

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

INTERNATIONAL BUSINESS MACHINES CORP.,

Petitioner

v.

VIRTAMOVE, CORP.,

Patent Owner

Case IPR2025-00591

U.S. Patent No. 7,784,058

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

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. BACKGROUND OF THE '058 Patent.....	2
A. Claims.....	2
B. Overview of the '058 Patent.....	2
C. Prosecution History of the '058 Patent	4
III. BACKGROUND OF THE TECHNOLOGY.....	5
A. Operating Systems (OS), CPUs, Kernel Mode, and User-Mode	5
B. Applications, Threads, and Processes	6
C. Linking and Shared Libraries.....	7
IV. IDENTIFICATION AND BASIS OF CHALLENGE	8
V. OVERVIEW OF THE ASSERTED PRIOR ART	10
A. Deianov.....	10
B. Ely.....	12
C. Levine.....	14
D. Sands.....	14
VI. LEVEL OF ORDINARY SKILL IN THE ART	15
VII. CLAIM CONSTRUCTION	15
VIII. SPECIFIC GROUNDS FOR PETITION	18
A. Ground I: Claims 1-4 and 18 Are Rendered Obvious by Deianov in view of Levine and/or Sands	18
1. Motivation to Combine	18

2.	Claim 1	25
a.	1[Pre]: “A computing system for executing a plurality of software applications comprising:”	25
b.	1[a]: “a processor;”	26
c.	1[b]: “an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor; and,”	27
d.	1[c]: “a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode and”	31
e.	1[c][i]: “wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications and 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.	1[c][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 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.	1[c][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.”	42
3.	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.”	44
4.	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.”	44
5.	Claim 4: “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, use system calls to access services in the operating system kernel.”	45
6.	Claim 18: “A computer system as defined in claim 2 wherein SLCSEs are not copies of OSCSEs.”	45
B.	Ground II: Claims 1-4, 18 Are Rendered Obvious by Ely in view of Levine and/or Sands	45
1.	Motivation to Combine	46
2.	Claim 1	49
a.	1[Pre]: “A computing system for executing a plurality of software applications comprising:”	50
b.	1[a]: “a processor;”	50
c.	1[b]: “an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor; and,”	51
d.	1[c]: “a shared library having shared library critical system elements (SLCSEs) stored therein	

	for use by the plurality of software applications in user mode and”	54
e.	1[c][i]: “wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications and 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,”	56
f.	1[c][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 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”	58
g.	1[c][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.”	60
3.	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.”	62
4.	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.”	63

5.	Claim 4: “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, use system calls to access services in the operating system kernel.”.....	63
6.	Claim 18: “A computer system as defined in claim 2 wherein SLCSEs are not copies of OSCSEs.”.....	64
IX.	SECONDARY CONSIDERATIONS	65
X.	PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION.....	66
A.	Discretionary Denial Under 35 U.S.C. § 325(d) is Not Warranted	66
B.	Discretionary Denial Under 35 U.S.C. § 314(a) is Not Warranted	67
XI.	MANDATORY NOTICES	68
A.	Real Party-In-Interest	68
B.	Related Matters.....	68
C.	Counsel Information.....	69
D.	Service Information.....	70
XII.	PAYMENT OF FEES	70
XIII.	CERTIFICATION	71
XIV.	CONCLUSION.....	71

TABLE OF AUTHORITIES

Cases	Page(s)
<i>10X Genomics, Inc. v. Bio-Rad Laboratories, Inc.</i> , IRP2020-00089, Paper 8 (PTAB Apr. 27, 2020)	17
<i>Advanced Bionics, LLC v. Med-El Elektromedizinische Geräte GmbH</i> , IPR2019-01469, Paper 6 (PTAB Feb. 13, 2020).....	66
<i>Amazon.com, Inc. v. VirtaMove, Corp.</i> , IPR2025-00561	69
<i>Apple Inc. v. Fintiv, Inc.</i> , IPR2020-00019, Paper 11 (P.T.A.B. Mar. 20, 2020).....	67, 68
<i>Becton, Dickinson, & Co. v. B. Braun Melsungen AG</i> , IPR2017-01586, Paper 8 (PTAB Dec. 15, 2017)	66
<i>Bowtech Inc. v. MCP IP, LLC</i> , IPR2019-00383, Paper 14 (PTAB Aug. 6, 2019).....	66
<i>CommScope Techs. LLC v. Dali Wireless, Inc.</i> , IPR2022-01242, Paper 23 (Feb. 27, 2023)	68
<i>Fasteners for Retail, Inc. v. RTC Indus., Inc.</i> , IPR2019-00994, Paper 9 (PTAB Nov. 5, 2019).....	66
<i>Google, LLC v. VirtaMove, Corp.</i> , IPR2025-00489	69
<i>Google, LLC v. VirtaMove, Corp.</i> , IPR2025-00490	69
<i>Red Hat, Inc. v. VirtaMove, Corp.</i> , No. 3:24-cv-04740 (N.D. Cal.).....	69
<i>Sotera Wireless, Inc. v. Masimo Corp.</i> , IPR2020-01019, Paper 12 (PTAB Dec. 1, 2020)	67
<i>Thryv, Inc v. Click-To-Call Techs., LP</i> , 140 S. Ct. 1367 (2020).....	68

Petition for *Inter Partes* Review of U.S. Patent No. 7,784,058

Case IPR2025-00591

<i>VirtaMove, Corp. v. Amazon.com, Inc.</i> , No. 7:24-cv-00030 (W.D. Tex.)	68, 69
<i>VirtaMove, Corp. v. Google LLC</i> , No. 7:24-cv-00033 (W.D. Tex.)	69
<i>VirtaMove, Corp. v. Hewlett Packard Enterprise Co.</i> , 2:24-cv-00093-JRG (E.D. Tex.)	15
<i>VirtaMove, Corp. v. Hewlett Packard Enterprise Co.</i> , No. 2:24-cv-00093	68
<i>VirtaMove, Corp. v. International Business Machines Corp.</i> , 2:24-cv-00064-JRG (E.D. Tex.)	15, 67
<i>VirtaMove, Corp. v. Microsoft Corp.</i> , No. 7-24-cv-00338 (W.D. Tex.)	69
<i>VirtaMove, Corp. v. Oracle Corp.</i> , 7-24-cv-00339 (W.D. Tex.)	69
<i>Western Digital Corp. v. SPEX Techs., Inc.</i> , IPR2018-00084, Paper 14 (PTAB Apr. 25, 2018)	17
<i>Wyers v. Master Lock Co.</i> , 616 F.3d 1231 (Fed. Cir. 2010)	65

Statutes

35 U.S.C. § 102	8, 9, 67
35 U.S.C. § 103	8, 10
35 U.S.C. § 119	8
35 U.S.C. § 301(a)(2)	17
35 U.S.C. § 314(a)	67
35 U.S.C. § 315(e)(1)	71
35 U.S.C. § 325(d)	66

Rules

Rule 42.104(a).....	71
---------------------	----

Other Authorities

37 C.F.R. § 42.10(b)	70
37 C.F.R. § 42.15(a)(1).....	70

LIST OF EXHIBITS

Exhibit No.	Description
1001	U.S. Patent No. 7,784,058 (the “’058 Patent”)
1002	Certified File History of U.S. Pat. No. 7,784,058
1003	Declaration of Todd C. Mowry Ph.D. in Support of Petition for <i>Inter Partes</i> Review of U.S. Patent No. 7,784,058
1004	<i>Curriculum Vitae</i> of Todd C. Mowry, Ph.D.
1005	U.S. Patent No. 6,529,985 (“Deianov”)
1006	David Ely, Stefan Savage & David Wetherall, <i>Alpine: A User-Level Infrastructure for Network Protocol Development</i> , 3D USENIX SYMP. ON INTERNET TECHS. & SYSS. (2001) (“Ely”)
1007	John R. Levine, LINKERS & LOADERS (1999) (“Levine”)
1008	U.S. Patent No. 5,291,601 (“Sands”)
1009	Declaration of the University of Texas at Austin Libraries regarding the “Alpine: A User-Level Infrastructure for Network Protocol Development” Reference
1010	Declaration of Rachel J. Watters on Authentication of Publication
1011	ANDREW S. TANENBAUM, MODERN OPERATING SYSTEMS (2d ed. 2001)
1012	HARVEY M. DEITEL, AN INTRODUCTION TO OPERATING SYSTEMS (1983)
1013	MICROSOFT COMPUTER DICTIONARY (5th ed. 2002)
1014	TALIGENT’S GUIDE TO DESIGNING PROGRAMS: WELL-MANNERED OBJECT-ORIENTED DESIGN IN C++ (1994)
1015	ABRAHAM SILBERSCHATZ & PETER B. GALVIN, OPERATING SYSTEM CONCEPTS (4th ed. 1994)
1016	STEPHEN A. RAGO, UNIX SYSTEM V NETWORK PROGRAMMING (1993)
1017	David M. Beazley, Brian D. Ward & Ian R. Cooke, <i>The Inside Story on Shared Libraries and Dynamic Loading</i> , COMPUT. IN SCI. & ENG’G (Sept./Oct. 2001)

Petition for *Inter Partes* Review of U.S. Patent No. 7,784,058
Case IPR2025-00591

Exhibit No.	Description
1018	FreeBSD Handbook, https://download.freebsd.org/doc/en/books/handbook/handbook_en.pdf (last visited Jan. 23, 2025)
1019	WILLIAM H. MURRAY, III & CHRIS H. PAPPAS, <i>MICROSOFT C/C++ 7: THE COMPLETE REFERENCE</i> (1992)
1020	Transcript of Deposition of Donn Rochette, Sept. 10, 2024
1021	W. RICHARD STEVENS, <i>TCP/IP ILLUSTRATED, VOLUME 1: THE PROTOCOLS</i> (1994)
1022	<i>Intel® Pentium® Pro Processor 200 MHz, 256K Cache, 66 MHz FSB</i> , INTEL, https://www.intel.com/content/www/us/en/products/sku/49952/intel-pentium-pro-processor-200-mhz-256k-cache-66-mhz-fsb/specifications.html (last visited Jan. 24, 2025)
1023	Joint Claim Construction and Prehearing Statement. <i>VirtaMove, Corp. v. Hewlett Packard Enterprise Co.</i> , 2:24-cv-00093, Dkt. 124 (Jan. 10, 2025 E.D. Tex.)
1024	Appendix A: Parties’ Agreed Terms. <i>VirtaMove, Corp. v. Google, LLC</i> , 7-24-cv-00033, Dkt. 63-1 (Oct. 22, 2024 W.D. Tex.)
1025	Declaration of Leslie Hesdorfer
1026	Stefan R. Savage, <i>Protocol Design in an Uncooperative Internet</i> (2002)
1027	Jonathan Michael Salz, <i>Telsa: A Transparent, Extensible Session-Layer Framework for End-to-end Network Services</i> (2002)
1028	Plaintiff’s Third Supplemental Objections and Responses to Defendants’ First Set of Common Interrogatories (Nos. 1-9). <i>VirtaMove, Corp. v. Hewlett Packard Enterprise Co.</i> , 2:24-cv-00093 (Nov. 18, 2024 E.D. Tex.)
1029	Declaration of The University of Washington at Seattle Libraries Regarding “Protocol Design in an Uncooperative Internet” by Stefan R. Savage
1030	Declaration of Sylvia D. Hall-Ellis, Ph.D.
1031	Wayback Machine – USITS 2001 Abstract, https://web.archive.org/web/20030122034313/http://www.usenix .

Petition for *Inter Partes* Review of U.S. Patent No. 7,784,058
Case IPR2025-00591

Exhibit No.	Description
	org/publications/library/proceedings/usits01/ely.html.
1032	The PDL Packet, The Newsletter on PDL Activities and Events Fall 2022
1033	<i>Windows Processor Requirements</i> , MICROSOFT, https://learn.microsoft.com/en-us/windows- hardware/design/minimum/windows-processor-requirements (last visited Feb. 6, 2025)

International Business Machines Corp. (“IBM” or “Petitioner”) requests *inter partes* review of claims 1-4 and 18 (“Challenged Claims”) of U.S. Patent No. 7,784,058 (the “’058 Patent”).¹

I. INTRODUCTION

The ’058 Patent concerns a computing system that stores replicas of important operating system (“OS”) kernel functions—called “critical system elements” (“CSEs”) by the ’058 Patent—in a shared library in OS user-space (*i.e.*, outside of OS kernel-space), so that these CSEs can be accessed by user applications executing on the computing system. Consistent with well-known implementations of shared libraries at the time, including UNIX-based ELFs and Windows-based DLLs, while the object code for these user-space CSEs is shared by various applications, multiple instances of the CSEs can be run in parallel in different data spaces allocated to each application.

But this use of shared libraries in user-space to house CSEs accessible to multiple applications simultaneously was well known in the prior art and is disclosed or rendered obvious by the Deianov and Ely references relied upon herein, neither of which was before the Patent Office during the original

¹ All emphasis in quotations is added unless otherwise noted. All citations herein are exemplary and not exhaustive.

prosecution of the '058 Patent. Petitioner requests the Challenged Claims be cancelled as unpatentable over the prior art on the grounds detailed below.

II. BACKGROUND OF THE '058 PATENT

The '058 Patent issued on August 24, 2010, from U.S. Application No. 10/946,536 (Ex-1002, "'536 Application"), which was filed on September 21, 2004. The '536 Application claims priority to provisional application No. 60/504,213 filed September 22, 2003. For purposes of this Petition only, Petitioner assumes the '058 Patent is entitled to a priority date of September 22, 2003, but reserves the right to challenge this or any other alleged priority date.

