 patent specification’s examples, the *function block* is part of the *superblock application*. See, e.g., Ex.1001, Fig.3B; Ex.1003, ¶79.



Ex.1005, Figs. 13 and 14 (annotated).

Abuan explains that “client application 1365 [*the superblock application*]...uses the video conferencing functions [*functionality*] of the video conference module 1302 [*the function block*].” Ex.1005, [0171]. Further, Abuan describes the application employing the videoconferencing module and other modular components as “a stand-alone application.” Ex.1005, [0105]. A POSITA would have understood Abuan’s reference to a “stand-alone application” to mean

that all the functionality of the application, including its module subcomponents (e.g., *the function block* functionality) is *compiled* into a single, unitary application. See Ex.1012, 1:26-29 (“Traditionally, an application's source files are compiled in object modules and then linked together with other object modules, generically called libraries, to form a complete stand-alone application.”); Ex.1003, ¶80.

For example, Abuan describes how an application accesses the functionality of the other software modules via defined application programming interfaces (APIs). Ex.1005, [0235]-[0237]. This structure puts functionality in the API-implementing component (e.g., *the function block*) and not in the API-calling component (e.g., *the superblock application*):

An API allows a developer of an API-calling component (which may be a third party developer) to **leverage specified features provided by an API-implementing component**.

Ex.1005, [0239]; *see also id.*, [0247]; Ex.1003, ¶81.

The use of APIs to define standard, programmatic ways for an application to access functionality of a software module was well-known and commonly used. Material Fact #3; Ex.1003, ¶82. Abuan explains that these API functions or methods are *compiled* into stand-alone applications. Ex.1005, [0239] (explaining that an “API can be... **compiled when an application is built**”).

Thus, it would have been obvious to a POSITA for *each of the superblock*

applications (Abuan's client application operating on each users' device) to include the module subcomponents as *a function block compiled therein* and performed within the execution context of Abuan's client application and not some other application. Ex.1003, ¶¶83-84.

Thus, Abuan renders obvious this limitation.

[1.3] receiving, by the superblock application on the computing device of the first user, a request to initiate a real-time communication session with the second user;

Abuan teaches that the video conference client 710 (part of the *superblock application*) receives *a request* from the user of the device to start a video conference with the user of another device, such as device 720 (*receiving...a request to initiate a real-time communication session with the second user*). See, e.g., Ex.1005, [0110]-[0111]. For instance, Abuan explains that “[t]he video conference client 710...receives the request to start the video conference when the user of the device 705 selects a user interface (UI) item...” Ex.1005, [0110]. Abuan explains that, based on a user's selection of “a user interface (UI) item,” the video conference client 710 (corresponding to the video conference client 1365), “sends---a video conference request...to the video conference server 715” to start the video conference session with the user of device 720 (*the second user*). Ex.1005, [0110]-[0111]; see also Ex.1005, [0118]-[0119], Fig. 8 (describing a “Video Conference option 854 (which can be implemented as a selectable icon)

[that] allows the user to start a video conference”). Ex.1003, ¶¶85-86.

As noted above in the discussion of limitation [1.0], Abuan’s Figure 7 illustrates one example sequence for commencing a video conference between two mobile devices: device 705 and device 720. Each mobile device is shown running an instance of Abuan’s video conference client 710 (*superblock application*). And Abuan explains that “the video conference client 710 receives (at operation 1) a request from a user of the device 705 to start a video conference with the device 720.” Ex.1005, [0110]; Ex.1003, ¶87. This is reflected in operation 1 of Figure 7, below:

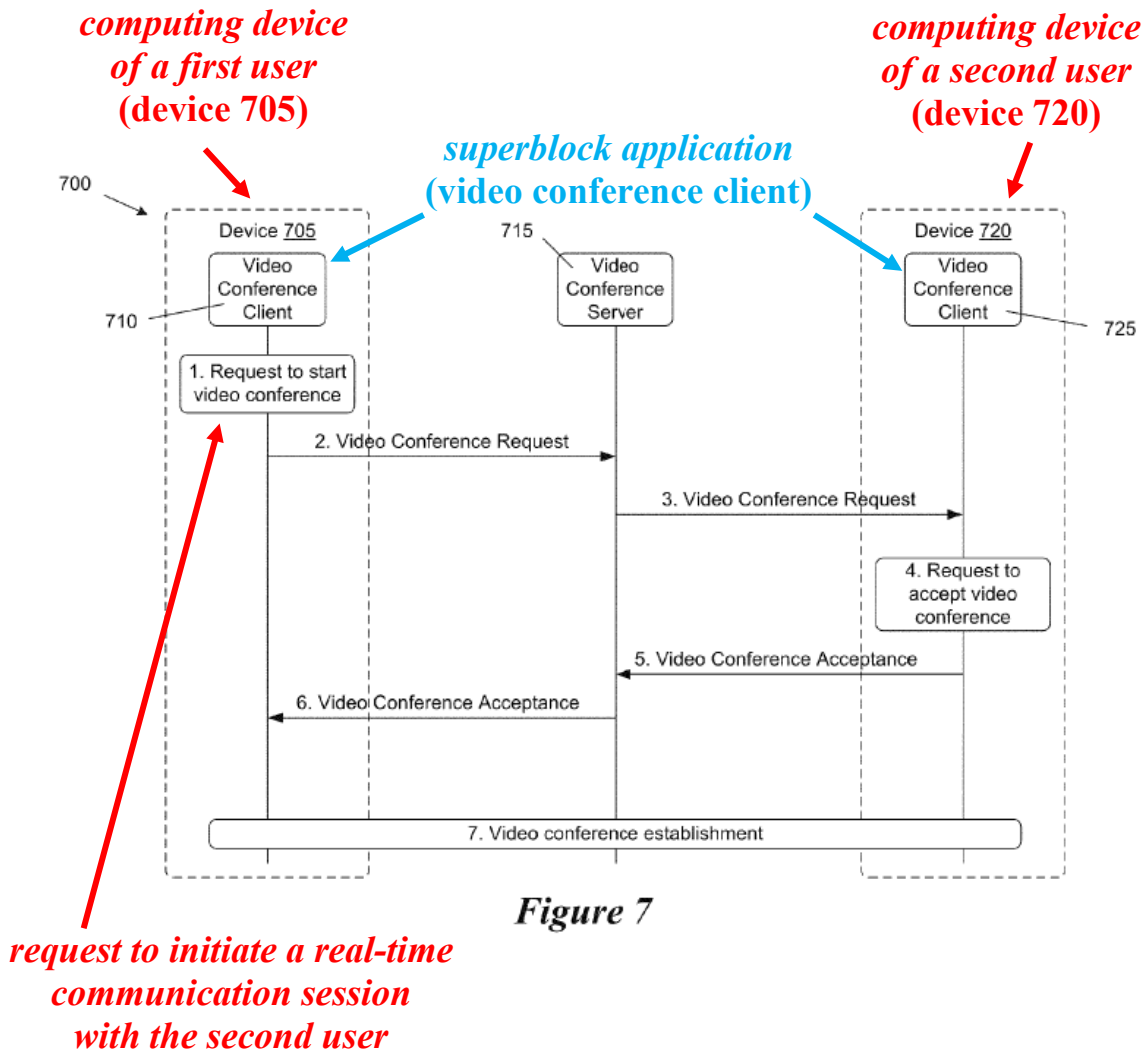


Figure 7

Ex.1005, Fig. 7 (annotated).

Thus, Abuan discloses or at least renders obvious *receiving, by the superblock application on the computing device of the first user, a request to initiate a real-time communication session with the second user.* Ex.1003, ¶¶88-89.

[1.4] communicating, by the superblock application on the computing device of the first user, the request to a remote resource;

As explained above at [1.3], based on a user's selection of the "user interface (UI) item," which initiates "*the request*," the video conference client 710 "sends---

a video conference request...to **the video conference server 715**” to start the video conference session with the user of device 720. Ex.1005, [0110]-[0111]. This is reflected in operation 2 of Figure 7, below. Ex.1003, ¶¶90-91.

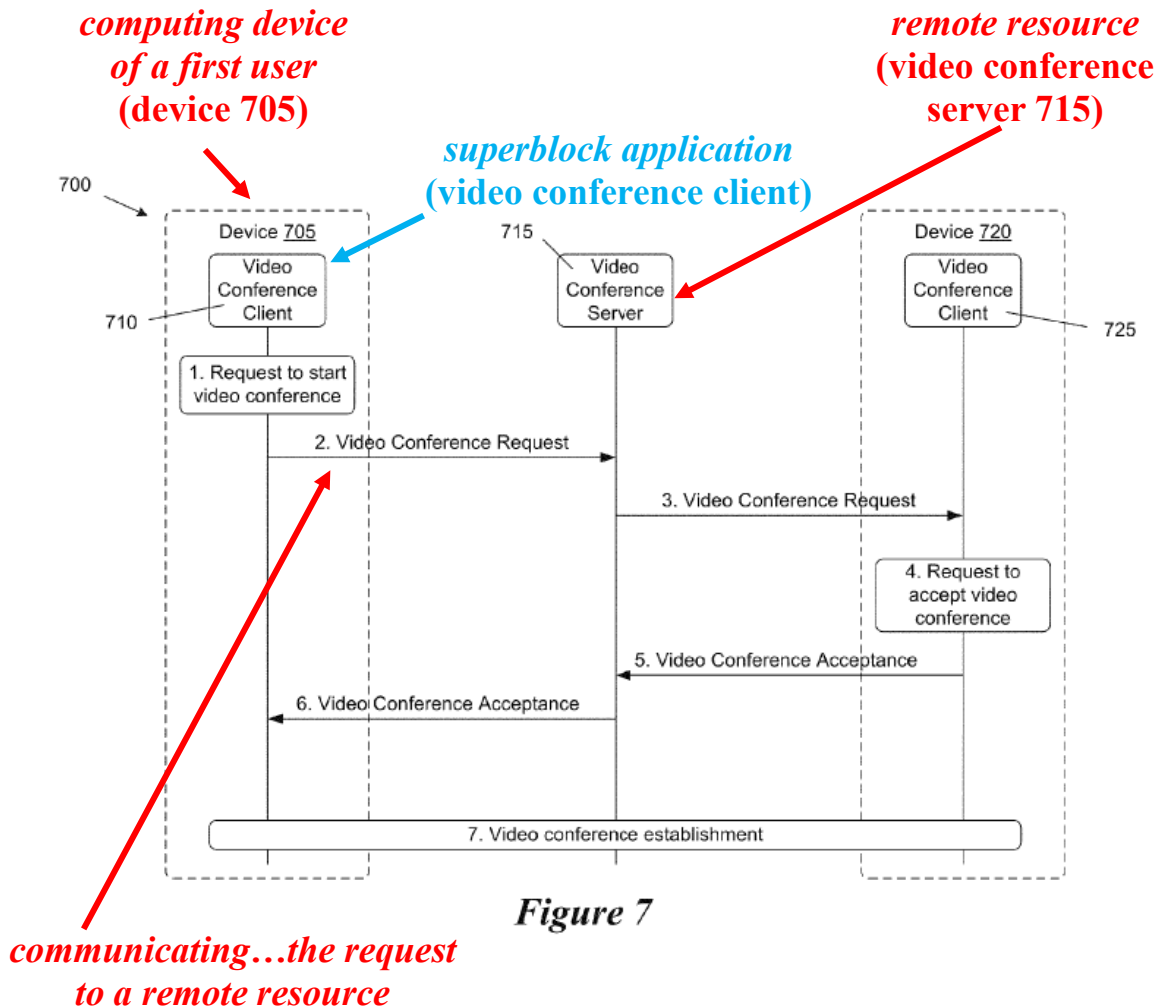


Figure 7

Ex.1005, Fig. 7 (annotated).

The video conference server 715 is shown to be a separate device from the “client device 705” running the video conference client application. Ex.1005, Fig. 7. The video conference server 715 is “a remote resource.” Ex.1003, ¶92. The video conference client application 710 transmitting *the request* to the video

conference server 715 to initiate the video conference with the user of the second device 720 teaches “*communicating, by the superblock application on the computing device of the first user, the request to a remote resource.*” Ex.1005, [0111], [0113]; Ex.1003, ¶¶92-93.

Thus, Abuan discloses or at least renders obvious this limitation.

[1.5] providing, by the superblock application on the computing device of the second user, a notification that the real-time communication session with the first user is being requested;

Abuan teaches that a video conference request sent to the video conference server, as discussed above in [1.4], is then sent from the video conference server to the video conference client of a second device (*the superblock application on the computing device of the second user*), where user interface is displayed to indicate that a request to start a video call by the first user (*providing...a notification that the real-time communication session with the first user is being requested*).

Ex.1005, [0108] (“one device (i.e., the device 705) requests a video conference and another device (i.e., the device 720) responds to such request.”). This is reflected in operation 4 of Figure 7, below. Ex.1003, ¶¶94-96.

