

Paper No. ___

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner,

v.

VIRTAMOVE, CORP.,
Patent Owner.

Case No. IPR2025-00490
Patent No. 7,784,058

**PETITION FOR INTER PARTES REVIEW
UNDER 35 U.S.C. §§ 311-319 AND 37 C.F.R. § 42.1 et seq**

TABLE OF CONTENTS

MANDATORY NOTICES..... xi

 A. Real Party-In-Interest xi

 B. Related Matters..... xi

 1. United States Patent & Trademark Office xi

 2. USPTO Patent Trial and Appeal Board xii

 3. U.S. District Court for the Eastern District of Texas xii

 4. U.S. District Court for the Western District of Texas..... xii

 5. U.S. District Court for the Northern District of California..... xiii

 C. Counsel and Service Information - § 42.8(b)(3) and (4)..... xiii

I. STANDING..... 1

II. GROUNDS FOR UNPATENTABILITY 1

III. THE '058 PATENT 2

 A. Background and Specification..... 2

 B. Person of Ordinary Skill in the Art (“POSA”)..... 7

 C. Prosecution History 8

IV. CLAIM INTERPRETATION 9

V. GROUND 1: Elnozahy+Draves Renders CLAIMS 1-18 Obvious 10

 A. Elnozahy (EX1006) 10

 B. Draves (EX1017)..... 12

 C. Elnozahy+Draves Combination 13

 D. Claim-by-Claim Analysis..... 15

 1. Claim 1 15

 a. [1PRE]: “A computing system for executing a plurality of software applications comprising:” 15

 b. [1A] “a) a processor;” 16

 c. [1B] 17

 i. [1B.1] “b) an operating system having an operating system kernel...” 17

ii.	[1B.2] “[OS kernel] having OS critical system elements (OSCSEs)...”	18
iii.	[1B.3] “[OSCSEs] for running in kernel mode using said processor”	21
d.	[1C]	24
i.	[1C.1] “c) a shared library having shared library critical system elements (SLCSEs) stored therein...”	24
ii.	[1C.2] “for use by the plurality of software applications in user mode...”	28
e.	[1D]	29
i.	[1D.1] “i) wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and...”	29
ii.	[1D.2] “are accessible to some of the plurality of software applications and...”	32
iii.	[1D.3] “when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications”	34
f.	[1E]	35
i.	[1E.1] “ii) wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and...”	35
ii.	[1E.2] “where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function, and...”	38
g.	[1F] “iii) wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is	

provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.”	39
2. Claim 2	41
3. Claim 3	43
4. Claim 4	44
a. [4A]	44
b. [4B]	44
i. File retrieval	45
ii. Initializing interrupt handling	47
5. Claim 5	50
a. Initializing Communication with Network-Interface Drivers	51
b. File retrieval	53
c. Interrupt setup	55
6. Claim 6	56
a. [6A]	56
i. File retrieval	56
ii. Packet arrival	57
b. [6B]	57
i. File retrieval	57
ii. Packet Arrival	58
7. Claim 7	58
8. Claim 8	59
9. Claim 9	60
10. Claim 10	61
11. Claim 11	62
a. Device access	62
b. Interrupt Delivery	63
c. Virtual memory mapping	63

12. Claim 12	64
13. Claim 13	65
14. Claim 14	65
15. Claim 15	66
16. Claim 16	68
17. Claim 17	68
18. Claim 18	69
VI. DISCRETIONARY DENIAL IS UNWARRANTED	70
A. §314(a).....	70
1. <i>Fintiv</i>	70
a. Stay Potential	70
b. Trial Timing	70
c. Litigation Investment	71
d. Issue Overlap.....	71
e. Litigation Defendants.....	71
f. Other Circumstances	71
2. Unified Re-exam	71
3. Amazon IPR	72
B. §325(d).....	73
1. Part One	73
2. Part Two	74
VII. CONCLUSION.....	74
VIII. CLAIM LISTING	78

TABLE OF AUTHORITIES

CASES

<i>Advanced Bionics, LLC v. Med-El Elektromedizinische Gerate GMBH,</i> IPR2019-01469, Paper 6 (Feb. 13, 2020).....	78, 79
<i>Apple Inc. v. Fintiv, Inc.,</i> IPR2020-00019, Paper 11 (Mar. 20, 2020).....	76
<i>Becton, Dickson & Co. v. B. Braun Melsungen AG,</i> Case IPR2017-01586, Paper 8 (December 15, 2017)	79
<i>BMW of North America, LLC v. Michigan Motor Techs., LLC,</i> IPR2023-01224, Paper 15 (Feb. 15, 2024).....	77
<i>Cellco P’ship v. Huawei Device Co.,</i> IPR2020-01117, Paper 10 (Feb. 3, 2021).....	79
<i>Google LLC v. Security First Innovations, LLC,</i> IPR2024-00215, Paper 15 (May 23, 2024)	9
<i>KSR Int’l v. Teleflex,</i> 550 U.S. 398 (2007)	21
<i>Markforged Inc. v. Continuous Composites,</i> IPR2022-00679, Paper 7 (Oct. 25, 2022).....	77
<i>PLR Worldwide Sales Ltd. v. Flip Phone Games, Inc.,</i> IPR2024-00209, Paper 9 (May 10, 2024)	20, 30
<i>Protect Animals With Satellites LLC v. OnPoint Sys., LLC,</i> IPR2021-01483, Paper 11 (Mar. 4, 2022).....	77
<i>Samsung Elecs. Co., Ltd. v. DoDots Licensing Soln’s. LLC,</i> IPR2023-00756, Paper 9, 21-22 (Oct. 11, 2023)	79
<i>Sand Revolution II, LLC, v. Cont’l Intermodal Group–Trucking,</i> IPR2019-01393, Paper 24 (June 16, 2020)	76
<i>Tesla, Inc. v. Graphite Charging Company LLC,</i> IPR2024-00388, Paper No. 15 (Aug. 27, 2024).....	78

Thorne Research v. Trs. of Dartmouth Coll.,
IPR2021-00491, Paper 18 (Aug. 12, 2021).....79

REGULATIONS

37 C.F.R. §42.100(b)9
37 C.F.R. §42.104(a).....1
37 C.F.R. §42.104(b)(3).....9

STATUTES

35 U.S.C. §102(b)2
35 U.S.C. §102(e)2
35 U.S.C. §282(b)9
35 U.S.C. §314(a) 76, 78
35 U.S.C. §315(d)78
35 U.S.C. §325(d)78

OTHER AUTHORITIES

Director’s Interim Procedure for Discretionary Denials (June 21, 2022) 76, 77

APPENDIX LISTING OF EXHIBITS

Exhibit	Description
1001	U.S. Patent No. 7,784,058
1002	Prosecution History of U.S. Patent No. 7,784,058
1003	Declaration of Samrat Bhattacharjee, Ph.D. (“Bhattacharjee”)
1004	Curriculum Vitae of Dr. Samrat Bhattacharjee
1005	[RESERVED]
1006	U.S. Patent App. Pub. No. 2003/0041118 (“Elnozahy”)
1007	U.S. Patent No. 6,263,376 (“Hatch”)
1008	U.S. Patent No. 6,260,075 (“Cabrerero”)
1009	U.S. Patent No. 6,212,574 (“O’Rourke”)
1010	U.S. Patent No. 5,481,706 (“Peek”)
1011	U.S. Patent No. 7,499,966 (“Elnozahy-966”)
1012	U.S. Patent App. Pub. No. 2004/0216145 (“Wong”)
1013	RESERVED
1014	RESERVED
1015	Excerpts from Charles Petzold, <i>Programming Windows 95</i> (Microsoft Press 1996) (“Petzold”)
1016	RESERVED
1017	U.S. Patent No. 6,349,355 (“Draves”)
1018	Excerpts from Silberschatz et. al, <i>Operating System Concepts</i> (Wiley 6 th ed. 2002) (“Silberschatz”)
1019	RESERVED
1020	Chart re: ’058 Patent accompanying Plaintiff VirtaMove Corp.’s Supplemental Preliminary Disclosure of Asserted Claims and Infringement Contentions, in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Sep. 26, 2024)
1021	U.S. Patent App. Pub. No. 2002/0095224 (“Braun”)
1022	U.S. Patent No. 6,173,336 (“Stoeckl”)
1023	RESERVED
1024	U.S. Patent No. 7,080,172 (“Schmalz”)
1025-1031	RESERVED
1032	U.S. Patent No. 5,375,241 (“Walsh”)
1033	U.S. Patent No. 6,698,015 (“Moberg”)
1034	Excerpts from Collin, <i>Dictionary of Computing</i> (Collin 4 th Ed. 2002) (“Collin”)
1035	Microsoft Computer Dictionary (Microsoft 5 th ed. 2002) (“Microsoft”)
1036	RESERVED

1037	U.S. Patent App. Pub. No. 2003/0154320 (“Calusinski”)
1038	U.S. Patent No. 7,437,483 (“Goossen”)
1039	U.S. Patent No. 7,100,162 (“Green-162”)
1040-1042	RESERVED
1043	U.S. Patent No. 6,792,492 (“Griffin”)
1044	U.S. Patent App. Pub. No. 2002/0116563 (“Lever”)
1045	U.S. Patent No. 6,594,698 (“Chow”)
1046	U.S. Patent No. 7,216,164 (“Whitmore”)
1047	U.S. Patent No. 6,988,271 (“Hunt”)
1048	Liss et. al, <i>Efficient Exploitation of Kernel Access to Infiniband: a Software DSM Example</i> , 11th Symposium on High Performance Interconnects, 2003 (“Liss”)
1049	Patel et. al., <i>A Model of Completion Queue Mechanisms Using the Virtual Interface API</i> , Proc. IEEE Int’l Conf. on Cluster Computing, 281-282 (IEEE 2002) (“Patel”)
1050	Scheduling Order in VirtaMove, Corp. v. Google LLC, 7:24-cv-00033-DC-DTG (W.D. Tex. June 17, 2024) (ECF 34)
1051	Federal Court Management Statistics–Profiles, U.S. District Courts–Combined Civil and Criminal (June 30, 2024)
1052	Google LLC’s Proposed Claim Terms for Construction, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 1, 2024)
1053	Plaintiff’s Disclosure of Proposed Claim Constructions, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 1, 2024)
1054	RESERVED
1055	RESERVED
1056	U.S. Patent No. 5,278,969 (“Pashan”)
1057	Excerpts of <i>Webster’s New World Dictionary</i> (4 th Ed. 2003)
1058	Joint Claim Construction Statement, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Dec. 18, 2024)
1059	Google’s Opening Claim Construction Brief, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Oct. 22, 2024)
1060	Request for <i>Ex Parte</i> Reexamination of U.S. Patent No. 7,784,058 (“Unified Reexam”)
1061-1064	RESERVED
1065	U.S. Patent No. 6,442,752 (“Jennings”)
1066	RESERVED

1067	VirtaMove’s Responsive Claim Construction Brief, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Nov. 12, 2024)
1068	VirtaMove’s Surreply Claim Construction Brief, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Dec. 13, 2024)
1069	U.S. Patent No. 7,213,247 (“Wilner”)
1070	RESERVED
1071	RESERVED
1072	U.S. Patent App. Pub. No. 2003/0037337 (“Yona”)
1073	U.S. Patent App. Pub. No. 2002/0083127 (“Agrawal”)
1074	U.S. Patent App. Pub. No. 2002/0004834 (“Guenther”)
1075	U.S. Patent No. 6,049,892 (“Casagrande”)
1076	U.S. Patent App. Pub. No. 2003/0023580 (“Braud”)
1077	U.S. Patent No. 6,917,627 (“Footer”)
1078	U.S. Patent No. 6,361,335 (“Calanni”)
1079	U.S. Patent App. Pub. No. 2002/0078135 (“Venkatsubra”)
1080	U.S. Patent No. 6,594,690 (“Cantwell”)
1081	U.S. Patent No. 5,394,547 (“Correnti”)
1082	U.S. Patent No. 5,931,925 (“McNabb”)
1083	U.S. Patent No. 5,966,543 (“Hartner”)
1084	U.S. Patent No. 7,171,494 (“Karamanolis”)
1085	U.S. Patent No. 6,029,160 (“Cabrera”)
1086	U.S. Patent No. 6,931,501 (“Narayanaswamy”)
1087	U.S. Patent No. 5,630,141 (“Ross”)
1088	U.S. Patent App. Pub. No. 2003/0007488 (“Rao”)
1089	Order Granting Defendant Google LLC’s Motion to Transfer Venue, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Jan. 22, 2025)
1090	Order Cancelling Markman Hearing, <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Jan. 17, 2025)
1091	U.S. Patent No. 7,937,559 (“Parameswar”)
1092	U.S. Patent No. 7,200,761 (“Freeman”)
1093	U.S. Patent App. Pub. No. 2002/0027909 (“Brinkerhoff”)
1094	U.S. Provisional Patent Application No. 60/504,213
1095	Redline Comparison between U.S. Provisional Patent Application No. 60/504,213 and specification of U.S. Patent No. 7,784,058
1096	U.S. Patent No. 6,728,839 (“Marshall”)
1097	RESERVED
1098	U.S. Patent No. 5,815,701 (“Slavenburg”)
1099	U.S. Patent No. 5,581,768 (“Garney”)

1100	European Patent Application EP1164480 (“Temple”)
1101	U.S. Patent Application Pub. No. 2002/0129085 (“Kubala”)
1102	U.S. Patent No. 6,874,144 (“Kush”)
1103	U.S. Patent No. 5,630,076 (“Saulpaugh”)
1104	U.S. Patent No. 5,983,021 (“Mitrovic”)
1105	U.S. Patent No. 5,305,461 (“Feigenbaum”)
1106	U.S. Patent App. Pub. No. 2002/0032821 (“Garrigues”)
1107	Plaintiff VirtaMove Corp.’s Supplemental Preliminary Disclosure of Asserted Claims and Infringement Contentions, in <i>VirtaMove, Corp. v. Google LLC</i> , 7:24-cv-00033-DC-DTG (W.D. Tex.) (Sept. 06, 2024)

MANDATORY NOTICES

A. Real Party-In-Interest

Petitioner is the Real Party-in-Interest.¹

B. Related Matters

1. United States Patent & Trademark Office

The application from which U.S. Patent No. 7,784,058 issued claims priority to provisional application No. 60/504,213, filed September 22, 2003.

The following U.S. patent applications claim the benefit of priority to U.S. Patent 7,784,058:

(i) U.S. Patent Application 11/432,843 (U.S. Patent No. 7,757,291), filed May 12, 2006; (ii) U.S. Patent Application 11/380,285 (U.S. Patent No. 7,774,762), filed April 26, 2006; (iii) U.S. Patent Application 12/075,842 filed March 13, 2008.

U.S. Patent No. 7,784,058 is the subject of *Ex Parte* Reexamination No. 90/019,676, requested by Unified Patents, LLC, filed on September 23, 2024.

¹ Google LLC is a subsidiary of XXVI Holdings Inc., which is a subsidiary of Alphabet Inc. XXVI Holdings Inc. and Alphabet Inc. are not real parties-in-interest to this proceeding.

2. USPTO Patent Trial and Appeal Board

Concurrently with the present petition, Petitioner is filing IPR2025-00489, also challenging U.S. Patent No. 7,784,058.

Petitioner is also filing IPR2025-00487 and IPR2025-00488 challenging U.S. Patent No. 7,519,814, which is also asserted in *VirtaMove, Corp. v. Google LLC*, Case No. 7:24-cv-00033, listed below.

U.S. Patent No. 7,784,058 is the subject of *Amazon.com, Inc. v. VirtaMove, Corp.*, IPR2025-00561, filed on January 30, 2025, by different petitioner Amazon.com, Inc.

3. U.S. District Court for the Eastern District of Texas

(i) *VirtaMove, Corp. v. Hewlett Packard Enterprise Company*, Case No. 2:24-cv-00093;

(ii) *VirtaMove, Corp. v. International Business Machines Corporation*, Case No. 2:24-cv-00064.