A. Claims

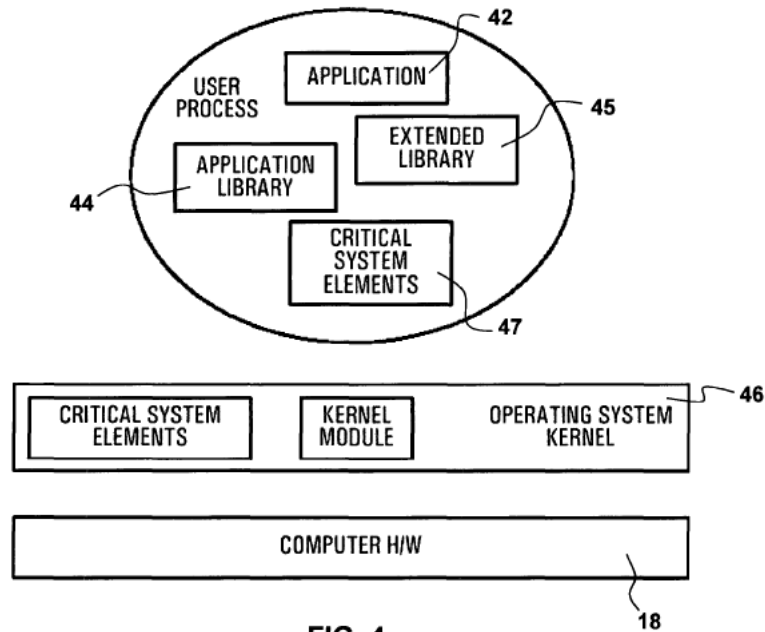
The '058 Patent has 18 claims. Claims 1-4 and 18 are challenged herein, of which claim 1 is independent. Claims 2-4 each depend from claim 1, while claim 18 depends from claim 2.

B. Overview of the '058 Patent

The '058 Patent "relates to a computing system, and to an architecture that affects and extends services exported through application libraries." Ex-1001, 1:15-18. The '058 Patent explains that "[c]omputer systems are designed in such a way that software application programs share common resources," and "[i]t is traditionally the task of an operating system to provide mechanisms to safely and effectively control access to shared resources." Ex-1001, 1:21-24. The '058 Patent notes that this centralized control of shared resources by the OS "creates a

limitation caused by conflicts for shared resources.” Ex-1001, 1:26-28. According to the ’058 Patent, this is particularly true of resources “critical to software applications, hereafter called critical system elements (CSEs).” Ex-1001, 1:25-27. CSEs include resources like networking services and file system services normally found in the OS kernel, the core part of the OS that interfaces between applications and system resources like memory and hardware devices. Ex-1001, Fig. 1, 6:6-28.

As its proposed solution, the ’058 Patent describes the replication of CSEs into shared libraries “which provides a means of attaching or linking a CSE service to an application having access to the shared library.” Ex-1001, 5:29-34. These replicated CSEs are called “shared library critical system elements” (“SLCSEs”). Ex-1001, 2:10. These SLCSEs are located outside the OS kernel, in user-space. Ex-1001, 3:19-24. An exemplary implementation of this architecture is illustrated in Figure 4 of the ’058 Patent, reproduced below.



Ex-1001, Fig. 4.

These SLCSEs (designated 47 in Figure 4) “form a part of at least some of the software applications by being linked thereto,” and “run in a context of said software application without being shared with other software applications.” Ex-1001, 2:14-21.

As discussed below, however, this architecture is taught by both Deianov and Ely, and is merely an application of dynamically linked shared libraries that were known in the art since at least the 1980s.

C. Prosecution History of the '058 Patent

The application that issued as the '058 Patent went through several rejections and amendments. In the Examiner’s rejection prior to allowance, the Examiner rejected the pending claims over the O’Rourke reference. The applicant

argued that O'Rourke failed to disclose that the SLCSEs stored in O'Rourke's shared library are functional replicas of OSCSEs. Ex-1002, 327-28. The applicant also argued O'Rourke's SLCSEs were not a part of the application as claimed. Ex-1002, 328-329.

The application was allowed on May 5, 2010, following an Examiner's Amendment. Ex-1002, 354-359. The Examiner's Amendment added limitations of a processor and a requirement that the SLCSEs related to the same predetermined function can be provided to two of a plurality of applications simultaneously. Ex-1002, 355-456.

The Deianov and Ely references discussed herein, however, disclose or render obvious all the elements allegedly missing from the prior art during the original prosecution, as discussed below.

III. BACKGROUND OF THE TECHNOLOGY

The '058 Patent implicates several areas of computer architecture, which are briefly discussed below for context.

A. Operating Systems (OS), CPUs, Kernel Mode, and User-Mode

An OS is software whose job includes providing a convenient programming abstraction and managing shared resources such as hardware devices. Ex-1011, 25; Ex-1012, 3.

Computer systems are generally controlled by a hardware component called the central processing unit (“CPU”). Ex-1013, 4; Ex-1011, 45. The CPU does this by fetching, under control of the OS, instructions from memory when executing a program (also known as an application), decoding them, and then executing them. Ex-1011, 45; Ex-1013, 3; Ex-1011, 29.

The OS and CPU typically have two modes of operation: “kernel mode” and “user mode.” Ex-1011, 46. “Kernel mode” is “[t]he context in which the kernel portion of an operating system executes.” Ex-1001, 6:32-33. The kernel is “[t]he core of an operating system—the portion of the system that manages memory, files, and peripheral devices; maintains the time and date; launches applications; and allocates system resources.” Ex-1013, 5; Ex-1012, 8-9. “User mode” is “the context in which applications execute.” Ex-1001, 6:31. User-mode generally involves execution of only a subset of OS instructions and access to a subset of hardware features. Ex-1011, 47.

B. Applications, Threads, and Processes

Because computer users often use multiple applications simultaneously, computers must provide a way for these applications to share system resources. Ex-1011, 29-30. Long before the ’058 Patent’s priority date, computer systems facilitated running multiple applications concurrently on the same computer

through various techniques, including threads and processes. Ex-1011, 95-182; Ex-1014, 4.

“Threads” are the minimal set of “register” (special memory locations) values needed to execute a particular invocation of an application. Ex-1011, 105-124. To give each application the illusion it has the machine’s memory to itself, OSs run applications in separate “processes,” where each process has its own address space. Ex-1011, 95-104. OSs maintain separate mappings of virtual addresses to physical addresses for individual processes. Ex-1011, 51, 226-229. Each process typically contains at least one thread. Ex-1011, 105. When a user-mode program executes, its process runs in user-mode. Ex-1011, 541. When programs invoke functionality requiring the OS’s kernel, the thread running the program may transition between user-mode and kernel-mode while completing the task. Ex-1011, 482.

C. Linking and Shared Libraries

Since at least the 1980s, programmers have utilized library routines when writing software. Ex-1020, 7 (26:8-16); Ex-1015, 18; Ex-1014, 3; Ex-1016, 3-44; Ex-1012, 7; Ex-1019, 3. Library routines allow applications to use common functions, like I/O operations and networking functions, without writing the code for these functions directly into the applications. Ex-1012, 7; Ex-1017, 2. Instead, these functions are written once and stored in libraries, which applications

incorporate through “linking.” Ex-1017, 1-2. Linking resolves library routine calls in an application to ensure that applications access the correct instructions for execution. Ex-1017. In allowing multiple applications to share library routines, libraries reduce redundancies and save memory. Ex-1017, 2.

Starting around the late 1980s, both UNIX and Windows began supporting “dynamic linking,” where library routines are linked to applications at runtime. Ex-1017, 2; Ex-1016, 4. Advantageously, dynamic linking allows storing a single library routine copy in physical memory, while each application runs its own library routine instance in user-space. Ex-1017, 2. Dynamic shared libraries are called ELF libraries (Executable and Linkable Format) in UNIX and DLLs (dynamic-link libraries) in Windows. Ex-1003, ¶59.

IV. IDENTIFICATION AND BASIS OF CHALLENGE

Petitioner’s challenge is based on the following prior art, none of which was considered by the Patent Office during the original prosecution of the ’058 Patent:

- **Deianov (Ex-1005)** – U.S. Patent No. 6,529,985 (“Deianov”), filed February 4, 2000 and issued on March 4, 2003, is prior art at least under 35 U.S.C. § 102(a).²

² Citations to 35 U.S.C. §§ 102, 103, and 119 are to the pre-AIA versions applicable here.

- **Ely (Ex-1006)** – *Alpine: A User-Level Infrastructure for Network Protocol Development* by David Ely, *et al.* (“Ely”), is an article that was first published on March 28, 2001. The attached Declaration of Sean O’Bryan (Ex-1009) shows that Ely was included in “Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems (USITS ’01), March 26-28, 2001, San Francisco, California, USA,” which was indexed and available to the public at the University of Texas libraries in mid-2001. Ex-1009, 1, Ex. B; *see also* Ex-1030, 16-20. Ely was also available to the public on USENIX’s website. Ex-1031. Ely is prior art under at least 35 U.S.C. § 102(b). Ely was also cited by multiple thesis papers written in 2002, confirming it was available to the authors of those papers in that timeframe. Ex-1026, 116; Ex-1027, 70; Ex-1029.
- **Levine (Ex-1007)** – *Linkers & Loaders* by John R. Levine (“Levine”), is a textbook that was first published on October 11, 1999. The attached Declaration of Rachel J. Watters shows that Levine was indexed and publicly available at the University of Wisconsin-Madison libraries as of early 2000. Ex-1010, 2. Levine is prior art at least under 35 U.S.C. § 102(b).
- **Sands (Ex-1008)** – U.S. Patent No. 5,291,601 (“Sands”), filed March 27, 1992, and issued on March 1, 1994, is prior art at least under 35 U.S.C. § 102(b).

Petitioner requests cancellation of the Challenged Claims on the grounds below:

Ground	Claims	Proposed Statutory Rejection
I	1-4 and 18	Obvious under §103 in view of Deianov, or alternatively obvious under §103 in view of Deianov in combination with Levine and/or Sands
II	1-4 and 18	Obvious under §103 in view of Ely, or alternatively obvious under §103 in view of Ely in combination with Levine and/or Sands

Section VIII details the statutory grounds for unpatentability of each of the Challenged Claims. Additional evidence is provided in the accompanying Declaration of Dr. Todd Mowry (Ex-1003) and other supporting exhibits, which are cited and discussed below. The sources of background information cited below are all dated mid-2002 or earlier. Ex-1025.

V. OVERVIEW OF THE ASSERTED PRIOR ART

A. Deianov

Deianov discloses a computer system running a multi-tasking OS that includes OSCSEs—referred to in Deianov as “system calls”. Ex-1005, 1:37-38. Deianov discloses that its “system calls” are “subroutine[s], the object code of which is located in an operating system,” and that “[e]xamples of operations executed by system calls include reading data from a file, opening a network communication channel, and allocating computer memory to a specific process.”

Ex-1005, 1:37-40. These are the same types of CSEs discussed in the '058 Patent.

Ex-1001, 6:11-28.

To improve system performance and efficiency, Deianov teaches the use of CSEs in the form of “system call wrappers.” These system call wrappers replicate the functions of system calls, but operate outside of the OS kernel in user-space and form part of an application during its execution. Ex-1003, ¶81. This is illustrated in Figure 1 of Deianov, replicated and highlighted below:

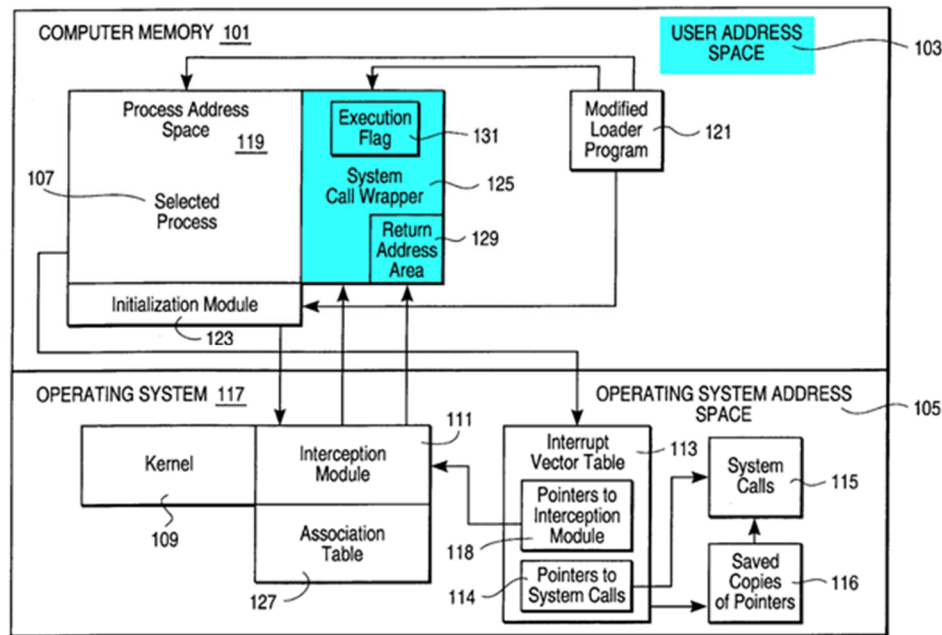


FIG. 1

Ex-1005, Fig. 1 (annotated).

These system call wrappers, which when stored in a user-space shared library are referred to as SLCSEs in the '058 Patent, are alternative object code that may be executed in place of a system call (OSCSE) and that replicate or extend the

functionality of the system call to user-space. Ex-1005, 1:64-2:18; Ex-1003, ¶80.

Deianov’s SLCSEs (system call wrappers) execute in user-space “in order to avoid the expense and other hazards associated with executing system call wrappers in operating system address space.” Ex-1005, 3:19-27, 13:7-11. Deianov further discloses that the SLCSEs are accessible to the applications running in Deianov’s systems by designating associated processes for utilizing the SLCSEs, and that during application execution, an instance of the SLCSE is unique to that application. Ex-1005, 4:37-40, 6:59-66; Ex-1003, ¶83. Figure 1 of Deianov shows that the SLCSE is in process address space.