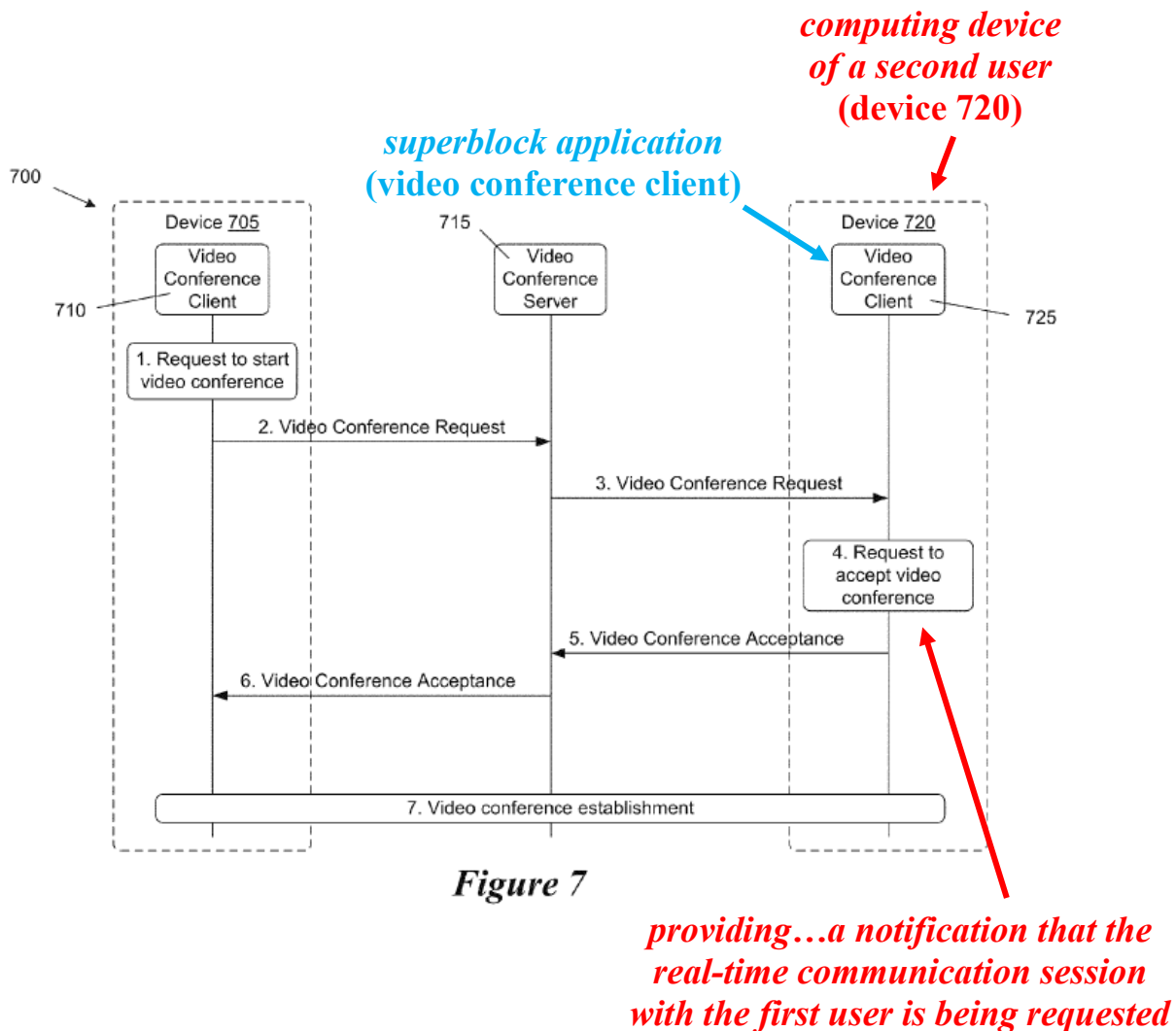


Figure 7

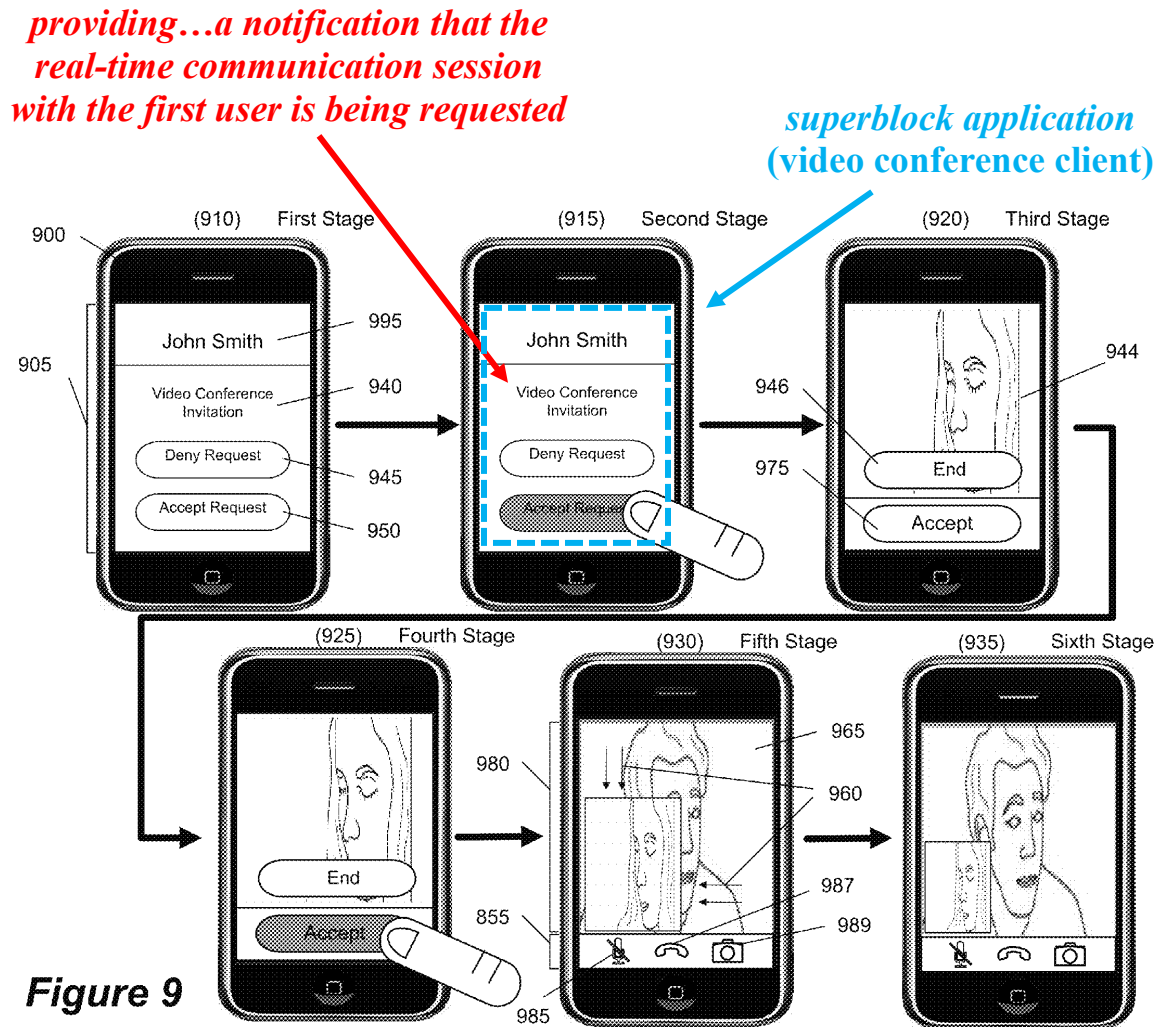
Ex.1005, Fig. 7 (annotated).

Abuan explains that the video conference client operating on device 720 (i.e., the superblock application on the computing device of the second user) receives a video conference request, the user interface displays a notification.

Ex.1005, [0112] (“When the video conference client 725 of some embodiments receives the video conference request, a user interface is displayed on the device 720 to indicate to the user of the device 720 that the user of the device 705 sent a

request to start a video conference and to prompt the user of the device 720 to accept or reject the video conference request.”). Ex.1003, ¶97.

An example user interface for notifying a user receiving a video conference request (the *second user*) is illustrated in Figure 9, below. The user interface displays a notification with items including “a name field 995, a message field 940, and two selectable UI items 945 and 950.” Ex.1005, [0130]. “Upon seeing the ‘Video Conference Invitation’ notation displayed in the message field 940, the invite recipient may deny or accept the request by selecting the Deny Request option 945 or Accept Request option 950 in the UI, respectively.” Ex.1005, [0131]; Ex.1003, ¶¶98-99.



Ex.1005, Fig. 9 (annotated).

Thus, Abuan discloses or at least renders obvious this limitation.

[1.6] identifying, by the superblock application on the computing device of the second user, rendering information; and

First, Abuan specifically contemplates that an application can identify the capabilities of the hardware device¹ on which it is running through the use of API

¹ The dependent claims of the '362 patent provide additional details that confirm

calls. Ex.1005, [0243] (“an API call can report to an application the capabilities or state of a hardware device”). Abuan explains that the client hardware device includes common output devices such as “speaker 1430” and “a display.” Ex.1005, [0184], [0252]. The capabilities or state of the hardware device (*computing device of the second user*) is *rendering information* because it provides information related to input capabilities, output capabilities, processing capability, and related information that would be relevant to audio input and audio and video output for a video conference call. *See* Ex.1005, [0243]; Ex.1003, ¶¶100-102. It would have been obvious to a POSITA for Abuan’s stand-alone application (*superblock application*) to determine their capabilities (*rending information*) based at least in part on the output capabilities of output devices such as the speaker and display. Ex.1003, ¶102.

Second, Morrison provides additional implementation details regarding how an application would identify rendering information, such as information related to user interface data and display information. Morrison discloses that a display management module 510 determines what portion of the window of the platform

rendering information entails hardware capabilities, including information corresponding to a mic (claim 6), speaker (claim 7), window (claim 8), and camera (claim 14).

and application(s) 508 can be allocated to display video call from the video call client 506, i.e., whether the platform and application(s) 508 can provide a window to display a video call from the video call client 506. When the display management module 510 receives “user interface data from both the video call client 506 and the platform and application(s) 508, [it] may determine what portion of the display to allocate to each of the video call client 506 and the platform and application(s) 508” “based on [] user preferences or settings.” Ex.1007, [0049]; Ex.1003, ¶¶103-104.

Thus, Abuan alone or in view of Morrison renders obvious this limitation.

[1.7] rendering, by the superblock application on the computing device of the second user, the real-time communication session with the first user using the rendering information.

First, Abuan teaches that the video conference client of the second user device (*the superblock application on the computing device of the second user*) shows a display area containing the video from the first user (*rendering...the real-time communication session with the first user*) after a call request has been accepted. Ex.1005, [0112] (“a user interface is displayed on the device 720” (referencing Fig. 7)); *see* Ex.1005, [0149] (“FIGS. 7, 8, 9, 10, and 11 show several ways of establishing a video conference”); Ex.1003, ¶¶105-106. Abuan explains that “after the remote user has accepted the video conference request,” the UI changes to “show a **display area 1192 that contains the video from a camera of**

the remote user behind the display area 1180.” Ex.1005, [0148] (referencing Fig.

11); *see* Ex.1005, [0149] (“FIGS. 7, 8, 9, 10, and 11 show several ways of establishing a video conference”). Further, once a user has accepted a video conference request, “the UI 805 can display a display area 870 (e.g., a display window 870) that contains the video from a camera of the remote device” to create “a PIP display 880 that has a foreground inset display 860 of the local user's video and a background main display 870 of the remote user.” Ex.1005, [0124] (referencing Fig. 8); *see* Ex.1005, [0149] (“FIGS. 7, 8, 9, 10, and 11 show several ways of establishing a video conference”). Ex.1003, ¶106.

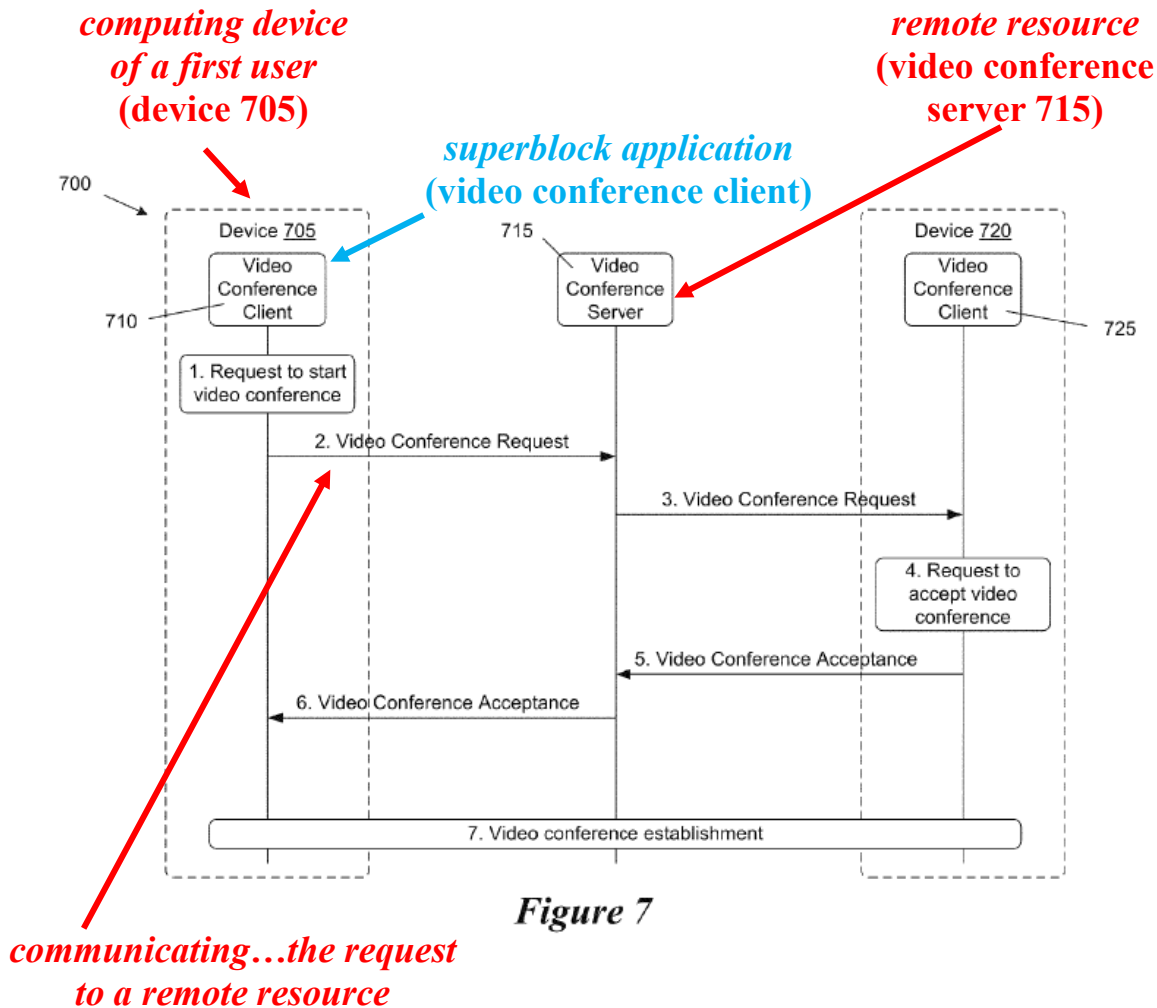
Second, as discussed above in [1.6], Morrison discloses that a display management module 510 determines what portion of the window of the platform and application(s) 508 can be allocated to display video call from the video call client 506. In Morrison, the display management module 510 “**render[s]** one as an overlay in front of a portion of the other,” such as “display[ing] a reduced-size **video call** UI 104 as an overlay to the activity UI 110.” Ex.1007, [0049], [0023]; Ex.1003, ¶107. When the display management module 510 receives “user interface data from both the video call client 506 and the platform and application(s) 508, [it] may determine what portion of the display to allocate to each of the video call client 506 and the platform and application(s) 508” “based on [] user preferences or settings. Such [] preferences or settings could include preferences or

settings...for **rendering** one [user interface] as an overlay in front of a portion of the other.” Ex.1007, [0049]; Ex.1003, ¶¶107-108.