4. U.S. District Court for the Western District of Texas

(i) *VirtaMove, Corp. v. Google LLC*, Case No. 7:24-cv-00033 (pending transfer to Northern District of California per Order dated January 22, 2025, *see* EX1089);

(ii) *VirtaMove, Corp. v. Amazon.com, Inc. et al*, Case No. 7:24-cv-00030 (pending transfer to Northern District of California per Order dated January 22, 2025, *see* Docket Entry No. 87);

(iii) *VirtaMove, Corp. v. Microsoft Corp.*, Case No. 7:24-cv-00338;

(iv) *VirtaMove, Corp. v. Oracle Corp.*, Case No. 7:24-cv-00339.

5. U.S. District Court for the Northern District of California

(i) *Red Hat, Inc. v. VirtaMove, Corp.*, No. 5:23-cv-04740.

C. Counsel and Service Information - § 42.8(b)(3) and (4)

Lead Counsel	Elisabeth H. Hunt, Reg. No. 67,336
Backup Counsel	Anant K. Saraswat, Reg. No. 76,050 Gregory S. Nieberg, Reg. No. 57,063
Service Information	<u>E-mail</u> : EHunt-PTAB@wolfgreenfield.com ASaraswat-PTAB@wolfgreenfield.com GNieberg-PTAB@wolfgreenfield.com <u>Post and hand delivery</u> : Wolf, Greenfield & Sacks, P.C. 600 Atlantic Avenue Boston, MA 02210-2206 <u>Telephone</u> : 617-646-8000 <u>Facsimile</u> : 617-646-8646

A power of attorney is submitted with the Petition. Counsel for Petitioner consents to service of all documents via electronic mail.

Petitioner requests *inter partes* review and cancellation of claims 1-18 of U.S. Patent No. 7,784,058 (“the ’058 patent,” EX1001).

I. STANDING

Petitioner certifies that the ’058 patent is available for *inter partes* review and that Petitioner is not barred or estopped from requesting *inter partes* review of the challenged claims. 37 C.F.R. §42.104(a).

II. GROUNDS FOR UNPATENTABILITY

Claims 1-18 are unpatentable under Elnozahy (EX1006) in view of Draves (EX1017).

The ’058 patent was filed September 21, 2004, and claims priority to Provisional Application No. 60/504,213 (“’213 Provisional,” EX1094), filed September 22, 2003. However, the ’058 patent is not entitled to the ’213 Provisional’s filing date because the ’213 Provisional does not provide written description support for the ’058 patent’s claims. As just one example, written description support for Element [1E.1], which recites that an SLCSE “*instance*” is provided to applications without being “*shared*” (*see* claim listing *infra* §VIII), is not in the ’213 Provisional, but was added to the specification of the ’058 patent. *See* EX1095 (automatically-generated redline comparison of ’213 Provisional and ’058 patent specification), 3-4 (showing addition of description of how “CSEs are not shared among applications”); EX1003 (“Bhattacharjee”), ¶¶31-35. Thus,

Elnozahy is prior art under pre-AIA 35 U.S.C. §102(b) based on Elnozahy's February 27, 2003 publication date. Furthermore, if the '058 patent were entitled to the September 22, 2003 priority date, Elnozahy is prior art under (at least) pre-AIA §102(e) based on Enozahy's August 23, 2001 filing date.

Draves issued February 19, 2002 and is prior art under (at least) pre-AIA §102(b).

Elnozahy's issued patent, Elnozahy-966 (EX1011), was of record during prosecution but was not considered in combination with Draves. Bhattacharjee, ¶¶36-37.

III. THE '058 PATENT

A. Background and Specification

A typical computer system may run multiple “application[s]” (or “application programs”), *e.g.*, email or word-processing, that must make shared use of the computer system's resources like hardware or files. EX1001, 1:21-31; EX1035, 378; Bhattacharjee, ¶38. The computer's “operating system” (“OS”) “traditionally...provides mechanisms to...control [the applications'] access to shared resources.” EX1001, 1:22-24. Additionally, the '058 patent states that the OS “‘normally’ supplie[s]” “service[s] or part[s] of a service” that are “critical to the operation of a software application”; the patent calls such services, or parts thereof, “critical system elements” (“CSEs”). EX1001, 2:1, 6:6-9. Examples of

CSEs include services such as networking, access to files, “memory allocation” and “device access.” EX1001, 5:38-45, 6:12-28, 9:33-35; Bhattacharjee, ¶39.

CSEs are “normally” found in the OS’s “kernel.” EX1001, 5:21-23. The “kernel” is the “core” of an OS and performs tasks like “manag[ing] memory, files, and peripheral devices” (*e.g.*, disks, network interfaces). EX1035, 300, 398; Bhattacharjee, ¶40. Conventional computer processors typically run in a special “kernel mode” when executing the kernel; running in this mode gives the kernel unrestricted access to the computer’s resources, which enables the kernel to perform its tasks. EX1001, 6:32-36; Draves, 1:23-47; Bhattacharjee, ¶41. In contrast, when running applications, processors typically run in “user mode,” where access to certain resources is restricted to prevent applications from interfering with each other or damaging the system—application code cannot run in kernel mode. EX1001, 6:33-36; Draves, 1:23-47; Bhattacharjee, ¶42. However, applications often need to use CSEs that are typically provided by the kernel; to access kernel services, applications typically make “system calls” to request that the OS kernel perform a needed service (like a CSE) for the application, using the processor executing in kernel mode. EX1001, 5:38-58, 6:66-7:16; Bhattacharjee, ¶¶43-44.

The above-described system-call-based technique has known disadvantages, including that (i) switching processor modes takes time and thus may impact

performance; and (ii) all applications must use the same version of a CSE, *i.e.*, whatever version the computer system's OS kernel provides. Bhattacharjee, ¶45; EX1098, 1:29-32; EX1001, 6:31-36, 5:54-6:3. Thus, it was known to implement some CSEs in user mode instead. EX1001, 7:23-25 ('058 patent admitting this was known); Bhattacharjee, ¶46. In some known user-mode CSE implementations, the CSE was provided by a "process[]" that executed in user mode but was separate from any application that used the CSE. EX1001, 7:23-30. In other words, when a user-mode application needed a CSE, it would request that a different **user-mode** process, rather than the kernel, perform the CSE. Bhattacharjee, ¶47. Different user-mode processes provided different user-mode CSEs, or a single user-mode process provided all user-mode CSEs. EX1001, 7:23-61; Figs. 2a-2b; Bhattacharjee, ¶48. As illustrated in Fig. 2a (below), however, communication between a user-mode application needing a CSE and a user-mode process providing that CSE still passed through the kernel—that is, the application issued a request to the kernel, which then invoked the user-mode CSE process. EX1001, 7:31-52; Bhattacharjee, ¶48.

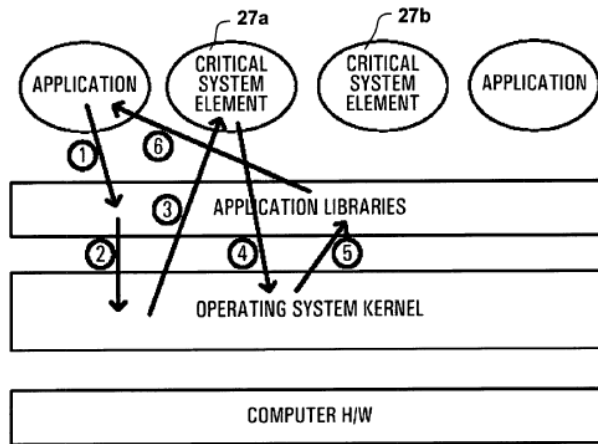


FIG. 2a
Prior Art

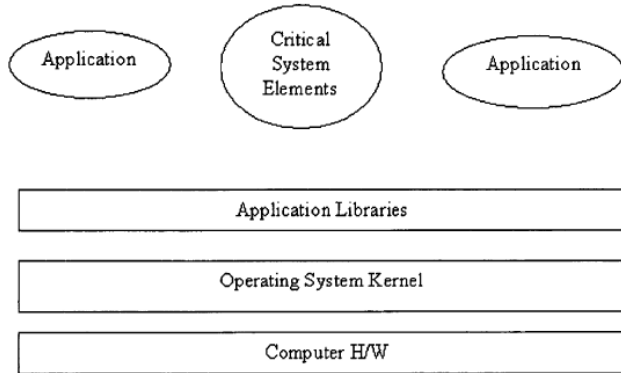


FIG. 2b
Prior Art

In contrast to the above-described scheme requiring communication with a separate user-mode CSE process via the kernel, the '058 patent discloses “replicat[ing] [CSEs] in user mode” by “placing CSEs similar to those in the OS in shared libraries.” EX1001, 5:22-34; Bhattacharjee, ¶49. “The CSE library includes replicas or substantial functional equivalents or replacements of kernel functions.” EX1001, 8:27-28. POSAs understood that a “function” is a predefined

set of instructions that carry out a specific action (e.g., a CSE). Bhattacharjee, ¶50; EX1007, 1:22-33. Placing a CSE in a shared library “provides a means of attaching or linking a CSE service to an application having access to the shared library.” EX1001, 5:29-31, 9:15-27; Bhattacharjee, ¶51. The “functions” in the shared CSE library “can be **directly** called by the applications...and as such can be **run in the same context** as the applications.” EX1001, 8:31-33;² *see also id.*, 9:41-42 (“FIG. 4 shows that the invention allows for [CSEs] to exist in the same context as an application.”), FIG. 4. In other words, the user-mode application itself performs the CSE (by calling a function from the library), rather than asking another user-mode process to perform the CSE. Bhattacharjee, ¶52.

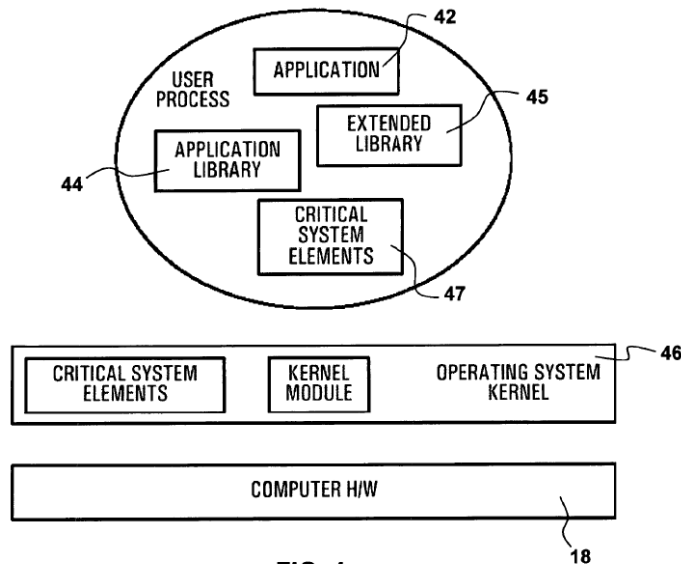


FIG. 4

² Emphases added throughout unless otherwise indicated.

The patent refers to user-mode CSEs in a shared library as “SLCSEs,” and to CSEs in the kernel as “OS [operating system] critical system elements (OSCSEs).” EX1001, 2:7-12. The patent mentions the well-known “dynamic linked library (DLL)” as an example of a “shared library.” EX1001, 2:45-51; Bhattacharjee, ¶53.

The patent contends that the ability to “execute” CSEs “in the same context as an application” “contrast[s]” with prior-art user-mode CSEs that were allegedly only implemented as a “shared service.” EX1001, 1:46-54. However, the patent admits that “sharing [] code” between applications via a shared library in user mode was “**common practice.**” EX1001, 3:30-33; *see also* EX1001, 7:3-5. The patent also admits that it was “typical” to provide “functions” in “libraries which applications link with.” EX1001, 6:37-45. Nowhere does the patent explain what is purportedly novel about using linking and shared libraries to make CSEs available to applications. Bhattacharjee, ¶¶54-55. In fact, as demonstrated below, Elnozahy and Draves in combination disclosed doing just that, well before the ‘058 patent’s filing.

B. Person of Ordinary Skill in the Art (“POSA”)

A POSA as of the ‘058 patent’s September 22, 2003 earliest claimed priority date would have had at least a bachelor’s degree in computer science, computer engineering, or a related field, with three years of academic and/or industry experience in the areas of “computing system[s]” and “application libraries.”

EX1001, 1:15-17. More education may substitute for less experience.

Bhattacharjee, ¶¶56-58.

C. Prosecution History

As detailed where relevant *infra* § V.D, the examiner rejected the originally-filed claims over Cabrero (EX1008), after which the applicants amended claim 1 to require SLCSEs to be “*functional*” replicas of OSCSEs. EX1002, 216-222, 235, 265-266, 273, 278; Bhattacharjee, ¶¶59-60. The examiner then rejected the claims over O’Rourke (EX1009) in view of Peek (EX1010). EX1002, 296-297. The applicants traversed the rejection, as also detailed *infra* § V.D. EX1002, 324-325; Bhattacharjee, ¶61. Subsequently, for reasons not recorded, the examiner discussed two different references, Elnozahy-966 (EX1011) and Wong (EX1012), in an examiner interview, and then allowed the claims without further discussing O’Rourke or Peek. EX1002, 348-351; Bhattacharjee, ¶62. The allowed claims contain an examiner’s amendment to Element [1E] (*see* claim listing *infra* §VIII) to recite “*at least a first*” and “*at least a second*” of the plurality of software applications. EX1002, 350-351. The examiner also incorporated into claim 1 originally-filed dependent claim 6, which is essentially limitation [1F]’s “wherein” clause. EX1002, 350-352; *infra* §VIII; Bhattacharjee, ¶63.

The examiner stated that neither Elnozahy-966 nor Wong disclosed the amended language of claim 1. EX1002, 352-353. But neither of those references

explicitly discloses shared libraries or an SLCSE in a shared library, and there is no indication that the examiner considered whether these features would have been obvious. Bhattacharjee, ¶¶64-68. There is also no evidence the examiner considered combining Elnozahy-966 with a reference like Draves (which discloses shared libraries, *see infra* §V.B) instead of Wong.

IV. CLAIM INTERPRETATION

Claim terms are construed herein using the standard used in civil actions under 35 U.S.C. §282(b), in accordance with the ordinary and customary meaning as understood by POSAs and the patent’s prosecution history. 37 C.F.R. §42.100(b). In the concurrent district-court proceeding, the parties (Petitioner “Google” and Patent Owner “VirtaMove”) have proposed constructions of various claim terms, including competing constructions for some terms and agreed constructions for others. EX1052-EX1053. As discussed further within the Ground, this Petition presents alternative mappings of the prior art under both parties’ proposed constructions of disputed terms, thus demonstrating unpatentability regardless of which proposed construction is correct. *Google LLC v. Security First Innovations, LLC*, IPR2024-00215, Paper 15, 6 (May 23, 2024) (finding this approach “complies with” Rule 42.104(b)(3)).

V. GROUND 1: ELNOZAHY+DRAVES RENDERS CLAIMS 1-18 OBVIOUS

A. Elnozahy (EX1006)

Elnozahy is directed to a “web server that integrates portions of operating system code to execute substantially within user space.” Elnozahy, [0008]. Except in Elnozahy’s claim 9 (discussed *infra* §V.D.1.b), POSAs understood that Elnozahy uses “web server” to refer to server software, as opposed to the computer the software runs on, which Elnozahy calls the “server device.” Elnozahy, [0004] (“Typically, web-based services are implemented by installing specialized software, referred to herein as a web server, on a server device”), [0015], [0017]; Bhattacharjee, ¶69. Elnozahy explains that conventionally, when the web-server software receives a “request” (*e.g.*, for a web page) from a “client” (*e.g.*, an end-user computer), the request was processed by a “Transmission Control Protocol/Internet Protocol (‘TCIP/IP’) block”; TCP/IP was (and is) a well-known networking protocol. Elnozahy, [0005]; Bhattacharjee, ¶70. This TCP/IP block was conventionally invoked by the “operating system kernel”; this block “extracts an HTTP formatted request” from one or more “packets” in the request and then passes the HTTP request to the server. Elnozahy, [0005], Fig. 1; Bhattacharjee, ¶71. HTTP, or “Hypertext Transfer Protocol,” was (and is) a well-known protocol for formatting requests from a browser to a web server and for formatting the server’s response. EX1035, 259; Bhattacharjee, ¶72.