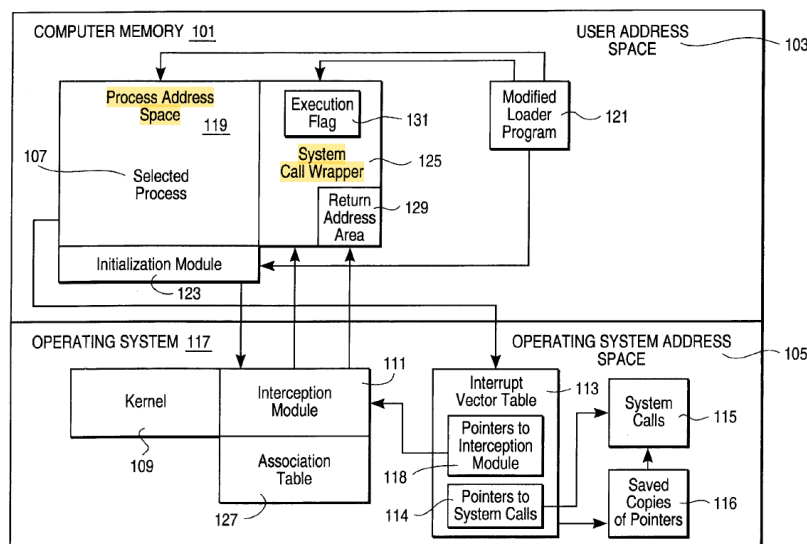


FIG. 1

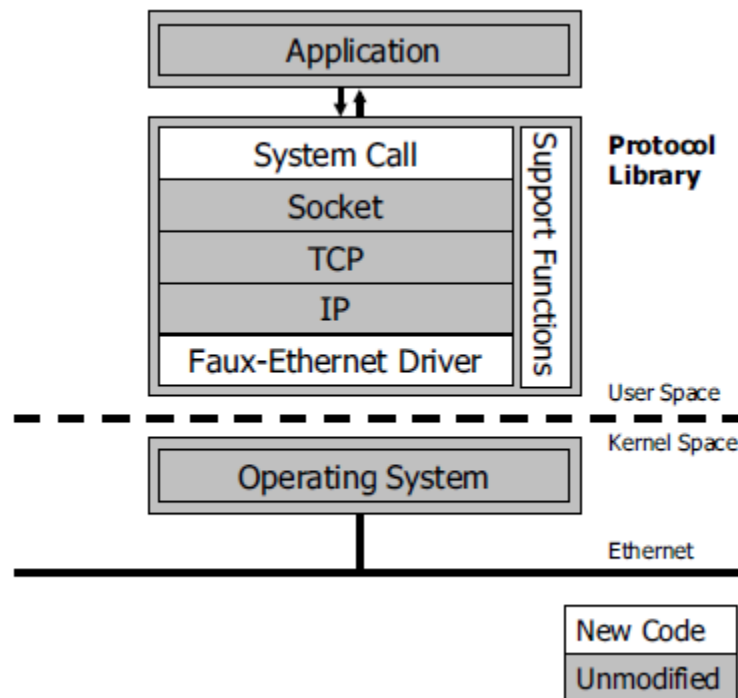
Ex-1005, Fig. 1. (annotated).

B. Ely

Ely discloses Alpine, a computer system with a “user-level networking infrastructure” that improves system operation by moving networking stack

services—identified as CSEs in the '058 Patent—from the kernel to user-space.

Ex-1006, 1; Ex-1001, 6:11-13. Networking stack services include communication protocols, like UDP, TCP, and IP, that typically reside in the kernel, which makes modification of these protocols challenging because it complicates debugging and risks crashing the system. Ex-1006, 1, Fig. 2. Accordingly, as shown in Figure 2 below, Alpine replicates these OSCSEs into a user-level networking stack, stored in a dynamic library as SLCSEs.



Ex-1006, Fig. 2.

These SLCSEs work on top of a UNIX-based OS, such as FreeBSD or Linux. Ex-1006, 1, 8, 12. Ely further discloses that these SLCSEs can be used

simultaneously by any application executing in the system, where each application runs its own instance of the SLCSE. Ex-1006, 7-8; Ex-1003, ¶90.

C. Levine

Levine is a textbook that details the use of dynamic linkers and loaders as critical tools for utilizing shared library components, like SLCSEs, stored in shared libraries in user-space. Ex-1007, 10. Levine confirms that it was known in the art well before the alleged invention of the '058 Patent that computer systems provided components like SLCSEs in “shared libraries for programs to use, so that all the programs that use a library can share a single copy of it.” Ex-1007, 18. Levine also discloses that in systems handling multiple programs simultaneously, each program has its own address space, which allows for instances of shared library components like SLCSEs to be utilized by multiple applications concurrently in their respective application spaces. Ex-1007, 52.

D. Sands

Like Levine, Sands discloses a system for linking to shared libraries, and is further evidence that the use of shared libraries was well known in the art before the '058 Patent's priority date. Ex-1008. Sands's system is capable “of running several programs concurrently,” and “each of the programs contain references to a library” of shared library components. Ex-1008, 4:12-17. Sands discloses that when a “user invokes a program,” “memory space for the program is allocated”

and “the relocatable file representing the desired program is loaded from the disk storage unit” “into the computer memory.” Ex-1008, Fig. 3A, Fig. 3B, 5:61-66. Linked components are loaded into the program’s address space at runtime, forming a part of the application. Ex-1008, Abstract; Ex-1003, ¶102. Sands teaches that components such as SLCSEs may be utilized by multiple simultaneously running programs without modifying the code of the component itself. Ex-1008, Claim 1.

VI. LEVEL OF ORDINARY SKILL IN THE ART

As of September 2003, a POSITA would have been an individual with a bachelor’s degree in computer science or a similar degree and at least two years of experience in operating systems or computer systems software. Ex-1003, ¶33. This level of skill in the art is reflected by the references cited in this Petition, the state of the art, and the opinion of Professor Todd Mowry set forth in his Declaration. *Id.*, ¶31-34. In this Petition, reference to a POSITA refers to an individual with these qualifications.

VII. CLAIM CONSTRUCTION

In the related, consolidated litigation *VirtaMove, Corp. v. Hewlett Packard Enterprise Co.*, 2:24-cv-00093-JRG (E.D. Tex.) (lead case) and *VirtaMove, Corp. v. International Business Machines Corp.*, 2:24-cv-00064-JRG (E.D. Tex.)

(member case) (the “Related IBM Litigation”), the parties have agreed to the following constructions for the ’058 Patent:

Term	Claim(s)	Agreed Construction
Preambles	1	The preambles are limiting.
processor	1	Physical computer processor.
user mode	1	The context in which applications execute.
kernel mode	1	The context in which the kernel portion of an operating system executes. Application code cannot run in kernel mode.
instance / unique instance	1	A single, specific occurrence or copy.

Ex-1023, Appendix A. The parties also agreed that for the ’058 Patent’s claims, various terms and their antecedents refer to the same entity based on the plain claim language. *See id.*

Beyond these agreed upon terms, Patent Owner (“PO”) has proposed that, except for the terms “critical system elements,” “functional replicas,” and “shared library,” all terms in the Challenged Claims be given their plain and ordinary meaning. *See* Ex-1023, Appendix C. PO has taken the position that “critical system elements” means “any service or part of a service, ‘normally’ supplied by an operating system, that is critical to the operation of a software application,”

“functional replicas” means “substantial functional equivalents or replacements of kernel functions,” and “shared library” means “an application library whose code space is shared among all user mode applications.” *Id.*, Appendix C.³

Pursuant to 35 U.S.C. § 301(a)(2), which permits Petitioner to cite to the Board “statements of the patent owner filed in a proceeding before a Federal court or the Office in which the patent owner took a position on the scope of any claim of a particular patent,” and without agreeing that Petitioner’s constructions are correct, Petitioner submits that PO’s proposed constructions in the related litigation should apply here, and applies those constructions to the prior art discussed herein. Fairness dictates that PO should be held to its claim-construction positions taken in the related litigation. *See, e.g., Western Digital Corp. v. SPEX Techs., Inc.*, IPR2018-00084, Paper 14 at 12 (PTAB Apr. 25, 2018); *10X Genomics, Inc. v. Bio-Rad Laboratories, Inc.*, IRP2020-00089, Paper 8 at 12 (PTAB Apr. 27, 2020).

Petitioner also notes that the Asserted Claims are unpatentable under its competing constructions for disputed terms, Ex-1023, Appendix C, which are based on the various lexicographical definitions identified in the ’058 Patent at 6:6-61, and applies those definitions in its analysis as well.

³ Petitioner challenges certain claim terms of the ’058 Patent in the related district court litigation as indefinite for having an unbounded scope, and nothing in this Petition is or should be construed as a waiver of that argument.

VIII. SPECIFIC GROUNDS FOR PETITION

A. Ground I: Claims 1-4 and 18 Are Rendered Obvious by Deianov in view of Levine and/or Sands

As discussed below, the Challenged Claims are obvious in view of Deianov, alone or in combination with either or both of Levine and Sands.

1. Motivation to Combine

A POSITA would have been motivated to combine Deianov with the shared library teachings of Levine and/or Sands to implement SLCSEs as components of shared libraries for simultaneous use by a plurality of applications running on Deianov's system. Ex-1003, ¶¶108-120.

Deianov teaches a multitasking system for selective interception of system calls by specified processes, enabling the execution of user-space CSEs referred to in Deianov as system call wrappers. *E.g.*, Ex-1005, 3:22-10:44. Deianov further discloses that these CSEs are stored in user-space memory available for use by all applications simultaneously. Ex-1005, Fig. 1. Therefore, in Deianov's system, CSEs reside in and are available to applications in user-mode. Ex-1005, 4:46-53. Deianov describes the benefits of utilizing CSEs in user-space, including the ability to customize operating system functionality and preventing risks associated with executing untested modified CSEs in OS address space. Ex-1005, 2:3-3:27. Dr. Mowry also explains that this architecture provides additional benefits such as increased speed and efficiency. Ex-1003, ¶¶109-119.

As Dr. Mowry explains, a POSITA reading Deianov would have understood and found obvious that Deianov's user-space CSEs are dynamically linked at runtime from a shared library to provide applications with their own private CSE instances, which was the most widely used method to provide different applications with access to commonly reused code. Ex-1003, ¶109.

Additionally, a POSITA would have been motivated to incorporate into Deianov a shared library as generally taught by Levine and Sands to house resources that can be linked to and simultaneously used by different applications at runtime, for multiple reasons. Ex-1003, ¶110. Levine and Sands share a similar architecture to Deianov in that they relate to multitasking computing systems, and both teach the use of shared libraries to provide commonly utilized files to applications simultaneously in a manner that increases system speed and efficiency. Ex-1003, ¶¶111-119. As Deianov teaches the use of shared system resources by different applications, a POSITA would have been motivated to implement into Deianov's system the use of shared libraries with dynamic runtime linking as taught by Levine and/or Sands in order to realize Deianov's goal of providing CSEs to applications as an alternative to system calls for the reasons highlighted in both Levine and Sands, including rapid linking and efficient runtime performance when used by multiple concurrently running applications. Ex-1007, 18; Ex-1008, 2:24-37; Ex-1003, ¶¶111-119. As Deianov concerns multiple

applications simultaneously executing CSEs, Ex-1005, 5:60-61, using a dynamically linked shared library as taught by Levine and/or Sands would have been an obvious choice to achieve increased system efficiency and performance. Ex-1003, ¶¶111-119.

For example, Levine discusses the implementation and operation of shared libraries and how they are linked with applications to provide unique private instances of library components. *E.g.*, Ex-1007, Preface, Chapters 1-2, 8-10. Levine discloses that in systems handling multiple application programs simultaneously, each application has its own address space which allows for unique private instances of the same library components to be utilized by multiple applications concurrently in their respective application spaces. Ex-1007, 52; Ex-1003, ¶111. The benefits of this approach as described by Levine are increased performance and the prevention of conflict between applications using the same library resource. *Id.* These benefits would be equally desirable in Deianov's system. *Id.*

Sands similarly discloses a system of linking to shared libraries where private instances of the same component can be used by different applications simultaneously. *E.g.*, Ex-1008, Abstract, 2:24-3:54, 4:9-9:49, claim 1. Sands discloses a system capable “of running several programs concurrently,” where “each of the programs contain references to a library.” Ex-1008, 4:12-17. Sands

discloses that this shared library includes library items, like routines and data, and a library symbol table. Ex-1008, 2:58-60. These routines are “commonly used by several or all of the programs in the system.” Ex-1008, 4:53-57. Sands discloses that when a “user invokes a program,” “memory space for the program is allocated” and “the relocatable file representing the desired program is loaded from the disk storage unit” “into the computer memory.” Ex-1008, Fig. 3A, Fig. 3B, 5:61-66. “Addresses of library items that are referenced in the relocatable file are found in the library symbol table, and the addresses of the library items are inserted directly into the program code of the relocatable file in computer memory.” Ex-1008, 2:64-68. The benefit of this approach as described by Sands is a more efficient and higher performance shared library system: a benefit that a POSITA would have been motivated to apply to Deianov. Ex-1008, 2:24-37; Ex-1003, ¶112.

The ’058 Patent defines shared libraries, and Petitioner agrees in its proposed construction in the Related IBM Litigation, as “[a]n 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” wherein an “application library” is “[a] collection of functions in an archive format that is combined with an application to export system elements.” Ex-1001, 6:46-53. The

'058 Patent gives a “dynamic link library” as an example of a “shared library.”

Ex-1001, 2:48-49. Levine describes ELF libraries as “dynamic shared libraries” that utilize dynamic linking. Ex-1007, 100-01. These ELF libraries are in an address space in a commonly understood archive or library format such that they “can be shared among multiple processes,” while each process may be associated with a different application. Ex-1007, 100; Ex-1003, ¶97. Further, Sands discloses shared libraries which are linked to applications and loaded at runtime. Ex-1008, Abstract. This is how dynamic linking works, confirming that the shared libraries in Sands are dynamic link libraries. Section III.C. Like Levine, the shared libraries and files therein disclosed in Sands are in an address space in a commonly understand archive or library format such that they are available to “several or all of the programs in the system.” Ex-1008, 4:56-59. These dynamic libraries are collections of functions in an archive format which different applications understand and pull information from. Ex-1003, ¶58-59. Levine’s and Sands’s shared libraries are libraries utilized by applications whose code space is shared among all user-mode applications, since the libraries are not restricted to certain applications. Ex-1003, ¶51-59.

Given the specific need to provide commonly utilized resources to a plurality of applications for simultaneous utilization as taught by Deianov, Ex-1005, 5:55-62, 9:31-37, but without a detailed disclosure of a mechanism to do so,

it would have been obvious to a POSITA that shared libraries like those disclosed in Levine and Sands were ideal for this purpose and would provide Deianov with the benefits of shared libraries with components simultaneously available to multiple applications. Ex-1003, ¶¶111-119. Levine and Sands both teach that a shared library allows applications to simultaneously access shared library resources, and using a shared library in Deianov would allow Deianov's system call wrapper to be effectively utilized by a plurality of applications. Ex-1008, 4:12-16; claim 1; Ex-1007, 52, 100. A POSITA would have understood that storing system call wrappers in a shared library would efficiently share versions of system call wrappers among the applications that utilize them, while also using a well-known and commonly implemented technology. Ex-1003, ¶¶115-119.