Thus, Abuan alone or in view of Morrison renders obvious this limitation.

Limitation [2.0]

As discussed above in the analysis of [1.4], Abuan discloses or at least renders obvious *communicating, by the superblock application on the computing device of the first user, the request to the remote resource*, as reflected in operation 2 of Figure 7, below. Ex.1003, ¶¶109-110.



Ex.1005, Fig. 7 (annotated).

Further, Abuan renders obvious *communicating, by the superblock application on the computing device of the first user, the request to the function block on the computing device of the first user* because Abuan explains that the video conference client application sends instructions from a user of the application to the video conference module 1302, including instructions to start a video conference. Ex.1005, [0172] (“client application 1365... sends instructions to the video conference module 1302... to start a conference” and “routes

instructions from a user... to the video conference module 1302,” referencing Fig.

13), [0187] (“the instructions are received from the client application 1365 or are received from a user through a user interface...,” referencing Fig. 15); *see also* Ex.1005, [0186] (Fig. 15 illustrates “operation of the video conference manager 1304”); Ex.1003, ¶111.

As discussed above in [1.1], Abuan’s video conferencing client application, shown in Figure 13 as client application 1365, is part of *the superblock application*. *See* Ex.1005, [0171] (“The client application 1365 may be an application that uses the video conferencing functions of the video conference module 1302....”). Abuan’s client application 1365 (*superblock application*) “uses the video conferencing functions of the video conference module 1302” (*function block*), as reflected below in Figure 13. Ex.1003, ¶112.

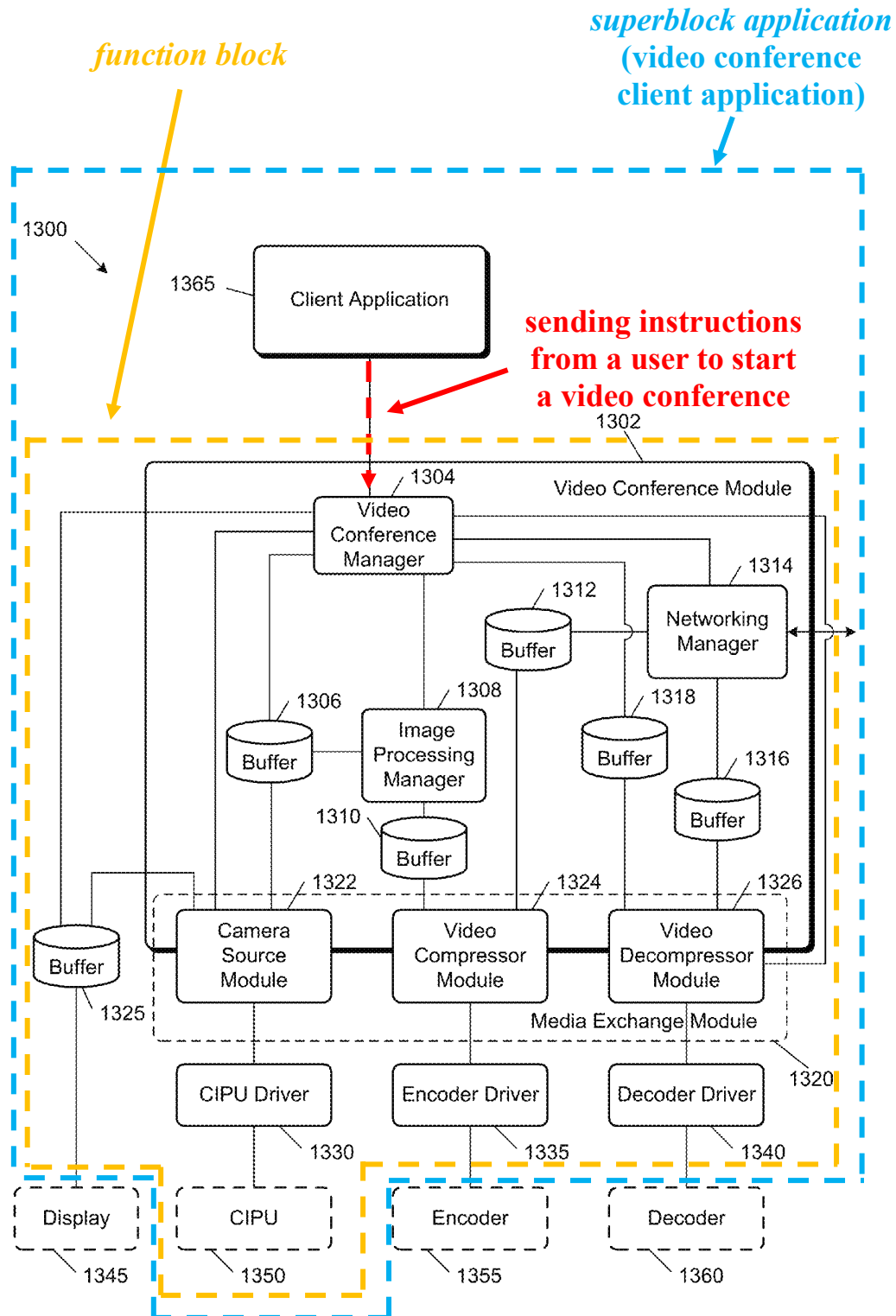


Figure 13

Ex.1005, Fig. 13 (annotated).

Finally, as discussed above at [1.1], Abuan teaches that API calls are used (*using one or more application programming interface (API) calls*) to allow a client program (*superblock application*) to access services provided by the video conference module 1302 (*function block*). See Ex.1005, [0243] (“the application or client program may incorporate calls to functions or methods provided by the SDK and provided by the API or use data types or objects defined in the SDK and provided by the API”). Ex.1003, ¶¶113-114.

Thus, Abuan renders obvious this limitation.

Limitation [2.1]

As discussed above in [1.5], Abuan teaches that a video conference request sent to the video conference server, which is then sent from the video conference server to the video conference client of a second device (*the superblock application on the computing device of the second user*), where user interface is displayed to indicate that a request to start a video call by the first user (*providing...the notification that the real-time communication session with the first user is being requested*). See Ex.1005, [0108]. This is reflected in operation 4 of Figure 7, below. Ex.1003, ¶¶115-116.

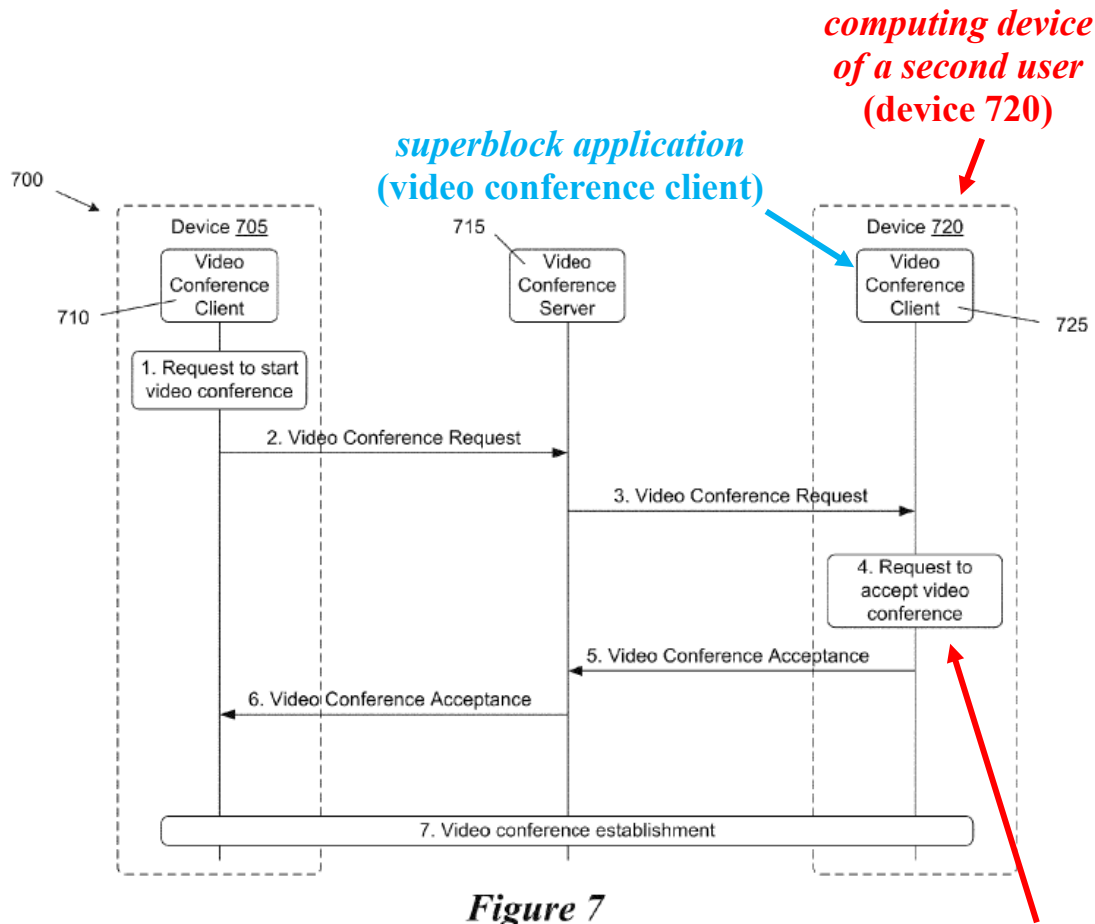


Figure 7

providing...the notification that the real-time communication session with the first user is being requested

Ex.1005, Fig. 7 (annotated).

Abuan explains, with reference to Figure 7 above, that a “video conference server 715 forwards (at operation 3) [a] video conference request [from device 705] to the video conference client 725 of the device 720” to establish a video conference between device 705 and device 720. Ex.1005, [0110]-[0111]. This communication between the video conference client 725—operating on device 720—and the video conference server 715, is performed by the video

conference module 1302 on that device. More specifically, Abuan discloses that “[e]ach of the video conference clients 710 and 725...communicates with the video conference server 715 over a network (e.g., a cellular network, a local area network, a wireless network, a network of networks, the Internet etc.) through a network interface such as the network interface 650.” Ex.1005, [0109]; Ex.1003, ¶117. Abuan explains that the networking manager 1314 of the video conference module 1302 (or network layer 640 of the video conference module 625) “establishes the connections between the dual camera mobile device and the other device of the video conference at the start of the video conference.” Ex.1005, [0179]; *see also* [0104]. Thus, a POSITA would have found it obvious that the video conference module 1302 (part of *the function block*) receives the video conference request from the video conference server 715 and then provides the video conference request to the video conference client 725 (*providing the notification by the function block on the computing device of the second user*). Ex.1003, ¶¶117-118.

Thus, Abuan renders obvious this limitation.

Limitation [2.2]

As discussed above in [1.6], Abuan’s application identifies the capabilities of the hardware device using an API call. Ex.1005, [0243]; Ex.1003, ¶¶119-120. Abuan further explains that when instructions to start a video conference are received,

modules of the video conference module 1302 are initialized. Ex.1005, [0188]. For instance, “initializing the CIPU 1350 includes instructing the CIPU 1350 to start processing images captured by **one or both cameras of the dual camera mobile device.**” Ex.1005, [0189]. “To initialize the audio processing manager 1415, some embodiments instruct the audio processing manager 1415 to begin encoding audio data captured by **the microphone 1425** and decoding audio data stored in the buffer 1410 (which was received from the other device) in order to output to **the speaker 1430.**” Ex.1005, [0189]; Ex.1003, ¶120.

Further, Abuan explains that “changes to video conference settings may also be received from the client application 1365[.]” Ex.1005, [0191]; Ex.1003, ¶121. For instance, “a change to an image capture setting” may be sent from the client application 1365 to the video conference module 1302, including “switching cameras (i.e., switching which camera on the dual camera mobile device will capture video), focus adjustment, exposure adjustment, displaying or not displaying images from one or both cameras of the dual camera mobile device, and zooming in or out of images displayed on the dual camera mobile device, among other setting changes.” Ex.1005, [0193]. These changes, communicated from the client application 1365 (*superblock application*), provide information on which hardware components to utilize (*provid[ing]...rendering information*), such as which camera to use for video input. Ex.1003, ¶121.

Accordingly, a POSITA would have found it obvious that by initializing modules of the video conference module 1302 (*the function block*), including modules that provide video and audio processing, *the rendering information* associated with the capabilities of the hardware device used for that video and audio processing is provided to the video conference module 1302. Ex.1003, ¶122.