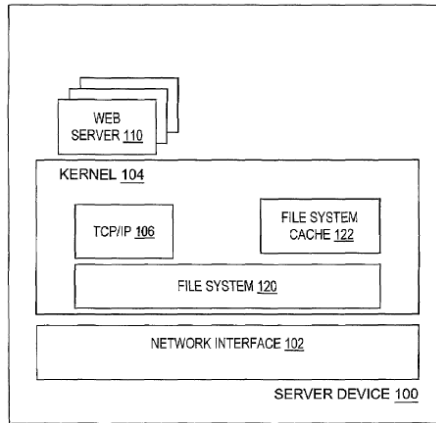


FIG. 1
PRIOR ART

Elnozahy, Fig. 1

In contrast to the above-described scheme that uses the kernel, Elnozahy discloses a web-server application that includes a “user space transmission protocol library that enables the server to perform its own network processing of requests and responses.” Elnozahy, [0008], Fig. 2. This library, which Elnozahy also refers to as a “network protocol library,” includes “TCP/IP library routines that provide a user space TCP/IP protocol stack” to the web server. Elnozahy, [0022]; Bhattacharjee, ¶73. The web server also uses “kernel extension drivers” to communicate directly with a “network interface” (*i.e.*, a hardware device that lets a computer access a network), without need for “kernel calls.” Elnozahy, [0020]; EX1035, 363; Bhattacharjee, ¶74. Additionally, the operating system has its own “protocol stack,” also called a “TCP/IP block” or “TCP/IP Stack,” that “interfaces with [a] general purpose network interface” that is separate from the interface used

by the web-server application; this “general purpose” interface is used by “other applications” or used for “other functions” than those performed by the web server.

Elnozahy, [0005], [0017], [0025]; Bhattacharjee, ¶75.

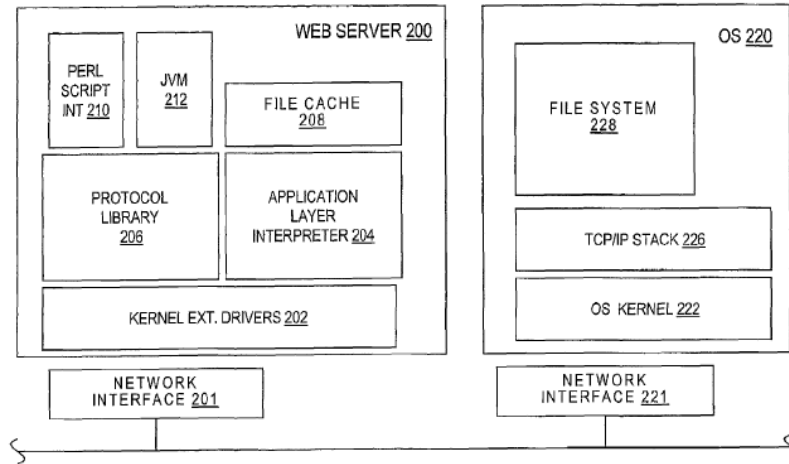


FIG. 2

Elnozahy, Fig. 2.

B. Draves (EX1017)

Draves is directed to allowing “sharing of executable program modules between kernel-mode threads [*i.e.*, tasks] and user-mode threads.” Draves, 4:51-57, 4:65-5:5; EX1035, 518; Bhattacharjee, ¶76. Draves accomplishes this sharing by creating a “shared range of addresses in memory” that appears in both the “user address space and the kernel address space” and placing “shared executable components” in that address space; “address space” refers to a range of memory

locations. Draves, 4:65-5:5; EX1035, 20; Bhattacharjee, ¶77. One exemplary “program module” Draves discusses that is “sharable” between user- and kernel-mode threads using Draves’s technique is a “DLL (dynamic link library)” (an alternate spelling for what the ’058 patent calls a “dynamic linked library (DLL)”). Draves, 3:57-61; EX1001, 2:48 Bhattacharjee, ¶77.

C. Elnozahy+Draves Combination

POSAs would have been motivated to implement Elnozahy’s user-space TCP/IP “protocol library” as a shared library usable by multiple applications, for the reasons explained below.

Elnozahy expressly contemplates “other applications” running on the same computer as the web-server application. Elnozahy, [0017]. Elnozahy mentions the “other applications” using the “general purpose interface 221,” and also mentions that a “protocol stack 226,” which is labeled “TCP/IP Stack 226” in Figure 2, “interfaces with the general purpose network interface.” Elnozahy, [0017], [0025]; Bhattacharjee, ¶¶78-79. Thus, Elnozahy discloses other applications aside from the web server running on the same computer as the web server and also using TCP/IP, consistent with a POSA’s background knowledge that it was well-known to run multiple applications using TCP/IP on a single computer. Bhattacharjee, ¶80; EX1072, [0007] (email); EX1073, [0031]-[0032] (instant messaging); EX1074, [0057] (examples including email, news, and terminal emulation);

EX1075, 1:26-40 (“file transfer protocol”); EX1076, [0044] (known for multiple TCP/IP applications differentiated by an assigned “port” to run on same server); EX1077, 3:39-62 (disclosing “server” running “several different application programs” using TCP/IP).

Furthermore, Elnozahy discloses that its user-space protocol library provides benefits such as “improved” “performance” due to reduced “context switching and protocol overhead.” Elnozahy, [0015]. Additionally, POSAs understood that it was “desirable to implement applications in a modular fashion, and to re-use modules already implemented by other applications.” EX1069, 2:21-23; Bhattacharjee, ¶81. Thus, in view of (i) Elnozahy’s own teachings about the user-space protocol library’s advantages and (ii) POSAs’ understanding about the desirability of module re-use, POSAs would have been motivated to implement Elnozahy’s user-space TCP/IP stack in a shared “protocol library” (Elnozahy, [0022]) that at least some of the “other applications” (Elnozahy, [0017]) use. Bhattacharjee, ¶82.

Additionally, Draves discloses that a DLL is a type of “potentially [] sharable program module.” Draves, 3:57-61. POSAs understood that a DLL was a library and was a well-known and customary way to share functionality between applications, as the ’058 patent itself acknowledges. EX1001, 2:45-50; EX1015,

959-960; EX1032, 1:28-35. POSAs would thus have been motivated to implement Elnozahy's user-space protocol library as a DLL. Bhattacharjee, ¶¶83-84.

In the resulting Elnozahy+Draves combination, Elnozahy's user-space "protocol library" (Elnozahy, Abstract, [0008]) is implemented as a DLL as taught in Draves (Draves, 3:57-61). This DLL is shared with Elnozahy's "web server" as well as with "other applications" running on the same computer that also use TCP/IP, as discussed above. Elnozahy, [0017]. POSAs would have reasonably expected success in achieving such an implementation, for multiple reasons. First, as discussed above, it was known for multiple applications using TCP/IP to run on the same server. Bhattacharjee, ¶85; EX1076, [0044]; EX1077, 3:39-62. Second, as also discussed above, it was known to share functionality between applications using a shared library such as a DLL. Bhattacharjee, ¶86; EX1001, 2:45-50; EX1015, 959-960; EX1032, 1:28-35.

D. Claim-by-Claim Analysis

1. Claim 1

a. [1PRE]: "A computing system for executing a plurality of software applications comprising:"

Elnozahy discloses a "server device" on which the "web server application" "reside[s]," where the "server device" also "*execute[s]*" other "*applications.*" Elnozahy, [0015], [0017]. In Elnozahy+Draves, the "web server" and at least some other "applications" use Elnozahy's user-space TCP/IP protocol library.

Supra §V.C. POSAs understood that a “server device” is a *computing system*. Bhattacharjee, ¶87; EX1078, 1:33-35. Elnozahy also discloses that its invention may be “implemented as a set of *computer executable instructions (software)*.” Elnozahy, [0016]. Thus, POSAs understood that Elnozahy’s “server device” is a *computing system for executing a plurality of software applications*, e.g., the applications using the user-space TCP/IP library. Bhattacharjee, ¶88.

b. [1A] “a) a processor;”

Elnozahy claims a “web server” system comprising a “*processor*.” Elnozahy, claim 9. Elnozahy also says that running “server class operating systems” may require a “server *processor*” to switch between executing different processes. Elnozahy, [0007]. POSAs understood, or at least would have found obvious, that the processor is a physical computer processor because it was customary for processors in computer systems to be implemented as physical processors. Bhattacharjee, ¶¶89-90; EX1096, 1:34-39 (describing a “processor” as a typical “hardware unit” in “computer architecture”). POSAs also understood that the “web server” comprising a “processor” in Elnozahy’s claim 9 is an example of a “server device” in Elnozahy’s written description. Elnozahy, [0004], [0017]; Bhattacharjee, ¶91. Thus, POSAs understood that Elnozahy’s “server device,” *i.e. computing system (supra* §V.D.1.a), includes a *processor* under the agreed litigation construction (EX1058, 6). Bhattacharjee, ¶92.

c. [1B]

i. [1B.1] “b) an operating system having an operating system kernel...”

Elnozahy’s “server device” ([0017]) *i.e. computing system (supra §V.D.1.a)*, includes an “*operating system* [OS].” Elnozahy, [0025], Fig. 2; Bhattacharjee, ¶93. POSAs understood that Elnozahy’s OS is a conventional OS, which is “[t]he software that controls the allocation and usage of hardware resources such as memory, central processing unit (CPU) time, disk space, and peripheral devices” (EX1035, 378), consistent with Elnozahy’s discussion of the “operating system” as including a “file system” that “retrieve[s] data from disk or other non-volatile storage facility” (Elnozahy, [0025]). Bhattacharjee, ¶¶94-95. Thus, Elnozahy’s “operating system” meets Google’s litigation construction. EX1058, 4.³

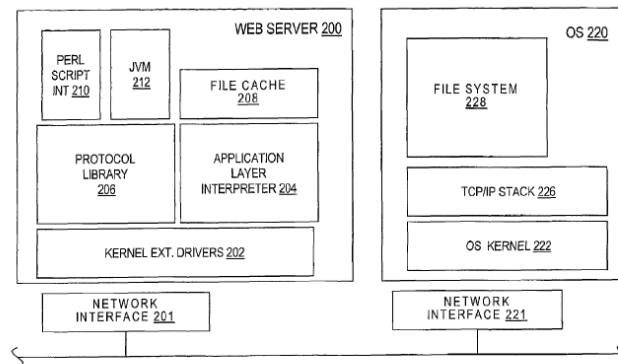


FIG. 2

Elnozahy, Fig. 2

³ VirtaMove asserted no construction necessary.

Elnozahy’s OS has a “*kernel*.” Elnozahy, [0025]. POSAs understood that Elnozahy’s kernel is a conventional kernel, which POSAs understood is the “core... portion” of an operating system “that manages memory, files, and peripheral devices; maintains the time and date; launches applications; and allocates system resources” (EX1035, 257), consistent with Elnozahy’s description of the kernel as handling “processing of interrupts,” starting and prioritizing application threads, and mapping hardware to applications. Elnozahy, [0020], [0027]; Bhattacharjee, ¶¶96-97. Thus, Elnozahy’s “kernel” meets Google’s litigation construction. EX1058, 4.⁴

Thus, Elnozahy’s server device *comprises an operating system having an operating system kernel*. Bhattacharjee, ¶98.

ii. [1B.2] “[OS kernel] having OS critical system elements (OSCSEs)...”

The ’058 patent says a *critical system element* (CSE) is “[a]ny service or part of a service, ‘normally’ supplied by an operating system, that is critical to the operation of a software application.” EX1001, 6:6-8; Bhattacharjee, ¶99.

Examples of CSEs include “[n]etwork services including TCP/IP.” EX1001, 6:12-13; *see also* EX1001, 3:22-30 (noting that a “kernel may provide a TCP/IP service” and describing this as an example of a CSE); Bhattacharjee, ¶100.

⁴ VirtaMove asserted no construction necessary.

Elnozahy explains, consistent with a POSA’s understanding, that in a “conventional web server architecture” such as shown in Figure 1, the “kernel...invoke[s]...network protocol routines indicated in FIG. 1 by the Transmission Control Protocol/Internet Protocol (TCP/IP) block 106,” which Figure 1 shows as being part of the kernel. Elnozahy, [0005], Fig. 1; Bhattacharjee, ¶101.

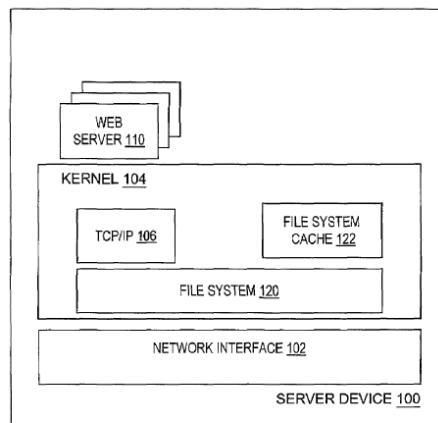


FIG. 1
PRIOR ART

Elnozahy, Fig. 1

The *kernel* in Elnozahy’s operating system is a conventional kernel. *Supra* §V.D.1.c.i ([1B.1]). Thus, POSAs would have understood Elnozahy to disclose an implementation of its *kernel* that includes a TCP/IP block as shown in Figure 1 (along with an alternate implementation where TCP/IP is implemented in a separate module from the kernel, as shown in Figure 2). Bhattacharjee, ¶102. At a

minimum, POSAs would have found it obvious to implement Elnozahy+Draves this way, because this was the “conventional” implementation, which Elnozahy does not teach away from, and would have been within a POSA’s skill. Elnozahy, [0005]; Bhattacharjee, ¶103. In such an implementation, the *OS kernel has the TCP/IP block*. Bhattacharjee, ¶104.

Elnozahy discloses that a TCP/IP block includes “routines” (plural) and “functions” (plural) including “extract[ing] an HTTP formatted request” from packets and “hand[ing] the extracted HTTP request to a web server application.” Elnozahy, [0005]. Thus, while the full scope of “normally supplied” and “critical” in the ’058 patent’s definition of *CSE* (EX1001, 6:6-8) are unclear (EX1059, 14-16), POSAs understood that at least the TCP/IP “routines” and “functions” in Elnozahy’s TCP/IP block in the kernel are *CSEs*. Bhattacharjee, ¶105; *PLR Worldwide Sales Ltd. v. Flip Phone Games, Inc.*, IPR2024-00209, Paper 9, 39 (May 10, 2024) (“even if full scope of” a term “is indefinite because the specification does not provide a sufficient boundary...that does not prevent us from making a determination that a teaching is well within that boundary”). Furthermore, these functions and routines are *CSEs* even if VirtaMove’s litigation construction were accepted (*see* EX1059, 14), since these functions/routines match examples of *CSEs* provided in the ’058 patent. *See* EX1067, 17 (VirtaMove claim-

construction brief citing “TCP/IP” and “network services” as examples of CSEs in patent); Bhattacharjee, ¶106.