Thus, modifying Deianov's invention to include the use of shared libraries as taught and espoused by Levine and/or Sands would merely be a combination of prior art elements according to known methods. Further, as Dr. Mowry explains, because the use of dynamic libraries as taught by Levine and Sands was well known in the art and could be realized through commonly known and understood programming techniques that required no special hardware or OS modifications, the result of that combination would have been predictable. Ex-1003, ¶¶115-119. For these same reasons, this combination would be use of a known technique to improve similar devices in the same way, and also the application of a known

technique to improve a known device ready for improvement to yield predictable results. Ex-1003, ¶¶115-119. Use of dynamic shared libraries as one of a known, finite number of data structures would also have been obvious try for these same reasons, including because Deianov does not disclose any restrictions for the storage of its system call wrappers, shared libraries are designed for the storage of code like system call wrappers, and the use of shared libraries for providing instances of system call wrappers to a plurality of applications was a known technique at the time with well-known benefits. Ex-1016, 3-4; Ex-1003, ¶¶111-119. The use of a shared library with dynamic linking, as disclosed by Levine and Sands, would have been an obvious choice to implement Deianov's goal, as dynamic shared libraries conserve resources and allow for multiple applications to simultaneously access code stored therein without conflict or fault. Ex-1003, ¶¶51-59; Ex-1017, 2. A POSITA would also have had a reasonable expectation of success in this implementation because, as Dr. Mowry explains, to implement a shared library in Deianov, a programmer would simply need to write and compile code with a shared library flag and use a linker command to combine the desired files into a shared library, without needing any hardware modification. Ex-1005, 9:31-38; Ex-1003, ¶119. This would only require routine computer programming skills that were well-known to a POSITA at the time. Ex-1003, ¶¶31-34.

2. Claim 1

Deianov alone or in view of Levine and/or Sands renders claim 1 obvious.

Ex-1003, ¶¶122-155.

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

Deianov discloses or renders obvious the preamble. Ex-1003, ¶¶123-125.

First, Deianov discloses or renders obvious a computing system. Deianov explains that its invention pertains to a “multitasking operating system” and that “[m]ultitasking operating systems ... are widely utilized in commercial *computing systems*.” Ex-1005, 1:8-16. Moreover, Figure 1 discloses “computer memory” and an “operating system” in the preferred embodiment of the invention.

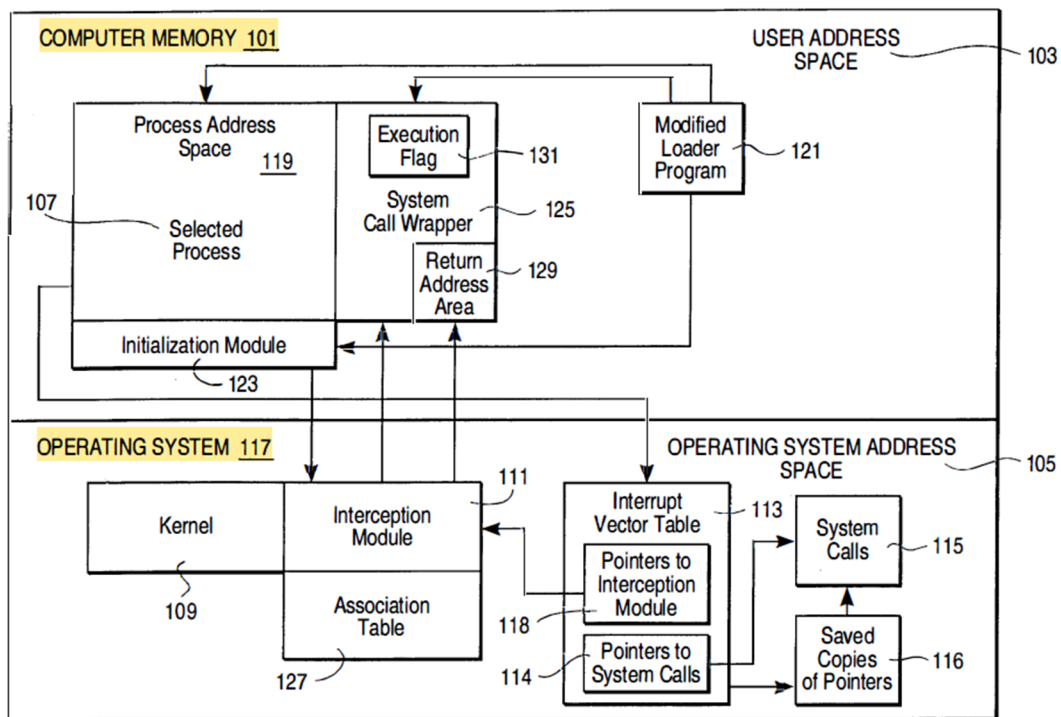


FIG. 1

Id., Fig. 1 (annotated). A POSITA would understand and find obvious that Figure 1 of Deianov depicts a computing system. Ex-1003, ¶123.

Second, Deianov discloses or renders obvious a computing system in which a plurality of applications are executed. For example, Deianov discloses that “[t]he present invention relates generally to intercepting system calls made to *multitasking* operating systems,” *Id.*, 1:8-9, and that in multitasking OSs, “application programs execute.” *Id.*, 2:61-62. Deianov further discloses that a “loader program” is utilized to “execute computer programs.” *Id.*, 3:37-42. A POSITA would understand and find obvious that Deianov’s reference to “multitasking” and “computer programs” in the plural means that Deianov’s system executes more than one application program. Ex-1003, ¶125.

b. 1[a]: “a processor;”

Deianov discloses or renders obvious this limitation. Ex-1003, ¶¶126-128.

PO agreed in the Related IBM Litigation that a processor is a “physical computer processor.” Ex-1023, Appendix A. Deianov explains that “[i]n order to make a system call, arguments are programmatically loaded into specific registers of the *central processing unit* of the computer on which the operating system is executing.” Ex-1005, 1:45-48. A CPU is a physical computer processor. Ex-1003, ¶126. Deianov is a system in which system calls are made, thus disclosing that a processor is present. Ex-1005, Abstract; Ex-1003, ¶126.

Further, Deianov states that “processes execute in user address space 103 under control of the operating system.” Ex-1005, 8:12-14. Deianov further explains that “a single process can simultaneously execute multiple threads of execution.” Ex-1005, 9:31-32. A POSITA would have understood and found obvious that a computing system executing processes and threads would include a processor because these are instantiated by object code run by a processor. Ex-1003, ¶128. Moreover, Deianov discloses UNIX and Windows as examples of multitasking OSs like those implemented in Deianov. Ex-1005, 1:13-14. Both UNIX and Windows run on a processor. Ex-1003, ¶127.

c. 1[b]: “an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor; and,”

Deianov discloses or renders obvious this limitation. Ex-1003, ¶¶129-135.

First, Deianov discloses an OS having an OS kernel. Figure 1 shows an operating system 117 having a kernel 109 therein. Ex-1005, Fig. 1.

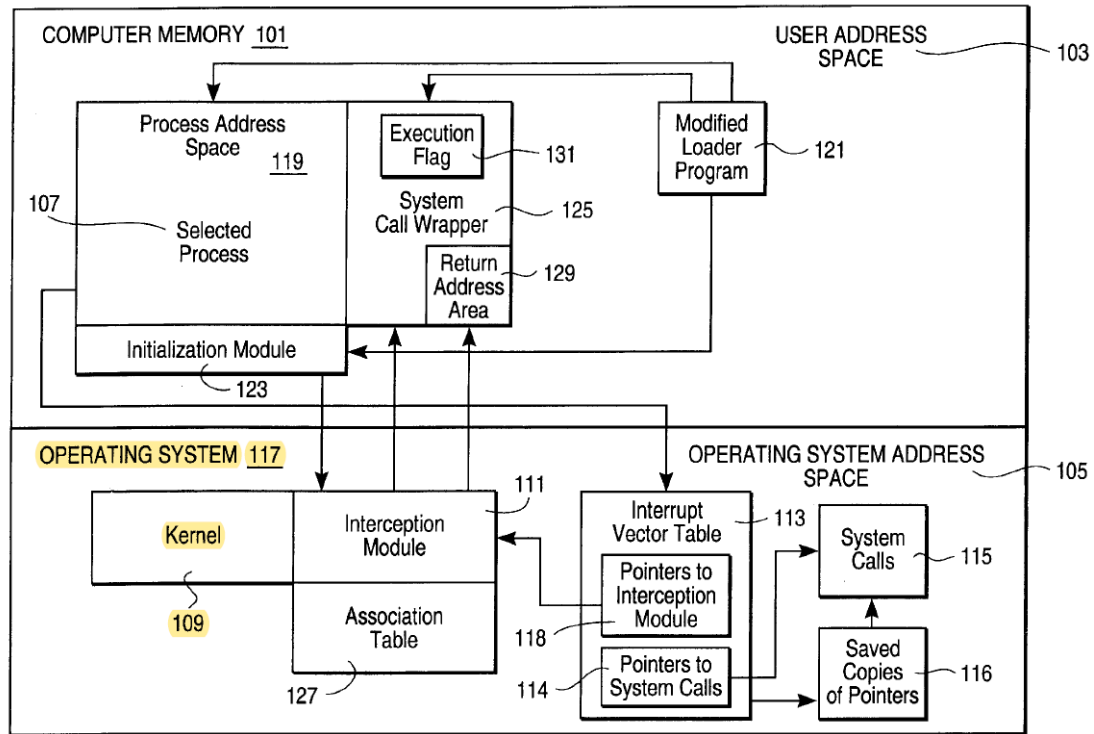


FIG. 1

Id. (annotated).

Second, Deianov discloses or renders obvious an OS kernel having OSCSEs for running in kernel-mode using the processor. The '058 Patent explains, and PO agreed in the Related IBM Litigation, that a critical system element is “[a]ny service or any part of a service, ‘normally’ supplied by an operating system, that is critical to the operation of a software application.” Ex-1001, 6:6-8; Ex-1023, Appendix C. It further explains, and PO agrees, that “kernel mode” as “[t]he context in which the kernel portion of an operating system executes ... Application code cannot run in kernel mode.” Ex-1001, 6:32-33; Ex-1023, Appendix A. As

Deianov teaches, its application code does not run in kernel-mode or in the context of an executing kernel. Ex-1005, 2:61-63; Ex-1003, ¶135.

Deianov discloses or renders obvious OSCSEs in the form of “system calls.” *E.g.*, Ex-1005, 3:30-11:45. System calls are “subroutine[s], the object code of which is *located in an operating system*” which “perform[] some system operation, such as the system hardware or software resource.” Ex-1005, 1:31-37. “Examples of operations executed by system calls include reading data from a file, opening a network communication channel, and allocating computer memory to a specific process.” Ex-1005, 1:37-40. The ’058 Patent discloses that CSEs perform functions such as “memory allocation,” which is a system function disclosed by Deianov. Ex-1001, 9:34-35. Deianov’s system calls are also located in and supplied by Deianov’s OS. Ex-1005, 1:31-32; Ex-1003, ¶¶132-134. Further, they are critical to the operation of software applications utilizing them because they provide an interface for applications to access the OS kernel and its underlying resources which user-mode applications need to function as intended. Ex-1003, ¶134; Ex-1011, 47. Therefore, Deianov’s system calls provide services normally provided by the OS, such as network services, that are critical to the operation of a software application and listed as an example of CSEs by the ’058 Patent, and thus constitute critical system elements. Ex-1005, 1:37-40; Ex-1001, 6:11-13; Ex-1003, ¶134. Deianov’s system calls are also “located in an operating system,” and

therefore they constitute OS critical system elements (OSCSEs). Ex-1005, 1:31-32.

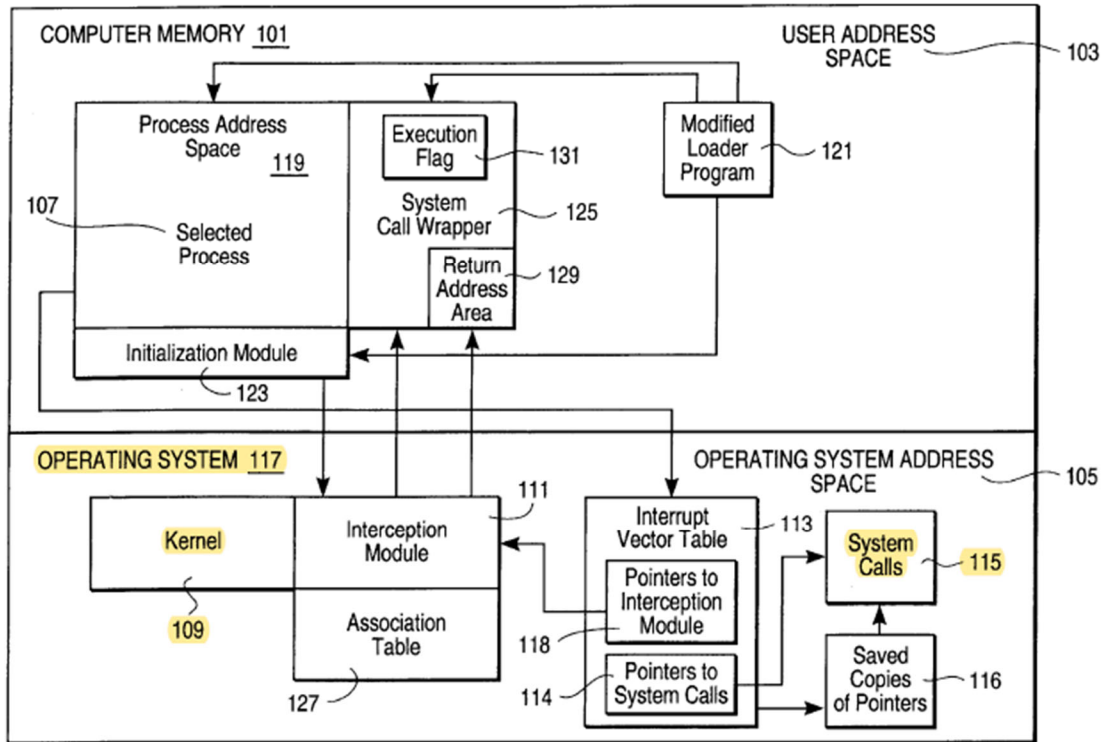


FIG. 1

Ex-1005, Fig. 1 (annotated).