Further, Morrison explicitly teaches information, including information related to the display of the device, is provided *by the superblock application...to the function block*. Morrison explains that when the display management module 510 receives “user interface data from both the video call client 506 and the platform and application(s) 508, [it] may determine what portion of the display to allocate to each of the video call client 506 and the platform and application(s) 508” “based on [] user preferences or settings.” Ex.1007, [0049]; Ex.1003, ¶¶123-124.

Thus, Abuan alone or in view of Morrison renders obvious this limitation.

Limitation [2.3]

As discussed above in [1.7], Abuan teaches that the video conference client of the second user device shows a display area containing the video from the first user (*rendering...the real-time communication session with the first user*) after a call request has been accepted. *See* Ex.1005, [0148], [0124]. And as discussed above in [2.2], Abuan explains that in establishing a video conference, modules of

the video conference module 1302 are initialized. Ex.1005, [0188]; Ex.1003, ¶¶125-126.

Abuan further explains that after “all of the required modules have been initialized, the process 1500 generates (at 1520) composite images for displaying on the dual camera mobile device (i.e., local display).” Ex.1005, [0190]; Ex.1003, ¶127. And components of the video conference module 1302 (*function block*) further process the video call information received using the hardware components, including “encoding audio data captured by the microphone 1425 and decoding audio data stored in the buffer 1410 (which was received from the other device) in order to output to the speaker 1430.” Ex.1005, [0189]. Thus, Abuan teaches outputting the audio and video elements of the video call (*rendering...the real-time communication session*) using the information about the hardware device providing the audio and video data (*using the rendering information*). Ex.1003, ¶127.

Further, as discussed above in [1.7], Morrison teaches that its display management module 510 (*function block*) “**render[s]** one as an overlay in front of a portion of the other,” such as “display[ing] a reduced-size **video call** UI 104 as an overlay to the activity UI 110.” Ex.1007, [0049], [0023]. When the display management module 510 receives “user interface data from both the video call client 506 and the platform and application(s) 508, [it] may determine what portion of the display to allocate to each of the video call client 506 and the platform and

application(s) 508” “based on [] user preferences or settings. Such [] preferences or settings could include preferences or settings...for **rendering** one [user interface] as an overlay in front of a portion of the other.” Ex.1007, [0049]; Ex.1003, ¶128. Accordingly, Morrison similarly teaches that *rendering...the real-time communication session* is performed in the *superblock application* by the *function block*, using the rendering information related to the display. Ex.1003, ¶¶128-129.

Thus, Abuan alone or in view of Morrison renders obvious this limitation.

Claim 3

Abuan describes a process 1500 that begins with “receiving (at 1505) instructions to start a video conference.” Ex.1005, [0187]. For example, the video conference may be started by the user interacting with the user interface of the client application (“*superblock application*”) shown in Figures 8-10. *See, e.g.*, Ex.1001, [0110]. The user’s interaction, in turn, causes the client application to start initializing the modules required for a videoconference. Ex.1005, [0187] (“instructions to start a video conference...are received...from a user through a user interface...and forwarded to the video conference manager 1304 by the client application 1365”). Ex.1003, ¶¶130-131.

Abuan describes initializing the videoconference manager 1304 and other modules (“*initializing...the respective function block*”) in response to the request to start a videoconference. Ex.1005, [0188] (“the process 1500 **initializes (at 1510) a**

first module that interacts with the video conference manager 1304” including “the CIPU 1350, the image processing manager 1308, the audio processing manager 1415, and the networking manager 1314.”); *see also id.* [0189]-[0190] (initialization continues until “all... required modules have been initialized”); Ex.1003, ¶132.

Since the module initialization occurs in response to user’s interaction with the client application user interface, the initialization is performed “*by the superblock application.*” Ex.1003, ¶132. Since the client application is running and presenting a user interface to the user, it would have been obvious that the initialization occurs when the client application is loaded and running (“*following initialization of the superblock application on the computing device*”). Ex.1003, ¶132. For example, Abuan describes how an application is “read into memory for processing by a processor.” Ex.1005, [0234]; *see also id.* [0233] (“applications are implemented as software processes that are specified as a set of instructions...executed by one or more processing unit(s)”; Ex.1003, ¶¶132-133.

Claim 4

As discussed above in [1.2], Abuan teaches that API calls are used to allow a client application (*superblock application*) to access services provided by the video conference module 1302 (e.g., *function block*), where the services are compiled into stand-alone applications. *See* Ex.1005, [0243], [0239]. Abuan further discloses

that API calls, such as the API calls used within stand-alone applications to access services provided by the libraries compiled with the application, include parameters that affect how the API-implementing component provides the service. More specifically, these include *configurable parameter[s]* that are configured by the API-implementing component (i.e., *the superbloc application*). Ex.1005, [0241] (explain that a “parameter can be a constant, key, data structure, object, object class, variable, data type, pointer, array, list or a pointer to a function or method or another way to reference a data or other item to be passed via the API.”); Ex.1003, ¶¶134-135.

More specifically, Abuan describes various types of *configurable parameter[s]* that are used to configure aspects of the video conference, including “a profile setting for configuring a camera” to use “high resolution video,” “low resolution video,” “still images,” or various power levels. Abuan, [0068], [0058]; Ex.1003, ¶136. Additionally, Abuan explains that the client application, or the video conference module, receives, through the user interface, changes to “video conference settings” that control aspects of the video conference. Ex.1005, [0191] (“The changes to video conference settings may also be received from the client application 1365 or other modules in the video conference module 1302”).

Those changes include changes to “image capture settings,” such as “switching cameras,” “focus adjustment, exposure adjustment,” or a change to

network settings. Ex.1005, [0192]-[0193]. Each of these settings is an example of a *configurable parameter* that is provided to the function block and controls aspects of the video conference. Ex.1003, ¶137. And Abuan's API-calling components, which send function calls from the application to video conference module 1302, "pass[] data and control **information using parameters** via the API calls or invocations" for purposes of making changes such as those Abuan describes regarding the video conference settings. Ex.1005, [0241]; *see also* Ex.1005, [0235]-[0236]. Abuan renders obvious that the application *configures...at least one configurable parameter of [its] respective function block* because API calls are sent from the application to the video conference module 1302. *See* Ex.1005, [0241]. Ex.1003, ¶¶137-138.

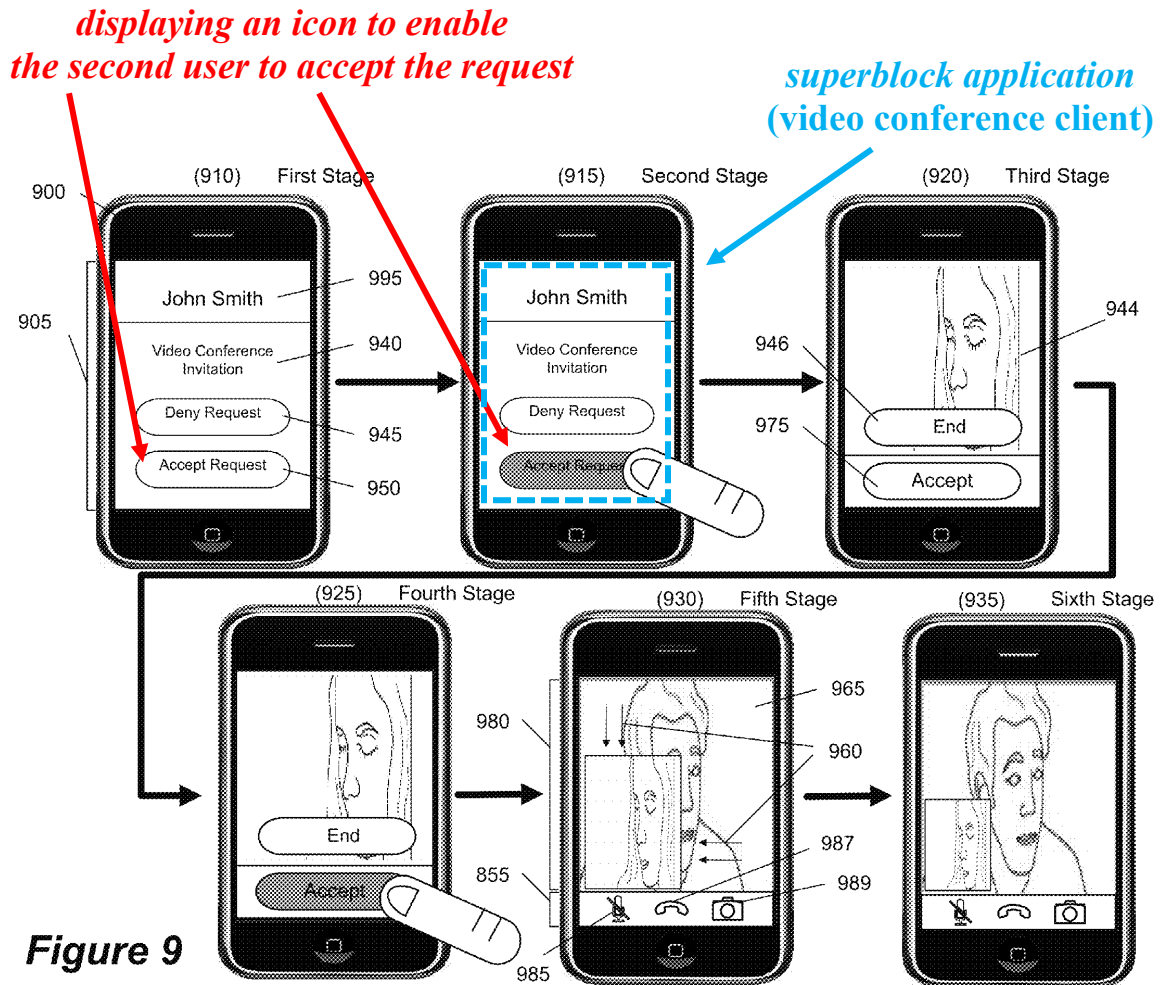
Thus, Abuan renders obvious this claim.

Claim 5

As explained above at [1.5], Abuan's disclosure of UIs indicating incoming calls or requests for video conferences discloses "*providing the notification.*" *See* Ex.1005, Fig. 9; Ex.1003, ¶¶139-140.

Abuan explains that information regarding the incoming call, as well as selectable UI options to accept or deny a video conference request (*an icon to enable the second user to accept the request*), are displayed (*displaying*) to the user (*the second user*) when a video conference is requested. Ex.1005, [0130] (Fig. 9

illustrates “the UI 905 when the invite recipient receives an invitation to a video conference from the invite requestor, John Smith.”). Abuan explains that the UI includes a message field 940 that “displays an invite from the invite requestor to the invite recipient.” Ex.1005, [0130]. When a video conference is requested, “selectable UI items 945 and 950 (which can be implemented as selectable buttons) provide selectable Deny Request and Accept Request options 945 and 950 for the invite recipient to use to reject or accept the invitation.” Ex.1005, [0130]; Ex.1003, ¶141.



Ex.1005, Fig. 9 (annotated).

As reflected in Figure 9, above, the “accept request” selectable UI icon is *an icon to enable the second user to accept the request* because by selecting accept request, “the invite recipient has agreed to join the video conference.” Ex.1005, [0131]. Ex.1003, ¶¶142-143.

Thus, Abuan renders obvious this claim.

Claim 6²

Abuan specifically contemplates that an application can identify the capabilities of the hardware device on which it is running through the use of API calls. Ex.1005, [0243] (“an API call can report to an application the **capabilities or state of a hardware device**”). Abuan specifically contemplates that its “dual camera mobile device” includes a “microphone 1425.” Ex.1005, [0184]; Ex.1003, ¶¶144-145. Thus, a POSITA would have recognized that in reporting capabilities of a hardware device to the video conferencing application, such information would have included information corresponding to the microphone contemplated by Abuan and utilized for audio input during a video conference. Ex.1003, ¶145.

Further, Abuan explains that when instructions to start a video conference are received, modules of the video conference module 1302 are initialized. Ex.1005, [0188]. “To initialize the audio processing manager 1415, some embodiments instruct the audio processing manager 1415 to begin encoding audio

² Claim 1 does not recite a step of “*providing the rendering information.*” The language “*wherein providing the rendering information includes...*” found in claims 6-8 and 14 is understood to refer to providing information to the function block on the computing device of the second user, like the similar *providing* step in [2.2].

data captured by **the microphone 1425** and decoding audio data stored in the buffer 1410 (which was received from the other device) in order to output to the speaker 1430.” Ex.1005, [0189]; Ex.1003, ¶146.

Accordingly, a POSITA would have found it obvious that by initializing modules of the video conference module 1302, including modules that provide audio processing, *the rendering information* associated with the capabilities of the hardware device used for that audio processing is provided, including *information corresponding to a mic of the computing device*, to the video conference module 1302. Ex.1003, ¶¶147-148.

Thus, Abuan renders obvious this claim.

Claim 7

Abuan specifically contemplates that an application can identify the capabilities of the hardware device on which it is running through the use of API calls. Ex.1005, [0243] (“an API call can report to an application the **capabilities or state of a hardware device**”). Abuan specifically contemplates that its “dual camera mobile device” includes a “speaker 1430.” Ex.1005, [0184]. Thus, a POSITA would have recognized that in reporting capabilities of a hardware device to the video conferencing application, such information would have included information corresponding to the speaker contemplated by Abuan and utilized for audio output during a video conference. Ex.1003, ¶¶149-150.

Further, Abuan explains that when instructions to start a video conference are received, modules of the video conference module 1302 are initialized.

Ex.1005, [0188]. “To initialize the audio processing manager 1415, some embodiments instruct the audio processing manager 1415 to begin encoding audio data captured by the microphone 1425 and decoding audio data stored in the buffer 1410 (which was received from the other device) in order to output to the **speaker 1430**.” Ex.1005, [0189]. Ex.1003, ¶151.

