

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

DROPBOX, INC.,
Petitioner,

v.

ENTANGLED MEDIA, LLC
Patent Owner.

Case No. IPR2024-00285
U.S. Patent No. 8,484,260

DECLARATION OF PATRICK D. MCDANIEL, PH.D.

U.S. PATENT NO. 8,484,260 (CLAIMS 1-8)

Contents

I.	ENGAGEMENT	1
II.	PROFESSIONAL BACKGROUND.....	1
III.	MATERIALS REVIEWED	6
IV.	DESCRIPTION OF THE RELEVANT TIMEFRAME, THE RELEVANT FIELD, AND A PERSON OF ORDINARY SKILL IN THE ART.....	7
V.	TECHNOLOGY BACKGROUND.....	7
A.	What are “File Systems”?	8
B.	The Evolution of File Systems	12
1.	From Files to Folders	12
2.	Remote Storage for File Systems.....	13
3.	Sharing Files with Peers.....	15
C.	The Fundamentals of File Systems	18
1.	The File System API.....	19
2.	Modifying File Systems	21
D.	Distributed File Systems in Peer-to-Peer Networks	24
VI.	OVERVIEW OF THE ’260 PATENT	26
VII.	OVERVIEW OF THE PRIOR ART	28
A.	Havewala	28
B.	Adams.....	32
C.	Saridakis	34
D.	Rothman	35
VIII.	CLAIM CONSTRUCTION	36
IX.	ANTICIPATION	37
X.	OBVIOUSNESS.....	37
XI.	OPINIONS REGARDING PATENTABILITY	39
XII.	MY OPINIONS AS TO THE CHALLENGED CLAIMS OF THE ’260 PATENT.....	41

A.	My Opinions as to Claims 1, 4-8 concerning Havewala in view of Adams	41
1.	Claim 1	41
a.	1[P].....	43
b.	1[A].....	53
c.	1[B]	56
d.	1[C]	66
e.	1[D].....	70
f.	1[E]	76
g.	1[F].....	80
h.	1[G].....	85
i.	1[H].....	90
2.	Claim 4.....	93
3.	Claim 5.....	95
4.	Claim 6.....	98
5.	Motivation to Combine Havewala and Adams.....	99
6.	Claim 7.....	103
a.	7[P].....	104
b.	7[A]-[H].....	105
7.	Claim 8.....	107
a.	8[P].....	108
b.	8[A].....	108
c.	8[B]	109
d.	8[C]-[J]	110
B.	My Opinions as to Claims 2 and 3 based on Havewala in view of Adams and Saridakis.....	112
1.	Claim 2	112
2.	Claim 3	114

3.	Motivation to Combine Havewala with Adams and Saridakis.....	115
C.	My Opinions as to Claims 2 and 3 based on Havewala in view of Adams, Saridakis, and Rothman.....	116
1.	Claim 1-8.....	116
2.	Motivation to Combine Havewala, Adams, Saridakis, and Rothman	121
XIII.	CONCLUDING STATEMENTS.....	123

EXHIBIT LIST

EXHIBITS FILED BY PETITIONER	
Ex-1001	U.S. Patent No. 8,484,260 (the “’260 patent”)
Ex-1002	File History of the ’260 patent
Ex-1003	Declaration of Dr. Patrick McDaniel
Ex-1004	CV of Dr. Patrick McDaniel
Ex-1005	U.S. Patent Application Publication 2007/0016621 (“Havewala”)
Ex-1006	U.S. Patent Application Publication 2002/0046232 (“Adams”)
Ex-1007	U.S. Patent No. 8,874,691 (“Saridakis”)
Ex-1008	U.S. Patent Application Publication 2005/0289218 (“Rothman”)
Ex-1009	Dropbox, Inc.’s <i>Sotera</i> Stipulation
Ex-1010	The Linux Information Project, “Metadata Definition.” 2006.
Ex-1011	Wayback printout, “NTFS Master File Table (MFT)”, NTFS.com
Ex-1012	Eberspächer, Jörg and Schollmeier, Rüdiger. “Peer-to-Peer Systems and Applications.” (2005), Chapter 5 First and Second Generation of Peer-to-Peer Systems.
Ex-1013	Frystyk, Henrik. “The World-Wide Web.” (1994).
Ex-1014	Wayback printout, “POSIX® 1003.1 Frequently Asked Questions (FAQ Version 1.10)”
Ex-1015	Sandberg, Russel. “The Sun Network Filesystem: Design, Implementation and Experience.” (2001).
Ex-1016	Wayback printout – “Filesystems”

Ex-1017	Webpage printout – “Multics”
Ex-1018	Sandberg, Russel, et al. “Design and implementation of the Sun network filesystem.” Proceedings of the summer 1985 USENIX conference. 1985.
Ex-1019	[Reserved]
Ex-1020	Wayback printout – Napster
Ex-1021	Shuler, Rus. “How Does the Internet Work?” (2002).
Ex-1022	Fox, Geoffrey and Pallickara, Shrideep. “Peer-to-Peer Interactions in Web Brokering Systems.” (2002).
Ex-1023	Wayback printout – Introduction to Linux Loadable Kernel Modules
Ex-1024	Webpage printout – Deploying your hardware and software systems
Ex-1025	Wayback printout – Invasion of the Data Snatchers
Ex-1026	Warrick, R. Drake. “File Management: My Computer and Explorer.” (2002).
Ex-1027	MacManus, Richard. “Microsoft has 97% of OS market, says OneStat.com.” (2006)
Ex-1028	Alan Freedman, Computer Desktop Encyclopedia, Ninth Edition, Osborne/McGraw-Hill, 2001.

I, Patrick D. McDaniel, Ph.D., declare as follows:

I. ENGAGEMENT

1. I have been retained by Dropbox, Inc. (“Dropbox” or “Petitioner”) in connection with the above-captioned petition for *Inter Partes* Review (“IPR”) of U.S. Patent No. 8,484,260 to Erik Caso and Michael Abraham (“the ’260 Patent,” Ex1001). I understand the ’260 Patent is currently assigned to Entangled Media, LLC (“Patent Owner” or “PO”).

2. I have been asked by Petitioner to offer opinions regarding the ’260 Patent, including the unpatentability of claims 1-8 (which I may refer to subsequently as the “challenged claims”) in view of certain prior art. This declaration sets forth the opinions I have reached to date regarding these matters.

3. I am being compensated by Petitioner at my standard hourly consulting rate for my time spent on this matter. My compensation is not contingent on the outcome of the IPR or on the substance of my opinions.

4. I have no financial interest in Petitioner or Patent Owner.

II. PROFESSIONAL BACKGROUND

5. My qualifications are set forth in my curriculum vitae, a copy of which is attached as an Exhibit 1004. Exhibit 1004 also includes a list of my publications and the cases in which I have testified at deposition, hearing, or trial within the past four years. As set forth in my curriculum vitae:

Dropbox Exhibit 1003
Dropbox v. Entangled Media

6. I earned my Ph.D. in Computer Science from the Computer Science and Engineering Department at University of Michigan, Ann Arbor in 2001. I earned my Bachelor of Science degree in Computer Science from Ohio University in 1989, and my Master of Science degree, also in Computer Science, from Ball State University in 1991.

7. Since 2022, I have been Tsun-Ming Shih Professor of Computer Sciences in the School of Computer, Data and Information Sciences at the University of Wisconsin-Madison. Prior to my move to Wisconsin, I was the William L. Weiss Professor of Information and Communications Technology in the School of Electrical Engineering and Computer Science at the Pennsylvania State University in University Park, PA. I was also the director of the Institute for Network and Security Research, and founder and co-director of the Systems and Internet Infrastructure Security Laboratory, a research laboratory focused on the study of security in diverse network and computer environments. My research efforts primarily involve computer systems, network management and authentication, systems security, and technical public policy.

8. I frequently teach and do research on filesystems and storage, particularly as it relates to security. For example, Chapter 6 of my PhD dissertation, “Case Studies: Virtual Private Filesystems in AMirD”, described a

peer-to-peer distributed file system that allowed multiple devices to synchronize and share metadata indices and file data blocks amongst devices. I continue to do research in filesystems to this day, and in my most recent work (appearing in the technical report Securing Cloud File Systems using Shielded Execution) has developed techniques for securing filesystem metadata on untrusted hardware. I also frequently teach filesystems design in my systems programming course.

9. Prior to 2017, I was an Assistant Professor (2004-2007), Associate Professor (2007-2011), Full Professor (2011-2015), and Distinguished Professor (2016-2017) of Computer Science and Engineering and the William L. Weiss Professor of Information and Communications Technology (2017-2022) at the Pennsylvania State University. Since 2004, I have taught courses in the field of computer systems, systems programming, mobile device security, networks, and network and computer security at both the undergraduate and graduate level. I created and maintained several of these courses at Penn State University. I have begun teaching courses at the University of Wisconsin-Madison in the Fall of 2023 teaching network and systems security.

10. From 2003-2009, I was also an Adjunct Professor at the Stern School of Business at New York University in New York, NY. At the Stern School of Business, I taught courses in computer and network security and online privacy.

11. I am a Fellow of the Association for Computing Machinery (ACM, the leading professional association for computer science), a Fellow Institute for Electrical and Electronics Engineering (IEEE, the leading professional association for engineering), and Fellow of the American Association for the Advancement of Science (AAAS, the leading professional association for science).

12. I am the director of the NSF Frontier Center for Trustworthy Machine Learning (CTML), a research organization exploring security and privacy of machine learning (and AI in general). Funded by the US National Science Foundation, the CTML is led by the University of Wisconsin-Madison and has members from Stanford, the University of California, Berkeley, The University of California-San Diego, and the University of Virginia.

13. I was the Program Manager (PM) and lead scientist for the Cyber Security Collaborative Research Alliance (CRA). The CRA is led by Penn State University and includes faculty and researchers the Army Research Laboratory, Carnegie Mellon University, Indiana University, the University of California-Davis, and the University of California-Riverside. This initiative is a major research project aimed at developing a new science of cyber-security for military networks, computers, and installations.

14. I have served as an advisor to several Ph.D. and master's degree candidates, several of whom have gone on to become professors at various institutions such as North Carolina State University, Purdue University, the University of Toronto, the University of Oregon, and the Georgia Institute of Technology and the University of Florida. I am currently an advisor to seven Ph.D. students.

15. Before joining Pennsylvania State University as a professor, I was a software developer and project manager for companies in the networking industry including Applied Innovation, Inc., and Primary Access Corporation. I was also a senior researcher at AT&T Research-Labs. As part of my duties in these industrial positions, I informed, reviewed, and formed corporate policies and practices relating to the security of mobile devices, networks, and software systems (including filesystems)

16. I have published extensively in the field of mobile security, network and security management, computer systems, authentication, storage and filesystems systems security, applied cryptography and network security. In addition to writing several articles for industry journals and conferences, I have authored portions of numerous books related to computer systems, applied cryptography and network security. I have served on the editorial boards of

several peer-reviewed journals including ACM Transactions on Internet Technology, for which I was the Editor-in-Chief. I was also an Associate Editor for ACM Transactions on Information and System Security and IEEE Transactions of Software Engineering, two highly regarded journals in the field.

17. Additional information regarding my background, qualifications, publications, and presentations is provided in my CV, which I understand has been submitted as Exhibit 1004.

III. MATERIALS REVIEWED

18. In forming my opinions, I have reviewed the '260 Patent and considered each of the documents listed in the Exhibit List above and all other materials discussed herein. In reaching my opinions, I have relied upon my experience in the field and also considered the viewpoint of a person of ordinary skill in the art as of the time of the earliest claimed priority date of the '260 Patent, *i.e.*, May 5, 2009. As explained below, I am familiar with the level of a person of ordinary skill in the art regarding the technology at issue as of that time and all of the opinions I have formed have been assessed from the perspective of the person of ordinary skill in the art.

IV. DESCRIPTION OF THE RELEVANT TIMEFRAME, THE RELEVANT FIELD, AND A PERSON OF ORDINARY SKILL IN THE ART

19. I understand that the '260 Patent was filed on March 19, 2012, and claims priority to Provisional Application No. 61/175,489, filed on May 5, 2009. Thus, for purposes of my analysis, I have treated the effective filing date for the challenged claims as May 5, 2009. I reserve the right to update my analysis should Patent Owner assert a different priority date.

20. I have received and understand the specification, claims, and file history of the '260 Patent.

21. In my opinion, a person having ordinary skill in the art ("PHOSITA") at the time of the effective filing date of the '260 Patent would have had at least a Bachelors or Masters degree in computer science or electrical engineering or a related degree and with at least two years training or experience with networking and file systems. Additional work or research experience can substitute for less or a different education, and vice-versa. My opinions presented herein are as viewed through the eyes of a PHOSITA prior to May 5, 2009.

V. TECHNOLOGY BACKGROUND

22. Before presenting the relevant facts and my opinions relating to my charge by the court and as a means for providing context and background for my later discussions, I present an overview of file systems. Specifically, I first define

what file systems are and their primary functions (as relevant to this report), then I describe the evolution of file systems, covering topic such as flat versus hierarchical file systems, centralized versus distributed file systems, client-server versus peer-to-peer network models, and network proxies. I conclude with a discussion surrounding the fundamentals of file system functions, how are they extended to support new features, and how distributed filesystems in peer-to-peer models operate with such extensions. Note that this exposition is by no means exhaustive, but simply outlines basic terminology in sufficient depth to frame this report and my opinions.

A. What are “File Systems”?

23. A *file system* (commonly abbreviated as “fs”) is the methods and data structures that an operating system uses to keep track of files on Storage device (e.g., hard disk, thumb drive). Ex-1016, p.1. There are many kinds of different file systems, including minix, xia, ext3, ext2, jfs, xfs, zfs, msdos, vfat, among others. *Id.*, pp.1-3. While the specific features provided by these varying kinds of file systems is broad (some of which I cover in subsequent sections of this report), they all perform the same basic function: to organize and keep track of files.

24. While some of the types of file systems above differ in some respects, there is a set of common structures and features that they all share (and broadly all modern file systems today). These structures and features are core file system

technologies; said differently, in the absence of these features, it could be argued that such technology could not be classified as a “file system.” These features include storage, filenames, directories (*i.e.*, folders), and metadata.