Deianov also discloses an OS kernel having OSCSEs (system calls 115). Deianov discloses that “an operating system kernel 109 executes in operating system address space 105,” Ex-1005, 11:55-56, and that system calls 115 are also in OS address space. Ex-1005, Fig. 1. These system calls belong to the kernel, *i.e.*, the kernel “has” them, because the system calls provide privileged functionality that is exclusive to the kernel. Ex-1003, ¶129-131; Ex-1011, 47. Because the system calls are exclusive to the kernel, they are compiled to run in,

and do run in, kernel-mode, *i.e.*, they are executed by the kernel and therefore are run in the context in which the kernel executes. Ex-1003, ¶135. Moreover, a POSITA would understand that system calls are not application code as they are instead subroutines that reside in and run in kernel-space. Ex-1003, ¶¶131-135. A POSITA would also have understood and found obvious that the OSCSEs execute using Deianov's CPU (*i.e.*, "said processor"), since it is the CPU that executes instructions in Deianov's system. Ex-1003, ¶135; Ex-1005, 1:45-54.

d. 1[c]: "a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode and"

Deianov, alone or in combination with Levine and/or Sands, renders obvious this limitation. Ex-1003, ¶¶136-142.

First, Deianov renders obvious a shared library having shared library critical system elements (SLCSEs) stored therein. Ex-1003, ¶¶136-139. Deianov discloses the use of system call wrappers. *E.g.*, Ex-1005, 3:30-11:45. System call wrappers are alternative object code that may be executed by applications running on Deianov's system in place of system calls and which replicate the functionality of the system calls. Ex-1005, 1:64-2:18; Ex-1003, ¶136. These system call wrappers, as they perform the same critical functions as system calls (which are CSEs), are therefore also CSEs. Section VIII.A.2.c; Ex-1003, ¶139.

Deianov discloses that at runtime of an application, a system call wrapper is loaded into process address space. Ex-1005, 4:34-37. While Deianov's use of the word "load" indicates that the system call wrapper is loaded from memory, Deianov does not specifically explain where in memory it is loaded from. A POSITA reading Deianov, however, would understand and find obvious that the system call wrapper is stored in a shared memory space accessible to all applications, *i.e.*, in a library, since system calls are not restricted to any particular applications. Ex-1005, 5:49-11:49; Ex-1003, ¶137.

Additionally, a POSITA, in view of Levine and Sands, would find it obvious to store Deianov's system call wrappers in a shared library, a well-known existing technology, to allow the plurality of software applications to dynamically utilize the same system call wrappers simultaneously as taught by, and consistent with the goal of, Deianov. Ex-1003, ¶¶138-140; Ex-1016, 3-4; Ex-1017; Ex-1019, 3-4. As discussed in Section VIII.A.1. above, incorporation of dynamic libraries into Deianov would merely be a combination of prior art elements with a predictable result, would further use a known technique to improve a known device ready for improvement to yield predictable results, and would be an obvious combination to try.

Second, Deianov's SLCSEs are used by the plurality of software applications in user-mode. Deianov discloses that system call wrappers (SLCSEs)

are located in “user address space,” and that applications execute in that space. Ex-1005, Fig. 1, 5:49-11:45; Ex-1003, ¶141. Thus, Deianov’s applications execute in user-mode. The ’058 Patent describes “user mode” as the “context in which applications execute.” Ex-1001, 6:31. Deianov also teaches that system call wrappers are accessible to multiple applications. Ex-1005, 9:33-38. Levine and Sands also disclose that their shared libraries, in which SLCSEs are stored, have components that are accessible to either one or a plurality of applications. Levine discloses that “ELF shared libraries ... can be shared among multiple processes.” Ex-1007, 100. The components of these libraries are thus shared among multiple processes, and each of these processes may be associated with a different application. Ex-1003, ¶¶45-50. Sands discloses that the shared “library 24 is shared by the programs 16, 18, 20, 22 and any other programs which may be run on computer 10.” Ex-1008, 4:65-67. Applications are a form of program, and thus each of these disclosed applications may access the shared library and consequently its components, *i.e.*, SLCSEs. Ex-1003, ¶142. A POSITA implementing shared libraries in Deianov’s system would thus have found it obvious to make the SLCSEs accessible to a plurality of software applications in user-mode. Ex-1003, ¶¶141-142.

- e. **1[c][i]: “wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications and 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,”**

Deianov, alone or in combination with Levine and/or Sands, renders obvious this limitation. Ex-1003, ¶¶143-148.

First, the '058 Patent describes the term “replica” as used to “denote a CSE having similar attributes to, but not necessarily and preferably not an exact copy of a CSE in the operating system.” Ex-1001, 1:66-2:1. PO explained during prosecution that SLCSEs are “functional replicas” and “in other words, replacements” for OSCSEs. Ex-1002, 288. PO further agreed in the Related IBM Litigation that “functional replicas” means “substantial functional equivalents or replacements of kernel functions.” Ex-1023, Appendix C. Deianov’s system call wrappers are code that serve as an “alternative” to their corresponding system calls, and thus are functional replicas of the same. Ex-1005, 1:64-2:2. They perform the same functions, differing only in how that functionality is executed. Ex-1005, 1:55-2:18, 8:12-28; Ex-1003, ¶144. Deianov explains that system call wrappers serve as functional alternatives for system calls: “[w]hen system calls are made, the interception module executes and determines whether the process that made the system call is a selected process, associated with a system call wrapper.

If the process is a selected process, the system call wrapper executes. If the process is not a selected process, the system call is made.” Ex-1005, Abstract. Deianov does not restrict which processes may be selected for interception, such that the system call wrappers are available to all applications and their associated processes. *E.g.*, Ex-1005, 3:30-11:45, Ex-1003, ¶144. By executing the system call wrapper in place of the system call, the system call wrapper replaces it. *Id.* Moreover, a POSITA would have understood and found obvious that these SLCSEs stored within a shared library are accessible to all applications, including the plurality of software applications, for the same reasons discussed in Section VIII.A.2.d. Ex-1003, ¶¶136-142.

Second, Deianov alone or in view of Levine and/or Sands renders obvious that “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 application.” Ex-1003, ¶¶145-148. PO argued during prosecution that “forms a part” means that the SLCSE “reside[s] in the same address space as application code.” Ex-1002, 329. This is also explained in the ’058 Patent’s specification. Ex-1001, 2:10-15. Deianov discloses that when a system call wrapper (SLCSE) is called by a process, the system call wrapper is “loaded into their process address space.” Ex-1005, 4:32-45. A process associated with an application shares an address space, so when executed, the system file wrapper resides in the same

address space as application code. *Id.*; Ex-1003, ¶¶45-50, 145. Thus, Deianov discloses that when an SLCSE 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. Indeed, Figure 1 shows the system call wrapper residing as a part of the process.

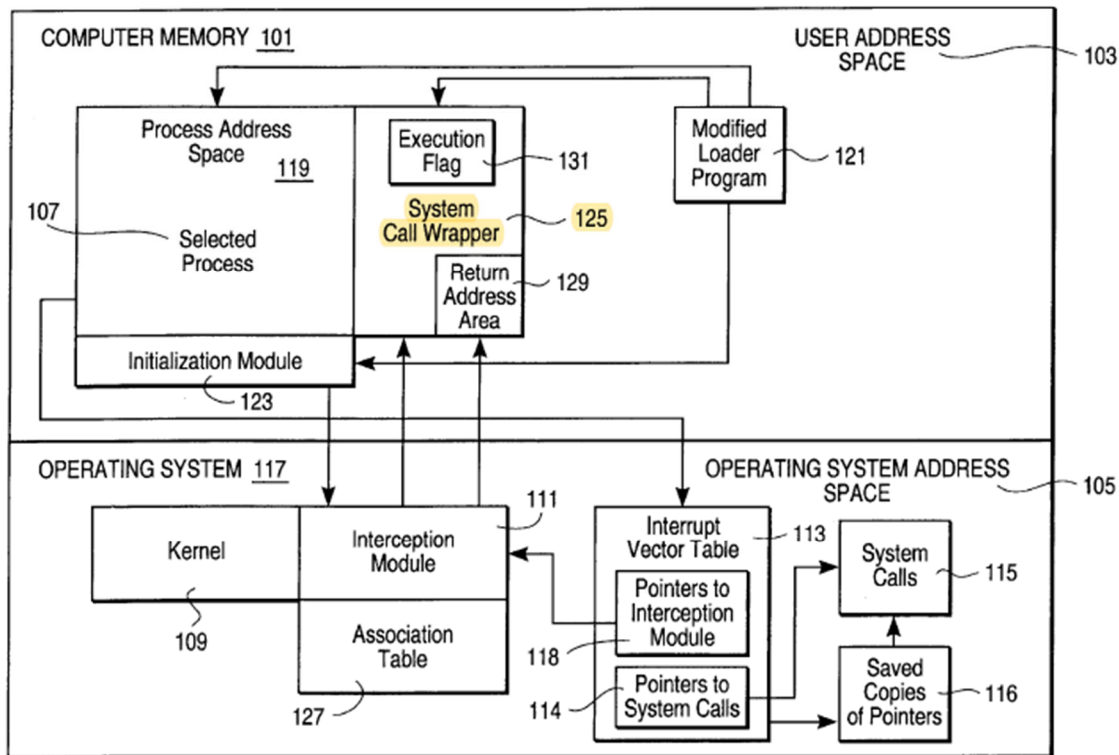


FIG. 1

Ex-1005, Fig. 1 (annotated).

Further, the '058 Patent discloses that SLCSEs may “form a part of at least some of the software applications by being linked thereto.” Ex-1001, 2:14-15. Further, in the related Google litigation, PO agreed that “[an SLCSE] forms a part of the one or more of the plurality of software applications” means “it is linked to

one or more of the plurality of software applications.” Ex-1024, 1. The components stored in Levine’s and Sands’s shared libraries, like system call wrapper SLCSEs, are accessed by and linked to one or more of the plurality of software applications. Ex-1003, ¶146. Levine discloses that ELF libraries contain “dynamically linked files,” and when these files are linked to a program, result in a “dynamically bound program,” *i.e.*, application. Ex-1007, 101, 106; Ex-1003, ¶146-148. Sands discloses a system in which a linker is utilized to link programs (*i.e.*, applications) to library components directly at run-time. Ex-1008, Abstract; claim 1. Through this linking, the SLCSE instance resides in the same address space as application code. Ex-1003, ¶¶146-148. Thus, a POSITA reading Levine and Sands would have found it obvious to implement a shared library in Deianov’s system that uses dynamic linking as taught by Levine and/or Sands such that an SLCSE resides in the same address space as the application using it, meeting Petitioner’s proposed construction of forming a part in the Related IBM Litigation, as their components “literally form a part of the application such that it resides in the same address space as application code, in contrast to a proxy that is exclusive of the application.” Ex-1023, Appendix C; Ex-1003, ¶146-148.

A POSITA would have been motivated to incorporate Levine’s and/or Sands’s teachings regarding shared libraries into Deianov’s system for the reasons discussed in Section VIII.A.1. Indeed, it was well known that shared libraries

allow for a commonly utilized file therein to be accessed by a plurality of applications simultaneously, increasing system efficiency and performance. Section VIII.A.2.d.; Ex-1012, 7 Ex-1017, 1-2. Shared libraries allow this utilization to occur through linking applications to files stored within the shared library. Section III.C.

- f. **1[c][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 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”**

Deianov alone, or in view of Levine and/or Sands, renders obvious this limitation. Ex-1003, ¶¶149-152.

Deianov teaches that “the system call wrapper 125 [SLCSE] executes in the process address space 119 of the selected process 107. Therefore, the system call wrapper 125 *is actually a part of the process 107.*” Ex-1005, 8:34-37. Each process is associated with its respective application. Ex-1003, ¶¶45-50. Therefore, by being a part of the process associated with the application, a POSITA would understand and find obvious that Deianov’s SLCSE runs in the context of that process’s application. Ex-1005, 5:49-10:17; Ex-1003, ¶149. Further, each process address space is separate from other process address spaces, such that the SLCSE

executing in the process address space of a particular process is not shared with other processes. *Id.*; Ex-1003, ¶150; Ex-1011, 95-104. The process address space is part of the associated application's address space. Ex-1003, ¶150. Deianov further discloses that "multiple processes 107 can execute simultaneously." Ex-1005, 5:60-61. Because each process may be associated with a unique application, multiple applications may execute simultaneously. Ex-1003, ¶151. By loading the SLCSE into the application's address space, each application may access and utilize a unique instance of SLCSE with the same function. Ex-1003, ¶¶150-151. Indeed, different instances of the same SLCSE have the same function because they are the same SLCSE. Ex-1003, ¶151. Moreover, Deianov teaches that all processes are run "under control of the operating system," such that, because processes are associated with applications, the plurality of applications are running under the OS. Ex-1005, 8:12-14; Ex-1003, ¶151.

This limitation is also obvious further in view of Levine and/or Sands, which teach that the best and most efficient implementation of a shared library allows for simultaneous use of instances of the same SLCSE by different applications running under the system. For example, Levine teaches that ELF libraries utilize "position independent code" (PIC). Ex-1007, 100. "The idea [behind PIC] is simple, separate the code from the data and generate code that won't change regardless of the address at which it's loaded. That way the code can be shared among all

processes, with only data pages being private to each process.” Ex-1007, 62.

When an ELF component (*i.e.*, SLCSE) linked to a process is run, that SLCSE is loaded into the process. Ex-1007, 103-06. The ’058 Patent explains “[i]n accordance with this invention, the code shared is by all applications ... However, this shared code does not imply that the CSE itself is shared ... Each application has its own unique data space. This indivisible data space ensures that CSEs are unique to an application ... In this manner CSEs are not shared among applications even though the code is shared.” Ex-1001, 3:30-44. These “CSEs” refer to SLCSEs. Ex-1001, 3:16-44. Each process disclosed by Levine may be associated with a different application. Ex-1003, ¶¶45-50. Therefore, with the data private to each process, each application associated with a process has an instance of that SLCSE loaded in its private address space to run in the context of that application without data from the execution of that SLCSE being shared with other of the plurality of software applications which have different processes from the other applications. Ex-1007, 62; Ex-1003, ¶151. A POSITA would have understood that each of these instances of the same service performs the same function as the other instances because they are instances of the same code which a set function. Ex-1003, ¶151. Deianov also discloses that all processes are run “under control of the operating system,” such that, because processes are associated with applications, the plurality of applications are running under the OS.

Ex-1005, 8:12-14; Ex-1003, ¶151. Therefore, it would have been obvious to a POSITA that a first of the plurality of applications may have an instance of an SLCSE provided to run in the context of that application without being shared with other of the plurality while a second application running under the OS has use of a unique instance of the CSE for performing the same function.