The '058 patent also says “[a] CSE is a dynamic object providing some function that is executing instructions used by the applications.” EX1001, 6:8-10. Neither party has asserted that this language imposes a claim requirement—nevertheless, this would have been an obvious implementation of Elnozahy. A “dynamic object” is an “object,” *i.e.* a set of instructions and/or data, that can be created by a program and then erased. Bhattacharjee, ¶107; EX1021, [0053]; EX1007, 1:22-33; EX1035, 458. A function is also a set of instructions. *Supra* §III.A; EX1007, 1:22-23; Bhattacharjee, ¶107. POSAs would have been motivated to implement Elnozahy’s TCP/IP block functions/routines as dynamic objects, and reasonably expected success, because dynamically loading objects was one of two known options, *i.e.* static and dynamic. Bhattacharjee, ¶108; *KSR Int’l v. Teleflex*, 550 U.S. 398, 416-17 (2007).

Finally, as discussed above, Elnozahy’s *OS kernel has* the TCP/IP stack. Therefore, the CSEs in the TCP/IP block in the OS are *OS critical system elements (OSCSEs)*. Bhattacharjee, ¶109.

iii. [1B.3] “[OSCSEs] for running in kernel mode using said processor”

Elnozahy discusses in its “BACKGROUND” the well-known “user space” and “kernel space” that “identify conceptual spaces defined by the operating

system that have different levels of protection.” Elnozahy, [0007]. Elnozahy’s “BACKGROUND” also discusses the known problem of having to perform “*context* switching” when “an application program operating in user space invokes a process that executes in the kernel space.” Elnozahy, [0007]. Thus, POSAs understood that Elnozahy uses the term “kernel space” to refer to the context in which the kernel portion of an operating system executes, and further that application code cannot run in kernel space (because a context switch is required to invoke a kernel-space process), thus meeting the parties’ agreed litigation construction of *kernel mode*. EX1058, 6; Bhattacharjee, ¶¶110-111.

At a minimum, in Elnozahy+Draves, POSAs would have found it obvious to implement Elnozahy’s computing system with a “kernel mode” as disclosed in Draves, because Draves describes such a mode as “common.” Draves, 1:23-39. Draves further explains that this “common” “kernel mode” is distinct from the “user mode” that is used by “application programs.” Draves, 1:25-28, 1:37-40; Bhattacharjee, ¶112. POSAs would reasonably have expected success in implementing Elnozahy+Draves with the “kernel mode” disclosed in Draves as it was “common for a computer processor and associated [OS]” to have kernel and user modes. Draves, 1:23-39; Bhattacharjee, ¶112.

Elnozahy says in its “BACKGROUND” section when discussing Figure 1 that the “kernel...invoke[s] one or more network protocol routines indicated in

FIG. 1 by the [TCP/IP] block 106” that is inside “Kernel 104” in Figure 1.

Elnozahy, [0005]. Elnozahy does not teach away from this “BACKGROUND” implementation; thus, in Elnozahy/Draves, a POSA would at least have found it obvious to implement Elnozahy’s system so that the OS kernel invokes the TCP/IP block that is part of the OS kernel itself. Furthermore, Draves notes that “critical operating system components” “execute from,” i.e. run from, “kernel mode” (also called “system mode”). Draves, 1:23-48. Thus, the OSCSEs (which are in the TCP/IP block, *see supra* §V.D.1.c.ii ([1B.2])) are *for running in kernel mode*.

Bhattacharjee, ¶113.

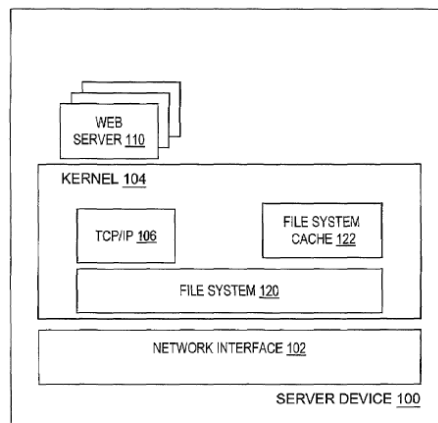


FIG. 1
PRIOR ART

Elnozahy, Fig. 1

Furthermore, POSAs understood that the OSCSEs run in kernel mode *using said processor* because running software such as a TCP/IP block is the purpose of a processor. Bhattacharjee, ¶114.

Therefore, Elnozahy’s OSCSEs are *for running in kernel mode using said processor*. Bhattacharjee, ¶115.

d. [1C]

i. [1C.1] “c) a shared library having shared library critical system elements (SLCSEs) stored therein...”

The ’058 patent’s specification defines a “*shared library*” as:

An application library code space shared among all user mode applications. The code space is different than that occupied by the kernel and its associated files. The shared library files are placed in an address space that is accessible to multiple applications.

EX1001, 6:49-53. “Code space” refers to physical memory space. EX1001, 3:39-43 (“same physical memory space, that is, shared code space”); Bhattacharjee, ¶116. The ’058 patent states that a DLL is an example of a “shared library.” EX1001, 2:47-50 (“A shared library or dynamic linked library (DLL) refers to an approach...”); Bhattacharjee, ¶117.

In Elnozahy+Draves, Elnozahy’s user-space “protocol *library* 206” (Elnozahy, [0022]) is implemented as a DLL, as Draves discloses (Draves, 3:57-61). *Supra* §V.C. Furthermore, Draves teaches that a “DLL (dynamic link *library*)” is an example of a “*shareable* program module.” Draves, 3:57-61. Thus, in Elnozahy+Draves, the “protocol *library*” implemented as a DLL is a *shared library*. Bhattacharjee, ¶118.

Furthermore, in an obvious implementation of Elnozahy+Draves, the protocol library implemented as a DLL satisfies the '058 patent's specification's above-quoted definition of *shared library*, which Google's litigation construction adopts (EX1059, 16). Consistent with a POSA's understanding, Draves describes "DLLs," which are "*shareable*," as residing in a single physical memory *address space*, *i.e. code space*, that is *different from that occupied by the kernel and its associated files* and that is mapped to a "code portion" in the "virtual *address space*" of two different applications, and thus *accessible to multiple applications*. Draves, 2:28-31 (describing "typical approach" involving "allocat[ing] some portion of the memory to the operating system or kernel and another portion for application or user processes."), 3:52-4:32 (describing "code portions 25" and "data portion 26" of "DLL" as mapped to separate region of memory from "kernel," as shown in Fig. 5), Fig. 5; *see also* EX1015, 959-966; Bhattacharjee, ¶119.

The library also meets VirtaMove's litigation construction, which requires "[a]n application library whose code space is shared among all user mode applications" (EX1068, 11), for the following reasons. First, POSAs understood that the shared physical memory space into which the DLL is loaded is the DLL's (library's) code space. Bhattacharjee, ¶120; Draves, 2:28-31, 3:52-4:32; EX1067, 19 (VirtaMove arguing that "'code space' is a space occupied by code"). Second,

POSAs understood that this code space is shared among all user mode applications in the same manner described in the '058 patent. The '058 patent says the “manner” in which “the code” for an SLCSE “is shared by all applications on the same compute platform” is that “all applications may physically execute the same set of instructions” representing an SLCSE that reside in “the same physical memory space, that is, shared code space.” EX1001, 3:30-44; Bhattacharjee, ¶120. Likewise, as discussed above, POSAs understood that the code representing a DLL is stored in a memory location that all user-mode applications may access to execute the same code. Bhattacharjee, ¶¶119-120; Draves, 2:28-31, 3:52-4:32, Fig. 5; EX1015, 959-966.

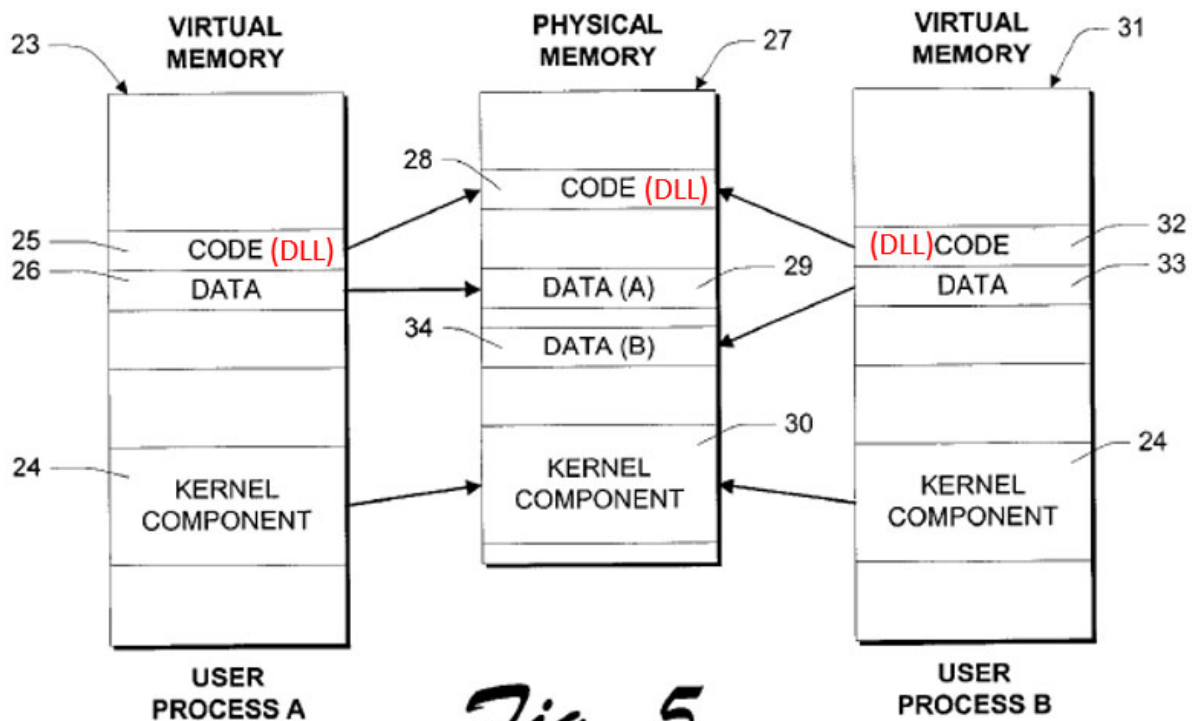


Fig. 5
Prior Art

Draves, Fig. 5

Finally, because the “protocol library” includes “TCP/IP library routines that provide a user space TCP/IP protocol stack” (4:38-40), and because Elnozahy states “[p]rotocol library 206 may be implemented as a fully general TCP/IP library capable of handling all of protocol supported events and tasks” (Elnozahy, [0022]), POSAs understood Elnozahy to disclose that the protocol library has all of the same TCP/IP “routines” and “functions” that are in the operating system’s TCP/IP block, and that are *CSEs* for the reasons discussed *supra* §V.D.1.c.ii for [1B.2]. Bhattacharjee, ¶121. The “protocol library” containing the user-mode

TCP/IP “routines” and “functions,” which are *CSEs* as discussed above, is implemented as a DLL in Elnozahy+Draves. *Supra* §V.C. POSAs understood that a library “*store[s]*” its contents. EX1035, 309 (“library” is “[i]n programming, a collection of routines stored in a file”); Bhattacharjee, ¶122; EX1065, 1:46-49. Because these CSEs are in a shared library, i.e., the protocol library, these CSEs are *SLCSEs*. EX1001, 2:45-47 (“In accordance with this invention, SLCSEs are placed in shared libraries[.]”); Bhattacharjee, ¶122. Thus, Elnozahy+Draves’s protocol library is a *shared library having SLCSEs stored therein*. Bhattacharjee, ¶123.

ii. [1C.2] “for use by the plurality of software applications in user mode...”

Elnozahy discusses in its “BACKGROUND” the well-known “user space” and “kernel space” that “identify conceptual spaces defined by the operating system that have different levels of protection.” Elnozahy, [0007]. Elnozahy’s “BACKGROUND” also discusses the known problem of having to perform “*context switching*” when “an *application* program operating in user space invokes a process that executes in the kernel space.” Elnozahy, [0007]. Thus, POSAs understood that Elnozahy uses the term “user space” to refer to the context in which applications execute, i.e., *user mode* under the parties’ agreed litigation construction. EX1058, 6; Bhattacharjee, ¶¶124-125.

The SLCSEs are in Elnozahy’s “protocol library,” which in Elnozahy+Draves is a DLL (*shared library*). Elnozahy, Abstract, [0008]; *supra* §§V.D.1.d.i ([1C.1]), V.C. Elnozahy says the protocol library “includes TCP/IP library routines that provide a *user space* TCP/IP protocol stack to web server 200.” Elnozahy, [0022]. Furthermore, in Elnozahy+Draves, *the plurality of software applications* use the protocol stack provided by the protocol library. *Supra* §§ V.D.1.a ([1Pre]), V.C. Thus, the SLCSEs in the protocol library are *for use by the plurality of software applications in user mode*. Bhattacharjee, ¶126.

e. [1D]

i. [1D.1] “i) wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and...

Elnozahy’s “TCP/IP library routines” and “functions” in the “protocol library” are *SLCSEs stored in the shared library*. Elnozahy, [0022]; *supra* §V.D.1.d.i ([1C.1]).

The ’058 patent says, “the term replica used herein is meant to denote a CSE having similar attributes to, but not necessarily and preferably not an exact copy of a CSE in the [OS].” EX1001, 1:65-2:1; Bhattacharjee, ¶127. The specification provides an example of “*replication*” of an OSCSE using an SLCSE where TCP/IP services are provided by a shared library. *See* EX1001, 5:22-25. Specifically, the specification says “[e]mbodiments of the invention enable the *replication* of

critical system elements normally found in an *operating system* kernel,” and describes an “example” in which the “CSE that is part of a shared library” is a “TCP/IP stack,” and where the “TCP/IP services in the CSE are the same as those provided in the Linux kernel.” EX1001, 5:22-55, 5:34-55; Bhattacharjee, ¶128. Likewise, in Elnozahy+Draves, TCP/IP routines/functions (*SLCSEs*) are in the “protocol library” that is implemented as a shared library (a DLL) and that provides “a user space TCP/IP protocol stack” providing the same services as the operating system’s TCP/IP block. *Supra* §V.D.1.d.i ([1C.1]); Elnozahy, [0022]; Bhattacharjee, ¶129. Thus, although the full scope of “functional replicas” is unclear to POSAs (EX1059, 18-20), POSAs understood that at least Elnozahy’s TCP/IP “routines” and “functions,” which are claimed *SLCSEs* (*supra* §V.D.1.d.i ([1C.1])), qualify as *functional replicas* of their kernel-mode counterpart functions/routines, which are claimed *OSCSEs* (*supra* §V.D.1.c.ii ([1B.2])), because the relationship Elnozahy describes between the user-mode TCP/IP routines/functions and their kernel-mode counterparts matches the *SLCSE/OSCSE* replication example in the ’058 patent’s specification. Bhattacharjee, ¶130; *PLR*, IPR2024-00209, Paper 9, 39 (Board may determine that teaching is within claim boundaries even if boundaries are unclear).

Additionally, the “functions” and “routines” in the user-mode protocol library in Elnozahy+Draves perform the same operation as their kernel-mode

counterparts. Thus, although the full scope of “similar attributes” in the ’058 patent’s definition of “replica” is unclear to POSAs (EX1059, 19-20), POSAs would have understood that the user-mode protocol library functions/routines have the same relevant attributes (*i.e.*, services provided) as their kernel-mode counterparts, in the same manner described in the ’058 patent, and are thus *functional replicas of OSCSEs* even though the outer boundaries of this language are unclear. Bhattacharjee, ¶131.

The “functions” and “routines” in the Elnozahy+Draves user-mode protocol library are also *functional replicas of OSCSEs* even if VirtaMove’s litigation construction, “substantial functional equivalents or replacements of kernel functions” (EX1067, 22), were accepted, because in Elnozahy+Draves the SLCSE/OSCE relationship matches the example described in the patent, as discussed above. Bhattacharjee, ¶132.