25. *Storage*, in the context of file systems, refers to components of a computer that are used to ensure that the data within a file system (and the computer broadly) is preserved when the system is shut down and restored upon start. Computer storage (*e.g.*, hard disk and solid-state drives) is typically characterized as *non-volatile*, in that the stored information is retained even after the system is shut down (and thus, power is removed). This is unlike computer memory (*e.g.*, RAM) which loses its information when the system shuts down. File systems are not limited to a single storage medium; they can, for example, be distributed across multiple storage mediums, wherein portions of the file system are “mounted” across multiple hard disk drives, for example.

26. The core feature of all modern file systems is the notion of a *filename*, a collection of alphanumeric (and some special) symbols used in association with a collection of data. In most common file systems, filenames follow a name-dot-extension convention; filenames are preceded with a collection of symbols (often assigned by the user), a period, and an extension (typically assigned by an application). This convention allows files to be recognizable by users (or other

applications) and linked with a particular application so that when, for example, clicked on by a user, the appropriate application is opened and loads the contents of the file from storage into memory. For example, the filename “Document.docx” would be interpreted to be a Microsoft Word document.

27. Ostensibly all modern file systems can be classified as, “hierarchical file systems”, wherein the file system organizes files into a tree-like structure. This organization is made possible by directories (sometimes also called “folders”), a structure which catalogues the contents inside of it. Directories can have directories inside of them (*i.e.*, subdirectories), sub-subdirectories within subdirectories, and so on. Often, the terms “parent” and “child” are used to describe the relationship between a directory (the “parent”) and a file or subdirectory inside of it (the “child”). In hierarchical file systems, files can be retrieved by navigating from the “root” parent directory (the directory in a file system that contains all files and subdirectories), through child subdirectories, and finally to the desired files. This process, navigating from parents to children to find files, is called a *file path* (or path). The path of a file describes both the location of a file within the file system and how the file should be organized when shown to the user (*e.g.*, as being “inside” of a folder). Technically speaking, file paths are

built from inode numbers, which inform the operating system where the data is physically located in computer storage.

28. Information such as file names and paths are stored alongside files as *metadata* (*i.e.*, data about data). This metadata is used by both the file system and operating system for a variety of functions, such as how files should appear to users (*i.e.*, their names), who owns the file (so that unauthorized users cannot access or modify file contents), their path, how much storage space the file consumes, *etc.* File system metadata can be read via the Unix system (a popular computer operating system whose origins date back to 1969) call¹ “stat.” A system call (detailed in Section V.C) is a method by which applications can programmatically request a service from the operating system. Given a file name, stat returns information such as the current storage device the file is located on, its size, when it was created, among other pieces of information. Notably, this bookkeeping represents critical pieces of information for file systems; if metadata for a file were to be deleted, the file would be “lost” (in the sense that the operating system would not be aware of the existence of the file, which would require

¹ A system call (detailed in Section 1.4) is a method by which applications can programmatically request a service from the operating system.

complex data recovery tools to rediscover the file on computer storage, if it were even possible).

B. The Evolution of File Systems

29. It is challenging to lay out the history of file systems since many of the earliest file systems were tightly coupled with the operating system (that is, it would be hard to treat the operating system and file systems as different entities, which is a paradigm that started in roughly the 1970s). Nonetheless, the evolution of files systems can be characterized through three epochs: migration from a flat to hierarchical file organization scheme, the ability to retrieve data remotely, and the seamless sharing of data across multiple devices.

1. From Files to Folders

30. Perhaps the earliest file system can be traced back to the DECTape (1964) and CP/M (1973) operating systems. The file systems inside of these operating systems organized files in a flat manner, *i.e.*, there were no directories or folders to organize files within. This design was partially due to the storage capacities of computers at that time. While the systems did not exhibit any technical reason to be limited to a flat file organization, storage space was scarce, and thus, as a matter of practical limits, directories would not be particularly useful given the small number of files that could be saved to storage. However, as storage

sizes increased, flat file systems rapidly became unwieldly from the sheer number of files applications, users, and operating systems would create.

31. To better managing an ever-growing number of files, the Multics operating system (Ex-101), dating back to 1965, introduced hierarchical file systems. Multics influenced the design of many modern operating system date (*e.g.*, Unix), including its hierarchical file system design, support for long file names, storage quotas, *etc.* After Multics, essentially every popular modern file system implements a directory hierarchy, filenames, metadata, among other features. To support this organization schema, file metadata would be augmented to include where a file existed within the hierarchy, so that users and applications could focus on a desired set of files.

2. Remote Storage for File Systems

32. With the rise in popularity of computing systems and networks, file systems were further extended to allow access to storage over a network. In 1984, Sun Microsystems introduced the Network File System (NFS), a *distributed file system* that allowed users to access files stored remotely. Ex-1018. File systems using network protocols, such as NFS, are “distributed” in the sense that the methods to retrieve data are identical for both local and remote storage mediums. Thus, many distributed file systems are “transparent” in that users are not readily aware if files are retrieved via local or remote storage mediums. Specifically, such

remote files are called *virtual files*, files whose content is stored remotely, with metadata such as file names, path, size, *etc.* stored locally. *See, e.g.*, Ex-1018, 12:32-13:11. These files support the transparency features provided by distributed file systems, in that virtual files appear structurally identical to all other local files in the file system.

33. While distributed file systems using simple network protocols offer many advantages, they are limited by: (1) files must be transferred over the network when a user or application uses them, which may be slow, and (2) the accessibility of files stored remotely is dependent on the connection to the remote computing device; a loss in connectivity implies users will no longer have access to the remote files (in other words, a single point of failure).

34. As networking technology improved, many computing technologies became increasingly reliant on accessing data remotely. The sheer amount of available data was problematic for modern computers for at least two reasons: (1) storage mediums simply did not scale to the vast amount of desirable data (thus necessitating remote access via protocols such as NFS), and (2) the traditional client-server model, wherein multiple computers (clients) connected to a central computer (the server) for a particular service, became cost-prohibitive, as serving many clients mandated powerful (and expensive) computers to act as servers. One

solution was to distribute work evenly (as much as possible) across clients, wherein clients would also serve the requests of others.

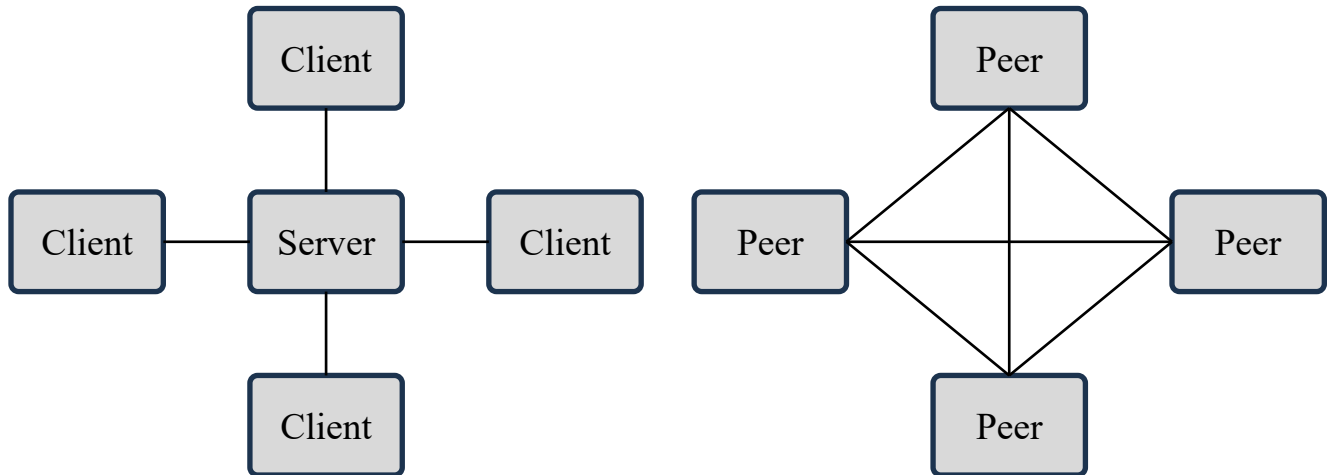


Figure 1 - Client-Server vs. Peer-to-Peer Models

3. Sharing Files with Peers

35. This solution became known as *peer-to-peer* (P2P) computing. As the name implies, instead of “clients” and “servers”, all participating computers are considered “peers”, as shown in the Figure above. P2P protocols have been used in many applications, with the earliest popular application being Napster, a platform for sharing music among peers, in 1999. Ex-1020. Many file-system-like applications (such as PAST and IgorFs) rapidly emerged in the early 2000s with the popularity of P2P, offering the ability to store, retrieve, and use data much like classical distributed network protocols like NSF that relied on the client-server model.

36. While P2P applications offered many benefits over the traditional client-server model, they were not without limitation. Perhaps the largest assumption made in any P2P application was that all peers are accessible to all other peers (that is, even if two peers are across the planet, their traffic will eventually be routed to one another). As the Internet matured, this assumption was found to rarely hold true. New techniques for providing security to computers connected to the Internet, such as firewalls, network address translation (NAT), dynamic IP addresses, among others, implied an asymmetry broadly incompatible with P2P: a host that can connect to others does not necessarily mean that others can connect to the host. In this context, network address translation maps multiple private IP addresses into a single public IP address, allowing multiple devices in, *e.g.*, a home network to use a single IP address to access the Internet. Dynamic IP addresses change from time to time, which is problematic for P2P networks since IP addresses were used to identify peers (and were largely assumed to be static).

37. Clearly, this asymmetry assumption is detrimental to P2P protocols functioning properly. Thus, many P2P applications adapted to this maturation of the Internet to include networking techniques to allow computers to send data to one another even in circumstances where they cannot be directly connected to. Specifically, a *proxy server* is a computer that acts as an intermediary between

other computers by relaying messages across computers. While proxies have many uses (many of which are outside the scope of this report), in the context of P2P file system, a proxy server would allow, for example, two clients to send and receive files from one another even in circumstances where they may not be able to directly connect to one another (*e.g.*, firewalls, NAT, dynamic IPs, *etc.*). Proxy-server-based solutions take on many forms, such as: *relay servers*, that relay data between peers (which can be general or application-specific); *Session Traversal Utilities for NAT* (STUN) servers, which assisted clients in traversing NAT; *Virtual Private Network* (VPN) services, which creates a virtual network such that clients appear to be on the same local network; among many other solutions. The most popular solution were relay servers, for their simplicity (in that users were not required to perform extensive software configuration, such as with STUN and VPN services).

38. Relay-based proxies often used the *Hypertext Transfer Protocol (Secure)* (HTTP(S)), which is one of the most common protocols used across Internet. Ex-1021. In the context of P2P networks, peers (who were unable to directly communicate with one another) would connect to the proxy server via HTTPS (just as they would when navigating the Internet) wherein communications between the peers would be relayed via the proxy server. This architecture is a

hybrid between the traditional client-server model and peer-to-peer model in that the central server is leveraged only when a peer-to-peer connection is not possible. Leveraging proxies to address some of the inherent limitations of P2P applications was an obvious solution. Ex-1022.

C. The Fundamentals of File Systems

39. File names, directories, metadata, *etc.* are primitives that are found in nearly any modern file system. File systems implement a series of functions to support the use and organization of these data structures. That is, applications can call these methods to create, rename, move, open, close, *etc.* files within the computer as desired. The collection of these methods is called the file system *application programming interface* (API). APIs are a core construct of computers in that they define the standards on how applications can communicate with one another.

40. Interfaces provided by the operating system (which includes file system interfaces) are called “system calls” (often abbreviated as a “syscall”). As the name suggests, a syscall is a particular service provided by the operating system for applications to use. These operating system interfaces are distinguished as: (1) applications can assume such interfaces will always exist for any collection of operating systems that implement the same standardized interfaces (*e.g.*, POSIX), and (2) commonly involve controlling hardware and thus, are thoroughly

scrutinized for correctness (as an incorrectly implemented syscall could cause the operating system to stop functioning, thereby “crashing” the system). POSIX (Portable Operating System Interface) is a set of standards which define the APIs for the operating system and filesystem in UNIX operating systems.

Consequently, software that modifies system calls fundamentally change the way the operating system/file system operates (discussed later in Section V.C.2).

1. The File System API

41. The purpose of a file system is to: (a) manage the locations of files and folders in non-volatile storage, (b) allow access to file data for applications to use. To support these basic features, all file systems (dating back to Multics) provide at least the following interfaces: open, close, read, write, metadata management, and directory management. While there are other interfaces, these basic interfaces are commonly modified by software to provide new features. Notably, while references to documentation are for Linux (a popular, Unix-like operating system), all modern operating systems (*e.g.*, Windows) provide file system interfaces that broadly provide identical functionality.

42. To “open” and “close” files are perhaps the most elementary concept in file systems. These are signals sent to the file system from applications that they wish to access data stored in storage. The file system then translates and sends these requests to the operating system to allocate the resources necessary (*e.g.*,

reserving space in memory) to retrieve (or modify) the data at the locations recorded in the file metadata. The file is considered “open” when a data structure (commonly referred to as a “pointer”, “handle”, or “descriptor”) is returned from the file system to the application that contains the necessary information for the application to perform input or output operations on the desired file.

Correspondingly, a file is considered “closed” when application notifies the file system that it wishes to terminate access to the file. On close, the operating system deallocates resources and the file system updates the metadata of the file.

43. When a file is open, applications can then either “write” or “read” data to or from the file, respectively. Reading data from a file involves, as the name describes, retrieving the data associated with the file, while writing involves modifying (or adding to) the file data. Both operations update the file metadata, in that writing data to a file would update “last file modification” metadata (and potentially the file size), while reading data from a file would update “last file access” metadata, among others.

44. As described in Section V.A, metadata records a variety of information about files, including their location on storage, their path within the file system hierarchy, event timestamps (such as opening, writing, *etc.*), among other pieces of information. In general, managing file metadata is the sole

responsibility of the file system (in that file systems do not expose interfaces to modifying metadata, with exceptions for non-critical data, such as file timestamps). Instinctively, critical information such as the location of a file on storage should not be trivially modifiable by applications in that consequences of mistakes are severe (*e.g.*, the file could become “lost” to the operating system and thus unretrievable). If desirable, applications can also extend metadata structures to include arbitrary information.