Accordingly, a POSITA would have found it obvious that by initializing modules of the video conference module 1302, including modules that provide audio processing, *the rendering information* associated with the capabilities of the hardware device used for that audio processing is provided, including *information corresponding to a speaker of the computing device*, to the video conference module 1302. Ex.1003, ¶¶152-153.

Thus, Abuan renders obvious this claim.

Claim 8

Abuan specifically contemplates that an application can identify the capabilities of the hardware device on which it is running through the use of API calls. Ex.1005, [0243] (“an API call can report to an application the **capabilities or state of a hardware device**”). Abuan specifically contemplates that its “dual camera mobile device” includes a “display 1345.” Ex.1005, [0170]. This display

displays “images of a video” during a video conference. *See* Ex.1005, [0170]; Ex.1003, ¶¶154-155. Thus, a POSITA would have recognized that in reporting capabilities of a hardware device to the video conferencing application, such information would have included information corresponding to the display contemplated by Abuan and utilized for video output during a video conference. Ex.1003, ¶155.

Further, Abuan explains that when instructions to start a video conference are received, modules of the video conference module 1302 are initialized, including “the image processing manager 1308.” Ex.1005, [0188]. “Some embodiments initialize the image processing manager 1308 by instructing the image processing manager 1308 to start retrieving images from the buffer 1310 and processing and encoding the retrieved images.” Ex.1005, [0189]. “When all of the required modules have been initialized, the process 1500 generates (at 1520) composite images for displaying on the dual camera mobile device (i.e., local display).” Ex.1005, [0190]; Ex.1003, ¶156.

Accordingly, a POSITA would have found it obvious that by initializing modules of the video conference module 1302, including modules that provide video processing, *the rendering information* associated with the capabilities of the hardware device used for that video processing is provided, including *information*

for displaying a window within the superblock application, to the video conference module 1302. Ex.1003, ¶157.

Further, Morrison provides additional implementation details regarding how an application would identify rendering information, such as information related to user interface data and display information. Morrison discloses that a display management module 510 determines what portion of the window of the platform and application(s) 508 can be allocated to display video call from the video call client 506, i.e., whether the platform and application(s) 508 can provide a window to display a video call from the video call client 506, and receives “user interface data from both the video call client 506 and the platform and application(s) 508[.]” Ex.1007, [0049]; Ex.1003, ¶¶158-159.

Thus, Abuan alone or in view of Morrison renders obvious this claim.

Claim 9

As discussed above in claim 8, Abuan specifically contemplates that its “dual camera mobile device” includes a “display 1345” that displays “images of a video” during a video conference. Ex.1005, [0170]. Abuan further explains that its “image processing manager 1308 [operates by] retrieving images from the buffer 1310 and processing and encoding the retrieved images.” Ex.1005, [0189]. Accordingly, a POSITA would have understood that Abuan’s display window is *a*

video window because it outputs processed images during a video conference.

Ex.1003, ¶¶160-162.

Thus, Abuan renders obvious this claim.

Claim 10

Morrison provides further implementation details regarding the display of a video during a video conference, including situations where it would be desirable to resize and move the displayed video. For instance, “displaying 108 the video call UI 104 and activity UI 110 adjacent to each other may involve reducing a display size of the video call UI 104.” Ex.1007, [0024]. “Displaying 112 the video call UI 104 as an overlay to a portion of the activity UI 110 may involve reducing a display size of the video call UI 104.” Ex.1007, [0025]. This is reflected in Morrison’s Figure 1, below. Ex.1003, ¶¶163-164.

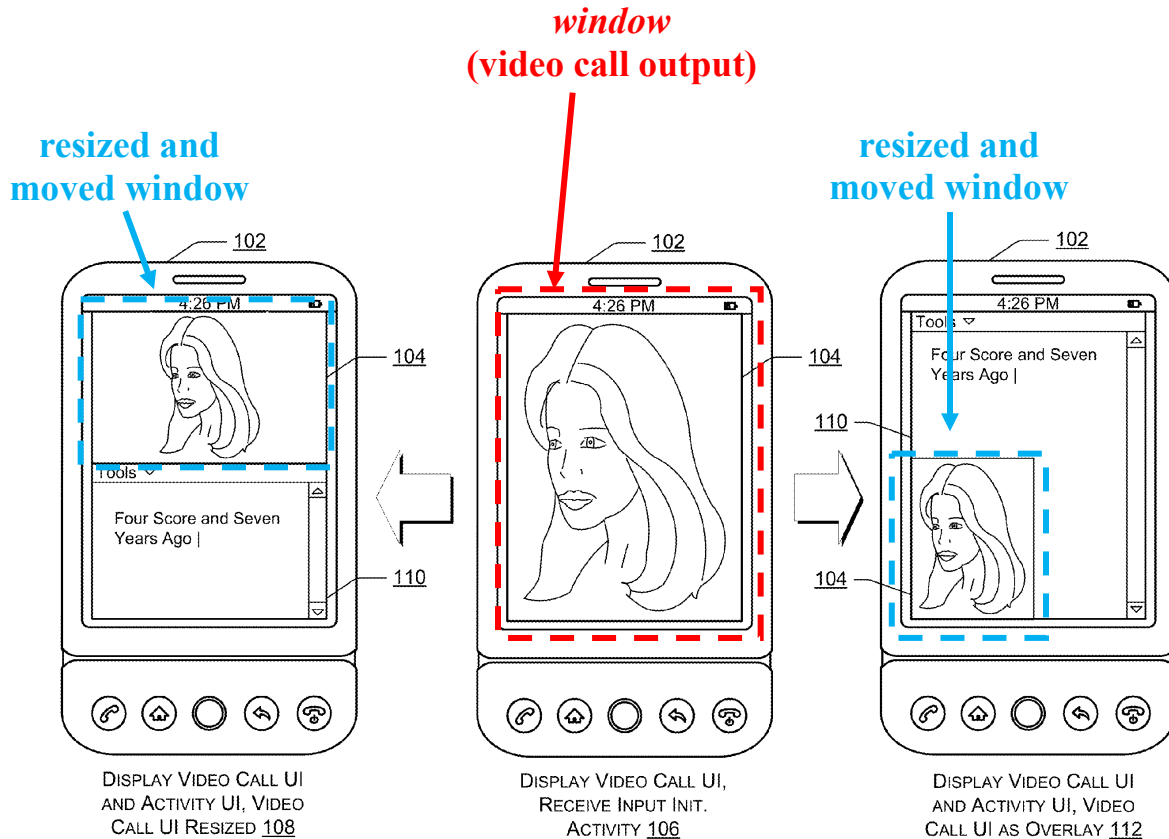


FIG. 1

Ex.1007, Fig. 1 (annotated).

A POSITA would have understood that *the window is resizable by the second user* because Morrison explains that the different arrangements that are illustrated in Figure 1 are selected based on a user's preferences or input. Ex.1007, [0049] (“In addition to determining and rendering, the display management module 510 may **receive user input to resize one of the video call UI 104** or activity UI 110 and may adjust the portions of the display allocated to each of the video call client 506 and the platform and application(s) 508.”), [0023] (“In response to receiving the input, the telecommunication device 102 may either display 108 an

activity UI 110 adjacent to a reduced-size video call UI 104 or display 112 a reduced-size video call UI 104 as an overlay to the activity UI 110.”); *see also* Ex.1007, [0023] (“The telecommunication device 102 may select among these display arrangements based on user preferences or default, suggested, or required settings.”), [0025] (“Such a reduction in display size may be performed automatically and may reduce the display size to resultant display size specified by user preferences or other settings.”); Ex.1003, ¶¶165-166.

Thus, Abuan in view of Morrison renders obvious this claim.

Claim 11

As discussed above with respect to claim 10, Morrison contemplates that a video display may be resized during a video call, for instance, in response to input provided by the user. *See* Ex.1007, [0049] (“the display management module 510 may receive user input to resize one of the video call UI 104 or activity UI 110 and may adjust the portions of the display allocated to each of the video call client 506 and the platform and application(s) 508.”), [0023]-[0025]. And as discussed above with respect to claims 2 and 8, Morrison discloses that a display management module 510 (*function block*) determines what portion of the window of the platform and application(s) 508 can be allocated to display video call from the video call client 506[.]” Ex.1007, [0049]. In this way, Morrison’s display management module 510 receives input from the application indicating that the

video display has been resized by the user and renders the updated, resized video display (*sending...an update for the window, wherein the update indicates that the window has been resized*). See Ex.1007, [0049]; Ex.1003, ¶¶167-168.

Further, Morrison explains that when a display size has been reduced, “the telecommunication device 102...may send a notification 308 of the display size reduction to the video call service 302.” Ex.1007, [0034]. And Abuan explains that the networking manager 1314 of the video conference module 1302 (part of *the function block*) (or network layer 640 of the video conference module 625) “establishes the connections between the dual camera mobile device and the other device of the video conference at the start of the video conference.” Ex.1005, [0179]; *see also* [0104]. Accordingly, in the combination, the video conference application (*superblock application*) sends a notification that display size had changed (*sending...an update for the window, wherein the update indicates that the window has been resized*) to the video conference module 1302 (part of *the function block*) to update the server. Ex.1003, ¶¶169-170.

Thus, Abuan in view of Morrison renders obvious this claim.

Claim 12

As discussed above with respect to claim 10, Morrison provides further implementation details regarding the display of a video during a video conference, including situations where it would be desirable to resize and move the displayed

video. *See* Ex.1007, [0023]-[0025]. As reflected in Figure 1, below, it would have been obvious to a POSITA that resizing a window, as contemplated in Morrison, further includes moving a window as the location of the window changes as the size and orientation of the window are adjusted. Ex.1003, ¶171-173.

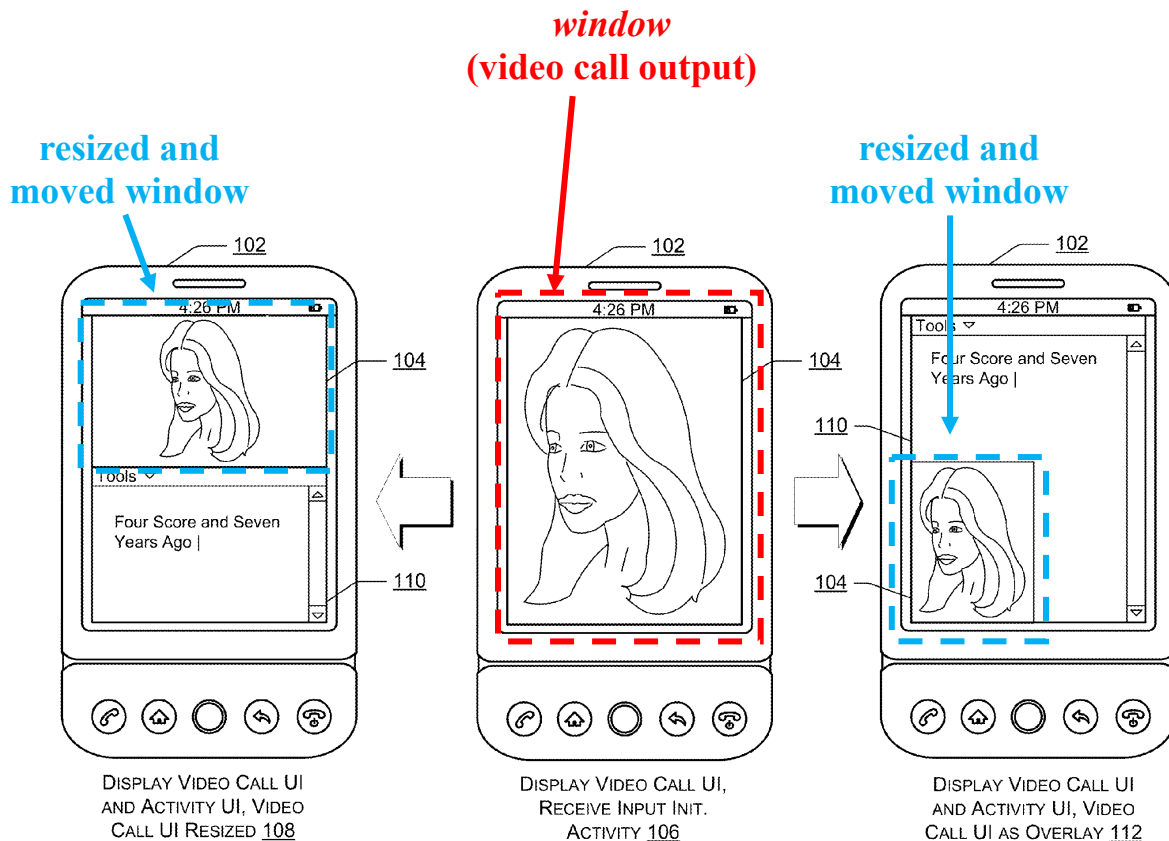


FIG. 1

Ex.1007, Fig. 1 (annotated).

Thus, Abuan in view of Morrison renders obvious this claim.

Claim 13

As discussed with respect to claim 12, it would have been obvious to a POSITA that resizing a window, per Morrison, would further include moving a

window because the location of a window changes as it is resized and moved. *See* Ex.1007, Fig. 1. And as discussed with respect to claim 11, Morrison’s display management module 510 receives input from the application indicating that the video display is adjusted, including what aspects of the display should be dedicated to the video call. *See* Ex.1007, [0049]. And as discussed above in claim 11, Morrison teaches that the display management module 510 renders the updated, moved video display. Ex.1007, [0049]; Ex.1003, ¶¶174-175. Further, as discussed above in claim 11, Morrison teaches updating the server in response to resizing the window. Ex.1007, [0034]. Morrison teaches that resizing and moving windows are related actions. *See* Ex.1007, [0023]-[0025], Fig. 1. Thus, in the combination, the video conference application (*superblock application*) sends a notification that display size had changed after being moved (*sending...an update for the window, wherein the update indicates that the window has been moved*) to the video conference module 1302 (part of the *function block*) to update the server. Ex.1003, ¶¶176-177.

Thus, Abuan in view of Morrison renders obvious this claim.