During prosecution, the applicants distinguished O’Rourke by arguing that a “functional replica” cannot be a mere “intermediary” to the kernel. EX1002, 323. POSAs understood that Elnozahy’s functions and routines in the user-mode protocol library (the *SLCSEs*) are not mere intermediaries. In characterizing O’Rourke as merely disclosing an “intermediary,” the applicants cited O’Rourke 6:12-19 and 10:12-33. These passages show that, in contrast to Elnozahy’s protocol library, which allows applications to bypass the kernel (Elnozahy,

[0020]), O'Rourke disclosed a user-mode "generic proxy object" that did nothing but invoke kernel-level code. *See* O'Rourke, 6:12-19 ("...a primary object" of O'Rourke is "to provide user mode proxies for kernel mode filters. Other objects... include ... providing a generic proxy object that may be used for virtually all kernel mode filters..."), 10:12-33 (describing various "prox[ies]" shown in O'Rourke Figure 3 as invoking "kernel mode" components); Bhattacharjee, ¶133.

Thus, Elnozahy's *SLCSEs stored in the shared library are functional replicas of OSCSEs* as claimed. Bhattacharjee, ¶134.

ii. [1D.2] "are accessible to some of the plurality of software applications and..."

POSAs would have understood that the *SLCSEs* in Elnozahy+Draves are *accessible to some of the plurality of software applications*, or alternatively would have found it obvious to implement Elnozahy+Draves this way, as explained below. Bhattacharjee, ¶135.

Elnozahy+Draves's *SLCSEs* are in a DLL used by *the plurality of software applications*. *Supra* §V.D.1.d.ii ([1C.2]). POSAs understood that a purpose of a DLL is to make its contents *accessible to* the applications using the DLL. Bhattacharjee, ¶136; EX1015, 959-60 (DLLs are "separate files containing functions that can be called by programs...[;] one purpose of [DLLs] is to provide

functions and resources that can be used by many different programs”), 965-966 (“Multiple applications can use the same DLL simultaneously...”).

Additionally, as explained below, POSAs understood that accessing the SCLSEs from Elnozahy+Draves’s protocol library involves *reading* them, which meets Google’s litigation construction of “*accessible*.” EX1059, 17; Bhattacharjee, ¶137. The Elnozahy+Draves *SLCSEs* are TCP/IP “functions” and “routine” in the user-space protocol library. *Supra* §V.D.1.d.i ([1C.1]). POSAs understood that a function/routine is a predefined set of instructions that carry out a specific action. *Supra* §III.A; Bhattacharjee, ¶138; EX1007, 1:22-29; EX1035, 458 (“routine” and “function” are used as synonyms). POSAs also understood that executing the instructions that make up a function/routine involves *reading* those instructions. Bhattacharjee, ¶¶138-139; EX1056, 4:18-21.

The Elnozahy+Draves SLCSEs also meet VirtaMove’s litigation construction, which requires the SLCSE to be accessible to “some applications,” and only requires “use” by the applications. EX1067, 20. POSAs understood that by reading and executing the instructions making up the SCLSEs as discussed above, Elnozahy+Draves’s applications *use* the SCLSEs. Bhattacharjee, ¶140.

Furthermore, because the *plurality* of applications collectively use each SLCSE, *some applications use* each SLCSE. Bhattacharjee, ¶140. Therefore, the

SLCSEs are *accessible to at least some of the plurality of software applications*.

Bhattacharjee, ¶141.

- iii. [1D.3] **“when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications”**

The '058 patent says SLCSEs “*form a part* of at least some of the software applications *by being linked* thereto.” EX1001, 2:10-15; *see also id.*, claim 16 (reciting “SLCSEs form a part of at least some of the plurality of software applications, by being linked thereto”); Bhattacharjee, ¶142. The parties’ agreed litigation construction of “*forms a part of...*” adopts the “*linked to*” requirement. EX1058, 6.

In Elnozahy+Draves, the SLCSEs are part of a DLL (“dynamic *link* library”) as Draves teaches. Draves, 3:58; *supra* §V.D.1.d.i ([1C.1]). POSAs therefore understood that when an application accesses an SLCSE, the SLCSE *forms a part of* that application because the DLL is linked to the application. Bhattacharjee, ¶¶143-144; EX1018, 745 (“A DLL is a file that gets mapped into a process’s address space such that any functions in the DLL appear to be *part of* the process”); EX1033, 4:47-49; EX1015, 965-966. Thus, *when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications* as claimed.

Bhattacharjee, ¶145.

f. [1E]

- i. [1E.1] “ii) wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and...”

“*Instance*” is not a term of art specific to CSEs, but rather customarily refers to an object or a copy of an object. Bhattacharjee, ¶146; EX1034, 186; EX1035, 276. The ’058 patent’s specification does not define an SLCSE *instance*; however, the specification says the “intent” of having applications “link” to the SLCSE library “is to *provide* an *application* with a unique *instance of a CSE*.” EX1001, 3:20-29; Bhattacharjee, ¶147. The specification also does not explain what it means for an *instance* of an SLCSE to *run in a context of an application*. However, the specification says the “ability to allow a [SL]CSE to execute in the same context as an application...allows...an ability to deploy multiple instances of a [SL]CSE.” EX1001, 1:46-54; Bhattacharjee, ¶148. The ’058 patent in turn explains that SLCSEs *run in a context* of an application because the SLCSEs reside in a “library” that is linked to the application. EX1001, 5:22-26 (“Embodiments of the invention enable the **replication** of critical system elements.... These **replicated CSEs** are then able to *run in the context of a software application*.”

Critical system elements are **replicated through the use of shared libraries.**”), 8:27-33, 9:15-20; Bhattacharjee, ¶148.

POSAs understood that in an obvious implementation of Elnozahy+Draves, SLCSEs are accessed using the same technique as in the '058 patent, which POSAs understood would yield the same result. Bhattacharjee, ¶149.

Elnozahy+Draves's SLCSEs are in a “dynamic *link* library,” which is a shared library “*link[ed]*” to applications, just as the '058 patent's SLCSE library is “linked” to applications. Draves, 3:57-61; *supra* §V.D.1.d.i ([1C.1]); EX1001, 3:20-29; Bhattacharjee, ¶150. Furthermore, POSAs understood that an obvious way for applications to invoke the functions/routines in the protocol library was to “call” them, just as the '058 patent's software applications “call” functions in the SLCSE library (EX1001, 8:27-33). EX1069, 2:23-29 (“Under traditional operating system implementations, such re-usable code modules (called ‘shared libraries’)... can be implemented in the user space, and any task that desires access to a function of the shared library may be allowed to directly call the function.”); Bhattacharjee, ¶150. As the '058 patent itself admits, as discussed above, the well-known consequence of calling a library's function is that *an instance of each operation is provided to each application from the shared library and is run in the context of that application.* EX1001, 5:22-25, 8:27-33, 9:15-20; Bhattacharjee, ¶150.

The '058 patent does not expressly state how SLCSE *instances* are provided to applications *without being shared* between applications. However, the patent explains that there is one copy of the computer “instructions” (or “code”) representing an SLCSE that is “shared by all applications,” but each application “uses [its own] separate data area[]” when running the SLCSE. EX1001, 3:30-42; Bhattacharjee, ¶151. “In this manner,” “CSEs are not shared among applications even though the code is shared.” EX1001, 3:42-44; Bhattacharjee, ¶151.

POSAs understood that in Elnozahy+Draves, applications access SLCSEs in the same manner described in the '058 patent. SLCSEs are provided to applications in a DLL, as Draves teaches. *Supra* §V.D.1.d.i ([1C.1]). Bhattacharjee, ¶152. Draves also teaches, consistent with a POSA’s background knowledge, that different applications that use a DLL share the code representing the DLL but use their own data areas in memory when running the code. Draves, 4:16-33 & Fig. 5 (showing two different programs A and B having their own data areas when accessing a DLL); Bhattacharjee, ¶153; EX1015, 965-966; EX1018, 745. At a minimum, POSAs would have found it obvious to implement Elnozahy+Draves such that applications share the code for operations but use application-specific data areas when running the code. Bhattacharjee, ¶154. The '058 patent admits that such an implementation was “common practice” (EX1001, 3:32-44), and POSAs understood that using this technique would beneficially

allow multiple applications to use the same operation; furthermore, POSAs would reasonably have expected success with such an implementation because it was common practice. Bhattacharjee, ¶154; EX1015, 966; EX1018, 745.

Thus, in Elnozahy+Draves, *an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications.* Bhattacharjee, ¶155.

- ii. **[1E.2] “where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function, and...”**

POSAs understood that the *plurality of software applications run under the operating system*, or at least would have found it obvious to implement Elnozahy+Draves this way, because this was the standard relationship between applications and OSs. Bhattacharjee, ¶156; EX1037, [0003] (“Computer systems typically include... one or more software application programs that run under the control of the operating system.”).

In Elnozahy+Draves, *SLCSEs* are in a DLL, which is a shared library. *Supra* §V.D.1.d.i ([1C.1]). Because DLLs are shared libraries, POSAs understood that multiple applications running under the computer’s OS can call (*i.e., have use of*) the same operation in a DLL. Bhattacharjee, ¶157; EX1015, 959-960, 965-966.

The '058 patent says an application gets a “*unique instance*” of an SLCSE by “linking” to the “library containing the [SLCSE].” EX1001, 3:25-29; Bhattacharjee, ¶158. The patent also states that because “[e]ach application has its own unique data space...[SLCSEs] are unique to an application.” EX1001, 3:36-39; Bhattacharjee, ¶159. In Elnozahy+Draves, each application is linked to the “protocol library” (implemented as a DLL) containing the SLCSEs. *Supra* §V.D.1.e.iii ([1D.3]). Furthermore, each application has its own unique data space in memory. *Supra* §V.D.1.f.i ([1E.1]). Thus, in Elnozahy+Draves, *at least a second of the plurality of software applications running under the [OS] have use of a unique instance of a corresponding critical system element for performing same function.* Bhattacharjee, ¶160.

- g. [1F] “iii) wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.”**

In Elnozahy+Draves, the “functions”/“routines” in the protocol library are claimed *SLCSEs*. *Supra* §V.D.1.d.i ([1C.1]). POSAs would at minimum have found it obvious to implement Elnozahy+Draves so that the functions/routines are invoked by reference to some *predetermined* identifier, *e.g.*, a function/routine name, as this was a known and typical way to invoke functions/routines from

libraries that was within a POSA's skill. Bhattacharjee, ¶161; EX1069, 10:55-58 (describing “[c]ode module” that “includes instructions that reference functions called ‘foo()’ and ‘goo()[]’”). Thus, each SLCSE relates to a *predetermined function*. Bhattacharjee, ¶161.

These SLCSEs are *provided to the plurality of the software applications*, including the *first* and *second* of the plurality, because they are in a DLL accessed by the applications. *Supra* §V.D.1.d.i-V.D.1.d.ii ([1C.1]-[1C.2]). Furthermore, the SLCSEs are provided to the applications *for running instances of a SLCSE*; the SLCSEs are in a DLL, and when an application invokes an SLCSE from the DLL, the application gets its own *instance* of the SLCSE that *runs* in that application's context. *Supra* §V.D.1.f.i ([1E.1]); Bhattacharjee, ¶162.

Finally, the SLCSEs are provided to the *first* and *second* of the *plurality of applications* for running their own *instances of the SLCSE* (i.e., *first* and *second instances*) *simultaneously*. “[S]imultaneous[]” appears nowhere in the '058 patent's specification. Element [1F] was originally filed as dependent claim 6. EX1002, 22. During prosecution, the applicants distinguished as-filed claim 6 over Cabrero by arguing that CSEs are “replicated and *can be accessed simultaneously* by...an application in user mode *by way of* the application using an instance of the CSE within a shared library,” which was in “contradistinction” to Cabrero, where CSEs are “independent applications that neither reside in the OS

nor in shared libraries.” EX1002, 241-242, 244; Bhattacharjee, ¶163. The applicants also said, “claim 6 *allows* different instances of an SLCSE...to be executed simultaneously by another one or more applications.” EX1002, 244. Thus, according to the applicants, Element [1F]’s “*for running...simultaneously*” is met by allowing two different applications to have simultaneous access to an SLCSE in a shared library. Bhattacharjee, ¶163. As discussed above, Elnozahy+Draves’s SLCSEs are functions/routines provided by a DLL, and POSAs understood that one purpose of DLLs was to allow multiple applications to access the same code in the DLL “*simultaneously*.” EX1015, 959-960, 965-966 (“Multiple applications can use the same DLL simultaneously”); Bhattacharjee, ¶¶163-164.

2. Claim 2

Elnozahy+Draves’s *first* and *second* applications are allowed to *simultaneously* run an *instance* of an SLCSE. *Supra* §V.D.1.g ([1F]). Each SLCSE is *stored in a shared library*. *Supra* §V.D.1.d.i ([1C.1]). The *first* and *second instances* together comprise *multiple* instances. If VirtaMove argues that “*multiple instances*” means three or more instances, this would have been obvious, because POSAs understood that it was typical for a single computer system to run three or more applications that use TCP/IP. Bhattacharjee, ¶165; EX1074, [0057] (example “TCP/IP based applications” include email, news, and terminal

emulation). POSAs would have been motivated to run at least three different TCP/IP applications on the same computer to achieve the efficiency benefit of using the same hardware for multiple tasks, and would reasonably have expected success given that Elnozahy itself discloses running multiple applications.

Elnozahy, [0017]; *supra* §V.A; Bhattacharjee, ¶166.

POSAs understood that when Elnozahy+Draves's SLCSE instances are run simultaneously, *i.e.*, *in operation*, they run *within the operating system*, as explained below. The '058 patent does not explain what it means for an SLCSE instance to run "*within*" the OS. However, the patent refers to "***applications*** [that] run[] under the [OS]" and that have "use of a unique ***instance*** of a corresponding critical system element." EX1001, Abstract, 2:18-21, 2:35-40; Bhattacharjee, ¶167. Furthermore, POSAs understood that the prior art sometimes referred to applications as running "within" an OS. EX1038, 1:18-20 (referring to "conventional computer systems having...software applications running within the [OS]"); EX1039, 1:6-15; Bhattacharjee, ¶167. Thus, POSAs understood that claim 2's reference to "*instances of an SLCSE...run[ning]...within the [OS]*" encompasses applications running within the OS and using their own instances of SLCSEs. Bhattacharjee, ¶168.

In Elnozahy+Draves, the first and second applications are provided with their own unique instances of an SLCSE. *Supra* §V.D.1.f.i ([1E.1]). POSAs

understood that in Elnozahy+Draves, the applications run within the OS because Elnozahy teaches that applications run in the kernel “context,” and the kernel is part of the OS, as discussed above for [1B.3] and [1B.1]. *See also* EX1018, 3 (“An *operating system*...provides a basis for application programs[.]”), 5 (“An [OS]... manages the execution of user programs[.]”); Bhattacharjee, ¶169.

Thus, in Elnozahy+Draves’s computer system, *in operation, multiple instances of an SLCSE stored in the shared library run simultaneously within the [OS]*. Bhattacharjee, ¶170. Additionally, *SLCSEs* in Elnozahy+Draves replicate the functionality of CSEs in the OS (i.e., *OSCSEs*) (*supra* §V.D.1.e.i [1D.1]) and are in shared libraries from where they can be run simultaneously (*supra* §V.D.1.g ([1F])), in contrast to the ’058 patent applicants’ characterization of Cabrero when distinguishing claim 2 during prosecution. EX1002, 242; Bhattacharjee, ¶171.

3. Claim 3

In an obvious implementation of Elnozahy+Draves, *OSCSEs* are part of the TCP/IP block that is inside the kernel. *Supra* §V.D.1.c.ii ([1B.2]). Thus, the *OSCSEs remain in the operating system kernel*. Bhattacharjee, ¶172. Furthermore, these *OSCEs correspond to* and are *capable of performing the same function as* the *SLCSEs*, because each *SLCSE* performs the same TCP/IP routine/function as an *OSCSE*. *Supra* §V.D.1.e.i ([1D.1]); Bhattacharjee, ¶173.