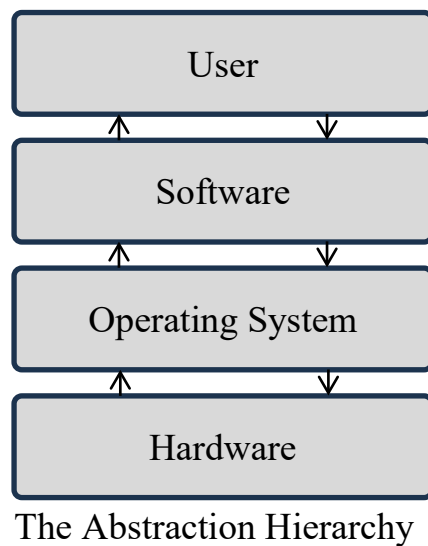
45. File systems also expose a series of interfaces for manipulating the path of a file in the filesystem hierarchy. Recall that the path of a file corresponds to how the file is organized within directories (which is different from its physical location on storage). Applications can move files or change their names,

2. Modifying File Systems

46. Since the 1960s, file systems have changed the way computers share and organize files. However, many of these changes do not require a developer to write their own file system from scratch when they wish to implement a new feature. Since the introduction of Unix and Windows 98, modifications done to file systems require simply writing and inserting the desired feature into the operating system. This capability of extending the features of file systems (and computers broadly) is one of the fundamental design philosophies of computers: the *principle*

of abstraction. Exact definitions vary from application to application, but the principle of abstraction generally states that software should be designed so that its functions are independent of its implementation. In other words, developers that wish to extend or build “on top of” pieces of software should be able to focus on what the software does and not how it does it.

47. The principle of abstraction is embodied in the design of modern computers; operating systems interface with the low-level functions of computer hardware to expose high-level features for the software above it to use, as shown in Figure below.



Here, computer hardware represents the least abstract layer of the computer hierarchy in that it (often quickly) performs a set of highly specific functions. The operating system sits above hardware, wherein it mediates access to hardware by

providing a set of services for software to use (*e.g.*, a file system). Software sits above the operating system and interfaces with the computer to provide services for users to interact with. Notably, software need not be aware of precisely how the operating system manages the hardware—it only needs to be aware of the services that are provided by the operating system, and so on. Thus, it is common to describe software as sitting, for example, “below” the file system to mean that the software modifies the underlying services a file system interfaces with to perform its function.

48. Operating systems interface with hardware through *device drivers*, software that controls hardware and provides an interface for the operating system to issue instructions to the underlying hardware. In Linux, device drivers are called loadable kernel modules (LKM) (Ex-1023), while in Windows they are called kernel-mode drivers. The kernel is the core component of the operating system that is responsible for managing system resources. To be precise, device drivers can run in user space (and thus, would be called “user-mode drivers” in Windows, for example). Without loss of generality, the file system device drivers in this declaration run in kernel-mode. In the context of file systems, device drivers could, for example, change the implementation of the write system call to exploit the design of the underlying storage medium (a notable example would be

traditional hard disk drives, wherein data would be written in physically close clusters to optimize performance). Thus, modifying the file system involves rewriting the relevant system call implementations provided by operating systems.

49. Such file system modifications can offer a myriad of benefits (especially in the context of a distributed file system. Some examples include virus detection, automatic encryption, transparent replication, storage optimization, among other features.

D. Distributed File Systems in Peer-to-Peer Networks

50. A distributed file system in a peer-to-peer network relies on some form of a networking protocol (to retrieve data stored remotely), and retrieves data is from arbitrary computers in the network (that is, data is not exclusively provided by a privileged set of computers, instead all computers in the network both provide and request data). These file systems all address the same fundamental challenges: (1) peer discovery (computers are aware of one another and the files that art part of the file system), and (2) data synchronization (file contents must be identical across peers). While proxies are not necessary in P2P networks, they are common in that they address many of the inherent limitations of P2P networks, as discussed in Section V.B.3.

51. The first step in any P2P network is joining the network. There are three approaches to joining a network. In “unstructured” networks, peers can

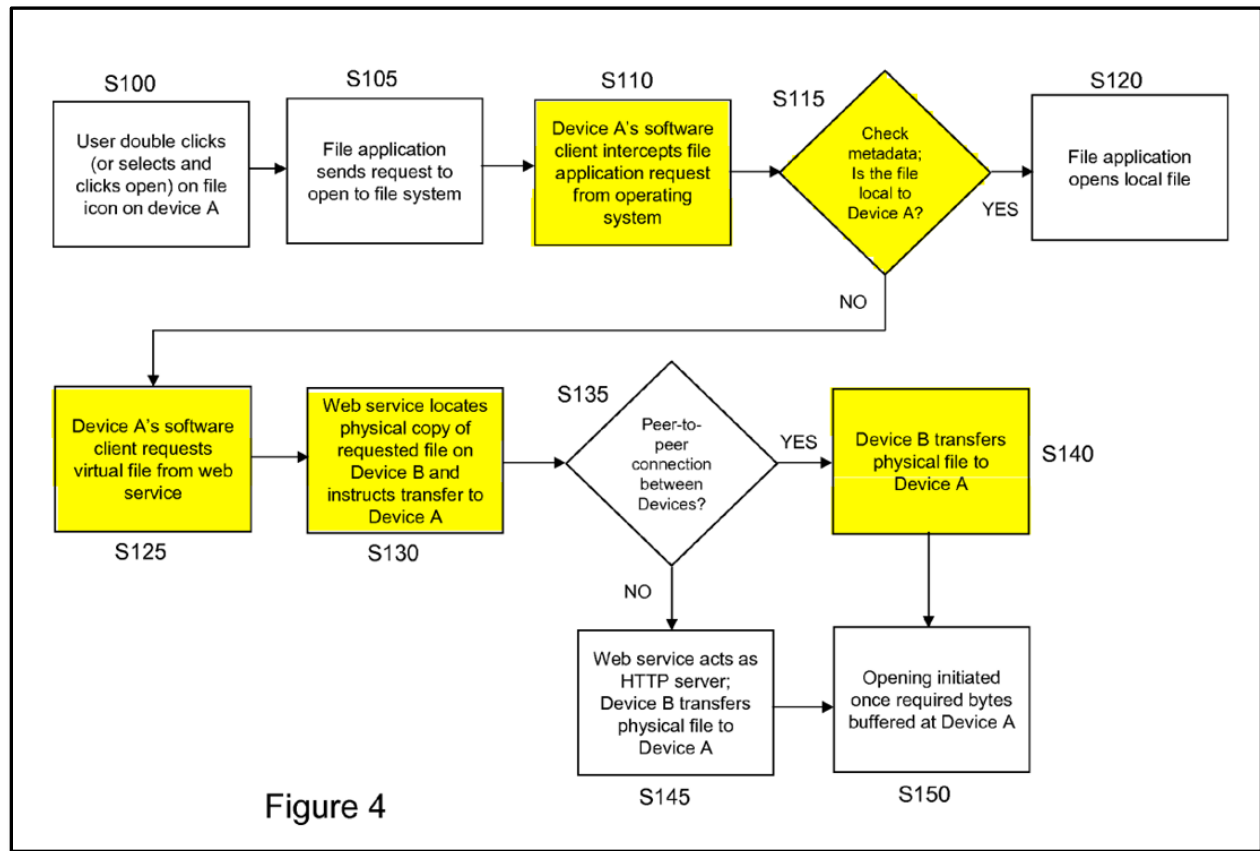
periodically broadcast their presence, thus forming arbitrary connections to other peers who receive the broadcast. While easy to build, unstructured networks are highly inefficient in that queries for data must be flooded throughout the network, inducing substantial costs in traffic overhead. To address this inefficacy, “structured” networks organize peers such that file ownership is assigned to specific peers based on network quality criteria (*e.g.*, distributed hash tables). However, structured networks often struggle with peer discovery when peers are often joining and leaving the network. The final approach addresses both limitations via a hybrid approach; a centralized server maintains a list of peers while using approaches for efficient queries commonly found in structured networks.

52. Once peers have joined the network, updates to files must be synchronized among peers. Data synchronization among peers is a well-known problem that contains multiple solutions. Popular solutions include serialization (only one computer can update a file at a time), centralization (all updates go through one computer and relayed to others), and timestamps (only updates to the latest version of a file are valid). Each of these solutions have their own limitations in that they can require a central authority (thereby straining resources),

disallow concurrent updates, or place limitations on the kinds of updates that are accepted (*e.g.*, only updates to the latest versions).

VI. OVERVIEW OF THE '260 PATENT

53. I have reviewed the '260 Patent and understand that Erik Caso and Michael Abraham are named as the inventors on this patent. The '260 patent claims a system and method for maintaining a file system that stores a physical file in one device and enables access of the physical file at other devices. The '260 patent states that data “synchronization across all devices for all content on all devices is limited or even prohibited by space restrictions.” Ex-1001 at 1:57-59. Because of these storage restrictions, the '260 patent purports that “there is a need in the art for a system and process that facilitates access to content that physically resides on any one of multiple user devices.” *Id.*, 4:7-9. The '260 patent proposes a system that “ensures that all [of] a user’s devices appear to share one single native file system containing all [of] the user’s files across all devices” while “not requir[ing] physical data replication across multiple devices.” *Id.*, 2:16-24. This was not new, as such systems were well known prior to the Critical Date. The '260 patent’s claimed system is shown in Figure 4:



54. The '260 patent illustrates in Figure 4 above that when a user attempts to operate on a virtual file at Device A, a software client intercepts the request (S110) to determine whether the file is locally stored on Device A. If the file is not local, the software locates and requests the corresponding physical file from the device where it is locally stored, here Device B (S125, 130). The corresponding physical file is then transferred from Device B to Device A (S140).

55. I note that unless otherwise stated, I added the coloring to the figures in this declaration. Bolded text has also been added throughout my declaration for emphasis unless otherwise indicated.

56. I have also reviewed the prosecution history of the '260 patent and note that the claims were rejected as being anticipated by U.S. Patent Publication No. 2006/0161585 to Clarke and also rejected as obvious by Clarke in view of U.S. Patent Publication No. 2007/0050069 to Aboulhosn. After an Examiner interview, claim 1 was amended to include “modifying the singular file system on the first device to make local files and virtual files appear indistinguishable to the singular file system, the local files and virtual files sharing a same location on the first device.” Ex-1002, p.50. The Applicant argued that the client in Clarke “distinguishes and discriminates locally stored object[s] and shared objects,” *id.*, p.54, that the file system in Aboulhosn “uses the file name extension ‘vf’ to discriminate and distinguish between virtual files and local files,” *id.*, p.55, and therefore, both Clarke and Aboulhosn were distinguishable from the amended claim 1.

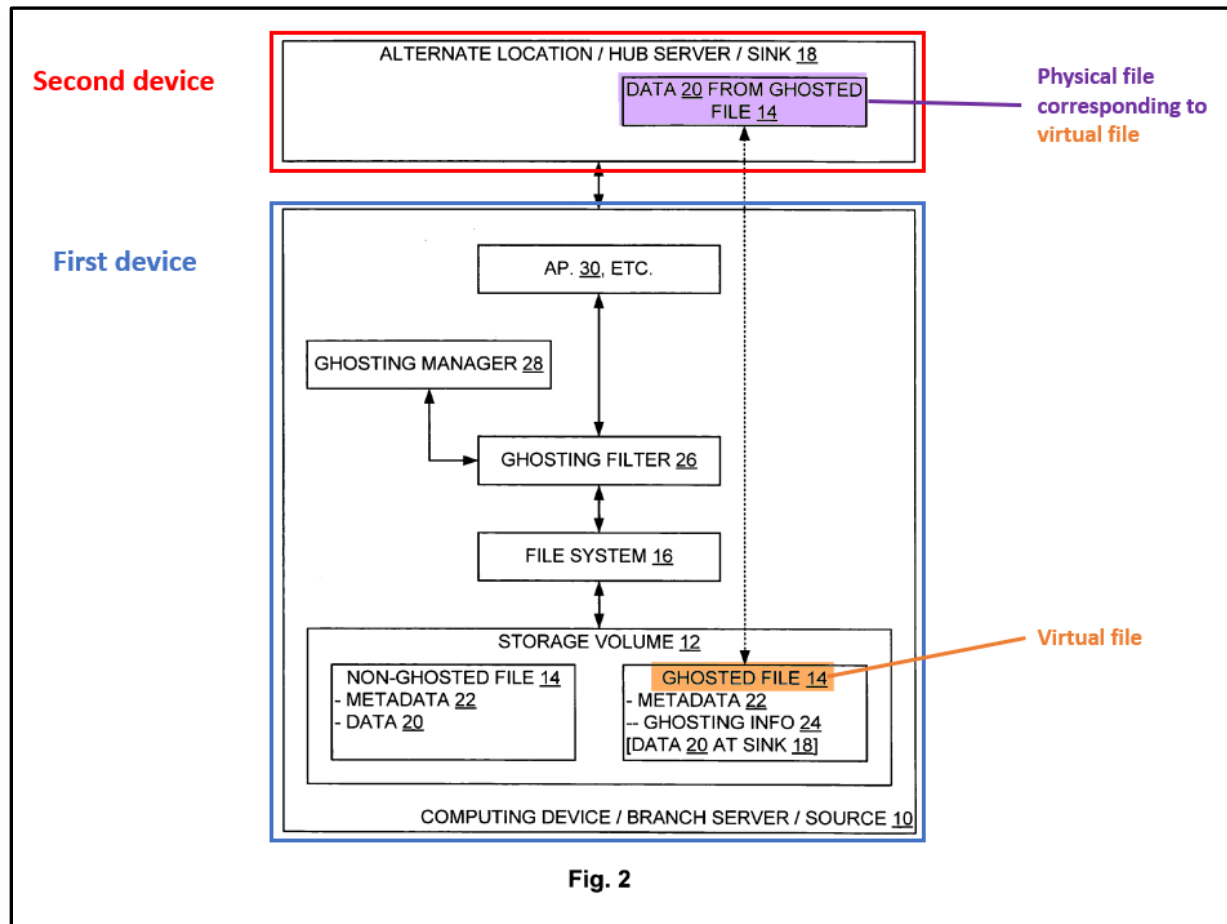
57. To gain allowance, a subsequent Examiner’s amendment incorporated the limitation of a dependent claim requiring determination of whether a file is a virtual file “by reviewing file metadata” into the independent claims.