Further, Sands discloses that, using the disclosed system of linking, concurrently running applications may be utilized such that an “invoked computer program is ready for execution and such that said executable library routines remain unmodified during execution of the invoked computer program and are available for use by other computer programs in said computer system.” Ex-1008, 4:12-14, claim 1. In Sands, one copy of a shared library containing library components is maintained in memory. Ex-1008, 4:67-68. Each application has its own memory space allocation, and the addresses of the shared library components “are inserted directly into the [application] code” such that each application has an instance of the library component (SLCSE) in its memory space. Ex-1008, 2:64-68; 5:24-66. Accordingly, a first of the plurality of applications may have an instance of an SLCSE provided to run in the context of that application without being shared with other of the plurality while a second application running under the OS has use of a unique instance of the CSE for performing the same function.

As previously discussed, a POSITA would be motivated to incorporate Levine's and Sands's shared library teachings into Deianov's system. Ex-1003, ¶¶109-121, 152; Sections VIII.A.1, VIII.A.2.d. Additionally, a POSITA would have been motivated to utilize Levine and Sands concurrently running SLCSE instances so that Deianov's system call wrappers accessed concurrently by multiple applications would not result in resource competition. Ex-1003, ¶¶109-121, 152; Ex-1005, 5:55-63.

- g. **1[c][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.”**

Deianov alone, or in view of Levine and/or Sands, renders obvious this limitation. Ex-1003, ¶¶153-155.

Deianov discloses that system calls have predetermined functions such as opening network communication channels and allocating computer memory, and that system call wrappers are utilized to implement that functionality in an alternative manner. Ex-1005, 1:34-39, 1:64-2:2; Ex-1003, ¶153. When stored in shared libraries, these system call wrappers constitute SLCSEs related to a predetermined function. Section VIII.A.2.d.; Ex-1003, ¶153. Deianov alone or in view of Levine and/or Sands also renders obvious that an SLCSE may be provided

to a first of the plurality of software applications for running that first instance of the SLCSE while an SLCSE for performing the same function may be provided to a second of the plurality of software applications for running the second instance of the SLCSE, as discussed above. Section VIII.A.2.f. Indeed, this is the entire purpose of using dynamic libraries as taught by Levine and Sands. Ex-1003, ¶¶109-121, 155.

Deianov also discloses that “multiple processes 107 can execute simultaneously.” Ex-1005, 5:60-61. Each process may be associated with a unique application, thus multiple applications may execute simultaneously. Ex-1003, ¶154. It would be obvious to a POSITA that in Deianov’s system, when an instance of an SLCSE (system call wrapper) is provided to an application while another instance of that SLCSE is simultaneously provided to another application, the multiple instances of an SLCSE stored in a shared library run simultaneously. Ex-1005, 5:60-61; Section VIII.A.2.f.; Ex-1003, ¶154.

This limitation would also be obvious in view of Levine and/or Sands, which both teach the use of shared libraries for simultaneous execution of applications and library components such as SLCSEs, for all the same reasons discussed in the Motivation to Combine Section and Section VIII.A.2.f. Sections VIII.A.1., VIII.A.2.f. As previously discussed, a POSITA would be motivated to incorporate shared libraries given the benefits of the use of shared libraries as taught by Levine

and Sands for all the reasons discussed in the Motivation to Combine Section and Section VIII.A.2.f. Ex-1003, ¶¶109-121, 155; Sections VIII.A.1, VIII.A.2.f.

3. 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 1 is obvious, as is claim 2. Section VIII.A.2; Ex-1003, ¶156. Petitioner explains above regarding element 1[c][iii] how Deianov alone or in combination with Levine/Sands renders obvious multiple instances of an SLCSE running simultaneously. Section VIII.A.2.g. These simultaneously running SCLSEs also run within the operating system. User-mode or kernel-mode both run “within the operating system,” as both are OS modes. Ex-1003, ¶156; Ex-1001, 6:31-36; Ex-1005, 2:44-66. Because system calls, which are critical to application operations, are CSEs, Ex-1001, 6:6-8, these user-mode system call wrappers when stored in shared libraries constitute SLCSEs, and when multiple simultaneously executing processes run those SLCSEs, multiple instances of the SLCSE are run simultaneously within the OS. Section VIII.A.2.d, VIII.A.2.g.; Ex-1003, ¶156.

4. 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 1 is obvious, as is claim 3. Section VIII.A.2; Ex-1003, ¶157. Deianov teaches that OSCSEs (system calls) remain in the OS kernel to be utilized

by processes which are not designated for system call interception. Ex-1005, 6:39-45; Section VIII.A.1.c.; Ex-1003, ¶157. Therefore, OSCSEs corresponding to and capable of performing the same function as SLCSEs remain in the OS kernel.

5. Claim 4: “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, use system calls to access services in the operating system kernel.”

Claim 1 is obvious, as is claim 4. Section VIII.A.2; Ex-1003, ¶158. Deianov discloses that “[i]t is commonly desirable for the system call wrapper 125 to utilize OS resources. Like any process, the system call wrapper 125 utilizes such resources by making a system call 115.” Ex-1005, 8:49-52. Deianov’s OS resources include the operations utilized through making system calls, which are services provided by Deianov’s OS kernel. Ex-1003, ¶58; Ex-1005, 8:49-53.

6. Claim 18: “A computer system as defined in claim 2 wherein SLCSEs are not copies of OSCSEs.”

Claim 2 is obvious, as is claim 18. Section VIII.A.3; Ex-1003, ¶159. Deianov discloses that system call wrappers are “*alternative* object code” to system calls, such that they are different from and not copies of the code. Ex-1005, 1:64-2:2; Ex-1003, ¶159.

B. Ground II: Claims 1-4, 18 Are Rendered Obvious by Ely in view of Levine and/or Sands

As discussed below, the Challenged Claims are obvious in view of Ely, alone or in combination with either or both of Levine and Sands.

1. Motivation to Combine

To the extent PO argues that multiple instances of library components running simultaneously is not disclosed or rendered obvious by Ely, a POSITA would have been motivated to include this functionality in Ely as taught by Levine and/or Sands.

Ely discloses Alpine, a computing system that places a networking stack — which is a set of OSCSEs per the '058 Patent—into a user-level shared library of SLCSEs to facilitate networking stack development. Ex-1001, 6:11-13; *e.g.*, Ex-1006, 1-12. Ely discloses that this shared library is a “dynamic library” linked to applications. Ex-1006, 7. Ely does not disclose all the details of its dynamic library implementation, but does disclose the importance of Alpine’s performance so that it may replicate OSCSE behavior in user-space—including the ability to run multiple applications and instances of the same CSE simultaneously—“in order to be a useful tool” for testing OSCSE modifications. Ex-1006, 1, 8. Given this express goal of Ely, a POSITA would have been motivated to modify Alpine to include dynamic libraries supporting simultaneously executing library components as disclosed by Levine and Sands because these library systems provide increased performance. Ex-1003, ¶¶161-173. Further, Dr. Mowry explains that support for simultaneously executing library components would have the additional benefit of preventing simultaneously executing applications from competing for the same

library component. Ex-1003, ¶¶165-167. Such competition can lead to application malfunction, such as when simultaneously executing applications must utilize networking stack protocols at the same time. *Id.*

As previously discussed, Levine and Sands both teach the use of a shared dynamically linked library for common user-space functions like SLCSEs. Section VIII.A.1. Levine discloses the implementation of dynamically linked shared libraries having SLCSEs therein utilized by simultaneously running applications where the applications each have use of unique instance of a corresponding SLCSE that is not shared amongst said applications. *Id.* Sands similarly discloses a system of linking applications to shared libraries having SLCSEs where multiple simultaneously running applications may each utilize a unique instance of a corresponding SLCSE for performing the same function concurrently without affecting the utilization of said SLCSEs by each concurrently running application. *Id.*

As Dr. Mowry explains, a POSITA would have found it obvious to incorporate Levine's and Sands's teachings regarding implementing dynamic shared libraries into Ely to realize Ely's goal of providing a tool able to mimic OSCSE capabilities to meet applications' functional requirements, but also to improve Ely's system by providing simultaneously running unique shared library component instances as taught by Levine and Sands. Ex-1003, ¶¶167-173. Ely

emphasizes that “Alpine must have reasonable performance in order to be a useful tool,” with Ely’s evaluation of Alpine’s performance being based on the efficiency of utilization of resources by multiple concurrently running applications. Ex-1006, 8-11. Given this concern, the resource utilization features of the dynamic library employed are critical to Alpine’s performance, because library functionality dictates the ability of applications to efficiently utilize critical shared code. Ex-1003, ¶168. Ely’s desire for higher performance with Alpine would have motivated a POSITA implementing Ely to look to references like Levine and Sands, which explain how to implement shared libraries supporting simultaneously executing applications to increase overall system performance. Section VIII.A.1.; Ex-1003, ¶¶161-173.

Additionally, Ely, Levine and Sands share similar architectures—they all have dynamic shared libraries—and are in the similar field of shared library computer systems. Ex-1003, ¶¶170-172. Given the closeness of this art and their compatibility, a POSITA would have been motivated to incorporate Levine’s and Sands’s teachings of multiple, private simultaneously running shared library components in user-space to improve Alpine’s performance such that it sufficiently mimics OSCSEs. *Id.* As Dr. Mowry explains, implementing such functionality in Ely would be trivial to a POSITA, and would involve only minor modifications to

Ely's existing dynamic shared library source code, as exemplified by modifications discussed by Ely itself. Ex-1006, 9-11; Ex-1003, ¶¶170-173.

This would amount to applying a known technique to a known system ready for improvement to yield predictable results. Ex-1003, ¶172. For these same reasons, modification of Ely to include multiple simultaneously running shared library components would also be combining prior art elements according to known methods to yield predictable results, Ex-1003, ¶170, and use of a known technique to improve similar devices in the same way, Ex-1003, ¶172.

Incorporating Levine's and Sands's teachings regarding dynamic linked libraries would also have been obvious to try. Ely either allows for simultaneous use of SLCSEs or it does not. Given Levine's and Sands's teaching of the benefit of simultaneity, simultaneity would have been obvious to try in Ely's system for the reasons previously discussed. A POSITA would have had a reasonable expectation of success in doing so given that implementation of these teachings involves well known and routine computer programming techniques with no hardware changes required. Ex-1003, ¶¶170-173. Nor would OS or application changes be required. *Id.*; Ex-1007; Ex-1008.

2. Claim 1

Ely in view of Levine and/or Sands renders obvious claim 1. Ex-1003, ¶¶174-200.

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

Ely discloses or renders obvious the preamble. Ex-1003, ¶¶175-178.

First, Ely discloses or renders obvious a computing system. Ely discloses that Alpine may be implemented on UNIX-based OSs including FreeBSD and Linux. Ex-1006, 1, 8, 12. Further, Alpine may be utilized on “200MHz Pentium-Pro PCs” running the FreeBSD OS. Ex-1006, 7-8. “PCs,” or “personal computers,” are a type of computing system. Ex-1003, ¶176. Moreover, Ely discloses a “hardware interface.” Ex-1006, 4. A POSITA would have understood that the combination of computer hardware and an OS constitutes a computing system because that combination provides an operable computer. Ex-1003, ¶175.

Second, Ely discloses or renders obvious a computing system in which a plurality of applications are executed. Ely discloses that “*applications* us[e] Alpine,” referencing plural applications. Ex-1006, 3. Further, Ely discloses using dynamic linking for applications. Ex-1006, 7. A POSITA would understand dynamic linking is used to delay binding of function calls until an application *executes*. *Id.*; Ex-1003, ¶178. Therefore, Ely discloses or renders obvious the preamble.

b. 1[a]: “a processor;”

Ely discloses or renders obvious this limitation. Ex-1003, ¶¶179-181.

Ely explains Alpine utilizes a “CPU.” Ex-1006, 5. A CPU (“central processing unit”) is a physical computer processor. Ex-1013, 4, 6. Further, Ely discloses that Alpine was tested on “200MHz Pentium-Pro PCs.” Ex-1006, 8-9. The “Pentium-Pro” is a processor, disclosing that a system Alpine is implemented on has a processor. Ex-1022. Moreover, Ely discloses processing data packets, executing functions, and executing applications. Ex-1006, 3, 5, 7. A POSITA would have understood and found obvious a computing system which processes data packets and executes functions and applications includes a processor. Ex-1003, ¶181.

c. 1[b]: “an operating system having an operating system kernel having OS critical system elements (OSCSEs) for running in kernel mode using said processor; and,”

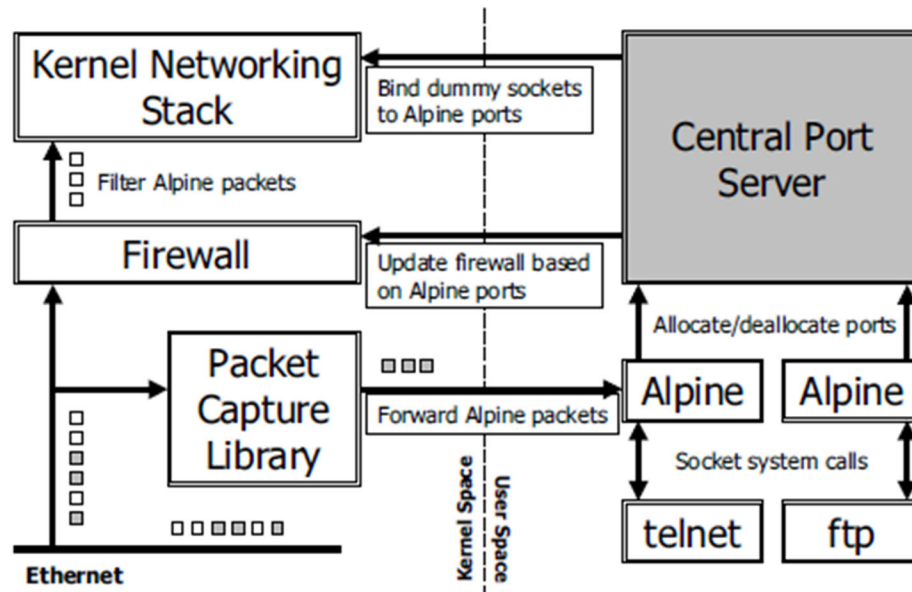
Ely discloses or renders obvious this limitation. Ex-1003, ¶¶182-186.