4. Claim 4

a. [4A]

“*[E]xclusive*” appears nowhere in the ’058 patent’s specification. However, the specification refers to “an application” being “provided” a “unique instance” of an SLCSE via “link[ing]” to the SLCSE library. EX1001, 3:25-29; Bhattacharjee, ¶174.

In Elnozahy+Draves, the *SLCSEs* are in a “DLL” that the *applications* “link[]” to. *Supra* §V.D.1.e.iii ([1D.3]). An instance of an SLCSE is *provided to one of the plurality of software applications* when the application calls the SLCSE from the linked library; additionally, when *SLCSEs* are called by an application, the SLCSEs run in the context of the application and are not shared. *Supra* §V.D.1.f.i ([1E.1]). Thus, the application *has exclusive use of* the *SLCSE* instance that is *provided* to the application by virtue of the application calling the SLCSE. Bhattacharjee, ¶175.

If VirtaMove argues that claim 4 requires an SLCSE to only be usable by one application, this would be incorrect; claim 4 depends from claim 1, which requires that the SLCSEs are in a *shared* library and are usable by the *plurality* of applications (*see* Elements [1C.1]-[1C.2]). Bhattacharjee, ¶¶176-177.

b. [4B]

POSAs understood a “system call” is a mechanism for “an application to invoke a kernel service.” EX1018, 463; Bhattacharjee, ¶178. In

Elnozahy+Draves, *one or more SCLSEs use system calls to access services in the [OS] kernel*, as explained below. Bhattacharjee, ¶178.

i. File retrieval

Elnozahy’s “Web server 200” includes an “HTTP interpreter” that can respond to received TCP/IP requests by providing an “HTML file.” Elnozahy, Abstract, [0028]-[0029]. The HTTP interpreter retrieves the file to be provided in one of two ways. If the file is in the server’s user-mode “file cache,” the interpreter retrieves the file from the cache. Elnozahy, [0029]. If not, the server “invoke[s] the operating system file system...to retrieve the [file] from disk.” Elnozahy, [0025], [0029]; Bhattacharjee, ¶179.

The ’058 patent says examples of CSEs include “[f]ile [s]ystem services that offer extensions to those supplied by the OS” such as “[a]ccess[ing] files that reside in different locations as though they were all in a single locality”; examples also include “[i]mplementation of file system optimizations for specific application behavior.” EX1001, 6:14-22; Bhattacharjee, ¶180.

POSAs understood that Elnozahy’s above-discussed feature of retrieving an HTML file either from the server’s file cache or from a disk using the “operating system file system” (Elnozahy, [0025]) matches examples of CSEs in the ’058 patent, including “[a]ccess[ing] files that reside in different locations as though they were all in a single locality” and “[i]mplementation of file system

optimizations for specific application behavior.” EX1001, 6:14-22; Bhattacharjee, ¶181. Furthermore, POSAs understood that the file-retrieval feature is provided in user mode, because Elnozahy describes retrieval as a task performed by the “user space” web server. Elnozahy, [0025], [0008]; *supra* §V.D.1.d.ii ([1C.2]); Bhattacharjee, ¶181.

Thus, POSAs would have found it obvious to implement this feature as a function/routine in a shared library such as a DLL for the same reasons of reusability discussed *supra* §V.D.1.d.i for Element [1C.1] for other user-mode features. Bhattacharjee, ¶182. Additionally, the file-retrieval feature has the same attribute of the service provided (*i.e.*, obtaining a file) as the corresponding operating-system feature discussed below that retrieves a file by reading it from disk, and is thus a functional replica of that OS feature, *i.e.*, of that *OSCSE*. Bhattacharjee, ¶182. In such an obvious implementation, where the user-space file-retrieval feature is implemented in a DLL linked to a (web-server) application, that feature (which is a functional replica of a corresponding operating-system feature) is an *SLCSE* for the same reasons discussed for claim 1 with respect to the functions/routines in the user-space TCP/IP library. Bhattacharjee, ¶182. Because this *SLCSE* is in a DLL, it is also *provided to one of the plurality of software applications having exclusive use thereof* for the same reasons discussed *supra*

§V.D.4.a ([4A]) with respect to SLCSEs in the user-mode protocol library that is also a DLL. Bhattacharjee, ¶182.

Elnozahy’s Figure 1, illustrating “selected features of conventional web server architecture” (Elnozahy, [0005]), shows the OS’s “File System” as part of the kernel. Elnozahy does not teach away from this “conventional” implementation; thus, POSAs would at least have found it obvious to implement Elnozahy+Draves so that the file system is part of the OS kernel (*supra* §V.D.1.c.ii ([1B.2])). POSAs would further at minimum have found it obvious to implement Elnozahy’s above-discussed file-retrieval feature, which is an *SLCSE* in the above-discussed obvious implementations of Elnozahy+Draves, to use a “kernel call” (Elnozahy, [0020]), which POSAs understood was a *system call*, to invoke the file system to read a file in the case when the desired file is not in the web server’s file cache. Using system calls was the customary way to invoke kernel services, and POSAs would have reasonably expected success in using this customary technique. Bhattacharjee, ¶183; EX1018, 463. In such an implementation, the file-retrieval feature, which is an *SLCSE*, *uses system calls* to access the file system, *i.e.*, a *service in the operating system kernel*. Bhattacharjee, ¶183.

ii. Initializing interrupt handling

The ’058 patent provides an example where “a critical system element in the context of an application program,” *i.e.* an *SLCSE*, “uses system calls to access

services in the kernel module” by relying on “[i]nterrupt handling...initialized through a system call” to enable the “kernel module” to “notif[y]” the SLCSE about an “interrupt.” EX1001, 8:45-59; [expert], ¶184.

Elnozahy discloses an embodiment where the web server “replac[es] conventional interrupt driven processing with a polling architecture” where “the server periodically monitors the network interface for new requests.” Elnozahy, [0008]; Bhattacharjee, ¶185. However, Elnozahy does not *require* a polling architecture. *E.g.*, Elnozahy, claims 1, 9 (independent claims not requiring polling). Furthermore, POSAs understood that the benefits of user-mode TCP/IP processing that Elnozahy discloses, *e.g.*, reduced context switching and optimized TCP/IP processing for specific applications (Elnozahy, [0006]-[0007]), would also accrue in an interrupt-based architecture. Bhattacharjee, ¶186. Moreover, POSAs understood using interrupts versus polling involved a tradeoff; polling might reduce latency (overall time between the start and end of an operation), at the possible expense of wasting processor resources. Bhattacharjee, ¶186; EX1048, 1; EX1049, 282. Similarly, Elnozahy discloses that when polling is used, the web server is “given a high priority compared to other processes,” which POSAs recognized might not be optimal in all circumstances (*e.g.*, when the computing system on which the web server application runs also runs other applications). Elnozahy, [0017]; Bhattacharjee, ¶186.

Thus, in Elnozahy+Draves, POSAs would have been motivated to alternatively implement the TCP/IP functions/routines in the protocol library, *i.e.* the *SLCSEs* (*supra* §V.D.1.d.i ([1C.1])), to use interrupts rather than polling in cases where minimizing latency is not required or where giving higher priority to the application using the *SLCSEs* is undesirable. POSAs would have reasonably expected success because interrupt-based processing was “conventional.” Elnozahy, [0005]; Bhattacharjee, ¶¶187-188.

POSAs understood that interrupt handling was conventionally a kernel responsibility; thus, initializing interrupt handling is a *service in the OS kernel*. Bhattacharjee, ¶189; EX1044, [0042]. Thus, in the above-described Elnozahy+Draves implementation, POSAs would have understood, or at least found obvious, that the user-space TCP/IP library functions/routines, *i.e.* the *SLCSEs*, rely on the kernel to initialize interrupt handling. Furthermore, POSAs would have found it obvious to implement Elnozahy+Draves such that interrupt handling is initialized through a “kernel call” (Elnozahy, [0020]), *i.e.* a *system call*, just as the ’058 patent describes (EX1001, 8:45-59), as this is how Elnozahy discloses accessing kernel services (*supra* §V.D.4.b.i). Bhattacharjee, ¶189.

In the above-discussed Elnozahy+Draves implementation, *SLCSEs* in the user-mode protocol library *use system calls to access services in the [OS] kernel* in the same manner the ’058 patent describes. Bhattacharjee, ¶189.

5. Claim 5

The '058 patent does not define “*interface*,” but the patent describes an embodiment where a “kernel module” acts as a “*device interface*” that “enables data exchange between a user mode CSE and a device driver in kernel mode.” EX1001, 9:60-66; Bhattacharjee, ¶¶190-191. The patent then notes that “services exported for *device interface* typically include” “[i]nitialization” and “[e]stablish[ing] a channel between a CSE in user mode and a specific device.” EX1001, 9:66-10:20, Fig. 5. These examples are consistent with how POSAs understood “interface” in the context of applications and device drivers. EX1035, 279 (“interface” is “[s]oftware that enables a program to work with...another program...or with the computer’s hardware”); Bhattacharjee, ¶191.

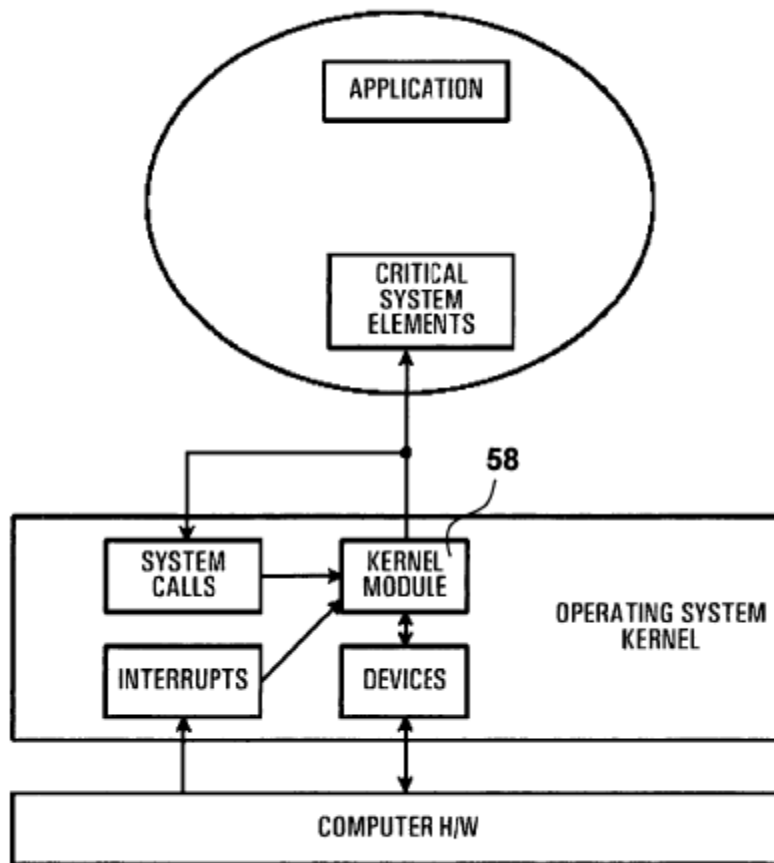


FIG. 5

EX1001, Fig. 5

Elnozahy+Draves meets claim 5 in three alternative ways, explained below.

Bhattacharjee, ¶192.

a. Initializing Communication with Network-Interface Drivers

Elnozahy discloses “kernel extension device *drivers*... configured to poll network interface 201 periodically to determine when new client requests have arrived.” Elnozahy, [0021]. “[I]f polling determines that a network packet has

been received, network processing of the packet is performed by” the “user space TCP/IP protocol library.” Elnozahy, [0028].

Elnozahy explains that the kernel “maps... the dedicated network interface to the web server” by allocating “memory” for “creating a shared buffer” accessible to both. Elnozahy, [0027]-[0028]; Bhattacharjee, ¶¶193-194. POSAs would have found it obvious to likewise have the kernel create a “shared buffer” (Elnozahy, [0027]) for communication between the TCP/IP library and the kernel extension drivers, since Elnozahy discloses this capability. Bhattacharjee, ¶194.

Furthermore, POSAs would at least have found it obvious to implement such memory allocation in a *kernel module*, *i.e.* a “set of functions that reside and execute in kernel mode as extensions to the [OS] kernel” under the agreed litigation construction (EX1001, 6:56-59; EX1058, 6), because implementing such functionality as part of a kernel extension was a known technique, within a POSA’s skill, that would have beneficially enabled Elnozahy’s described functionality. Bhattacharjee, ¶195; EX1102, 5:49-51 (“The kernel extension 30 implements...messaging queues, [and] memory pools...”); EX1099, 7:47-51; EX1100, [0053]; EX1101, [0115].

In such an obvious implementation, the *kernel module* providing the memory allocation service *serves as an interface between an SLCSE* in the “user-space protocol library” that “processes” the packet (Elnozahy, [0028]) and *a device*

driver (the kernel-extension driver), by “enabl[ing] data exchange between,” and “[e]stablish[ing] a channel between,” the SLCSE and the driver, as the ’058 patent’s describes. EX1001, 9:60-10:20; Bhattacharjee, ¶196. This SLCSE, like other SLCSEs, *runs in the context of an application* that invokes it (e.g., the web-server application). *Supra* §V.D.1.f.i ([1E.1]); Bhattacharjee, ¶196.

b. File retrieval

Elnozahy’s feature of retrieving an HTML file from either the file cache or from disk is a *SLCSE*; in the latter circumstance, the file-reading feature uses a system call to access the file system in the kernel. *Supra* §V.D.4.b.i ([4B]). POSAs understood that it was common to access physical “storage” such as a disk using a “driver.” EX1080, 1:15-17; Bhattacharjee, ¶197. Thus, POSAs would at minimum have found it obvious to implement Elnozahy+Draves so the file system uses a driver to read the file from disk; this was customary as discussed above, and thus POSAs would have reasonably expected success. Bhattacharjee, ¶197.

Elnozahy discloses that the file system is part of the kernel (*supra* §V.D.4.b.i), and Elnozahy is agnostic as to details of how this arrangement is implemented. POSAs understood it was customary to incorporate a file system into the kernel as an *extension* of the kernel, because different file systems that may each be implemented differently. Bhattacharjee, ¶198; EX1081, 5:27-31 (“Kernel extension mechanisms are well known... and are typically utilized to

dynamically load... file systems within an [OS] kernel.”); EX1082, 3:45-62; EX1083, 3:58-63. POSAs also understood that a file system is interfaced using a set of functions including functions to allow communication with the disk driver for tasks such as reading from or writing to the disk. Bhattacharjee, ¶199; EX1035, 213 (“file system” refers to “portion of an operating system that translates requests for file operations from an application...into...tasks that can be understood by the drivers”); EX1084, 3:32-37 (“The upper layers of the standard NFS [network file system] client protocol...implement the generic file system operations, such as read (), write (), open(), etc.”); EX1085, 14:4-6. Thus, POSAs would at minimum have found it obvious to implement Elnozahy’s file system interface as a kernel extension providing a set of functions for communicating with the disk driver; this was the standard implementation, as discussed above, and POSAs would thus reasonably have expected success. Bhattacharjee, ¶199.

In such an obvious implementation, the interface to the file system (which is part of the kernel) is a set of functions that reside and execute in kernel mode as extensions to the [OS] kernel, *i.e. a kernel module, that serves as an interface between an SLCSE* (the file-reading functionality) and *a device driver* (the disk driver). This SLCSE, like other SLCSEs, *runs in the context of an application* that invokes it (e.g., the web-server application). *Supra* §V.D.1.f.i ([1E.1]); Bhattacharjee, ¶200.

c. Interrupt setup

As discussed *supra* §V.D.4.b.ii, in an alternate obvious implementation of Elnozahy+Draves, the user-mode TCP/IP functions use interrupts to be notified of a packet's arrival. In such an implementation, "network processing of the packet is performed by the server's user space TCP/IP protocol library" (Elnozahy, [0028]) in response to an interrupt, rather than based on polling. Bhattacharjee, ¶201. POSAs would have found it obvious to implement interrupt handling, including setup and notification, using a *kernel module*, *i.e.* a "[a] set of functions that reside and execute in kernel mode as extensions to the [OS] kernel" (EX1001, 6:56-59; EX1058, 6), because implementing such functionality as part of a kernel extension was a known technique within a POSA's skill and would have beneficially enabled interrupt-based processing. Bhattacharjee, ¶201 EX1105, 5:38-41; EX1106, [0037].