VII. OVERVIEW OF THE PRIOR ART

A. Havewala

58. I understand that the Examiner was not aware of Havewala (Ex-1005) during prosecution.

59. Like the '260 patent, Havewala discloses a file system comprising ghosted (virtual) files that correspond to data (physical files) located at an alternate location or sink (second device) that contain the file data of the ghosted file located at a source (first device). Ex-1005, FIG. 2. Havewala's system "is described in the general context of computer executable instructions, such as program modules, being executed by a computer, such as a client workstation or a server." *Id.*, ¶34. Havewala provides a solution for a cluttered storage space containing "cold" files that "have not been accessed for some extended length" of time because, e.g., they are of limited interest to a user. *Id.* ¶10. Havewala recognizes that a user may not want to delete these files in their entirety because "most users are loathe to delete files to simply create space." *Id.* Therefore, to maintain the appearance of the presence of these files on a device, Havewala discloses "'ghost[ing]' a cold file by moving data from such cold files to the alternate location or to **copy the entire file to the alternate location**, while leaving the cold file in a smaller, 'ghosted' form on the volume." *Id.* ¶12; FIG. 2.

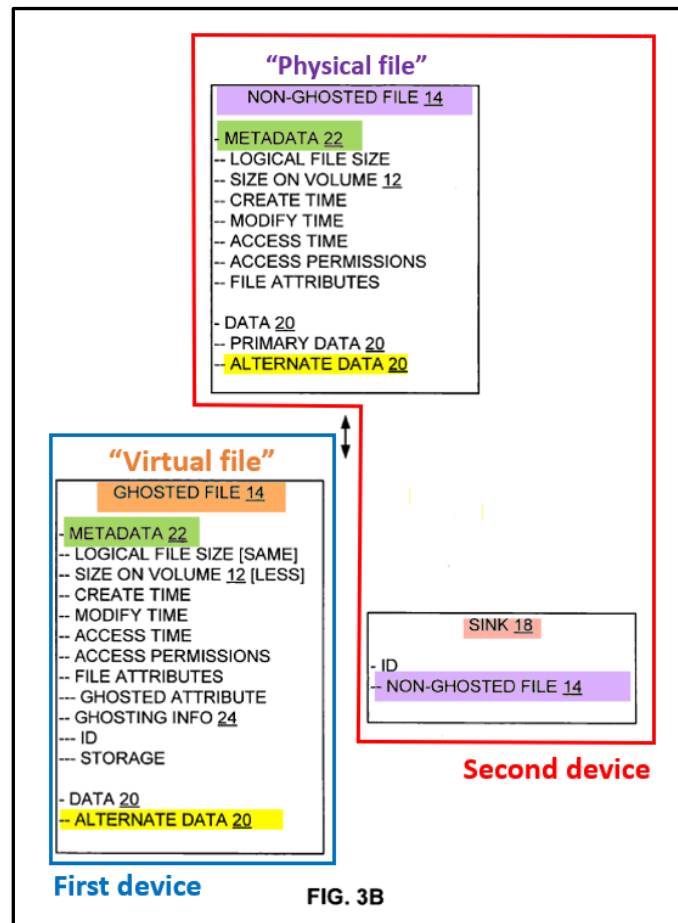


60. Havewala further discloses that the ghosted file resides in a storage volume at the source, in which ghosted and non-ghosted files are “organized by, accessed through, and otherwise controlled by a file system.” Ex-1005, ¶41.

When an application requests to operate on the ghosted file (virtual file) at the source, the ghosted file’s “corresponding data” (*i.e.*, physical contents) are obtained from the alternate location at which they are stored. Ex-1005, ¶60.

Havewala discloses that the ghosted file’s corresponding “data” can mean the “entirety” of ghosted file’s non-ghosted form, *i.e.*, physical file. *Id.*, at ¶54. For

instance, Havewala states that it “may be appreciated” by a PHOSITA that “while the primary data of the data 20 is removed from” the non-ghosted file at the source “to form the ghosted file 14” that “**the entirety** of such [non-ghosted file] is stored at the sink 18.”); *id.*, at FIG. 3B.



61. Havewala further discloses a ghosting filter and ghosting manager (together, corresponding to the claimed software client) that intercepts the operation request on a ghosted file at the first device and transfers the ghosted file’s corresponding data from the sink (second device) to the source (first device).

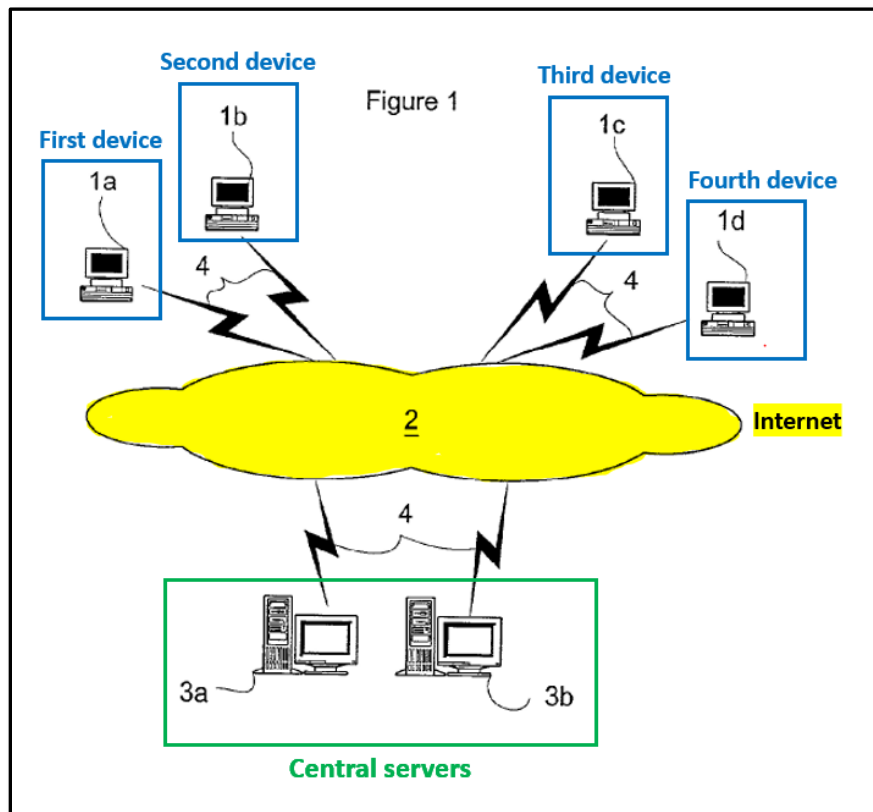
Id., ¶51. Havewala further acknowledges that this transfer of the physical file from the sink (second device) to the source (first device) can “be moved...by way of an appropriate transport mechanism and conduit.” *See id.*, ¶¶60, 74, 75. Once the ghosted file becomes reconstituted, the file can be operated on at the first device. *Id.*, FIG. 6.

B. Adams

62. I understand that the Examiner was not aware of Adams (Ex-1006) during prosecution.

63. Like the '260 patent, Adams discloses a system for “locating desired files in distributed file sharing over a computer network” (Ex-1006, Abstract) through a “peer-to-peer network connection.” The Adams system comprises of a “second client terminal being remote to [a] first terminal” and a “central index server” for transmitting the “location of [a] selected file” to client terminals. *Id.*, ¶25. Adams explains that a “limitation...of file sharing or exchange is that it is difficult to locate files of interest to a user.” *Id.*, ¶9. To address this problem, Adams discloses that a “central index server” that stores “the location of [a] file in the computer network.” *Id.*, ¶15. A client terminal (first device) retrieves files from another client terminal (second device) by “querying the central index” and “selecting a file from the available list of files” in the system. *Id.* ¶25. “[T]he location of the selected file” is “transmitted from the central index server to the

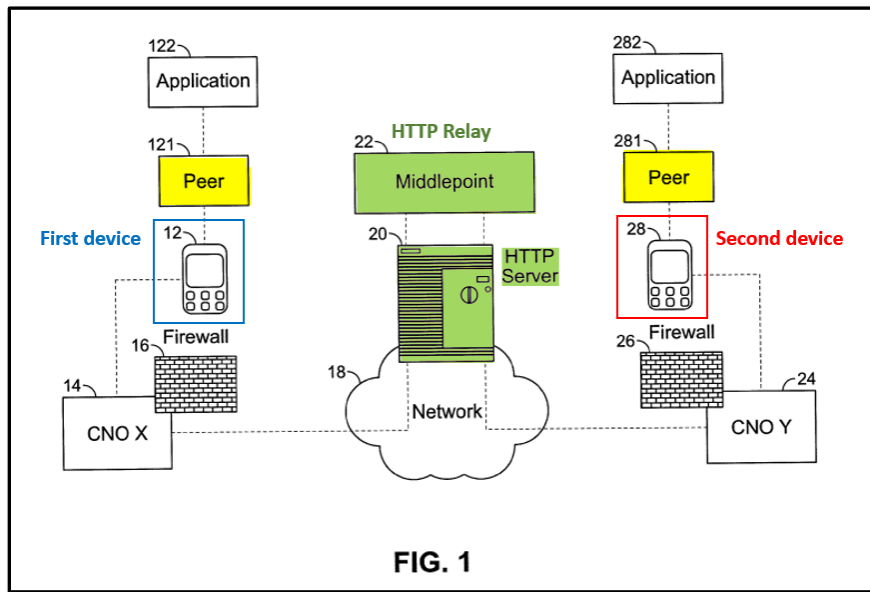
[requesting] client terminal” and the “client terminal making a peer-to-peer network connection to the [other] client terminal to obtain the selected file.” *Id.*; *see also, id.* FIG. 1. Adams discloses transferring the entirety of the file between client terminals once a peer-to-peer connection is brokered. *See e.g., id.*, Abstract (“The client terminal 1...can retrieve the file 6...from one of the [other] client terminals 1.”); ¶23 (“the selected file then being...downloaded to the first client terminal from the further client terminal”).



C. Saridakis

64. I understand that the Examiner was not aware of Saridakis (Ex-1007) during prosecution.

65. Like the '260 patent, Saridakis discloses a method of “provid[ing] a peer-to-peer communication channel between peers utilizing hypertext transfer protocol (HTTP).” Ex-1007, Abstract. Saridakis provides a solution for “circumventing network obstacles” that typically impair peer-to-peer communication channels such as firewalls. *Id.*, 2:48-53; 3:7-9. To establish a connection between multiple devices, Saridakis discloses “a first peer device communicating with a **relay** via a network including an obstacle, a second peer device communicating with the same relay via a network including another device, and a server coupled to the first and second peer devices.” *Id.*, 3:9-14. Saridakis further discloses that the “**server receives a HTTP request** from the first peer device” in which the “HTTP request includes data intended for another peer device.” *Id.*, 3:17-19. The server then “further **relays the aforementioned data to the intended peer device** establishing thus a virtual communication channel between the first peer device and the second peer device to enable sending and receiving of data.” *Id.*, 3:19-23. The HTTP Server thus acts as an HTTP relay.

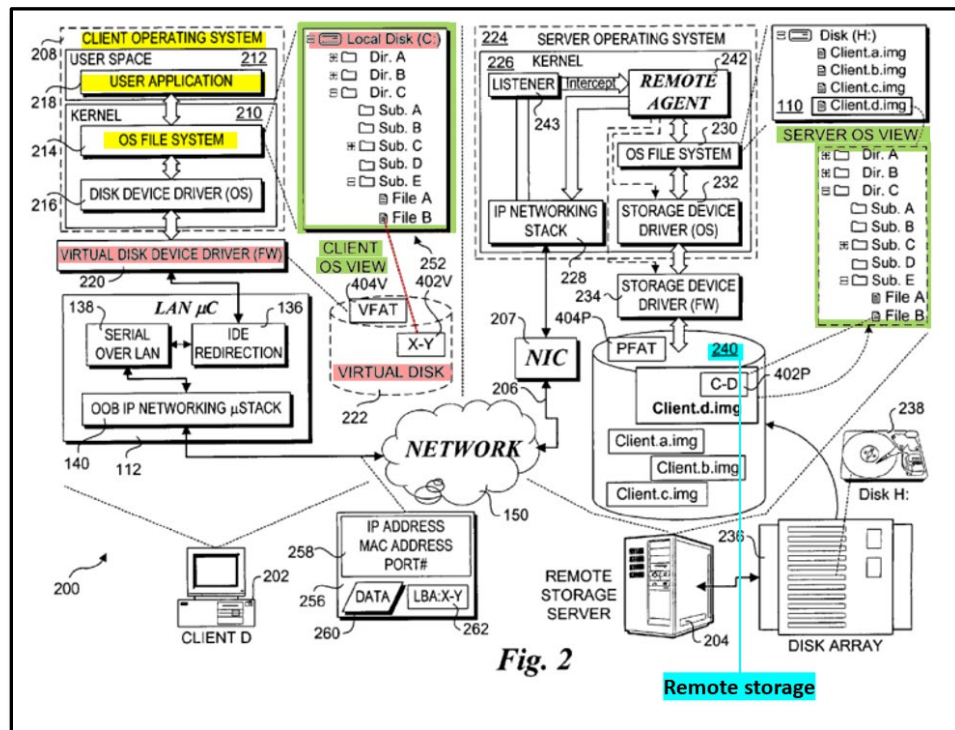


D. Rothman

66. I understand that the Examiner was not aware of Rothman (Ex-1008) during prosecution.

67. Like the '260 patent, Rothman discloses “techniques for enabling remote storage utilization as local disk resources.” Ex-1008, Abstract. As shown in Figure 2, for instance, a “virtual disk drive” that emulates a local disk drive “is facilitated via out-of-band (OOB) communications with a remote storage server in a manner **that is transparent to an operating system** (OS) running on the client.” *Id.* As shown in FIG. 2 below, “**virtual disk** [222] **appears to client operating system 208** (and thus to all of user applications 212) **as a local disk drive.**” *Id.*, ¶39. The “data ‘stored’ on the virtual disk are actually **physically stored** on

remote storage 240.” *Id.* Further, the file system of the client OS and server OS appear to contain the same files and share the same view. *Id.* FIG. 2.



VIII. CLAIM CONSTRUCTION

68. I understand that claim construction is the process of determining the meaning of a term or phrase in a patent claim. I understand that the proper construction of a term is how a PHOSITA would have understood the term based on its use in the claims, specification, and file history of the patent. I further understand that the claims are construed before the Board according to the same claim construction standard that applies in district courts. I have followed these principles in my analysis throughout this declaration.

69. For the purposes of this declaration, I do not believe there are any express constructions needed and have applied the plain and ordinary meaning of all terms.

IX. ANTICIPATION

70. I have been instructed as to the definition of “anticipation” in the context of the patent laws.

71. I understand that anticipation of a claim requires that every element of a claim be disclosed expressly or inherently in a single prior art reference, in combination, as claimed. I understand that a single prior art reference may anticipate claims without expressly disclosing a feature of the claimed invention if that feature is necessarily present, or inherent, in that reference. I understand that a reference is read from the perspective of one of ordinary skill in the art at the time of the invention.