First, Ely discloses an OS having an OS kernel. Ely discloses that Alpine runs on a UNIX OS, which a POSITA would understand has a kernel. Ex-1006, 1; Ex-1003, ¶182. Further, Ely discloses the OS has a kernel, describing that “[r]esources that are shared between the virtualized service and *the kernel* must be kept consistent.” Ex-1006, 2.

Second, Ely discloses or renders obvious an OS kernel having OSCSEs for running in kernel-mode using the processor. The ’058 Patent defines “kernel mode” as, which PO agreed in the Related IBM Litigation, “[t]he context in which

the kernel portion of an operating system executes ... Application code cannot run in kernel mode.” Ex-1001, 6:32-33; *See* Ex-1023, Appendix A. As Ely teaches, its application code does not run in kernel-mode or in the context of an executing kernel. Ex-1006, 4-5; Ex-1003, ¶185. Critical system elements (“CSEs”) are, and PO agrees, “[a]ny service or any part of a service, ‘normally’ supplied by an operating system, that is critical to the operation of a software application.” Ex-1001, 6:6-8. An example of CSEs provided by the ’058 Patent are “[n]etwork services.” A “networking stack” is a set of networking services. Ex-1003, ¶¶60-64; Ex-1021, 3-4. The networking stack includes TCP and IP services, listed as examples of CSEs in the ’058 Patent. Ex-1001, 6:12-15; Ex-1006, 2, Fig. 1. Other networking stack services include UDP. Ex-1006, 3; Ex-1021, 8. The networking stack is normally supplied by an OS, and as Ely states, “the networking stack is traditionally part of the operating system.” Ex-1006, 1; Ex-1021, 9. It is critical to the operation of applications that require network access as the stack provides such access. Ex-1003, ¶183; Ex-1021, 7. Therefore, networking stack services disclosed by Ely, including TCP, IP, and UDP, constitute CSEs. Ex-1006, 3; Ex-1003, ¶183. As the networking stack is “traditionally part of the operating system,” network stack services further constitute OSCSEs. Ex-1006, 1; Ex-1001, 1:60-62.

Ely also discloses a “Kernel Networking Stack,” *i.e.*, a kernel having a networking stack.



Ex-1006, Fig. 3; Ex-1003, ¶184-185. Because a network stack is made up of services like TCP, IP, and UDP, Ely discloses an operating system having an operating system kernel having OS critical system elements (OSCSEs). Ex-1003, ¶185. These OSCSEs are in kernel-space because they operate inside the kernel. Ex-1006, Fig. 3. Ely discloses that “Alpine provides a [network stack] that is also provided by the kernel.” Ex-1006, 5. Resources in kernel-space run in kernel-mode, *i.e.*, with higher privileges. Ex-1003, ¶185; Ex-1011, 47; Ex-1012, 4. Moreover, a POSITA would understand that network stack protocols are kernel-resident and therefore compiled to run in, and do run in, kernel-space. Ex-1003, ¶185. Finally, a POSITA would have understood and found obvious that the

OSCSEs execute using the computing system’s processor (*i.e.*, “said processor”).

Ex-1003, ¶186. Therefore, Ely discloses this limitation or at least renders it obvious.

d. 1[c]: “a shared library having shared library critical system elements (SLCSEs) stored therein for use by the plurality of software applications in user mode and”

Ely discloses or renders obvious this limitation. Ex-1003, ¶¶187-190.

First, Ely discloses a shared library having shared library critical system elements (SLCSEs) stored therein. Ex-1003, ¶187. Ely discloses that Alpine moves “the networking stack into a user-level library.” Ex-1006, 1. Networking stack services, including TCP, IP, and UDP are CSEs. Section VIII.B.2.c. The ’058 Patent defines, as does Petitioner in the Related IBM Litigation, a shared library as “[a]n 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,” wherein application library is defined as “[a] collection of functions in an archive format that is combined with an application to export system elements.” Ex-1001, 6:46-53. Ely discloses that the library disclosed is a dynamic library linked to applications. Ex-1006, 7. Dynamic libraries are collections of functions stored in an archive format—a library format accessible to multiple applications which exports system elements to applications.

Ex-1003, ¶187; Ex-1017, 2. Ely discloses this dynamic library is at user-level, *i.e.*, code space different from the kernel. Ex-1006, 1, Fig. 2. User-mode dynamic libraries are by their nature available to and shared by multiple user-mode applications Ex-1003, ¶190; Ex-1017, 2. For example, Levine describes that UNIX dynamic shared libraries (ELF libraries)—which is what Ely discloses—“can be shared among multiple processes,” and each process may be associated with different applications. Ex-1007, 100 Ex-1003, ¶¶45-50. For the same reasons as above, a POSITA would have understood that Ely’s shared library is an application library whose code space is shared among all user-mode applications. Ex-1003, ¶187. As Alpine’s shared library contains networking stack CSEs, these services constitute SLCSEs. Ex-1003, ¶¶187-189.

Second, the disclosed SLCSEs can be used by Ely’s multiple software applications in user-mode. The ’058 Patent describes “user mode” as the “context in which applications execute.” Ex-1001, 6:31. It was known that dynamic libraries are designed to be used by a plurality of applications. Ex-1003, ¶190; Ex-1017, 2; Ex-1019, 3-4. As all applications execute in user-mode, the SLCSEs taught by Ely are accessible to a plurality of software applications in user-mode. Ely also discloses that the shared library in which the SLCSEs are stored is at user-level, *i.e.*, user-mode. Ex-1006, 1.

- e. **1[c][i]: “wherein some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications and 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,”**

Ely discloses or renders obvious this limitation. Ex-1003, ¶¶191-194.

First, Ely discloses or renders obvious that some of the SLCSEs stored in the shared library are functional replicas of OSCSEs and are accessible to some of the plurality of software applications. Alpine is designed to replace the kernel networking stack to enable testing modifications to the networking stack without the downsides of development in the kernel. Ex-1006, 1, 8. Thus, Alpine’s networking stack consists of the same OS-independent networking protocols as the kernel networking stack, such as TCP, IP, and UDP, and therefore are replacement for those functions. Ex-1006, 2-3; Ex-1021, 6. Ex-1003, ¶191. Thus, Alpine’s SLCSEs perform just like kernel-level network stack services (OSCSEs). Ex-1006, 4-5. Further, components of dynamic libraries like those disclosed by Ely are available to multiple user-mode applications, *i.e.*, some of the plurality of software applications. Section VIII.B.2.d.

Second, Ely discloses or renders obvious that 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 application. Ex-1003, ¶193. The ’058

Patent discloses that SLCSEs may “form a part of at least some of the software applications *by being linked thereto.*” Ex-1001, 2:14-15. In the related Google litigation, PO agreed that “it forms a part of the one or more of the plurality of software applications” means “*it is linked* to one or more of the plurality of software applications.” Ex-1024, 1. Ely discloses that “Alpine can be used with unmodified application binaries by exploiting dynamic *linking.*” Ex-1006, 7. Utilizing environment variables (*e.g.*, LD_PRELOAD) “allows the Alpine dynamic library to be loaded before any other library” which “enables Alpine to be used with any *dynamically linked application.*” *Id.* Applications are dynamically linking to the services within Alpine’s shared library, *i.e.*, SLCSEs, such that the SLCSEs “form a part of at least some of the software applications by being linked thereto.” Ex-1001, 2:14-15; Ex-1003, ¶193. Through this linking, the SLCSE instance resides in the same address space as application code. *Id.* Finally, it was known in the art that dynamic shared libraries, and their components, are designed to be used by a plurality of applications. Section VIII.B.2.d.; Ex-1017, 2. This is further supported by Ely disclosing that “Alpine can be used with *any dynamically linked application.*” Ex-1006, 7. For these same reasons, Ely’s shared libraries meet Petitioner’s proposed construction of forming a part in the Related IBM Litigation. Ex-1023, Appendix C; Ex-1003, ¶193.

Ely also describes an embodiment of the Alpine infrastructure implemented on the Linux OS. Ex-1006, 12. This implementation would meet this limitation, and all other limitations of the Challenged Claims, for the same reasons as it does in its embodiment implemented on FreeBSD because Alpine “relies on almost no platform specific code” and can be ported to other operating systems to perform the same function. Ex-1003, ¶194.

- f. **1[c][iii]: “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 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”**

Ely alone or in combination with Levine and/or Sands renders obvious this limitation. Ex-1003, ¶¶195-197.

Ely teaches SLCSEs stored in shared libraries made available to a plurality of software applications. Section VIII.B.2.d. A POSITA would have understood and found obvious that each application has its own address space. Ex-1003, ¶199; Ex-1011, 95-104. Ely further discloses that a unique instance of the networking stack services (SLCSEs) from the shared library is provided to each application to run in the context of that application, as “the application and the user-level networking library run in the same process.” Ex-1006, 5. A POSITA would have

understood and found obvious that for applications and SLCSEs to run in the same process, a unique SLCSE instance is loaded into an application's address space, *i.e.*, in the context of the application, when called by that application. Ex-1003, ¶196. Moreover, as Ely's Alpine system allows for multiple applications to run simultaneously, Ex-1006, 6, a POSITA would have understood and found obvious that a unique SLCSE instance can be provided to at least a first of the plurality of software applications from the shared library, while at least a second of the plurality of software applications running under the OS can have its own unique instance of that SLCSE. Ex-1003, ¶196. The reason is that, as discussed above, each application has an SLCSE instance in its address space after calling the SLCSE and loading it into application address space. *Id.* Additionally, applications in Ely run under a single OS, such as FreeBSD. *Id.*; Ex-1006, 8. Further, each instance of the SLCSE performs the same function as the other instances, since it is the same SLCSE differing only in the application address space it is loaded into. Ex-1003, ¶196. Finally, a POSITA would have understood and found obvious that each instance of the SLCSEs in an application's address space is not shared with other applications. The reason is that each application has a private address space where the instance is loaded, and therefore is not shared with others of the plurality of applications. Ex-1003, ¶196; Ex-1011, 95-104.

This limitation would also be obvious in view of Levine and/or Sands. Section VIII.B.1; Section VIII.A.2.f. As explained in Section VIII.B.1., a POSITA would have been motivated to combine Ely with Levine’s and/or Sands’s teachings of simultaneity in order to bring out performance in Alpine that replicates OSCSEs when corresponding CSEs run in user-space. Ex-1003, ¶¶161-173, 197; Section VIII.B.1. As previously discussed, this modification merely combines prior art elements according to known methods to yield predictable results. Ex-1003, ¶170; Section VIII.B.1. It also amounts to use of a known technique to improve similar devices in the same way. Ex-1003, ¶172; Section VIII.B.1. Further, it would have been obvious to try with a reasonable expectation of success. Ex-1003, ¶173; Section VIII.B.1.

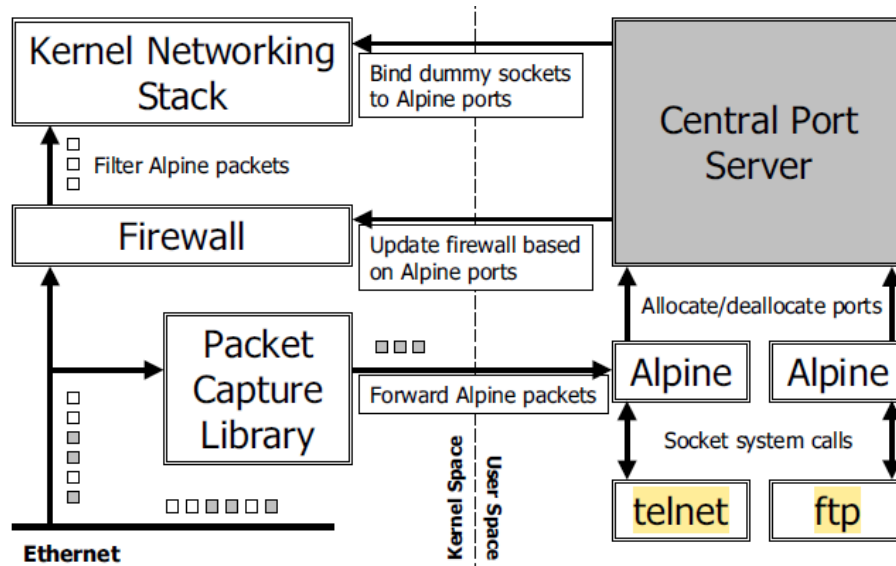
- g. **1[c][iii]: “wherein a SLCSE related to a predetermined function is provided to the first of the plurality of software applications for running a first instance of the SLCSE, and wherein a SLCSE for performing a same function is provided to the second of the plurality of software applications for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

Ely alone or in combination with Levine and/or Sands discloses or renders obvious this limitation. Ex-1003, ¶¶198-200.

Ely discloses or renders obvious SLCSEs stored in shared libraries. Section VIII.B.2.d. Further, Ely’s Alpine is a system in which multiple applications and corresponding SLCSEs may run simultaneously, exemplified by disclosing

“multiple applications try[ing] to update the firewall *concurrently*.” Ex-1006, 6.

Moreover, Figure 3 shows applications “telnet” and “ftp” accessing Alpine’s library simultaneously.



Ex-1006, Fig. 3 (annotated).

A POSITA would have understood and found obvious that each of these simultaneously running applications has its own application address space because applications always have their own address space. Ex-1003, ¶199; Ex-1011, 95-104. This is also obvious in view of Levine and/or Sands, as discussed above. Sections VIII.A.2.g., VIII.B.1. Further, a POSITA would have understood and found obvious that each simultaneously running application linked to an SLCSE would have its own instance of the SLCSE provided in its respective application address spaces running simultaneously, each with the same function. Ex-1003, ¶199. Indeed, Ely explains that “the application and the user-mode networking

library run in the same process,” such that each application has an instance of the networking library (SLCSEs) running in its process. Ex-1006, 5; Ex-1003, ¶199. This is also obvious in view of Levine and/or Sands as discussed above. Sections VIII.A.2.f, VIII.B.1. Finally, a POSITA would have understood and found obvious that these instances of an SLCSE stored in a shared library are provided to each of the plurality of applications simultaneously, and further may run simultaneously, as Ely discloses a system in which applications run simultaneously and SLCSEs are part of the applications. Ex-1006, 6; Ex-1003, ¶¶198-200. This is also obvious in view of Levine and/or Sands, as discussed above. Sections VIII.A.2.f, VIII.B.1.