In such an implementation, the *kernel module* providing the interrupt services *serves as an interface between an SLCSE* in the "user-space protocol library" that "processes" the packet (Elnozahy, [0028]) and *a device driver* (the kernel-extension driver) by "enabl[ing] data exchange between," and "[e]stablish[ing] a channel between," the SLCSE and the driver, as in the '058 patent's examples. EX1001, 9:60-10:20; Bhattacharjee, ¶202. This SLCSE, like

other SLCSEs, *runs in the context of an application* that invokes it (e.g., the web server application). *Supra* §V.D.1.f.i ([1E.1]); Bhattacharjee, ¶202.

6. Claim 6

a. [6A]

i. File retrieval

In Elnozahy+Draves, the “file system” in the kernel, which meets the claimed *kernel module*, is used to read a file from the disk. *Supra* §V.D.5.b (claim 5). POSAs understood that the typical and customary way to read a file from a disk involved issuing a request to the driver for the file and then waiting for an interrupt that indicates the file has been retrieved. Bhattacharjee, ¶203; EX1086, 1:12-25; EX1043, 1:12-21. Thus, in Elnozahy+Draves, POSAs would have at minimum found it obvious to implement the file system to read files by waiting for an interrupt; this was typical, as discussed above, and would have been within a POSA’s skill. Bhattacharjee, ¶204.

Furthermore, since the file system itself waits for an interrupt when reading a file, POSAs would have found it obvious to implement the file-retrieval functionality, which meets claim 5’s *SLCSE* and *runs in the context of an application program* (*supra* §V.D.5.b), to wait for *notification* of the event of the completed reading of the file *provided by* the file system. Bhattacharjee, ¶205. POSAs understood that it was logical for the file-retrieval functionality to wait for notification, given that the file system itself would be waiting for an interrupt, and

it would have been within a POSA's skill to implement such functionality using well-known programming techniques. Bhattacharjee, ¶205.

ii. Packet arrival

In the interrupt-based implementation of Elnozahy+Draves discussed *supra* §V.D.5.c, the functions/routines in the TCP/IP library, which are *SLCSEs* that *run in the context of the application program (supra §V.D.5.c)*, wait for *notification of the event of packet arrive provided by the interrupt handling kernel module*.

Bhattacharjee, ¶206.

b. [6B]

i. File retrieval

As discussed *supra* §V.D.6.a, an *event* in Elnozahy+Draves is the completion of reading the file from disk by the file system. POSAs understood that the completion of reading a file from disk is an *asynchronous event, i.e., one that may occur at any time*. Bhattacharjee, ¶207; EX1043, 1:12-17; EX1035, 38. POSAs also understood that this event *requires information to be passed to the SLCSE from outside the application*. The event requires the SLCSE to be notified that the file system has finished reading a file, and the file system is in the kernel and thus *outside* the web server application. *Supra* §V.D.4.b.i ([4B]).

Bhattacharjee, ¶208.

ii. Packet Arrival

POSAs understood that packet arrival is an *asynchronous event*, *i.e.*, one that may occur at any time. Bhattacharjee, ¶209; EX1043, 1:12-17; EX1035, 38.

POSAs also understood that this event *requires information to be passed to the user-mode TCP/IP library function, which is an SLCSE that runs in the context of an application program (supra §V.D.1.f.i ([1E.1])), from outside the application,* for two reasons: (1) the event requires the SLCSE to be notified of the packet’s arrival at the network interface, which is *outside* the application; and (2) in response to the event, the SLCSE processes the packet, which arrived from outside the application. Elnozahy, [0020]; Bhattacharjee, ¶209.

7. Claim 7

“*[N]otifying the SLCSE*” includes notifying about claim 6’s *event*.

Bhattacharjee, ¶210. In Elnozahy+Draves, an *event* of claim 6 may be (i) the competition of reading a file by the file system (*supra* §V.D.6.a.i); or (ii) the arrival of a packet (*supra* §V.D.6.a.ii). In either case, the relevant SLCSE is notified as the result of an interrupt. *Supra* §§V.D.6.b.i-V.D.6.b.ii. POSAs understood that it was customary to manage interrupts using an interrupt “*handler*” in the kernel. EX1087, 1:33-46 (discussing prior art interrupt handlers); EX1044, [0042]; Bhattacharjee, ¶211. POSAs would thus have found it obvious to implement Elnozahy+Draves to include an interrupt handler that processes

interrupts, and would reasonably have expected success in using this known technique. Bhattacharjee, ¶211.

POSAs would also have found it obvious to implement the interrupt handler to notify the relevant SLCSE—*i.e.* the file-retrieval feature (V.D.6.b.i) or a user-space TCP/IP library function (V.D.6.b.ii)—via an *upcall mechanism*, which is “[a] means by which a service in kernel mode executes a function in a user mode application context.” EX1001, 6:60-61; Bhattacharjee, ¶212. Specifically, POSAs would have implemented the interrupt handler to invoke the file-reading functionality using an upcall once the file had been read from disk, and to invoke the user-space library packet processing function using an upcall once the packet arrives. Using an upcall to inform user-mode processes about interrupts was a known technique that would have beneficially signaled the relevant SLCSE without the need to switch from kernel mode (where the interrupt handler runs) to user mode, and POSAs would have reasonably expected success using this known technique. Bhattacharjee, ¶213; EX1044, [0042]; EX1045, 26:36-53; EX1046, 14:55-59.

8. Claim 8

In an obvious implementation of Elnozahy+Draves, an interrupt handler invokes a user-mode *SLCSE* using an upcall. *Supra* §V.D.7 (claim 7). The interrupt handler in Elnozahy+Draves is part of the kernel. *Supra* §V.D.4.b.ii

([4B]), §V.D.7 (claim 7). Draves discloses, consistent with a POSA's background knowledge, that kernel-mode processes have access to all address space, including user-mode address space. Draves, 1:47-49 (“The kernel may access both the kernel system memory and the user process system memory.”); Bhattacharjee, ¶214. Thus, POSAs understood that the interrupt handler could access SLCSE instructions *resident in user mode space* and *execute those instructions* while remaining in *kernel mode*. POSAs would have been motivated to implement the interrupt handler this way because POSAs understood that having the kernel-mode interrupt handler execute the SLSCE instructions beneficially avoided the performance impact of switching to user mode. Bhattacharjee, ¶215; EX1088, [0002]; EX1098, 1:29-32.

9. Claim 9

Elnozahy+Draves's TCP/IP functions/routines in the user-mode protocol library are *CSEs* because they supply services normally supplied by the operating system. *Supra* §V.D.1.d.i ([1C.1]). Furthermore, the SLCSEs are in a user-mode library linked to the applications. *Supra* §V.D.1.f.i ([1E.1]). Thus, Elnozahy-Draves's user-mode TCP/IP functions/routines *provide one of the plurality of software applications access to operating system services*. Bhattacharjee, ¶216.

“A function overlay occurs when the implementation of a function that would normally be called, is replaced such that an extension or replacement

function is called instead.” EX1001, 8:62-64. Elnozahy+Draves’s *SLCSEs* include “functions” in the user-mode protocol library. *Supra* §V.D.1.d.i ([1C.1]). POSAs understood that these functions are “replace[ments]” for “function[s]” in the OS “that would normally be called” were the user mode library not available (EX1001, 8:62-64) because these functions were normally provided by the OS. *See supra* §V.D.1.c.ii ([1B.2]). Thus, in Elnozahy+Draves, *a function overlay is used to provide one of the plurality of software applications access to operating system services* because the sending and receiving functions normally supplied by the OS are replaced by user-mode functions. Bhattacharjee, ¶¶217-218.

10. Claim 10

During prosecution, the applicants asserted that then-pending claim 11, which became claim 10, “distinctly claims that user mode CSEs are part of an application, distinct from the microkernel design where user mode CSEs are independent applications.” EX1002, 247; Bhattacharjee, ¶219.

In Elnozahy+Draves, the *SLCSEs* are in a DLL. *Supra* §V.D.1.d.i ([1C.1]). The *SLCSEs* form a part of the applications by virtue of applications linking to the DLLs containing the *SLCSEs*; the *SLCSEs* are thus not “independent applications” of the type the applicants distinguished during prosecution. *Supra* §V.D.1.e.iii [1D.3]. Furthermore, because the *SLCSEs* are part of a dynamic *link* library, POSAs understood that they are *linked to particular software applications of the*

plurality of software applications, namely the applications using the DLL; this linking gives each application *a link that provides unique access to a unique instance of a CSE*. Bhattacharjee, ¶220; *supra* §V.D.1.f.ii ([1E.2]).

POSAs would have found it obvious to implement Elnozahy+Draves to link the DLL to applications *as the applications are loaded*. This was a standard way of linking shared libraries (as the '058 patent itself admits), and POSAs would thus have reasonably expected success in using this technique. Bhattacharjee, ¶221; EX1001, 9:18-22 (“These system elements are contained in a shared library. *As such* they are *linked* to a software application *as the application is loaded*.”); EX1047, 42:66-43:10.

11. Claim 11

As explained below, Elnozahy+Draves’s *SLCSES utilize kernel services supplied by the [OS] kernel* for the purposes recited in claim 11. Bhattacharjee, ¶222.

a. Device access

Elnozahy’s functionality of reading an HTML file either from the file cache or from disk is an *SLCSE*; when the file is to be read from disk, the file-reading functionality uses a system call to access the file system in the kernel. *Supra* §V.D.4.b.i ([4B]). POSAs understood a disk is a *device*; thus, the file-reading

functionality *utilizes a kernel service supplied by the operating system kernel, i.e. the file system, for device access.* Bhattacharjee, ¶223.

b. Interrupt Delivery

In obvious implementations of Elnozahy+Draves, *SLCSEs* are notified by the interrupt handler of interrupts that occur when the file system has finished reading a file from the disk or when a packet has arrived. *Supra* §V.D.6 (claim 6). Thus, the *SLCSEs utilize kernel services supplied by the operating system kernel for interrupt delivery.* Bhattacharjee, ¶224.

c. Virtual memory mapping

Draves discloses, consistent with a POSA's background knowledge, that virtual memory mapping between virtual and physical addresses is a standard OS kernel service. Draves, 1:48-60 (explaining in "BACKGROUND" that "The kernel is responsible for supervising the virtual memory system.... Each time a process uses a virtual memory address, the virtual memory system translates it into a physical address using a virtual-to-physical address *mapping.*"). Bhattacharjee, ¶225. Additionally, Draves discloses, consistent with a POSA's background knowledge, that different applications that linked to a DLL would have the library in their "virtual address space." Draves, 3:52-4:32, Fig. 5 (describing a prior art "sharable DLL" located in a region of physical memory mapped to a "code portion" in the "virtual address space" of two different applications);

Bhattacharjee, ¶225. Thus, POSAs would have found it obvious to implement Elnozahy+Draves so that user-mode TCP/IP functions/routines in a DLL, which are *SLCSEs* (*supra* §V.D.1.d.i ([1C.1])), rely on virtual memory mapping services supplied by the kernel—*e.g.*, for “map[ing]” the “network interface” to the “web server” (Elnozahy, [0027]-[0028]; *supra* §V.D.5.a (claim 5))—to achieve the known benefits of virtual memory, such as “isolating processes from each other” (Draves, 1:50-56); Bhattacharjee, ¶¶226-227. POSAs would reasonably have expected success since using virtual memory was well-known, as discussed above. Bhattacharjee, ¶227. In such an implementation, the *SLCSEs utilize a kernel service supplied by the operating system kernel for virtual memory mapping.* Bhattacharjee, ¶228.

12. Claim 12

Elnozahy+Draves’s TCP/IP “routines” and “functions” in the protocol library are *SLCSEs*. *Supra* §V.D.1.d.i ([1C.1]). They *include services related to TCP/IP (i.e., at least networking protocol processes)*. Bhattacharjee, ¶229.

Elnozahy+Draves also meets claim 12 if it requires *SLCSEs* to include services related to *management of files*. The ’058 patent’s specification never mentions “*management*,” but does mention “access[ing] files that reside in different locations” as an example of “File System services” that are *CSEs*. EX1001, 6:10-17. Elnozahy+Draves’s web server’s HTML file-retrieval feature,

which meets an *SLCSE* of claim 1 (as explained *supra* §V.D.4.b.ii ([4B])), reads HTML files from either cache or the disk, thus *including services related to the management of files*. Bhattacharjee, ¶230.

13. Claim 13

One obvious implementation of the user-space protocol library uses interrupts instead of polling. *Supra* §V.D.4.b.ii. In Elnozahy+Draves, POSAs would have been motivated to *modify* some TCP/IP functions/routines in the protocol library, *i.e.* *SLCSEs* (*supra* §V.D.1.d.i ([1C.1])), to use interrupts rather than polling *for a particular one of the software applications* for which minimizing latency is not required and where giving that application higher priority is not desirable. Bhattacharjee, ¶231. POSAs would have reasonably expected success because interrupt-based processing was “conventional” (Elnozahy, [0005]) and having different versions of functions/routines that accomplish the same functionality in different ways was a well-known software engineering technique within a POSA’s skill. Bhattacharjee, ¶232.

14. Claim 14

The’058 patent’s specification states that some “software *applications are provided with respective versions* of [CSEs]. ...[T]he system elements which are *application specific* reside in user mode[.]” EX1001, 3:57-60; Bhattacharjee, ¶233. As discussed *supra* §V.D.13 (claim 13), it would have been obvious to

implement SLCSEs with two versions used for different applications: (1) an interrupt-based version, and (2) a polling-based version. These different SLCSE versions are *SLCSEs that are application specific and reside in user mode* (in the user-mode library). EX1001, 3:57-60; Bhattacharjee, ¶234.

Elnozahy+Draves's kernel-mode versions of the TCP/IP functions/routines, which are OS *critical system elements* (OSCSEs), *reside in the operating system kernel*. *Supra* §V.D.1.c.ii ([1B.2]); Bhattacharjee, ¶235. POSAs also understood that these CSEs are *platform specific*. The '058 patent defines a “compute *platform*” as [t]he combination of computer hardware and a single instance of an operating system.” EX1001, 6:29-30. Elnozahy+Draves's OSCSEs are hardware- and OS instance-specific because they are part of the OS kernel running on the “server device” (Elnozahy, [0025]). *See supra* §§V.D.1.c.i-V.D.1.c.ii ([1B.1]-[1B.2]); Bhattacharjee, ¶236.

15. Claim 15

The '058 patent says that “[*a*]s a *device interface*, the kernel module enables *data exchange* between a user mode CSE and a device driver in kernel mode” using “virtual memory such that data is transferred in both directions without a copy.” EX1001, 9:60-66.

In Elnozahy+Draves, the file system, which is part of the kernel and can be used by an SLCSE to read a file from disk by invoking the disk driver, meets claim

5's kernel module. *Supra* §V.D.5.b. SLCSEs operate *in user mode*. *Supra* §V.D.1.d.ii ([1C.2]). POSAs understood that an SLCSE (such as the HTML-file-reading functionality discussed *supra* §V.D.5.b) using the file system to invoke the disk driver involves *data exchange between* the SLCSE and the driver in both directions; for example, the SLCSE conveys to the driver which file is to be read, and the driver conveys to the SLCSE when the file-reading is complete (*see supra* §V.D.6 (claim 6)). Bhattacharjee, ¶¶237-238.