X. OBVIOUSNESS

72. I have been instructed as to the definition of “obviousness” in the context of the patent laws.

73. It is my understanding that obviousness is a question of law based on underlying factual issues including (1) the scope and content of the prior art, (2) the differences between the prior art and the challenged claims, (3) the level of ordinary skill in the pertinent art, and (4) the existence of secondary considerations

such as commercial success, long-felt but unresolved needs, failure of others, copying, industry praise, etc.

74. It is also my understanding that I should first determine the scope and content of the prior art, then consider the differences, if any, between the prior art and each challenged claim; and that in doing so, although it is proper to consider differences between the claimed invention and the prior art, I should not focus on only the differences, because the test is whether the claimed invention as a whole would be obvious over all of the prior art. It is also my understanding that, in evaluating validity, each claim of the patent must be considered separately and in its entirety.

75. It is my understanding that the next step is to determine the level of ordinary skill in the art to which the claimed invention pertains at the time the invention was made.

76. It is my understanding that a PHOSITA is presumed to have knowledge of the relevant prior art at the time of the claimed invention; and that if I find that the available prior art shows, discloses, teaches or suggests each of the elements of the challenged claims, I should determine whether a PHOSITA would have had a reason to combine or coordinate these elements in the same manner as the challenged claim.

77. I understand that an obviousness evaluation can be based on a combination of multiple prior art references.

78. I understand that for a single reference or a combination of references to render obvious the claimed invention, a PHOSITA must have been able to arrive at the claimed invention by altering or combining the applied reference or references. I also understand that obviousness arises where it is determined that the claimed invention implemented a predictable variation of prior art elements or amounts to nothing more than the predictable use of prior art elements according to their established functions.

79. I understand that the prior art references themselves may provide an explicit suggestion, motivation, or reason to combine, but that the suggestion to combine may also arise from the state of the prior art generally. I also understand that when considering the obviousness of a patent claim, one should consider whether a teaching, suggestion, or motivation to combine the references exists to avoid impermissibly applying hindsight when considering the prior art. I understand this test should not be applied rigidly, but that the test can be important to avoid such hindsight.

XI. OPINIONS REGARDING PATENTABILITY

80. In my opinion, claims 1, 4-8 are rendered obvious by Havewala in view of Adams. Additionally, claims 2 and 3 are rendered obvious by Havewala in

view of Adams and Saridakis. Moreover, in my opinion, claims 1-8 are also rendered obvious by Havewala in view of Adams, Saridakis, and Rothman.

81. I have based my opinion on the following references:

- **Havewala:** U.S. Patent Application Publication No. 2007/0016621 titled “Moving Data From File on Storage Volume to Alternate Location to Free Space” to Sarosh C. Havewala *et al.* (“Havewala,” Ex. 1005), was filed July 14, 2005 and **published January 18, 2007**. I understand that Havewala is prior art to the ’260 Patent.

- **Adams:** U.S. Patent Application Publication No. 2002/0046232 titled “Organizing Content on a Distributed File-Sharing Network” to Colin J. Adams *et al.* (“Adams,” Ex. 1006), was filed September 17, 2001 and **published April 18, 2002**. I understand that Adams is prior art to the ’260 Patent.

- **Saridakis:** U.S. Patent No. 8,874,691 titled “System and Method for Establishing Peer to Peer Connections Between PCs and Smart Phones Using Networks with Obstacles” to Tito Saridakis (“Saridakis,” Ex. 1007), was filed June 22, 2005, **published December 28, 2006**, and issued October 28, 2014. I understand that Saridakis is prior art to the ’260 Patent.

- **Rothman:** U.S. Patent Application Publication No. 2005/0289218 titled “Method to Enable Remote Storage Utilization” to Michael A. Rothman *et. al.* (“Rothman,” Ex. 1008), was filed June 28, 2004 and **published December 29, 2005**. I understand that Rothman is prior art to the ’260 Patent.

82. In my opinion, the challenged claims of the ’260 Patent are unpatentable based on the following grounds:

Ground	Statute	Reference(s)	Claims
1	§103	Havewala, Adams	1, 4-8
2	§103	Havewala, Adams, Saridakis	2, 3
3	§103	Havewala, Adams, Saridakis, Rothman	1-8

XII. MY OPINIONS AS TO THE CHALLENGED CLAIMS OF THE ’260 PATENT

A. My Opinions as to Claims 1, 4-8 concerning Havewala in view of Adams

1. Claim 1

83. In my opinion, claim 1 is unpatentable because it is rendered obvious by Havewala in view of Adams. Claim 1 of the ’260 Patent recites the following. I address each of these limitations in my analysis below.

1[P] A process for operating on files located on multiple devices using a singular file system comprising:

Dropbox Exhibit 1003
Dropbox v. Entangled Media

1[A] accepting a request to operate on a file at a first device, wherein the file is selected from the singular file system on the first device;

1[B] modifying the singular file system on the first device to make local files and virtual files appear indistinguishable to the singular file system, the local files and virtual files sharing a same location on the first device;

1[C] intercepting the request by a software client on the first device;

1[D] determining by the software client if the file is physically located on the first device or if the file is a virtual file of a corresponding file physically stored on a second device by reviewing file metadata,

1[E] wherein a visual representation of the singular file system on the first device is identical to a visual representation of the singular file system on the second device; and

1[F] if the file is the virtual file of the corresponding file physically located on the second device, requesting by the software client on the first device that a peer-to-peer connection be brokered by a server-based web service between the first device and the second device;

1[G] if the peer-to-peer connection is brokered, transferring the corresponding physical file from the second device to the first device; and

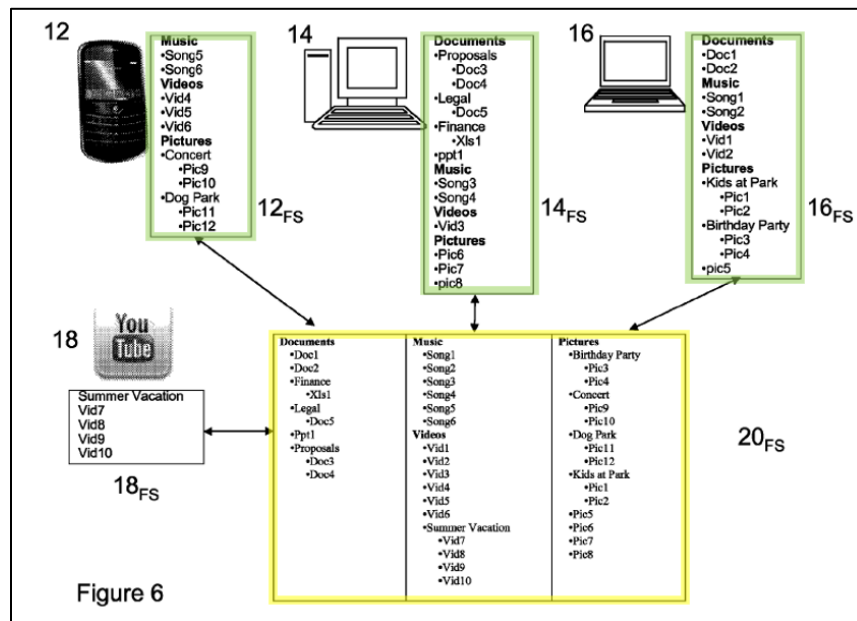
1[H] performing the operation on the transferred corresponding physical file at the first device.

a. 1[P]

84. Limitation 1[P] recites: A process for operating on files located on multiple devices using a singular file system comprising.

85. If the preamble is considered a limitation of the claim, it is my opinion that Havewala discloses it.

86. **Singular File System.** The '260 patent describes “a singular file system” as a system used to manage and display files across multiple devices, which is consistent with the terms plain and ordinary meaning of the terms “singular” and “file system.” *See, e.g.*, Ex-1001, 3:22-24, 5:67-6:1; 6:18-20. The '260 patent defines “file system” as “the component within an operating system responsible for managing all manner of files, as well as presenting them within a graphical user interface.” *Id.* 4:30-32. This file system includes a “set of standard file storage options, such as ‘My Documents’ for documents, ‘My Music’ for music, ‘My Pictures’ for pictures, and so on.” As shown in Figure 6 below, the '260 patent illustrates a *singular* file system using an “exemplary **unified file structure 20_{UFS}** [shown as 20_{FS} in yellow] created from the individual device file structures 12_{UFS}, 14_{UFS} and 16_{UFS} [green] and on-line service file structure 18_{UFS}” and “[t]his unified file structure 20_{UFS} [yellow] will replace file structures 12_{UFS}, 14_{UFS} and 16_{UFS} in devices 12, 14 and 16.” *Id.*, 9:15-22.



Ex-1001, Fig. 6.

87. In my opinion, Havewala discloses the same singular file system. Havewala discloses implementing a system for remotely accessing files in “a computing device [that] has a storage volume, a **file system** managing the storage volume, and a file stored on the storage volume by the file system and accessed through such file system.” Ex-1005, ¶18; *see id.* ¶9 (explaining in the context of the background of the invention that a storage volume includes “local computer files” and that a “such storage volume may be organized by, accessed through, and otherwise controlled by a file system running on the computing device, as is also known”). Havewala further explains that “a user or the like can browse **the entire volume 12 of the source 10 by way of the file system 16** even if some of the files 14 on the volume 12 are ghosted and some of the files 14 on the volume 12 are not

ghosted.” *Id.*, ¶51. The file system may “organize[] the volume 12 according to a **directory format** or otherwise” (*id.*, ¶50) and may display a file using “a graphic representation such as a ‘thumbnail’” (*id.* ¶52). Havewala further discloses that its system may be “practiced **in distributed computing environments** where tasks are performed by **remote processing devices** that are linked through a communications network.” Ex-1005, ¶34;

88. In particular, Havewala’s system contemplates a scenario where a first device—source 10—may include a ghosted (or virtual) file that “is present [on the source 10] in a reduced or ‘stub’ form” (*id.* ¶49) such that “source 10 contains all metadata 22 from the original, non-ghosted file 14, and also contains ghosting information 24, that may be employed to retrieve the data 20 for the file 14 from the sink 18” (*id.* ¶50). When the file is ghosted, the data with the metadata is transferred to a second device—alternate sink location 18. *See, e.g., id.* ¶54 (explaining when removing “primary data” from a non-ghosted file at the source and transferring the data to the sink, that “the entirety of such file 14 is stored at the sink 18”); *see also id.* ¶12.

89. Havewala refers to “source 10” and “alternate location 18” with different terminology throughout the specification. For example, Havewala refers to “source 10” as “computing device 10,” “storage device 10,” (Ex-1005, ¶43) and

“branch server 10” (*id.*, ¶46). Havewala refers to “alternate location 18” as a “sink 18” or “hub server 18.” *Id.*, ¶48. Havewala further explains that “alternate location 18 may be local to the computing device or may be **remote therefrom**” and “may be another volume 12 on the storage device 10, **another volume 12 on another computing device 10**, a file warehouse at a server, a long-term storage device at a remote server, or the like.” *Id.*, ¶43. Therefore, as discussed in this Declaration, “source 10” (and its associated uses of this term) refers to a first device (e.g., a computing device) including the storage volume storing the ghosted file in its common file system; and the “alternate location 18” (or its associated uses of this term) refers to a second device including the storage volume storing the non-ghosted file in its common file system.

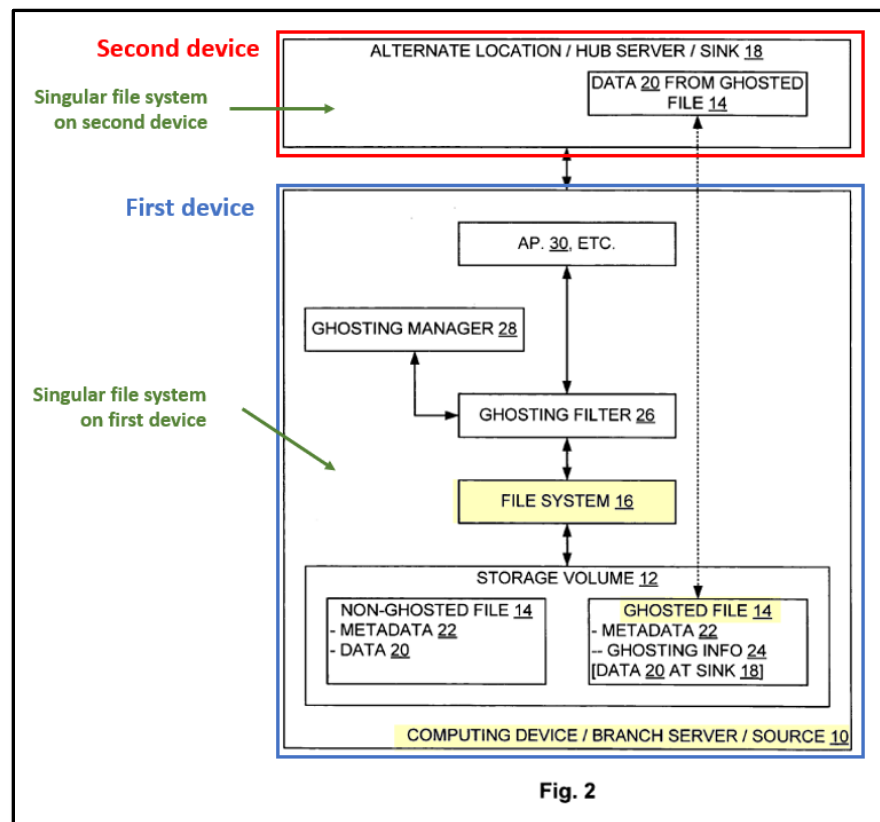
90. Havewala’s Disclosure of Implementing its Singular File System in Multiple Computing Devices. Havewala discloses that the alternate location may be “another volume 12 on **another computing device 10**” (Ex-1005, ¶43) and that “a computing device has a storage volume [and] a **file system** managing the storage volume” (*id.* ¶18). In other words, Havewala discloses that the source and sink may both be computing devices 10, that look and behave exactly the same, *i.e.*, they have the same file system stored on both devices. For example, Havewala’s system of ghosting a file is enabled through the installation of software programs

and program modules onto a computing device with a pre-existing operating system (e.g., Microsoft Windows computer). *See* Ex-1005, ¶34.