Claim 14

Abuan specifically contemplates that an application can identify the capabilities of the hardware device on which it is running through the use of API calls. Ex.1005, [0243] (“an API call can report to an application the **capabilities or**

state of a hardware device”). Abuan specifically contemplates that its “dual camera mobile device” includes two cameras. *See* Ex.1005, [0176], [0189] (“processing images captured by one or both cameras of the dual camera mobile device”), [0193] (“switching cameras (i.e., switching which camera on the dual camera mobile device will capture video)”); Ex.1003, ¶¶178-179. Thus, a POSITA would have recognized that in reporting capabilities of a hardware device to the video conferencing application, such information would have included information corresponding to which camera is used for video input as contemplated by Abuan and utilized for video input during a video conference. Ex.1003, ¶179.

Further, Abuan explains that when instructions to start a video conference are received, modules of the video conference module 1302 are initialized. Ex.1005, [0188]. “[I]nitializing the CIPU 1350 includes instructing the CIPU 1350 to start processing images captured by **one or both cameras of the dual camera mobile device**.” Ex.1005, [0189]. Ex.1003, ¶180.

Accordingly, a POSITA would have found it obvious that by initializing modules of the video conference module 1302, including modules that provide video processing, *the rendering information* associated with the capabilities of the hardware device used for that video processing is provided, including *information corresponding to a camera of the computing device*, to the video conference module 1302. Ex.1003, ¶¶181-182.

Claim 16

As discussed above in [1.2], Abuan explains that an API is implemented using an “API-calling component” to “access and use one or more functions, methods, procedures, data structures, classes, and/or other services provided by [an] API-implementing component.” Ex.1005, [0238]. In this way, Abuan explains, the “API allows a developer...to leverage specified features provided by an API-implementing component.” Ex.1005, [0239]. Accordingly, Abuan contemplates that the API calls made by a calling component (e.g., Abuan’s video conferencing application, which is part of the *superblock application*) are *compatible with an interface provided by* an implementing component (e.g., Abuan’s video conference module, which is part of the application’s *respective function block*). Ex.1003, ¶¶183-185.

Claim 17

Abuan describes how establishing a videoconference session includes establishing “a control communication channel [] for messaging between the local mobile device and a remote device.” Ex.1005, [0218]. This control communication channel is used for “sending and receiving requests, notifications, and acknowledgements,” plus “remote control instruction messages.” Ex.1005, [0218]. A POSITA would have recognized that such messages would be exchanged substantially instantly because they relate to the establishment, maintenance, and

control of the real-time videoconference communication session. As confirmation, Abuan explains that the control communication channel uses “protocols like a **real-time** transport control protocol (RTCP).” Ex.1005, [0218]. Abuan’s discussion of establishing a real-time messaging channel for controlling videoconference session renders obvious that “*the real-time communication session is an instant messaging service.*” Ex.1003, ¶¶186-182; *see also* Ex.1001, 17:34-35 (“instant messaging (IM) control module”); Material Fact #2.

Further, Abuan explains that its invention contemplates various types of *real-time communication session[s]*, including *an instant messaging service* over an instant messaging application. Ex.1005, [0171] (“The client application 1365 may be an application that uses the video conferencing functions of the video conference module 1302, such as...an instant messaging application.”); *see also* Ex.1005, [0270] (“**additional services can be provisioned**...such as data services like...**instant messaging**”); Ex.1003, ¶¶188-189.

Thus, Abuan renders obvious this claim.

Claim 18

Abuan explains that its invention contemplates various types of *real-time communication session[s]*, including *a voice call* such as over a voice-over-IP application. Ex.1005, [0171] (“The client application 1365 may be an application that uses the video conferencing functions of the video conference module 1302,

such as...a voice-over-IP (VoIP) application (e.g., Skype)"); Material Fact #1.

Further, as discussed above in claims 6-7, Abuan explains that video conference module 1302 (part of the *function block*) initializes modules for *voice call[s]*, including audio processing manager 1415, microphone 1425, speaker 1430, and performs “encoding [of] audio data captured by the microphone 1425 and decoding [of] audio data stored in the buffer 1410” Ex.1005, [0189]; Ex.1003, ¶¶190-193.

Thus, Abuan renders obvious this claim.

Claim 19

Abuan explains that its invention contemplates various types of *real-time communication session[s]*, including a *video call* over a video conferencing application. Ex.1005, [0171] (“The client application 1365 may be an application that uses the video conferencing functions of the video conference module 1302, such as a video conferencing application”). Ex.1003, ¶¶194-196.

Thus, Abuan renders obvious this claim.

Claim 20

Abuan’s teachings support that frameworks provided by Apple, which are intended to be used on the iOS operating system, can be implemented as a media exchange module. Ex.1005, [0044] (“One example of such a media exchange module 310 is the Core Media framework provided by Apple Inc.”); Ex.1043, 3:28-33 (describing “a mobile device executing the iOS™ operating system”

including a “media processing module” (framework) that “provides a low-level programming interface for managing and playing audiovisual media” such as “Core Media framework”); *see also* Ex.1023, 1; Ex.1037 (Wayback Machine capture showing “Core Media Framework Reference” from Apple iPhone developer documentation for “iOS application”); Ex.1003, ¶¶197-198. As explained in [1.1], Abuan’s video conferencing and processing module 1300 (including the client application 1365 and the various modules) is *the superblock application*, which includes media exchange module 1320. Ex.1005, [0181] (“is the same as the media exchange module 310 shown in FIG. 3, with more detail provided”). Apple’s Core Media framework is part of “a rich set of [APIs] and tools to create innovative applications for iPhone and iPod® touch.” Ex.1023, 1. Using Core Media framework—provided by Apple for applications running on iOS—as the media exchange module for Abuan’s video conferencing application 1365 means the application is *adapted to operate with iOS as an operating system*, as the framework provides interfaces for “iOS application[s].” Ex.1037; Ex.1003, ¶¶198-199.

Thus, Abuan renders obvious this claim.

Claim 22

Abuan describes mobile devices throughout its disclosure. *See, e.g.*, Ex.1005, Title, Fig. 1, [0001] (“portable devices, such as smartphones”), [0003]

(“mobile device”). For example, Abuan illustrates a communication setup process in Fig. 7 between two client devices 705 and 720, which may both be “dual camera mobile device[s].” Ex.1005, [0108]; Ex.1003, ¶¶200-202.

Thus, Abuan discloses or at least renders obvious this claim.

Claim 23

Abuan explains that its invention contemplates *mobile device[s]*, such as user device 705 and device 720 discussed above, including implementations where the mobile devices are mobile phones (i.e., *cellular phone[s]*). Ex.1005, [0032] (“Some embodiments of the invention provide a mobile device with two cameras that can take pictures and videos. Examples of mobile devices include mobile phones[.]”); Ex.1003, ¶¶203-205.

Thus, Abuan discloses or at least renders obvious this claim.

Claim 24

Abuan explains that its invention contemplates *mobile device[s]*, such as user device 705 and device 720 discussed above, including implementations where the mobile devices are *smart phone[s]*. Ex.1005, [0032] (“Some embodiments of the invention provide a mobile device with two cameras that can take pictures and videos. Examples of mobile devices include...smartphones[.]”). Ex.1003, ¶¶206-208.

Thus, Abuan discloses or at least renders obvious this claim.

Claim 25

Abuan explains that its invention contemplates *mobile device[s]*, such as user device 705 and device 720 discussed above, including implementations where the mobile devices are *tablet[s]*. Ex.1005, [0032] (“Some embodiments of the invention provide a mobile device with two cameras that can take pictures and videos. Examples of mobile devices include...tablet personal computers[.]”). Ex.1003, ¶¶209-211.

Thus, Abuan discloses or at least renders obvious this claim.

Claim 26

Abuan explains that its invention contemplates *mobile device[s]*, such as user device 705 and device 720 discussed above, including implementations where the mobile devices are *personal digital assistant[s]*. Ex.1005, [0032] (“Some embodiments of the invention provide a mobile device with two cameras that can take pictures and videos. Examples of mobile devices include...personal digital assistants (PDAs)[.]”). Ex.1003, ¶¶212-214.

Thus, Abuan discloses or at least renders obvious this claim.

Claim 27

Abuan explains that its invention contemplates *mobile device[s]*, such as user device 705 and device 720 discussed above, including implementations where the mobile devices are *laptop computer[s]*. Ex.1005, [0032] (“Some embodiments

of the invention provide a mobile device with two cameras that can take pictures and videos. Examples of mobile devices include...laptops[.]”). Ex.1003, ¶¶215-217.

Thus, Abuan discloses or at least renders obvious *the computing device of at least one of the first and second users is a laptop computer.*

Claim 28

While Abuan explains that its invention contemplates devices such as user device 705 and device 720 discussed above, Abuan further explains that its “embodiments are used in cases involving a video conference between a dual camera mobile device and another device, such as a single camera mobile device, **a computer**, a phone with video conference capability, etc.” Ex.1005, [0279]. And Morrison, which contemplates similar embodiments of a mobile device executing a video call, explains that its “telecommunication device 102 may be any of a smart phone, a tablet computer, a personal digital assistant (PDA), **a personal computer (PC)**, a laptop computer, a media center, a work station, etc.” Ex.1007, [0020]. Accordingly, a POSITA would have recognized that Abuan’s teachings would be applicable to *desktop computer[s]*. Ex.1003, ¶¶218-220.

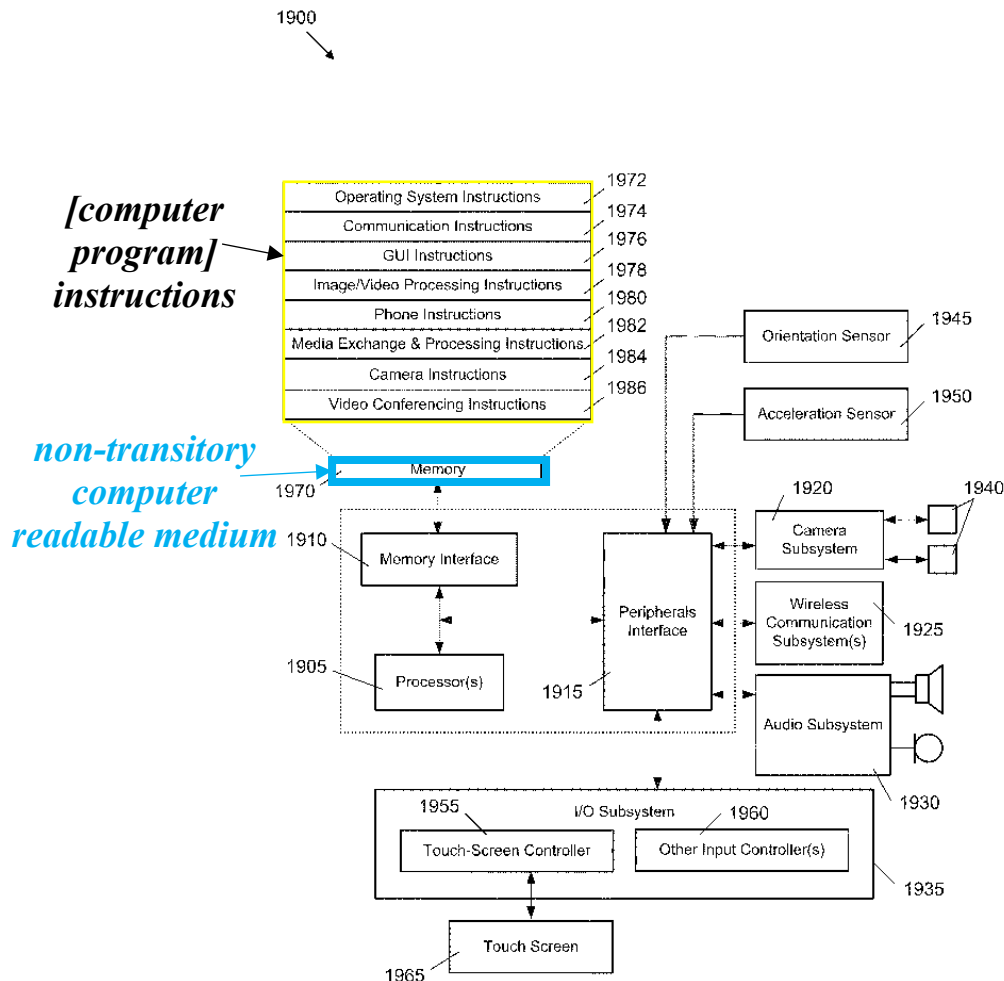
Thus, Abuan alone or in view of Morrison renders obvious this claim.

7. Independent Claim 75

[75.0] A non-transitory computer readable medium embodying a computer program comprising instructions for providing a real-time communication session using one or more servers connected over the internet to a superblock application intended for use on a computing device, the instructions comprising:

As discussed above in [1.0], Abuan illustrates “a method for managing a video conference between a first device and a second device.” Ex.1005, Abstract. Abuan’s methods “are implemented as software processes.” Ex.1005, [0235]. The software comprises program code embodied as computer program instructions residing in read-only memory. Ex.1005, [0234] (“In this specification, the term “software” is meant to include firmware **residing in read-only memory** or applications stored in magnetic storage which can be read into memory for processing by a processor.”). Ex.1003, ¶¶221-222.

Further, Abuan explains that the software (*computer program comprising instructions*) is stored in “a machine readable medium,” and “in a form readable by...a computer.” Ex.1005, [0248]; Ex.1003, ¶223.



Ex.1005, Fig. 19 (annotated).

Accordingly, Abuan’s “memory” storing computer program instructions for a video conference client application is a *non-transitory computer readable medium embodying a computer program comprising instructions*. Ex.1003, ¶224.

Further, as discussed above in [1.0], Abuan’s teachings pertain to the transmission of “captured images to one or more devices **during a real-time communication session** between the users of” mobile devices (*providing a real-time communication session*). Ex.1005, [0003]; Ex.1003, ¶225.