Draves discloses a technique “to allow user processes to pass valid memory pointers to kernel functions.” Draves, 5:14-15. POSAs understood that using memory pointers was a way to exchange data; for example, an SLCSE passes to the file system a pointer to the memory location of a desired file's name, and the file system passes the SLCSE a pointer to the memory location of a notification that the file has been read. EX1091, 17:30-32; EX1092, 4:10-13; EX1093, [0006]. POSAs would have been motivated and reasonably expected success to implement the data exchange this way, because it was a well-known way to exchange data. Bhattacharjee, ¶¶239-240. Draves further discloses that its pointer-sharing process *uses mapping of virtual memory*. Draves, 8:9-42 (“the user virtual address space...is mapped into the system virtual address space at an offset location” and “[t]he kernel is configured to correct for this offset when receiving pointers or other memory references from a user process”), Fig. 7; Bhattacharjee, ¶¶240-241.

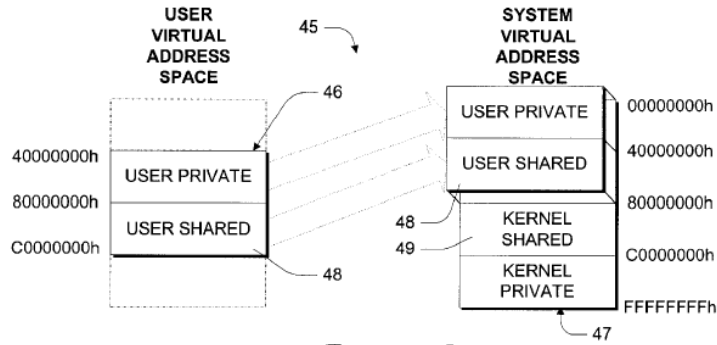


Fig. 7

Draves. Fig. 7

16. Claim 16

Elnozahy+Draves’s SLCSEs *form a part of* software applications because they are in a DLL *linked* to the applications. *Supra* §V.D.1.f.i ([1E.1]).

Bhattacharjee, ¶242.

17. Claim 17

In Elnozahy+Draves, SLCSEs *utilize kernel services supplied by the operating system kernel for device access, interrupt delivery, and virtual memory mapping*. *Supra* §V.D.11 (claim 11).

Elnozahy discloses that the “user space extensions” of, e.g., the “protocol stack...enables [the] web server to operate substantially independently of the operating system.” Elnozahy, [0025]. The “protocol stack” is provided by the “TCP/IP [protocol] library” (Elnozahy, [0022]), which contains *SLCSEs*. *Supra* §V.D.1.d.i ([1C.1]). Thus, the SLCSEs *otherwise execute without interaction from the operating system kernel*. Bhattacharjee, ¶¶243-244.

18. Claim 18

The user-mode HTML file-retrieval functionality discussed above for claim 5 is an *SLCSE*. *Supra* §V.D.5.b. Elnozahy discloses that the “protocol stack” provided by the “protocol library” enables the web server to operate “substantially independently” of the operating system, and describes one example of this independent operation as using a “user space file cache” to retrieve HTML files. Elnozahy, 5:5-21. POSAs thus understood, or at least found obvious, that there is an OS version of the functionality for reading HTML files, and that the user-mode HTML file-retrieval functionality is *not a copy* of OS version, *i.e.* an *OSCSE*, because only the user-space version (and not the *OSCSE*) would access the user-space file cache. Bhattacharjee, ¶245.

Additionally, Elnozahy says “network processing of packets” in the “user space TCP/IP protocol library” is invoked if the “polling” mechanism determines that data is available. Elnozahy, [0028]. Elnozahy says this polling mechanism is different from the interrupt-driven mechanism used in traditional TCP/IP stacks that are in the OS kernel; thus, the packet “processing” routines in the user-space protocol library are *not a copy* of the corresponding *OSCSE* routines. *See* Elnozahy, [0005], [0008]; Bhattacharjee, ¶246.

Finally, Elnozahy says the “user space protocol library” “eliminat[es]” certain “code” that might otherwise be found in a traditional TCP/IP block in the

OS kernel; thus, the *SLCSEs* in the “user space protocol library” are *not a copy* of the corresponding *OSCSEs* in the kernel’s TCP/IP block. Elnozahy, [0024]-[0025]; Bhattacharjee, ¶247.

VI. DISCRETIONARY DENIAL IS UNWARRANTED

A. §314(a)

1. *Fintiv*

“[T]he PTAB will not deny institution based on *Fintiv* if,” as here, “there is compelling evidence of unpatentability.” Director’s Interim Procedure for Discretionary Denials, 5 (June 21, 2022) (“Interim Procedure”) (discussing *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (Mar. 20, 2020) (precedential)). That should conclude the *Fintiv* analysis.

If considered, the *Fintiv* factors weigh against denial.

a. Stay Potential

This factor is currently neutral or favors institution. *Sand Revolution II, LLC, v. Cont’l Intermodal Group–Trucking*, IPR2019-01393, Paper 24, 7 (June 16, 2020) (informative). The district-court proceeding is currently temporarily stayed pending transfer. EX1089, 4-5.

b. Trial Timing

Petitioner’s motion to transfer the district-court proceeding to the Northern District of California was recently granted. EX1089. That district’s median time-to-trial is 47.9 months (EX1051 at 66), and no post-transfer trial schedule has yet

been set. Factor 2 therefore weighs against denial. *BMW of North America, LLC v. Michigan Motor Techs., LLC*, IPR2023-01224, Paper 15, 11 (Feb. 15, 2024).

c. Litigation Investment

There has been little investment in invalidity-related issues in the district-court litigation. A *Markman* hearing was canceled before the transfer decision EX1090, and “a significant portion of work remains to be done” including “fact and expert discovery and dispositive motions.” *Protect Animals With Satellites LLC v. OnPoint Sys., LLC*, IPR2021-01483, Paper 11, 14-15 (Mar. 4, 2022).

d. Issue Overlap

This Petition challenges 18 claims, whereas only seven are asserted in district court. *See* EX1020. This weighs against denial. *Markforged Inc. v. Continuous Composites*, IPR2022-00679, Paper 7, 32-33 (Oct. 25, 2022).

e. Litigation Defendants

This factor is at worst neutral. *Id.* at 33.

f. Other Circumstances

The Petition’s “strong showing on the merits” weighs against denial. Interim Procedure, 3-5.

2. Unified Re-exam

The ’058 patent is the subject of *Ex Parte* Reexamination No. 90/019,676 filed by Unified Patents, LLC (“Unified”) (the “Unified Reexam”). EX1060. Multiple Board panels have held that the discretionary-denial factors applicable when there

has been a previous **IPR petitions** on the same patent do not apply where, as here, the previous challenge was an **ex-parte reexam** by another party. *See, e.g., Tesla, Inc. v. Graphite Charging Company LLC*, IPR2024-00388, Paper 15, 6-8 (Aug. 27, 2024) (rejecting patent owner’s argument that petition should be discretionarily denied under §314(a) (or under §315(d)) because of prior reexam, collecting cases). Moreover, the Unified Reexam challenges only claims 1 and 12, while this Petition challenges all claims. EX1060, 2. Discretionary denial based on the Unified Reexam is therefore unwarranted.

3. Amazon IPR

The day before this Petition’s filing, Amazon.com, Inc. (“Amazon”) filed IPR2025-00561 challenging the ’058 patent over different prior art than this Petition. Google and Amazon have different allegedly infringing products, had no involvement in preparing each other’s IPRs, and thus have no “significant relationship.” *Ford Motor Co. v. Neo Wireless LLC*, IPR2023-00763, Paper 28, 7, 10 (Mar. 22, 2024) (director review); *Facebook, Inc. v. Express Mobile Inc.*, IPR2021-01457, Paper 10, 11-12 (Mar. 18, 2022) (no “significant relationship” where litigation defendants in “joint defense relationship” did not coordinate on IPRs); *Videndum Production Sols., Inc. v. Rotolight Ltd.*, IPR2023-01218, Paper 12, 7 (Apr. 19, 2024) (director review) (discretionary denial “not justified” where petitioners “do not have a significant relationship”). Moreover, by filing at

essentially the same time as Amazon, Google obtains no road-mapping benefit from Amazon's IPR, and is only exercising its own right to challenge the patent Google is accused of infringing. *Volkswagen Grp. of America, Inc. v. Carucel Investments, L.P.*, IPR2019-01105, Paper 8, 16-17 (noting also that “[t]he Trial Practice Guide[’s]” provisions on parallel petitions “*by a petitioner*” do not apply to petitions “filed by different petitioners”) (emphasis original). Discretionary denial based on Amazon's IPR is therefore unwarranted.

B. §325(d)

Considering the two-part framework of *Advanced Bionics, LLC v. Med-El Elektromedizinische Gerate GMBH*, IPR2019-01469, Paper 6 (Feb. 13, 2020) (precedential), no basis for denial exists under §325(d).

1. Part One

Elnozahy-966 (EX1011) (the issued-patent version of Elnozahy) was of record in prosecution, but Draves was not. That the Office has never considered this Petition's Elnozahy+Draves combination weighs heavily against discretionary denial. *Samsung Elecs. Co., Ltd. v. DoDots Licensing Soln's. LLC*, IPR2023-00756, Paper 9, 21-22 (Oct. 11, 2023) (no need to reach *Advanced Bionics* part two where only one reference of combination was of record); *Thorne Research v. Trs. of Dartmouth Coll.*, IPR2021-00491, Paper 18, 8-9 (Aug. 12, 2021) (similar). Furthermore, the Elnozahy+Draves combination is not cumulative of the

combination of Elnozahy-966 and Wong considered by the Examiner, because neither Elnozahy-966 nor Wong discloses shared libraries, whereas Draves does.

Supra §III.C.

2. Part Two

If the Board reaches *Advanced Bionics* step two because Elnozahy-966 was of record, “specific teachings...impact[ing] patentability” were “misapprehend[ed] or overlook[ed].” *Cellco P’ship v. Huawei Device Co.*, IPR2020-01117, Paper 10, 12 (Feb. 3, 2021). Furthermore, the three “*Becton, Dickson*” (“*BD*”) factors applicable to Part Two (*Advanced Bionics*, 10) weigh against denial.

Factor (c) weighs against denial because the examiner did not combine Elnozahy with Draves in a rejection.

Likewise, the examiner did not identify or explain any claim limitation that the Elnozahy+Draves combination failed to disclose. Allowing the claims was error because—as this Petition explains—Elnozahy and Draves in combination render all claims obvious. Factor (e) thus weighs against denial.

The prosecution did not include the expert testimony and evidence of a POSA’s background knowledge presented in this Petition. Thus, Factor (f) weighs against denial.

VII. CONCLUSION

The Board should institute review and cancel claims 1-18.

Dated: 2025-01-31

Respectfully submitted,
Google LLC

/Anant Saraswat/_____

Anant Saraswat, Reg. #76,050

CERTIFICATE OF SERVICE UNDER 37 C.F.R. § 42.6 (E)(4)

I certify that on January 31, 2025, I will cause a copy of the foregoing document, including any exhibits or appendices filed therewith, to be served via USPS Priority Mail Express at the following correspondence address of record for the patent:

Allen, Dyer, Doppelt + Gilchrist, PA
1135 East State Road 434
Suite 3001
Winter Springs, FL 32708

Date: January 31, 2025

/Dara Del Rosario/
Dara Del Rosario
Paralegal
WOLF, GREENFIELD & SACKS, P.C.

CERTIFICATE OF WORD COUNT

Pursuant to 37 C.F.R. § 42.24, the undersigned certifies that the foregoing Petition for *Inter Partes* Review contains 13,992 words excluding a table of contents, a table of authorities, Mandatory Notices under § 42.8, a certificate of service or word count, or appendix of exhibits or claim listing. Petitioner has relied on the word count feature of the word processing system used to create this paper in making this certification.

Date: January 31, 2025

/Dara Del Rosario/
Dara Del Rosario
Paralegal
WOLF, GREENFIELD & SACKS, P.C.

VIII. CLAIM LISTING

The following claim listing assigns element labels (e.g., [1A1], [1A2], etc.) to certain claims for clarity.

Claim 1
[1PRE] A computing system for executing a plurality of software applications comprising:
[1A] a) a processor;
[1B.1] b) an operating system having an operating system kernel...
[1B.2] having OS critical system elements (OSCSEs)...
[1B.3] for running in kernel mode using said processor; and,
[1C.1] c) a shared library having shared library critical system elements (SLCSEs) stored therein...
[1C.2] for use by the plurality of software applications in user mode and
[1D.1] i) wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and
[1D.2] are accessible to some of the plurality of software applications and
[1D.3] when one of the SLCSEs is accessed by one or more of the plurality of software applications it forms a part of the one or more of the plurality of software applications
[1E.1] ii) wherein an instance of a SLCSE provided to at least a first of the plurality of software applications from the shared library is run in a context of said at least first of the plurality of software applications without being shared with other of the plurality of software applications and
[1E.2] where at least a second of the plurality of software applications running under the operating system have use of a unique instance of a corresponding critical system element for performing same function, and
[1F] iii) wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for running a second instance of the SLCSE simultaneously.
Claim 2
A computing system as defined in claim 1, wherein in operation, multiple instances of an SLCSE stored in the shared library run simultaneously within the operating system.
Claim 3

A computing system according to claim 1 wherein OSCSEs corresponding to and capable of performing the same function as SLCSEs remain in the operating system kernel.

Claim 4

[4A] A computing system according to claim 1 wherein the one or more SLCSEs provided to one of the plurality of software applications having exclusive use thereof,

[4B] use system calls to access services in the operating system kernel.

Claim 5

A computing system according to claim 1 wherein the operating system kernel comprises a kernel module adapted to serve as an interface between an SLCSE in the context of an application program and a device driver.

Claim 6

[6A] A computing system according to claim 5 wherein the kernel module is adapted to provide a notification of an event to an SLCSE running in the context of an application program,

[6B] wherein the event is an asynchronous event and requires information to be passed to the SLCSE from outside the application.

Claim 7

A computing system according to claim 6 wherein a handler is provided for notifying the SLCSE in the context of one of the plurality of software applications through the use of an up call mechanism.

Claim 8

A computing system according to claim 7 wherein the up call mechanism in operation, executes instructions from an SLCSE resident in user mode space, in kernel mode.

Claim 9

A computing system according to claim 2, wherein a function overlay is used to provide one of the plurality of software applications access to operating system services.

Claim 10

A computing system according to claim 2 wherein SLCSEs stored in the shared library are linked to particular software applications of the plurality of software applications as the particular software applications are loaded such that the particular software applications have a link that provides unique access to a unique instance of a CSE.

Claim 11

A computing system according to claim 2 wherein the SLCSEs utilize kernel services supplied by the operating system kernel for device access, interrupt delivery, and virtual memory mapping.

Claim 12

A computing system according to claim 1, wherein SLCSEs include services related to at least one of, network protocol processes, and the management of files.

Claim 13

A computing system according to claim 10 wherein some SLCSEs are modified for a particular one of the plurality of software applications.

Claim 14

A computing system according to claim 13 wherein the SLCSEs that are application specific, reside in user mode, while critical system elements, which are platform specific, reside in the operating system kernel.

Claim 15

[15A] A computing system according to claim 5 wherein the kernel module is adapted to enable data exchange between the SLCSEs in user mode and a device driver in kernel mode,

[15B] and wherein the data exchange uses mapping of virtual memory such that data is transferred both from the SLCSEs in user mode to the device driver in kernel mode and from the device driver in kernel mode to the SLCSEs in user mode.

Claim 16

A computing system according to claim 1 wherein SLCSEs form a part of at least some of the plurality of software applications, by being linked thereto.

Claim 17

A computing system according to claim 2 wherein the SLCSEs utilize kernel services supplied by the operating system kernel for device access, interrupt delivery, and virtual memory mapping and otherwise execute without interaction from the operating system kernel.

Claim 18

A computer system as defined in claim 2 wherein SLCSEs are not copies of OSCSEs.