91. Havewala explains that its system is designed to overcome problems in the prior art, for example that typically when “data files of an organization or the like are to be replicated at multiple branch locations of the organization...a replication service would be employed to distribute a copy of each file” to each branch location. *Id.* ¶14. Havewala explains that this creates issues because this system increases the “amount of traffic over the network...perhaps to a point that exceeds available bandwidth” and also that, “as the overall size of all the files at the hub server increases, it may in fact become the case that each branch server does not have enough space available thereon to store all of such files.” *Id.*

92. To overcome this problem, Havewala’s system can be installed on multiple computing devices at those branch locations such that the computing devices would “be able to ‘ghost’ the non-relevant files at the branch server [e.g., first computing device/source 10] of the particular branch, such that each non-relevant file remains on the branch server [e.g., first computing device/source 10] in a smaller, ‘ghosted’ form.” Ex-1005, ¶16. The “data [of the ghosted file on branch server/first computing device] is stored at a sink such as an alternate location or a hub server [*i.e.*, second computing device].” Ex-1005, ¶17. If “the

ghosted file is needed at the branch server [e.g., first computing device/source 10], the data for such ghosted file may be retrieved from the hub server [i.e., second computing device/sink 18], the ghosted file may be reconstituted” and made available at the branch server/first computing device. *Id.*, ¶16. This is shown, for example, Figure 2 which I annotated below.



Ex-1005, Fig. 2.

93. Havewala explains that the system is designed to be utilized in, for example, situations where multiple different computers throughout a particular organization wish to have access a file, but the organization wishes to reduce at

least networking and storage constraints and store a file at only one computing device. In such a situation, Havewala's programs and program modules are stored on similar devices (computing device 10 acting as a source, a sink, or both), for example computing devices running Windows operating software. *See, e.g.*, Ex-1005, ¶¶9, 34-40, p.1 (Assignee: Microsoft Corporation). In such a situation, the same file system (e.g., the Window's file system as modified by Havewala) is installed and operates on the multiple devices to enable ghosting files between devices as discussed herein in more detail. The Windows file system would allow for a variety of operations on files at a computing device in which the Windows operating system was installed. Ex-1026. Havewala confirms that just like the source, files can be modified on both the first device and the second device. Ex-1005, ¶119; ¶127 ("if the data 20 of the file 14 at the source 10 is not modified but the corresponding data 20 at the sink 18 is modified"), ¶103 ("modifications to the file 14 that take place elsewhere, and/or the like"). Windows operating systems were widely popular well before the Critical Date, and it was routine and common to install the same Windows operating systems among a user's multiple devices. Ex-1027.

94. Therefore, consider a scenario where a file is ghosted on Computer A (e.g., first computing device/branch server 10) with Havewala's system installed

and the physical data of the file is stored on Computer B (e.g., second computing device/alternate location/branch hub 18) also with Havewala's system installed. A request to open the non-ghosted file on Computer B using, e.g., Windows file system modified by Havewala would result in the file being opened as usual because the physical contents of the file are present on Computer B. *See, e.g.*, ¶54 (“[W]here the hub server 18 replicates files 14 to the branch server 10, it may in fact be the case that while the primary data of the data 20 is removed from the remainder of the replicated file 14 at the source 18 to form the ghosted file 14, the entirety of such file 14 is stored at the sink 18.”), ¶52, Figs.3A-3B. Conversely, an attempt to open the ghosted file at Computer A (again using, e.g., Windows file system modified by Havewala) results in the physical file and/or associated data from Computer B being transferred to Computer A so that the file can be “reconstituted” and opened on Computer A. *See, e.g., id.* ¶ 49; *id.* ¶24 (“the entire file is stored at the sink”). As also discussed further below, Havewala explains that its system is installed such that any interaction or operation performed on a file (e.g., opening) is performed transparently, such that the computing device's user does not know that Havewala's system is being run or utilized when opening any particular file. *See id.* ¶64 (“[O]perations may be performed **transparently** or with

notice **to the user** or application 30, although **transparency is likely preferable to such user** or application 30.”).

95. Thus, in my opinion, Havewala discloses a singular file system at each computing device that utilizes the same file system interface (e.g., Windows file system modified by Havewala) presenting a “directory” of files available across multiple devices in a organization in such a way that a user does not know whether the file is ghosted or physically present on a particular computer device.

96. ***Operating on Files.*** Havewala discloses performing certain commands (*i.e.*, operations) on files, for example through “issuing an open command to the file system 16 with regard to the file 14.” ¶66; *see also id.*, ¶66-69 (further discussing “open” and “read” operations). Modifying a file always requires opening it. Therefore, in my opinion, Havewala discloses the required “operating on files.”

97. ***In My Opinion, Havewala Also Renders Obvious a Singular File System.*** While Havewala discloses a singular file system across at least the computing devices 10 and 18. To the extent PO argues that Havewala does not disclose a singular file system, it would have been obvious to a PHOSITA to implement Havewala’s invention using a **singular file system** across multiple devices. For example, an organization attempting to at least reduce overheads and

storage use associated with file duplication would install Havewala's system across its existing computing devices without altering the way that those devices operate. In addition to its other advantages, implementing Havewala's system in this way would provide a seamless, integrated user experience that would minimize disruption of processing workflows and maximize data availability to all users of the system. Indeed, it would be preferable to implement the software—as Havewala discloses—“transparently” such that users and employees need not be burdened to search through various file storage locations or remember or be notified as to whether a particular file is present on their computing device (*i.e.*, if the file is a non-ghosted file) or if the file is actually stored remotely (*i.e.*, the file is a ghosted file). In my opinion, a PHOSITA implementing Havewala's system in exactly the kind of situation that Havewala is designed for would have found it obvious to employ Havewala such that a user on any given computer in the network may, for example, browse for both non-ghosted and ghosted files in a single, coherent and common file system.

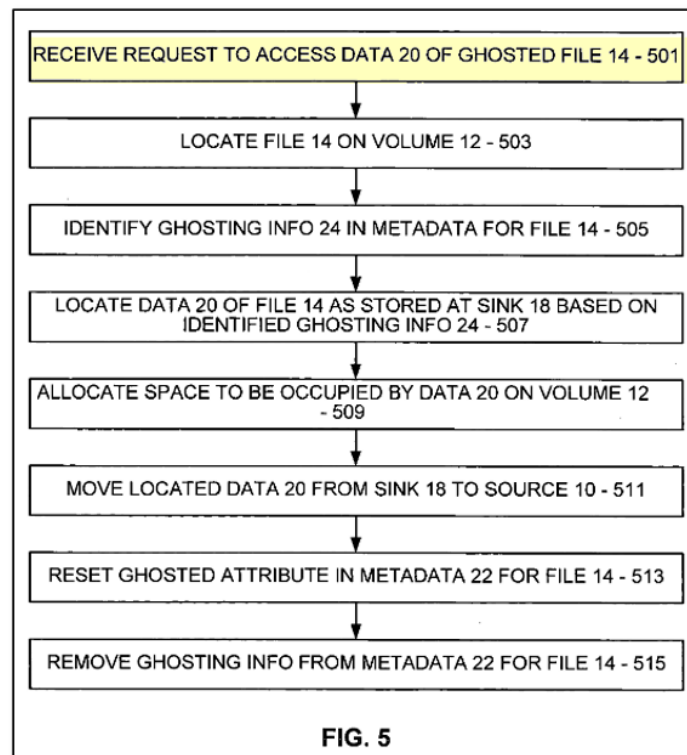
98. Therefore, in my opinion, Havewala discloses a **“process for operating on files [e.g., Havewala's “open” commands] located on multiple devices [e.g., first computing device 10 and second computing device 18] using a**

singular file system [e.g., file system 16 present at the first and second computing devices 10 and 18].”

b. 1[A]

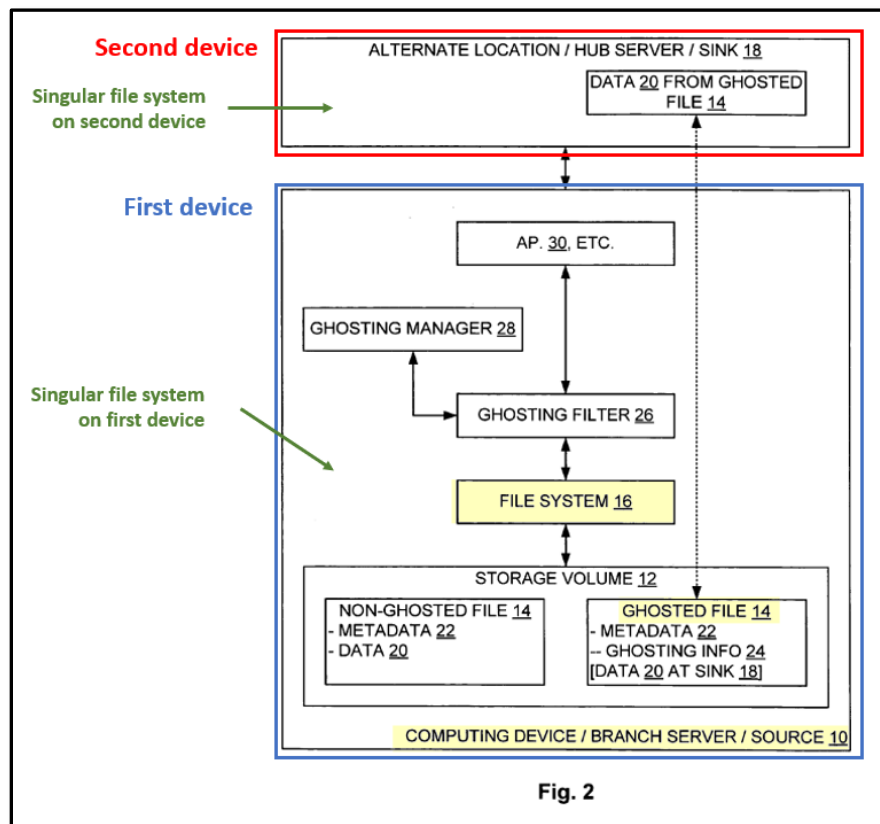
99. Limitation 1[A] recites: “accepting a request to operate on a file at a first device, wherein the file selected from the singular file system on the first device.”

100. As I detailed above, Havewala discloses a first device with ghosted and non-ghosted files and a singular file system on the first device to manage and perform operations on those files. Ex-1005, FIG 2. In one situation, for example the one discussed above, ghosted (*i.e.*, virtual) files reside at the first computing device or source/branch server 10. *Id.*, ¶50 (“[G]hosted file 14 as **present on the volume of a source 10** is stored in the same location on the volume 12 as the original file 14.”). To access the file, a user selects a file from the file system and an “application 30 begins the process by **issuing the open command with respect to a particular ghosted file 14** on the volume 12 and such **open command ultimately is received by the file system 16.**” *Id.*, ¶67; *see also id.* FIG. 2, FIG.5, ¶¶59, 60; *see id.* ¶66 (“issuing an open command to the file system 16 with regard to the file 14”). Figure 5, below, of Havewala confirms that its system “receive[s] [a] request to access” or open the file that is ghosted on the first device.



Ex-1005, FIG.5. Havewala also discloses example operations on files that are well known to a PHOSITA, for example an “open command.” Ex-1004, ¶67 (“[T]he application 30 begins the process by issuing the open command with respect to a particular ghosted file 14 on the volume 12, and such open command is immediately received by the file system 16.”). Havewala therefore, in my opinion, discloses that it accepts a request to open a file that has been selected from the singular file system on the first computing device or source/branch server 10.

101. As illustrated by Figure 2 below, Havewala’s singular file system on the first computing device (source/branch server 10) includes file system 16 and storage volume 12, which contains at least a ghosted file 14. *Id.*, ¶41.



Ex-1005, FIG.2.

102. Therefore, in my opinion, Havewala discloses “**accepting a request to operate on a file at a first device** [e.g., Havewala’s open command that is received and accepted by the first computing device or source/branch server 10], **wherein the file is selected from the singular file system** [e.g., ghosted file or non-ghosted file on file system 16] **on the first device** [e.g., computing device/source 10].”

c. 1[B]

103. Limitation 1[B] recites: “modifying the singular file system on the first device to make local files and virtual files appear indistinguishable to the singular file system, the local files and virtual files sharing a same location on the first device.”

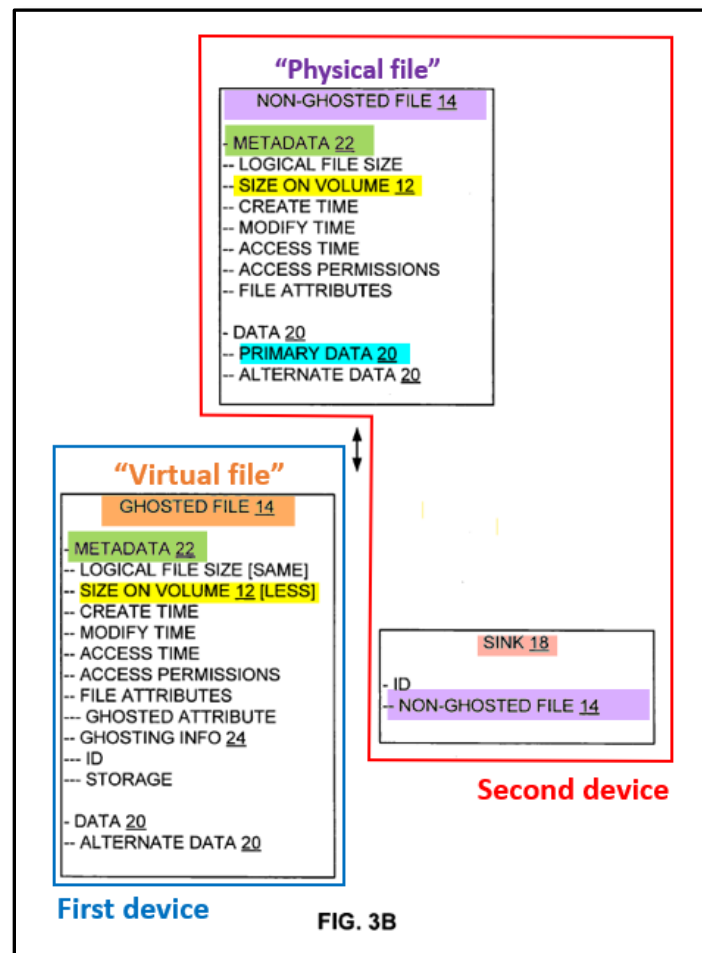
104. In my opinion, Havewala discloses this limitation.