Abuan explains that its computing devices, such as device 705 or device 720, include a networking manager as part of the video conference module within the video conference client (*superblock application*), as reflected in Figure 13 and shown in further detail in Figure 17. Abuan explains that “the networking manager 1700 manages network connections (e.g., connection establishment, connection monitoring, connection adjustments, connection tear down, etc.) between a dual camera mobile device on which it operates and a remote device in a video conference.” Ex.1005, [0208]; Ex.1003, ¶226.

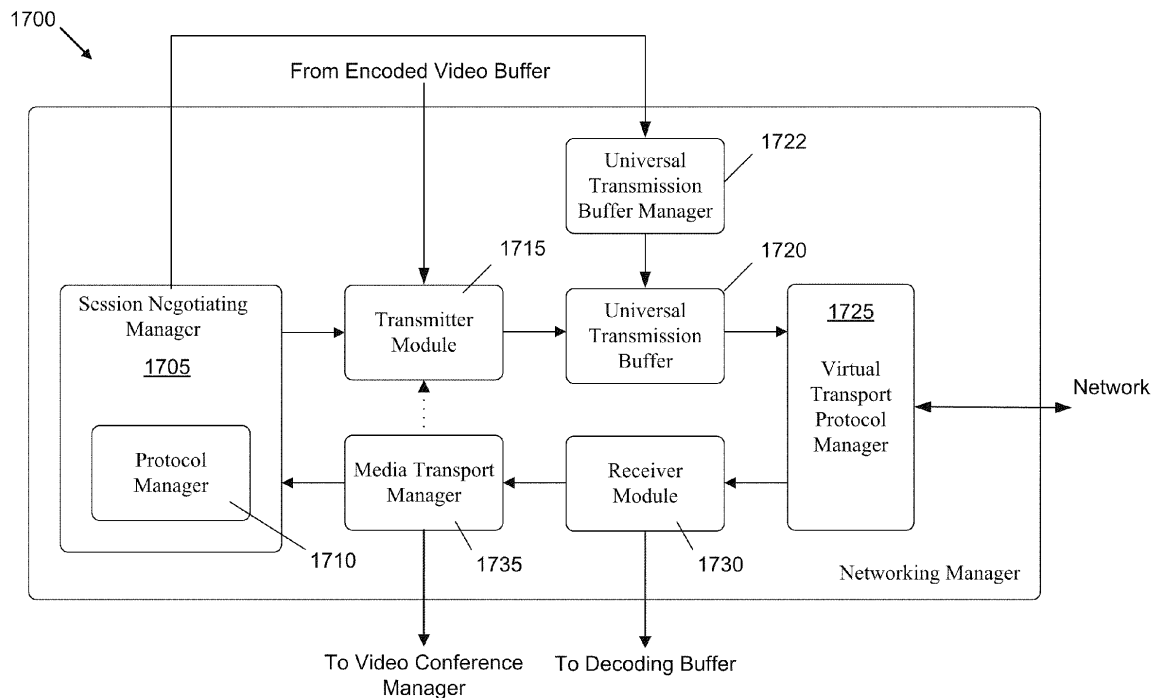


Figure 17

Ex.1005, Fig. 17.

As discussed above in [1.4], Abuan explains that a video conference is established between a first user and a second user using an intermediary video conference server. Ex.1003, ¶227.

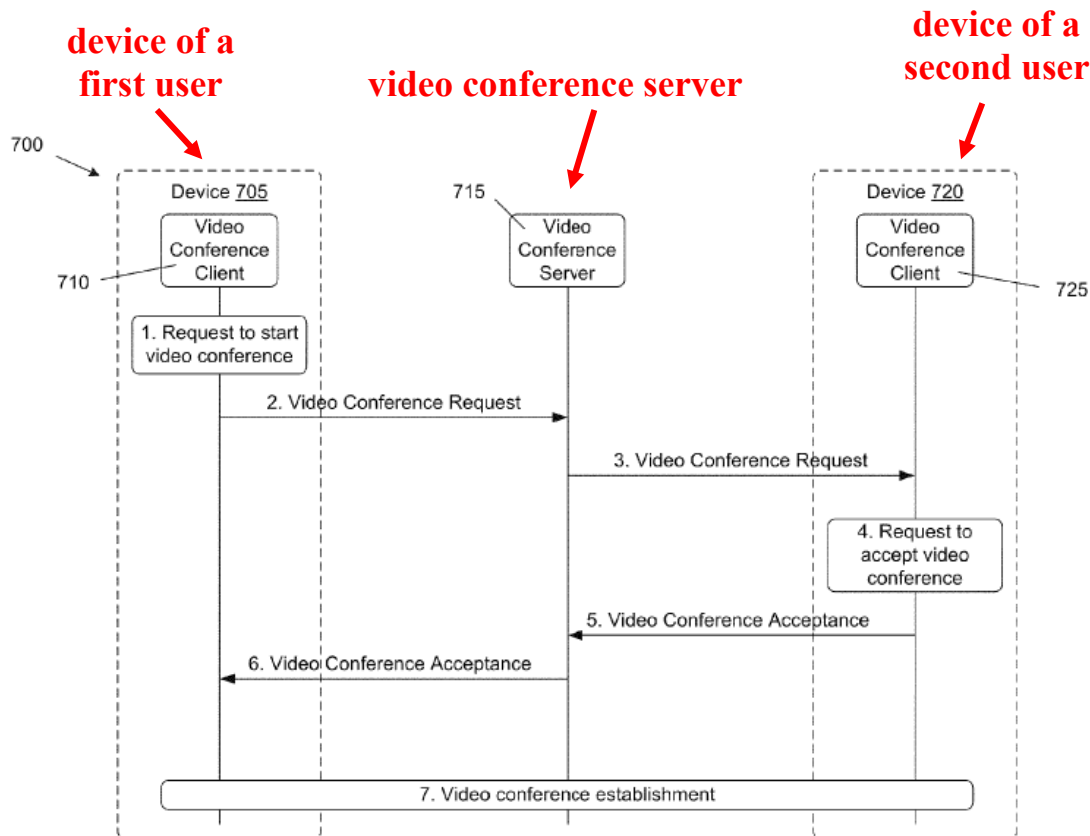


Figure 7

Ex.1005, Fig. 7 (annotated).

Abuan explains that its video conference client (*superblock application*) “sends (at operation 2) a video conference request, which indicates the device 720 as the recipient based on input from the user, to the video conference server 715.” Ex.1005, [0111]; Ex.1003, ¶228. The video conference server 715 (*one or more servers connected over the internet*) “forwards (at operation 3) the video conference

request to the video conference client 725 of the device 720.” Ex.1005, [0111]. The video conference server 715 also receives an acceptance from the second user in operation 5, and forwards the acceptance to the first user in operation 6, as reflected in Figure 7 above. Ex.1005, [0112]-[0113]; Ex.1003, ¶228.

Accordingly, Abuan discloses or renders obvious a *non-transitory computer readable medium embodying a computer program comprising instructions* (Abuan’s “memory” storing computer program instructions for a video conference client application) *providing a real-time communication session* (a real-time communication session such as a video conference) *using one or more servers connected over the internet* (e.g., video conference server 715) *to a superblock application intended for use on a computing device* (the video conference client operation on device 705 or 720). Ex.1003, ¶229.

[75.1] the superblock application adapted to interact with a function block through one or more application programming interface (API) calls in order to access the real-time communication session provided by the function block; and

Abuan renders obvious this limitation.

As discussed above in [1.1] and [2.0], Abuan renders obvious that *the superblock application [is] adapted to interact with a function block through one or more application programming interface (API) calls*. More specifically, Abuan teaches that API calls are used (*one or more application programming interface (API) calls*) to allow a client program (*superblock application*) to access services

provided by the video conference module 1302 (*function block*). Ex.1005, [0243]; Ex.1003, ¶¶230-231.

Abuan further explains that an API call is used “to initiate performance of an operation.” Ex.1005, [0243]. The operations performed by Abuan’s video conference client and functional modules include requesting, accepting, initializing, and ending a video conference. Ex.1003, ¶232. It therefore would have been obvious to a POSITA for Abuan’s client application to initiate such operations by calling one or more functions or methods defined in the API of the videoconferencing module, because Abuan proposes using API calls for video conference operations. Ex.1003, ¶¶232-233.

Thus, Abuan renders obvious this limitation.

[75.2] the function block compiled into the superblock application, wherein the function block is adapted to be compiled with the superblock application and is adapted to interact with the one or more servers connected over the internet to provide the real-time communication session within the superblock application.

Abuan renders obvious this limitation.

As discussed above in [1.2], Abuan discloses or renders obvious that *the function block [is] compiled into the superblock application*. Ex.1005, [0239], [0105]; *see also* Ex.1012, 1:26-29. As discussed at [1.0] Abuan explains that the video conference client for each respective user communicates with an intermediary server using a network connection such as the Internet (*one or more servers*

connected over the internet). Ex.1005, [0109], [0106], Fig. 7. The video conference clients are used to establish a real time communication session over the Internet (*to provide the real-time communication session within the superblock application*). See Ex.1005, [0003], [0033]-[0034]; Ex.1003, ¶¶234-235.

Further, Abuan's module subcomponents (*e.g., the function block* functionality) communicate with the video conference server 715 (*the one or more servers*) in establishing a real-time communication session such as a video conference (*to provide the real-time communication session within the superblock application*). More specifically, Abuan discloses that "[e]ach of the video conference clients 710 and 725...**communicates with the video conference server 715** over a network (*e.g., a cellular network, a local area network, a wireless network, a network of networks, the Internet etc.*) through a network interface such as the network interface 650." Ex.1005, [0109]. Abuan explains that the networking manager 1314 of the video conference module 1302 (part of *the function block*) (or network layer 640 of the video conference module 625) "establishes the connections between the dual camera mobile device and the other device of the video conference at the start of the video conference." Ex.1005, [0179]; *see also* [0104]. In this way, Abuan's video conference module 1302 (part of *the function block*) is adapted to interact with the one or more servers connected over the internet to provide the real-time communication session within the

superblock application by communicating with the video conference server over the Internet. Ex.1003, ¶236-23.

C. Ground 2

1. Eisenberg

Eisenberg describes a “system and method for initiating and supporting network video conferences[.]” Ex.1019, Abstract. Like Abuan, Eisenberg contemplates that “one or more instant messaging enabled clients can initiate a video conference.” Ex.1019, Abstract. Further, an intermediary server, “e.g., a video conferencing server, supports video conferences between video conference participants, a video conference being initiated between video conference participants in response to an instant message transmitted between the at least two client nodes.” Ex.1019, Abstract. Eisenberg recognizes that “[i]nstant messaging (IM) is becoming an increasingly popular utility on networks such as the Internet.” Ex.1019, [0003]. Eisenberg details that within such services, a “presence” feature is provided “which permits IM clients to know the status of other clients on the network serviced by the messenger service.” Ex.1019, [0003]. This allows users to communicate their availability or otherwise block incoming messages when unavailable. Ex.1019, [0003]; Ex.1003, ¶¶237-238.

Eisenberg is analogous art to the ’362 patent because it is in the same field of endeavor as the ’362 patent, namely, the “manner in which functionality is

accessed in certain environments, such as mobile device environments.” Ex.1001, 1:66-67; *see* Ex.1019, [0005]-[0006] (describing “the convenience of instant messaging with video conferencing capability to allow a user to initiate a video conference using IM” on “client nodes” or other types of “video conferencing system[s]”). Eisenberg and the ’362 patent also both generally relate to implementing functionality via software. *See* Ex.1001, Abstract (“providing additional functionality to existing software”); Ex.1005, [0027] (describing a device including “an IM client software module 16, a video conference initiation software module 18 and a video conferencing client software module 28.”), [0023] (“each IM client 12 includes a video conference module 18 which interfaces with both the IM client module 16 and the CTM server 20 to initiate a video conference”). Eisenberg and the ’362 patent both describe at length features for facilitating videoconferencing. Ex.1001, Abstract, 4:43-51 (purpose of a function block is to provide functions including “audio, video,...conferencing, meetings, and/or other functions”); Ex.1019, [0005] (“a video conference is initiated between video conference participants”), [0023] (“each IM client 12 includes a video conference module 18 which interfaces with both the IM client module 16 and the CTM server 20 to initiate a video conference”). Ex.1003, ¶239.

2. Reasons to Combine

A POSITA would have been motivated to include a presence indicator, as taught by Eisenberg, in Abuan's video conference client because doing so would have been the combination of prior art elements (Abuan's video conference client with Eisenberg's presence features) according to known methods to yield predictable results (providing information regarding the availability of a user before placing a call to them using Abuan's video conference client). Ex.1003, ¶240. Eisenberg explicitly teaches a presence feature for instant message clients, and more broadly contemplates a "system and method for initiating and supporting network video conferences." Ex.1019, Abstract. And Abuan similarly teaches an instant message client, such as its video conference client, that is able to establish video conferences over a network between two devices. Ex.1005, [0109]; Ex.1003, ¶240. A POSITA would have had a reasonable expectation of success in making such a combination because Eisenberg is directed to video conferencing, and compliments Abuan's teachings of establishing a video conference between two devices with additional information related to the users interacting with Abuan's video conference client. Ex.1003, ¶240.

3. Claim 15

As discussed above for [1.0], Abuan discloses "an application that uses the video conferencing functions of the video conference module 1302, such as a video

conferencing application, a voice-over-IP (VoIP) application (e.g., Skype), or **an instant messaging application.**” Ex.1005, [0171]. Eisenberg provides further implementation details regarding an instant messaging application, such as contemplated by Abuan. Ex.1003, ¶¶241-242. More specifically, Eisenberg explains that “[i]nstant messaging (IM) is becoming an increasingly popular utility on networks such as the Internet.” Ex.1019, [0003]. For such applications, Eisenberg notes that it is typical to include a *presence indicator* regarding whether a client is online or offline. Ex.1019, [0003] (“The messenger service also typically provides a ‘presence’ feature which permits IM clients to know the status of other clients on the network serviced by the messenger service. The status information for a client includes whether the client is online and available[.]”).