3. 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 1 is obvious, as is claim 2. Section VIII.B.2; Ex-1003, ¶¶201-204. As discussed above, Ely, alone or in combination with Levine and/or Sands, renders obvious multiple instances of SLCSEs running simultaneously in the OS. Section VIII.B.2.g.; Ex-1003, ¶198-200. Further, in Ely’s, Levine’s, and Sands’s systems, applications and SLCSEs run in user-mode, which is an OS mode. Sections VIII.A.3.; VIII.A.2.d; Ex-1003, ¶¶201-202. As previously explained, this additional limitation, and all other additional limitations, are also disclosed or

rendered obvious in the embodiment of Alpine implemented on Linux as Alpine performs the same regardless of the OS it is implemented on. Section VIII.B.2.e

4. **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 1 is obvious, as is claim 3. Section VIII.B.2; Ex-1003, ¶¶205-207.

Ely discloses it implements “[n]o kernel changes.” Ex-1006, 2. As the networking stack is kernel-resident, and no changes are made to the kernel by the Alpine system, the network stack services (OSCSEs) that perform the same function as SLCSEs remain in the kernel. Section VIII.B.2.c. Indeed, Ely explains one of the design constraints for Alpine is “that the kernel ... remain[s] unmodified.” Ex-1006, 4.

5. **Claim 4: “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, use system calls to access services in the operating system kernel.”**

Claim 1 is obvious, as is claim 4. Section VIII.B.2; Ex-1003, ¶¶208-209.

Ely teaches that Alpine (the shared library of SLCSEs) “make[s] a select call into the kernel to determine if there is anything to read out of a set of files.” Ex-1006, 7. The return of this call is a kernel-provided service. Ex-1003, ¶209. A POSITA would have understood the “kernel” is the OS kernel. Ex-1003, ¶¶182-186, 208-209.

6. Claim 18: “A computer system as defined in claim 2 wherein SLCSEs are not copies of OSCSEs.”

Claim 2 is obvious, as is claim 18. Section VIII.B.3; Ex-1003, ¶¶210-211.

Ely explains that Alpine is a system explicitly intended for modifying SLCSEs for testing purposes. Ex-1006, 1. Therefore, it would be obvious to a POSITA that the SLCSEs would be modified to be different from the kernel networking stack (OSCSEs). Ex-1003, ¶211. The SLCSEs are still functional replicas when used for experimentation because they perform the same function of enabling network communications. *Id.* Therefore, the components of Alpine (SLCSEs) are not copies of the kernel networking stack protocols (OSCSEs). *Id.*

Further, Ely’s Alpine shared library contains a FreeBSD network stack (SLCSEs), but also discloses that Alpine may easily be ported to Linux, which would have a Linux kernel networking stack (OSCSEs). Ex-1006, 1, 11-12; Ex-1003, ¶210. As previously explained, this Linux embodiment meets the limitations of Claims 1 and 2 for the same reasons as the FreeBSD embodiment Ely describes. Sections VIII.B.2.e, VIII.B.3. A POSITA would have understood and found obvious that the FreeBSD SLCSEs are not copies of the Linux OSCSEs in this disclosed implementation because the SLCSEs are from a different OS variation than the SLCSEs: the implementation of the network stack differs in each OS due to the differences in internal architecture and hardware compatibility requirements across differing operating systems. Ex-1003, ¶210. Therefore, the SLCSEs’ code

differs, making them not copies. Ex-1003, ¶210. Networking stack protocols specify *what* the protocols should do, not *how* to implement them, and thus the SLCSEs are still functional replicas of the OSCSEs because they perform the same function: enabling network communication. Ex-1003, ¶210; Ex-1021, 5-6, 10-12.

IX. SECONDARY CONSIDERATIONS

In a boilerplate discovery response served in the Related IBM Litigation, PO alleged that commercial success and long-felt need are secondary considerations that support non-obviousness of the Challenged Claims. Ex-1028, 11-15. In that discovery response, however, PO offers only conclusory arguments of commercial success and long-felt need with no evidence to support its allegations, including no showing of a nexus between the alleged commercial success or long-felt need and the claimed invention. *Id.* In addition, as Dr. Mowry explains, the asserted prior art demonstrates that there was no long felt need for the claimed inventions because the patented features were already well known in the prior art. Ex-1003, ¶212. Dr. Mowry, an expert in the field for 24 years is also not aware of any evidence of commercial success. Ex-1003, ¶¶34, 212. Moreover, the alleged secondary considerations cannot overcome the strong prima facie case of obviousness shown in this Petition. *See Wyers v. Master Lock Co.*, 616 F.3d 1231, 1246 (Fed. Cir. 2010).

Should PO attempt to meet its burden to show secondary considerations, Petitioner reserves the right to respond.

X. PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION

The Board should institute review of the Challenged Claims.

A. Discretionary Denial Under 35 U.S.C. § 325(d) is Not Warranted

All six *Becton, Dickinson* factors weigh in favor of institution. *Becton, Dickinson, & Co. v. B. Braun Melsungen AG*, IPR2017-01586, Paper 8 (PTAB Dec. 15, 2017); *see also Advanced Bionics, LLC v. Med-El Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 at 8 (PTAB Feb. 13, 2020). The Board has consistently “held that a reference that ‘was neither applied against the claims nor discussed by the Examiner’ does not weigh in favor of exercising [] discretion under § 325(d).” *Fasteners for Retail, Inc. v. RTC Indus., Inc.*, IPR2019-00994, Paper 9 at 7-11 (PTAB Nov. 5, 2019). Here, none of the prior art references relied upon by Petitioner were applied against the claims or discussed by the Examiner during the original prosecution. The Examiner erroneously concluded that the Challenged Claims were patentable because the Examiner did not consider the references and combinations presented herein. *Bowtech Inc. v. MCP IP, LLC*, IPR2019-00383, Paper 14 at 5 (PTAB Aug. 6, 2019) (petitioner did not need to explain how the Examiner erred “because the Examiner did not consider the combinations of the references asserted in the Petition at all”).

B. Discretionary Denial Under 35 U.S.C. § 314(a) is Not Warranted

Evaluation of the factors under *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 (P.T.A.B. Mar. 20, 2020) (precedential) and the clarifications made by the Director of the USPTO favors institution. *See* USPTO Memorandum, Interim Procedure for Discretionary Denials in AIA Post Grant Proceedings with Parallel District Court Litigation, 2 (June 21, 2022) (“Director’s Memo”).

To avoid overlap with parallel district court proceedings, and consistent with *Sotrea Wireless, Inc.*, Petitioner hereby stipulates “that, if the PTAB institutes *inter partes* review, Petitioner ‘will not pursue in [*VirtaMove, Corp. v. International Business Machines Corp.*, No. 2:24-cv-00064-JRG (E.D. Tex.) (the “Related IBM Litigation”)] the specific grounds [asserted in the *inter partes* review], or on any other ground ... that was raised or could have been reasonably raised in an IPR (i.e., any ground that could be raised under §§ 102 or 103 on the basis of prior art patent or printed publications).” *Sotera Wireless, Inc. v. Masimo Corp.*, IPR2020-01019, Paper 12 at 13-14 (PTAB Dec. 1, 2020) (precedential as to § II.A).

Other relevant factors weigh against discretionary denial under §314(a). First, it is early in the district court case. No claim construction opinions have been issued, no depositions on the merits have been taken, and no expert reports have been submitted. Second, while the Board’s final written decision may precede the trial date in the district court case, as jury selection is scheduled for

October 20, 2025, the likelihood of postponement is increased given that there are 13 cases scheduled for jury selection on that date.

Additionally, as demonstrated in Section VIII, the Petition presents “compelling” arguments for unpatentability of the Challenged Claims based on the same features and functionality described in the ’058 Patent specification. Director’s Memo at 4–5 (“the PTAB will not deny institution based on *Fintiv* if there is compelling evidence of unpatentability”); *CommScope Techs. LLC v. Dali Wireless, Inc.*, IPR2022-01242, Paper 23 at 3-5 (Feb. 27, 2023) (precedential) (evidence is compelling when it is “highly likely that the petitioner would prevail with respect to at least one challenged claim”). Thus, institution is consistent with the significant public interest against “leaving bad patents enforceable.” *Thryv, Inc v. Click-To-Call Techs., LP*, 140 S. Ct. 1367, 1374 (2020).

XI. MANDATORY NOTICES

A. Real Party-In-Interest

International Business Machines Corp. and Red Hat, Inc. are the real parties-in-interest.

B. Related Matters

The ’058 Patent is the subject of multiple parallel proceedings in the Northern District of California, Eastern District of Texas, and Western District of Texas. *VirtaMove, Corp. v. Hewlett Packard Enterprise Co.*, No. 2:24-cv-00093 (Lead Case); 2:24-cv-00064 (Member Case) (E.D. Tex.); *VirtaMove, Corp. v.*

Petition for *Inter Partes* Review of U.S. Patent No. 7,784,058

Case IPR2025-00591

Amazon.com, Inc., No. 7:24-cv-00030 (W.D. Tex); *VirtaMove, Corp. v. Google LLC*, No. 7:24-cv-00033 (W.D. Tex.); *VirtaMove, Corp. v. Microsoft Corp.*, No. 7-24-cv-00338 (W.D. Tex.); *VirtaMove, Corp. v. Oracle Corp.*, 7-24-cv-00339 (W.D. Tex.); *Red Hat, Inc. v. VirtaMove, Corp.*, No. 3:24-cv-04740 (N.D. Cal.). *VirtaMove, Corp v. Amazon.com, Inc.* and *VirtaMove, Corp. v. Google, LLC* have been ordered to be transferred to the Northern District of California. The '058 Patent is also subject to several proceedings before the Patent Office: *Amazon.com, Inc. v. VirtaMove, Corp.*, IPR2025-00561, *Google, LLC v. VirtaMove, Corp.*, IPR2025-00489 and *Google, LLC v. VirtaMove, Corp.*, IPR2025-00490. The Patent Office has also granted a request for *ex parte* reexamination of the '058 Patent from Unified Patents, LLC, with the reexamination control number 90/019,676. These actions may affect, or be affected by, decisions in this proceeding. Nothing herein should be taken as an endorsement of VirtaMove's positions in the co-pending litigations.

C. Counsel Information

Lead Counsel	Back-Up Counsel
James E. Marina, P.C. Reg. No. 41,969 jmarina@kirkland.com <u>Postal and Hand-Delivery Address:</u> KIRKLAND & ELLIS LLP 601 Lexington Ave., New York, NY 10022 Telephone: (212) 446-4877	Todd M. Friedman, P.C. Reg. No. 42,559 tfriedman@kirkland.com <u>Postal and Hand-Delivery Address:</u> KIRKLAND & ELLIS LLP 601 Lexington Ave., New York, NY 10022 Telephone: (212) 446-4786

Lead Counsel	Back-Up Counsel
	Brandon H. Brown, P.C. <i>pro hac vice</i> motion filed concurrently bhbrown@kirkland.com <u>Postal and Hand-Delivery Address:</u> KIRKLAND & ELLIS LLP 555 California St., San Francisco, CA 94104 Telephone: (415) 439-1670 Azaad O. Zimmermann Reg. No. 81,641 azaad.zimmermann@kirkland.com <u>Postal and Hand-Delivery Address:</u> KIRKLAND & ELLIS LLP 1301 Pennsylvania Ave., N.W. Washington, D.C. 20004 Telephone: (202) 389-3341

D. Service Information

IBM concurrently submits a Power of Attorney, 37 C.F.R. § 42.10(b), and consents to electronic service directed to the following email address: IBM-VirtaMoveIPR@kirkland.com.

XII. PAYMENT OF FEES

The undersigned authorizes the Office to charge the fee set forth in 37 C.F.R. § 42.15(a)(1) any additional fees that may be due in connection with this Petition to Deposit Account No. 506092.

XIII. CERTIFICATION

Petitioner certifies, pursuant to Rule 42.104(a), that the '058 Patent is available for IPR and that Petitioner is not barred or estopped from requesting an IPR of the Challenged Claims on the grounds identified in this Petition. Petitioner certifies: (1) Petitioner is not the owner of the '058 Patent; (2) Petitioner (or any real party-in-interest) has not filed a civil action challenging the validity of any claim of the '058 Patent; (3) this Petition is being filed within one year of service of a complaint against Petitioner asserting the '058 Patent; (4) estoppel provisions of 35 U.S.C. § 315(e)(1) do not prohibit this IPR; and (5) this Petition is filed after the '058 Patent was granted.

XIV. CONCLUSION

Petitioner requests institution of an IPR and cancellation of the Challenged Claims.

Dated: February 6, 2025

Respectfully submitted,

/s/ James E. Marina
James E. Marina, P.C.
Reg. No. 41,969

CERTIFICATE OF COMPLIANCE

This Petition complies with the type-volume limitations as mandated in 37 C.F.R. § 42.24, totaling 13,997 words. Counsel has relied upon the word count feature provided by Microsoft Word.

/s/ James E. Marina
James E. Marina, P.C.
Reg. No. 41,969

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

The undersigned hereby certifies that a copy of the foregoing document with accompanying exhibits, as listed in Petitioner's Exhibit List in this document, was served via overnight Federal Express directed to the attorney/agent of record for the patent at the following address:

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

A courtesy copy was also served via electronic mail on the attorneys of record listed below for the following related matter.

Reza Mirzaie (CA SBN 246953)
rmirzaie@raklaw.com
Marc A. Fenster (CA SBN 181067)
mfenster@raklaw.com
Neil A. Rubin (CA SBN 250761)
nrubin@raklaw.com
Amy E. Hayden (CA SBN 287026)
ahayden@raklaw.com
Christian W. Conkle (CA SBN 306374)
cconkle@raklaw.com
Jonathan Ma (CA SBN 312773)
jma@raklaw.com
Daniel Kolko (CA SBN 341680)
dkolko@raklaw.com
rak_virtamove@raklaw.com
RUSS AUGUST & KABAT
12424 Wilshire Boulevard, 12th Floor
Los Angeles, CA 90025
Telephone: (310) 826-7474

Qi (Peter) Tong (TX SBN 24119042)
ptong@raklaw.com
RUSS AUGUST & KABAT
4925 Greenville Ave, Suite 200
Dallas, TX 75206

Petition for *Inter Partes* Review of U.S. Patent No. 7,784,058
Case IPR2025-00591

/s/ James E. Marina
James E. Marina, P.C.
Reg. No. 41,969