105. *Virtual and Local Files*. The '260 Patent describes virtual files as files that “act as mock files and mock the property of the physical file.” Ex-1001, 8:5-6. These virtual files “appear to the user exactly as local files” and have “all that the physical file has except that it doesn’t take any space on the local device with the exception of the space required for meta-data storage.” *Id.*, 7:66-8:5. Havewala discloses the same kind of virtual files as described in the '260 patent.

106. Havewala’s “ghost files” correspond to the claimed “virtual files.” Havewala discloses that a local, non-ghosted file on a storage volume “include[s] data and metadata relating to the data.” Ex-1005, ¶18. To “ghost” a local file, “at least a portion of the data of the file is removed from the file and stored at an alternate location such that the data **does not occupy substantially any space on the volume** and the file is in a **reduced, ghosted format**.” *Id.*, Abstract; *see also id.* ¶¶12, 16, 18, 19, 44. Further, as shown in Figure 3B (annotated below), Havewala discloses that the **now-ghosted (i.e., virtual) file 14** has the same

metadata 22 (e.g., logical file size, create time, modify time, access time, access permissions, and file attributes) as the corresponding non-ghosted (*i.e.*, physical) file 14, but that the ghosted file uses less storage space on the first device because it does not include, for example, the “primary data 20” of the physical, “non-ghosted” file that is stored on the second device. Ex-1005, FIG. 3B; ¶50 (“Significantly, the ghosted file 14 as resident on the volume 12 of the source 10 contains all metadata 22 from the original, non-ghosted file 14.”); ¶53 (“Accordingly, when the non-ghosted file 14 is in fact ghosted, it may in fact be the case that only the primary data of the data 20 is removed from the remainder of non-ghosted file 14 to form the ghosted file 14, and that only such primary data is stored at the sink 18.”).²

² As I discuss below in Section IX(A)(5), Havewala discloses that “primary data” is transferred to and from the source (first device) and the sink (second device) to ghost and reconstitute a ghosted file. Primary data corresponds to the claimed physical file. Ex-1005, ¶¶52, 53.

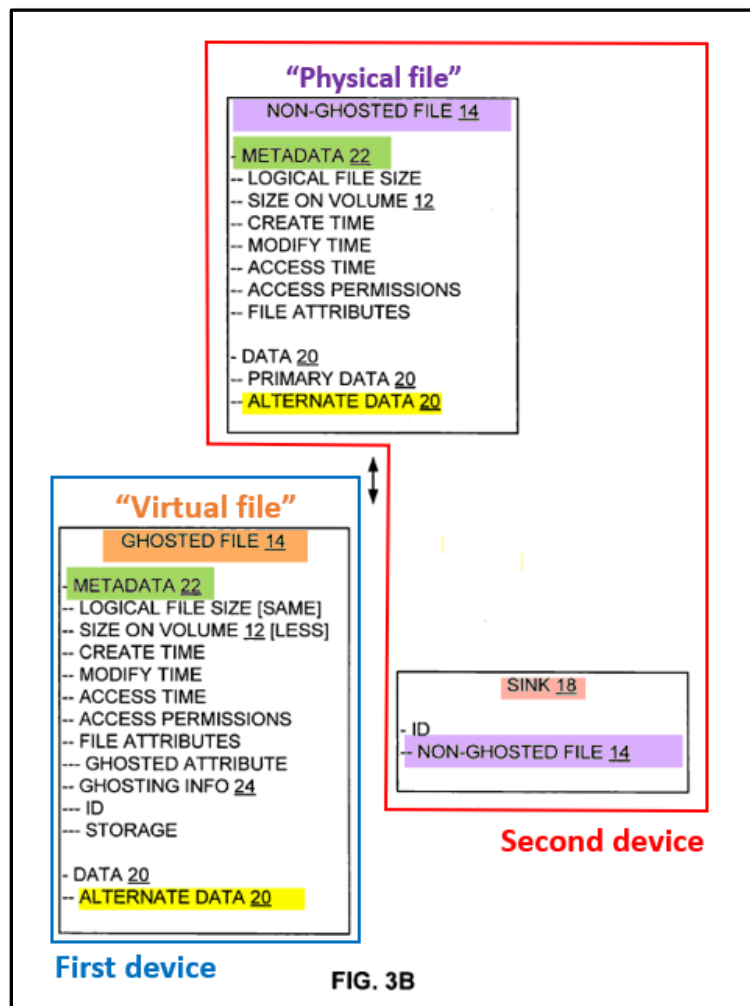


Ex-1005, FIG.3B.

107. *Indistinguishable to File System.* The '260 patent states that “virtual files **appear** in essentially all ways **the same** as a local file, making them **indistinguishable** unless such a distinction is enabled as a separate option.” Ex-1001, 6:38-40, 7:66. “For example, each virtual file includes a **thumb nail** with an icon, it would have the file name, the file size, the file type, the date it was created, the date it was modified, **all that the physical file has** except that it doesn’t take any space on the local device with the exception of the space required for **meta-**

data storage.” *Id.*, 7:66-8:5. The virtual file’s metadata also contains information on whether or not the file is in fact virtual. *See id.*, 8:35-38. (“The software client...checks [the file’s] metadata to **determine whether the selected file is a local file or a virtual file** (remote file).”) Havewala discloses the same.

108. As shown in FIG. 3B, Havewala discloses that a ghosted file “contains **all metadata** 22 from the **original, non-ghosted file** 14.” Ex-1005, ¶50. Similar to the ’260 virtual file, the ghosted file has the same metadata as a physical file such as “logical file size”, “create time”, “modify time”, and other “file attributes.” *Id.*, FIG. 3B. Further, the **ghosted file** on the first device (*i.e.*, **source/branch server 10**) retains the “**alternate data 20**” that is also possessed by the full **non-ghosted (physical) files** on the second device (*i.e.*, **alternate location/branch hub/sink 18**). Like the ’260 patent’s metadata, the “alternate data 20” that is retained on the first device “may include a graphic representation such as a ‘**thumbnail**’ that can be employed when display[ing] a representation of the file 14, perhaps by...the **file system**.” Havewala’s ghosted file’s metadata also includes “ghosting information” indicating that it is a ghosted (*i.e.*, virtual) file with a corresponding physical file which is located at an alternate location. *Id.*, ¶¶19, 20, 50.



109. The '260 patent describes that the file system indistinguishably accesses the virtual and physical files by “**transparently** perform[ing] numerous commands to request, locate and retrieve a physical copy of the virtual file without the knowledge of the requesting application or operating system.” Ex-1001, 8:60-64. To do so, the '260 patent states that “the software client intercepts the operation request,” retrieves the remote file “from the remote device to the local

device,” and performs “the operation...on the file at the local device.” *Id.*, 6:51-59.

110. In my opinion, Havewala discloses the same transparent operations. Havewala discloses that in operating on a ghosted file “[s]uch operations may be performed **transparently**...to the user or application” and “**transparency** is likely preferable to such user or application 30.” Ex-1005, ¶64. “Notably, the application 30 in issuing such open command is **not expected to have any knowledge** that the particular file 14 **is in fact ghosted.**” *Id.*, ¶67. For example, the ghosted file’s “ghosting information...is not typically used by the [] inquiring application 30 or the like” (*id.*, ¶58), and therefore ghosted files appear indistinguishable from physical files to the file system.

111. In making a call to access a ghosted file, the operating system and its file system does not know that the ghosted file is any different to a local file, and, instead, it is the ghosting filter that intercepts the command and locates and reconstitutes the ghosted file:

when a user or application 30 tries to access the removed data 20 of a ghosted file 14, the file system 16 upon receiving a request for such access will discover that the ghosted file 14 does not contain the removed data 20 and thus return an error which the **ghosting filter 26 will intercept**. Upon such interception, then, **the ghosting filter 26 employs the file system 16 to obtain the ghosting information 24**

from the metadata 22 for the ghosted file 14, and based on the obtained ghosting information 24 such ghosting filter 26 triggers reconstituting of such ghosted file 14, as will be set forth in more detail below, so that the request for access may ultimately be honored.

Ex-1005, ¶64.

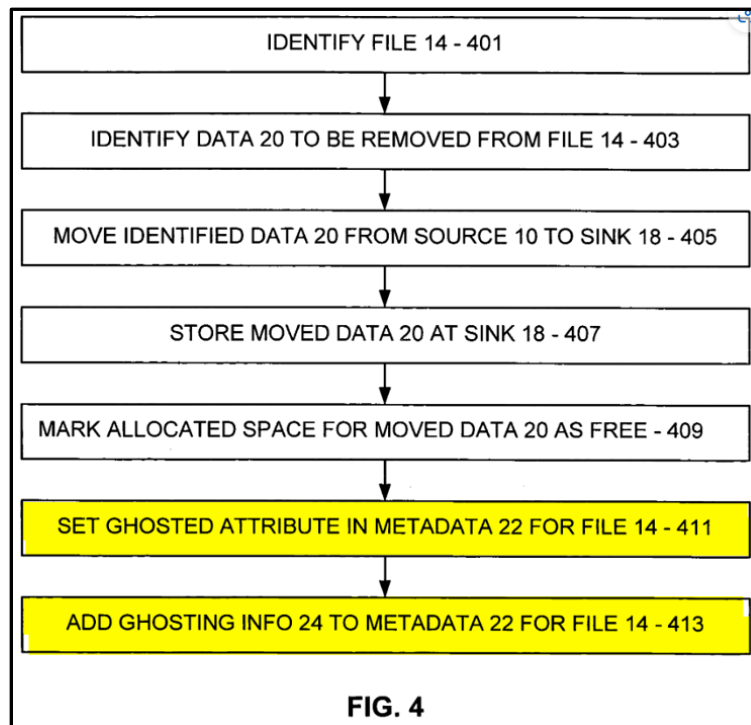
112. To the extent PO argues that Havewala does not disclose virtual and physical files indistinguishable to a file system, a PHOSITA would have understood that Havewala's ghosted and non-ghosted files—which are indistinguishable to the user and application—are necessarily also indistinguishable to the file system. Havewala discloses that the file system uses the same interface for operating on both ghosted and non-ghosted files. In other words, Havewala adheres to the POSIX—standards for how operating systems (which include file systems) should behave to maintain operational compatibility between them. Specifically, the same commands to open files, read data, write data, and close files are used for both ghosted and non-ghosted files. *Id.* ¶66, FIG. 6, FIG. 7. Those operations, for example a request to open the file, are processed by the file system in the same way regardless of whether the file is local or virtual. *See* Ex-1005, ¶68. If the file is virtual, the file system operates in its typical fashion and, before the file system's error command may be acted upon by the file system, operating system, application, or otherwise, Havewala's ghosting manager

and filter have intercepted the error, retrieved the ghosted file's data from the alternate location 18, and proceeded to open the reconstituted (or now-un-ghosted) file on the first device. *See id.* To the extent that PO argues that this is not inherent, a PHOSITA would have at least found it obvious for the reasons I discussed herein.

113. *Modifying The Singular File System To Make Local Files And Virtual Files Appear Indistinguishable To The Singular File System.* The '260 patent states that a “software client” “modifies the **natural behavior** of the operating systems of the individual registered devices in a way that, as it pertains to remote files, **prevents the operating system from doing what it normally does.**” Ex-1001, 7:43-50.

114. In my opinion, Havewala discloses modifying a file system in the exact same way. Havewala discloses that its system may be implemented by “computer-executable instructions, such a program module[], being executed by a computer.” Ex-1005, ¶34. Such program modules “include routines, programs, objects, components, data structures and the like that **perform particular tasks** or implement particular abstract data types.” *Id.* Havewala further discloses a “ghosting manager” and “ghosting filter” that are responsible for performing particular tasks related to, among other things, ghosting and reconstituting files.

Id., ¶¶64, 65. The ghosting manager “perform[s] the bulk of the **ghosting functionality** for the source 10 as is shown in connection with FIG. 4.” Ex-1005, ¶ 65. In particular, as shown in Figure 4, the ghosting manager amends “the metadata 22 for the now-ghosted file 14...to show that such file 14 is now ghosted (step 411),” (*id.*, ¶57), for example to include “ghosting information 24 (step 413).” *Id.*, ¶58.



Ex-1005, FIG.4.

115. Moreover, as discussed above, the operating system and file system are unaware if a file is ghosted because the ghosting filter intercepts an error when a command on the file is issued. The ghosting filter “aid[s]” the operations “that

are performed at the source 10 on behalf of a user or application and with respect to a ghosted file.” *Id.*, ¶64. “Such operations may be performed **transparently** or with notice to the **user or application 30**, although **transparency** is likely preferable.” *Id.*; *see also id.*, ¶66. In performing these operations, the ghosting filter “receives [the] ghosting information” implemented by the ghosting manager to “create[] a buffer and a handle corresponding to the ghosted file 14, and deliver[] the handle to application 30 as the **(normal) response to the open command therefrom.**”). *Id.*, ¶68. Therefore, the ghosting manager and ghosting filter modify the file system such that local and ghosted files appear indistinguishable to the operating system, and subsequently to its file system.

116. **Same location.** Havewala’s ghosted (virtual) files and non-ghosted (local) files share the same location at the source 10 (first device). The ’260 patent describes a “location” as a “directory/folder” within a “native file system (e.g., ‘My Document’, ‘My Pictures’, etc.). Ex-1001, 1:64-2:1. Havewala discloses that “a **ghosted file 14** as present on the volume of a **source 10** is stored in the **same location** on the volume 12 as the **original file 14.**” Ex-1005, ¶50. “Accordingly, a user or the like can browse the **entire volume 12 of the source 10** by way of the file system 16 even if some of the files 14 on the volume 12 **are ghosted** and some of the files 14 on the volume 12 **are not ghosted.**” *Id.*, ¶51. Moreover,

Havewala's file system "organizes the volume 12 according to a **directory** format." *Id.*, ¶50.

117. Therefore, in my opinion, Havewala discloses "**modifying the singular file system on the first device to make local files and virtual files appear indistinguishable to the singular file system** [*e.g.*, Havewala's file system is modified so that local file and the ghosted file have metadata that makes them indistinguishable to the file system], **the local files and virtual files sharing a same location on the first device** [*e.g.*, ghosted and non-ghosted files on the volume 12 of source 10].