To that end, Eisenberg’s teaching of providing a presence feature is advantageous for a video conference client, such as the one contemplated by Abuan, because Eisenberg explains that it allows users to communicate their availability or otherwise block incoming messages when unavailable. Ex.1019, [0003]; Ex.1003, ¶243. And as discussed above in Section XII.C.2, a POSITA would have been motivated to combine Eisenberg’s teaching of a presence feature in Abuan’s video conference client, and would have had a reasonable expectation of success in doing so because Eisenberg specifically contemplates such features in the context of video conferencing. Ex.1003, ¶¶243-244.

Thus, Abuan in view of Eisenberg renders obvious this claim.

D. Ground 3

1. Beilis

Beilis is related to “the filed of mobile communications,” and telecommunications applications running “on a mobile computing platform.” Ex.1041, [0003]. In an example, Beilis discloses an “android device,” that “run[s] a mobile platform such as the well-known android platform,” and is running a “telecommunications video application.” Ex.1041, [0024]. Beilis further explains that an application may command the telecommunications video application “to launch a video call.” Ex.1041, [0027]; Ex.1003, ¶245.

Beilis is in the same field of endeavor as the ’362 patent, namely, the “manner in which functionality is accessed in certain environments, such as mobile device environments.” Ex.1001, 1:66-67; *see* Ex.1041, [0003]-[0005] (describing the ways mobile applications run in different computing platforms, such as Android). Beilis and the ’362 patent also both generally relate to implementing functionality via software. *See* Ex.1001, Abstract (“providing additional functionality to existing software”); Ex.1041, [0011], [0013], [0024] (describing mobile device running software and software applications, such as telecommunications video applications). Beilis and the ’362 patent both describe features for facilitating videoconferencing. Ex.1001, Abstract, 4:43-51 (purpose of

a function block is to provide functions including “audio, video,...conferencing, meetings, and/or other functions”); Ex.1041, [0027]-[0028], [0039] (describing running a mobile application to “launch,” “render,” “transfer,” and “resize” a video call); Ex.1003, ¶246.

Beilis is reasonably pertinent to a problem allegedly addressed by the ’362 patent. The ’362 patent describes purported problems associated with using multiple applications on a mobile device, potentially requiring a user to “switch[] back and forth between the video window of [a] call and [a] superblock application.” Ex.1001, 4:22-26. The ’362 patent’s proposed solution is to provide functionality—such as communication functions—to a superblock application. Ex.1001, 4:43-61. Beilis similarly describes the complexities of having multiple applications running in parallel on a mobile device, where “a user must select the application and engage in manual task performance to replace the original display with new data from the other application.” Ex.1041, [0008]. Beilis integrates the capability to perform function calls while continuously occupying the full screen so that the user does not need to switch back and forth between applications. Ex.1041, [0029]; Ex.1003, ¶247.

2. Reasons to Combine

A POSITA would have been motivated to configure Abuan’s stand-alone video conferencing application available on the Android platform, as taught by

Beilis, because doing so would have been a predictable, routine combination of prior art element using known software development techniques to yield predictable results (Abuan’s video conferencing application configured to run on the well-established Android platform). Ex.1003, ¶248. A POSITA would have appreciated that configuring Abuan’s application to run on the Android platform is an implementation detail that would not change the underlying videoconferencing functionalities disclosed by Abuan. Ex.1003, ¶248. A POSITA would have had reasonable expectation of success in the combination because Beilis shows that video communications applications were available on, and operable within, the Android platform. Ex.1003, ¶248.

3. Claim 21

As discussed above, Abuan contemplates that its video conference client is “an application that may use the video conferencing functions” such as Skype. Ex.1005, [0105], [0171]. It was well-known that video conferencing applications, including Skype itself, were also available on the Android operating system. Ex.1022. And while Abuan provides frameworks provided by Apple Inc. as examples of its modules, Abuan does not limit itself to iOS as the only operating system it contemplates. Ex.1005, [0044]; *see also* Ex.1007, [0048]; Ex.1003, ¶¶249-250.

Beilis provides additional implementation details for operating mobile applications on different devices and platforms, including “android device[s]” running the “android platform.” Ex.1041, [0016]; Ex.1003, ¶251. Beilis explains that an android device may run a “telecommunications video application” on the android platform (“*Android as an operating system*”), which application is used to “launch” and “render” a video call. Ex.1041, [0024], [0027], [0029]. Ex.1003, ¶251. Thus, Beilis’s telecommunications application is “*adapted to operate with Android as an operating system for the computing device.*”

A POSITA would have been motivated to *adapt*[/] Abuan’s video conference client (“superblock application”) to run on the “Android operating system,” as taught by Beilis, because doing so would have been a predictable, routine engineering choice aimed at making the same mobile video-conferencing functionality available to the large and well-established Android ecosystem, without changing the underlying conferencing concepts disclosed by Abuan. Ex.1003, ¶252. Beilis shows that video communications applications were available on, and operable within, the Android platform, and that launching and conducting video calls through an Android application was a known and achievable implementation detail. Ex.1003, ¶¶252-253.

XII. Conclusion

Petitioner has established a reasonable likelihood that the Challenged Claims

are unpatentable.

Respectfully submitted,

Dated: January 13, 2026
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, Texas 75201
Customer No. 27683

/Theodore M. Foster/
Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456

XIII. Mandatory notices

A. Real party-in-interest

Pursuant to 37 C.F.R. § 42.8(b)(1), Petitioner certifies that the real party-in-interest is Cisco Systems, Inc.

B. Related matters

Pursuant to 37 C.F.R. § 42.8(b)(2), to the best knowledge of the Petitioner, the '362 Patent is or was involved in the following cases:

Case Heading	Number	Court	Filed
<i>Damaka, Inc. v Cisco Systems, Inc.</i>	2-25-cv-00593	EDTX	May 30, 2025
<i>Damaka, Inc. v. The Cigna Group</i>	2:25-cv-00594	EDTX	May 30, 2025

C. Lead and back-up counsel and service information

Lead Counsel

Theodore M. Foster
HAYNES AND BOONE, LLP
2801 Harwood St. Suite 2300
Dallas, TX 75201

Phone: (303) 382-6205
Fax: (214) 200-0853
ipr.theo.foster@haynesboone.com
USPTO Reg. No. 57,456

Back-up Counsel

Calmann J. Clements
HAYNES AND BOONE, LLP
2801 Harwood St. Suite 2300
Dallas, TX 75201

Phone: (972) 739-8638
Fax: (214) 200-0853
calmann.clements.ipr@haynesboone.com
USPTO Reg. No. 66,910

Back-up Counsel

David L. McCombs
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75201

Phone: (214) 651-5533
Fax: (214) 200-0853
david.mccombs.ipr@haynesboone.com
USPTO Reg. No. 32,271

Back-up Counsel

Brett Bostrom
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75201

Phone: (720) 484-3715
Fax: (214) 200-0853
brett.bostrom.ipr@haynesboone.com
USPTO Reg. No. 73,009

Back-up Counsel

Dagim Tilahun
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75201

Phone: (202) 654-4582
Fax: (214) 200-0853
dagim.tilahun.ipr@haynesboone.com
USPTO Reg. No. 75,722

Back-up Counsel

Matthew Beck
HAYNES AND BOONE, LLP
2801 N. Harwood St., Suite 2300
Dallas, TX 75201

Phone: (202) 654-4573
Fax: (214) 200-0853
matthew.beck.ipr@haynesboone.com
USPTO Reg. No. 79,791

Please address all correspondence in this proceeding to lead and back-up counsel. Petitioner consents to service in this proceeding by email at the addresses above on condition that Patent Owner likewise consents to email service.

XIV. Claims Appendix

- [1.0] A method for enabling an end to end connection for a first user and a second user to participate in a real-time communication session over the internet, the method comprising:
- [1.1] creating for storage on a server, a superblock application configured for use by a computing device of the first user and a superblock application configured for use by a computing device of the second user,
- [1.2] wherein each of the superblock applications has a function block compiled therein, wherein each respective superblock application communicates with its respective function block using application programming interface (API) calls;
- [1.3] receiving, by the superblock application on the computing device of the first user, a request to initiate a real-time communication session with the second user;
- [1.4] communicating, by the superblock application on the computing device of the first user, the request to a remote resource;
- [1.5] providing, by the superblock application on the computing device of the second user, a notification that the real-time communication session with the first user is being requested;
- [1.6] identifying, by the superblock application on the computing device of the

second user, rendering information; and

[1.7] rendering, by the superblock application on the computing device of the second user, the real-time communication session with the first user using the rendering information.

[2.0] The method of claim 1 wherein communicating, by the superblock application on the computing device of the first user, the request to the remote resource includes communicating, by the superblock application on the computing device of the first user, the request to the function block on the computing device of the first user using one or more application programming interface (API) calls;

[2.1] wherein providing, by the superblock application on the computing device of the second user, the notification that the real-time communication session with the first user is being requested includes providing the notification by the function block on the computing device of the second user; and

[2.2] wherein rendering, by the superblock application on the computing device of the second user, the real-time communication session with the first user using the rendering information includes: providing, by the superblock application on the computing device of the second user, the rendering information to the function block on the computing device of the second user; and

- [2.3] rendering, by the function block on the computing device of the second user, the real-time communication session with the first user using the rendering information.
- [3.0] The method of claim 1 further comprising initializing, by the superblock application on at least one of the computing devices of the first and second users, the respective function block following initialization of the superblock application on the computing device.
- [4.0] The method of claim 1 further comprising configuring, by the superblock application on at least one of the computing devices of the first and second users, at least one configurable parameter of the respective function block.
- [5.0] The method of claim 1 wherein providing the notification includes displaying an icon to enable the second user to accept the request.
- [6.0] The method of claim 1 wherein providing the rendering information includes providing information corresponding to a mic of the computing device of the second user.
- [7.0] The method of claim 1 wherein providing the rendering information includes providing information corresponding to a speaker of the computing device of the second user.
- [8.0] The method of claim 1 wherein providing the rendering information includes providing information for displaying a window within the superblock

application on the computing device of the second user.

[9.0] The method of claim 8 wherein the window is at least one of a video window and an instant message window.

[10.0] The method of claim 8 wherein the window is resizable by the second user.

[11.0] The method of claim 10 further comprising sending, by the superblock application on the computing device of the second user to the function block on the computing device of the second user, an update for the window, wherein the update indicates that the window has been resized.

[12.0] The method of claim 8 wherein the window is movable by the second user.

[13.0] The method of claim 12 further comprising sending, by the superblock application on the computing device of the second user to the function block on the computing device of the second user, an update for the window, wherein the update indicates that the window has been moved.

[14.0] The method of claim 1 wherein providing the rendering information includes providing information corresponding to a camera of the computing device of the second user.

[15.0] The method of claim 1 wherein the superblock application on the computing device of the first user is adapted to update a presence indicator indicating a status of the second user, wherein the status includes at least an offline status and an online status.

[16.0] The method of claim 1 wherein the API calls used by each superblock

application are compatible with an interface provided by its respective function block.

[17.0] The method of claim 1 wherein the real-time communication session is an instant messaging service.

[18.0] The method of claim 1 wherein the real-time communication session is a voice call.

[19.0] The method of claim 1 wherein the real-time communication session is a video call.

[20.0] The method of claim 1 wherein the superblock application on at least one of the computing devices of the first and second users is further adapted to operate with iOS as an operating system for the computing device.

[21.0] The method of claim 1 wherein the superblock application on at least one of the computing devices of the first and second users is further adapted to operate with Android as an operating system for the computing device.

[22.0] The method of claim 1 wherein the computing device of at least one of the first and second users is a mobile device.

[23.0] The method of claim 22 wherein the computing device of at least one of the first and second users is a cellular phone.

[24.0] The method of claim 22 wherein the computing device of at least one of the

first and second users is a smart phone.

[25.0] The method of claim 22 wherein the computing device of at least one of the first and second users is a tablet.

[26.0] The method of claim 22 wherein the computing device of at least one of the first and second users is a personal digital assistant.

[27.0] The method of claim 1 wherein the computing device of at least one of the first and second users is a laptop computer.

[28.0] The method of claim 1 wherein the computing device of at least one of the first and second users is a desktop computer.

[75.0] A non-transitory computer readable medium embodying a computer program comprising instructions for providing a real-time communication session using one or more servers connected over the internet to a superblock application intended for use on a computing device, the instructions comprising:

[75.1] the superblock application adapted to interact with a function block through one or more application programming interface (API) calls in order to access the real-time communication session provided by the function block; and

[75.2] the function block compiled into the superblock application, wherein the function block is adapted to be compiled with the superblock application

and is adapted to interact with the one or more servers connected over the internet to provide the real-time communication session within the superbloc application.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), Petitioner hereby certifies, in accordance with and reliance on the word count provided by the word-processing system used to prepare this Petition, that the number of words in this paper is 13,978. Pursuant to 37 C.F.R. § 42.24(d), this word count excludes the table of contents, table of authorities, mandatory notices under § 42.8, certificate of service, certificate of word count, appendix of exhibits, and any claim listing.

Dated: January 13, 2026

/Theodore M. Foster/

Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. § 42.6(e), the undersigned certifies that a true and correct copy of the foregoing was served on the Patent Owner as detailed below:

<i>Date of service</i>	January 13, 2026
<i>Manner of service</i>	FedEx
<i>Documents served</i>	PETITION FOR <i>INTER PARTES</i> REVIEW UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104; PETITIONER’S POWER OF ATTORNEY; PETITIONER’S EXHIBIT LIST; AND EXHIBITS Ex.1001-1042
<i>Persons Served</i>	Munck Wilson Mandala L.L.P 2000 McKinney Ave Ste 1900 Dallas, TX 75201

/Theodore M. Foster/

Theodore M. Foster
Lead Counsel for Petitioner
Registration No. 57,456