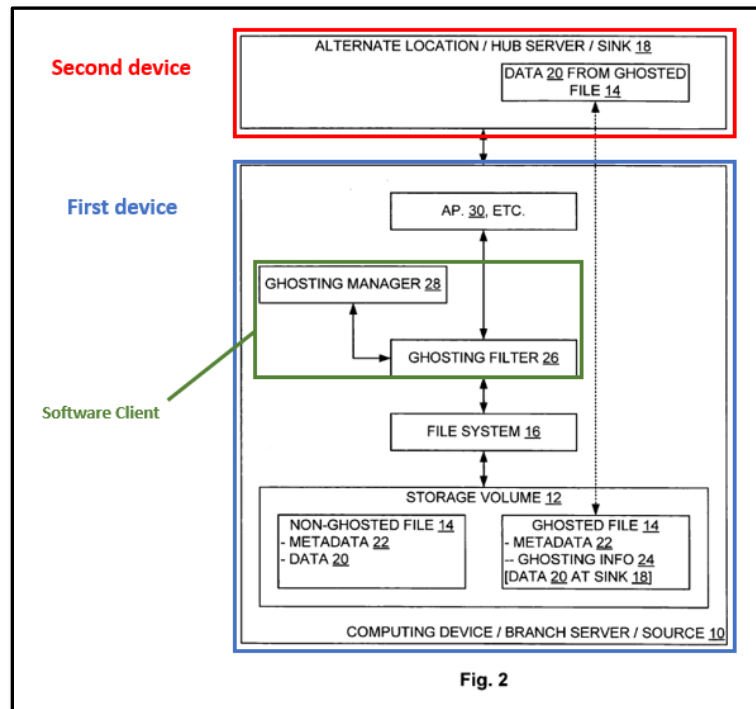
d. 1[C]

118. Limitation 1[C] requires: "intercepting the request by a software client on the first device."

119. The '260 patent states that a "Software Client (or the Client) is a piece of software that is installed and runs continuously on the device." Ex-1001, 4:46-47; *see id.*, 5:47-50. The '260 patent also states that the software client "modifies the natural behavior of the operating systems of the individual registered devices." *Id.* ¶46-50.

120. ***Havewala's Software Client.*** As I discuss in Section XII(A)(1)(c), Havewala discloses "program modules" that "perform particular tasks" in Havewala's system, including those performed by a ghosting filter and ghosting

manager. Ex-1005, ¶¶34, 64, 65, 140; *see id.* ¶34 (“[T]he invention is described in the general context of computer-executable instructions, such as program modules, being executed by a computer...includ[ing] routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types.”). Havewala discloses that the source 10 (*i.e.*, the first device) includes the “ghosting filter” that aids in ghosting files and the “ghosting manager” that “includes additional ghosting functionality and direct access to network resources such as the sink 18.” Ex-1005, ¶¶64, 65. This is also shown in Figure 2, below. A PHOSITA would have recognized that Havewala’s ghosting filter and ghosting manager are “programs” and correspond to the claimed software client because they alter the underlying behavior of standard file system operations, including by intercepting requests to operate on files. Specifically, as the ghost filter and manager intercepts the “open” command (one of the fundamental operations of any traditional file system).



EX-1005, FIG.2.

121. *Intercepting A Request To Operate on a File.* Havewala explains that:

when a user or application 30 tries to access the removed data 20 of a ghosted file 14, the file system 16 upon receiving a **request for such access** will discover that the ghosted file 14 does not contain the removed data 20 and thus return an error which the **ghosting filter 26 will intercept**. Upon such interception, then, the **ghosting filter 26 employs the file system 16 to obtain the ghosting information 24 from the metadata 22 for the ghosted file 14, and based on the obtained ghosting information 24 such ghosting filter 26 triggers**

reconstituting of such ghosted file 14 ... so that the request for access may ultimately be honored.”

Ex-1005, ¶64. Havewala further discloses that “the ghosting filter stores the read command in a queue or the like for awaiting further processing (step 613).” *Id.*, ¶71. Havewala’s ghosting filter—which is part of the software client on the first device—therefore intercepts a request to operate on a file through standard file system behaviors when a user requests access to a ghosted file that is not physically present on the first device.

122. Moreover, Havewala discloses that when the ghosting filter “receives [] ghosting information from such ghosted file” and “[b]ased on such ghosting information 24...the ghosting filter creates a buffer and a handle corresponding to the ghosted file 14, and delivers the handle to the application 30 as the (**normal**) response to the open command therefrom.” *Id.*, ¶68. Havewala further discloses that “when the ghosting filter 26 triggers [] reconstituting of [a] ghosted file 14, the ghosting manager 28 performs the bulk of such reconstituting functionality for the source 10.” *Id.*, ¶25. In sum, the ghosting filter and the ghosting manager work together to allow an application to still operate on a desired file despite a returned error, because the ghosting filter intercepts such error.

123. Therefore, in my opinion, Havewala discloses “**intercepting the request [e.g., ghosting filter 26 intercepting an access request] by a software**

client [e.g., Havewala’s ghosting manager and ghosting filter] **on the first device** [e.g., computing device/source 10].”

e. 1[D]

124. Limitation 1[D] requires: “determining by the software client if the file is physically located on the first device or if the file is a virtual file of a corresponding file physically stored on a second device by reviewing file metadata.”

125. In my opinion, Havewala discloses this limitation.

126. **Havewala Discloses Virtual Files on a First Device and Physical Files on a Second Device.** The ’260 patent further describes a “virtual file” as a file that “act as mock files and mock the properties of the physical file.” Ex-1001, 8:5-6. A claimed virtual file “doesn’t take any space on the local device with the exception of the space required for meta-data storage (less than 1 kilobyte per file).” *Id.*, 7:66-8:5; *see id.* (“each virtual file includes a thumb nail with an icon”).

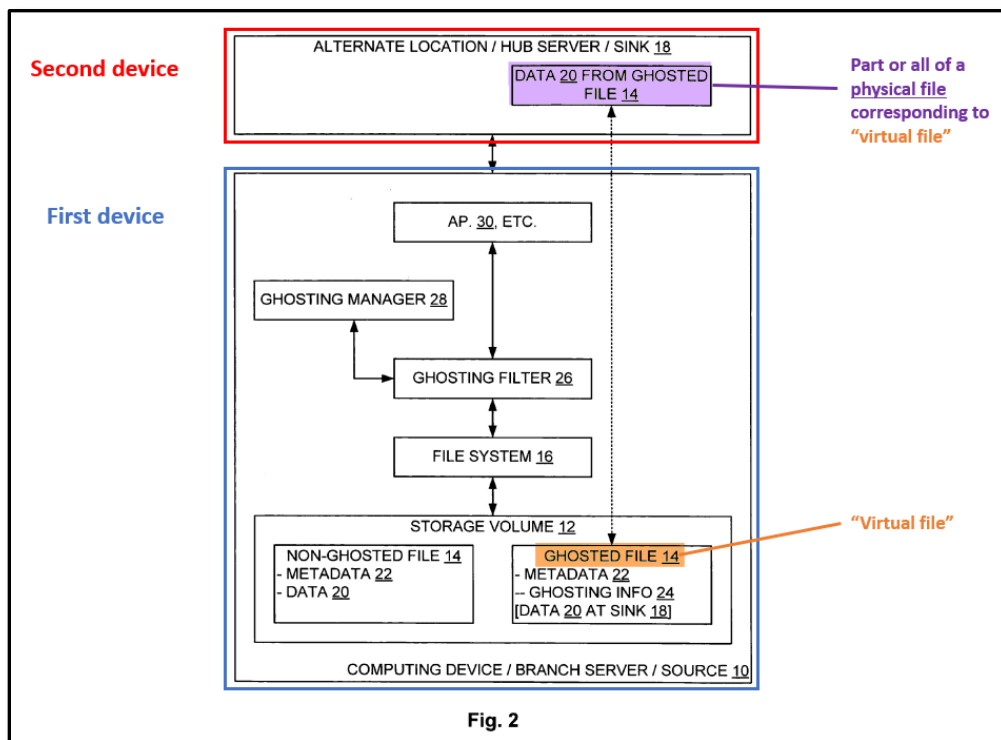
127. As I discussed in Section XII(A)(1)(e), Havewala discloses “a ghosted file 14 [*i.e.*, virtual file] as present on the volume of a source 10” (*i.e.*, the first device). Ex-1005, ¶50. Havewala’s ghosted file, like the ’260 patent’s virtual file, has had its “data [] moved from the file to the alternate location for being stored thereat such that the file is in a reduced, ghosted form.” Ex-1005, ¶¶19-20. As a result, “[s]pace on the volume formerly occupied by the moved data is thus marked

as being free.” *Id.* The data that remains on the source 10 (*i.e.*, the first device) is “alternate data,” such as the “metadata of the now-ghosted file,” including basic file information, “a graphic representation [of the ghosted file] such as a thumbnail” (*id.* ¶52), and ghosting information. *Id.*, ¶¶19-20.

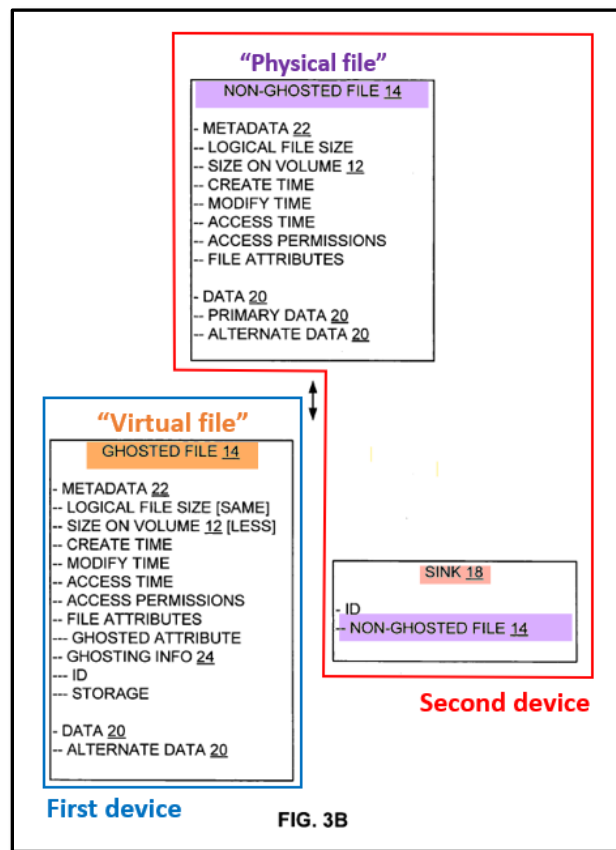
128. The ’260 patent describes a physical file as a “physically stored **data file[]**” that comprises the physical “content” of a virtual file “stored on a different device.” Ex-1001, 5:13-15, 2:7-9, 9:64-65, 8:32-33.

129. Havewala also discloses the same kind of physical file as the ’260 patent. When a non-ghosted file is ghosted (*i.e.*, local file becomes a virtual file), “the **primary data** of the data 20 is removed from the remainder of non-ghosted file 14 to form the ghosted file 14” and “such primary data is **stored at the sink 18.**” Ex-1005, ¶53. Removal of the “primary data” from a non-ghosted file to form a ghosted file at the source may also result in the “**entirety of the [non-ghosted] file [being] stored at the sink 18.**” *Id.*, ¶54. Moreover, “the primary data is the data 20 from the file 14 **as may be employed by an application**” to, for example, open the file. *Id.*, ¶52. Therefore, in my opinion, Havewala’s primary data is a physical file that is physically stored on a second device (e.g., sink).

130. As shown in Figures 2 and 3B, Havewala further discloses that the “primary data” or “entirety” of ghosted file’s corresponding non-ghosted file (*i.e.*, the physical file) is stored at a remote alternate location (e.g., sink).



Ex-1005, FIG.2.

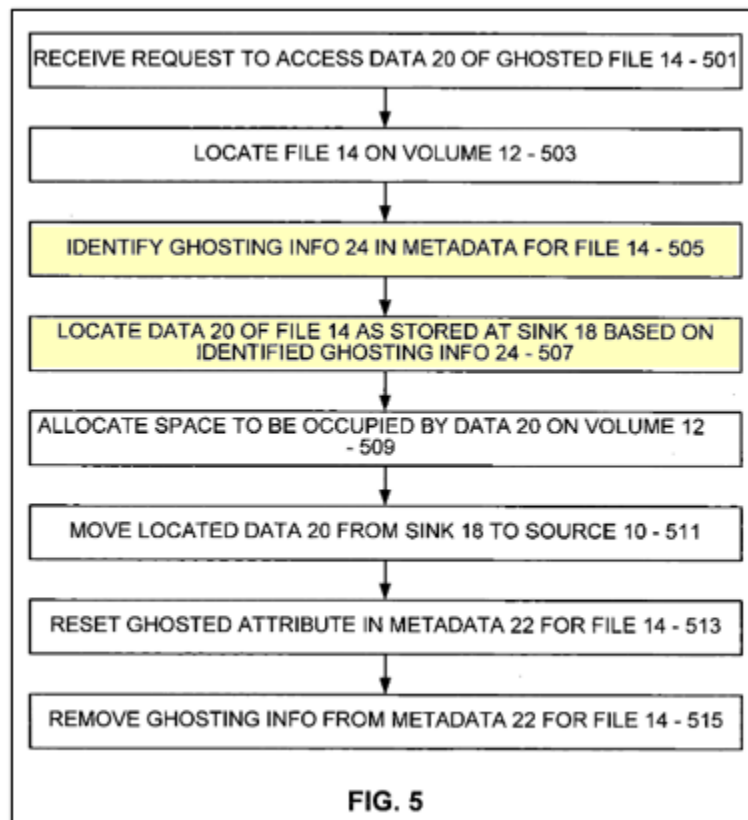


Id., FIG.3B.

131. **Determining Location of Files via Metadata.** Havewala discloses that once a file is ghosted (*i.e.*, a physical file is transferred to another device and a virtual file is left in its place), “the **metadata** of the now-ghosted file is **amended to include ghosting information** including information that may be employed to retrieve the **moved data for the file from the alternate location**” (*id.*, ¶19) as well as information that “may include the ID stored with and identifying the **data 20** at the **sink 18**...[and] **how to locate the sink 18**” (*id.*, ¶58). *See also id.*, ¶57, 58, FIGs. 3A-B. The ’260 patent similarly describes the claimed “virtual file” as

having metadata from which “the software client on Device A determines...the remote device on which [the virtual file] is stored (Device B).” Ex-1001, 8:41-43.

132. As I discussed in Section XII(A)(1)(d), Havewala’s ghosting filter (a component of the software client) intercepts a request to operate on a ghosted file. Havewala further discloses that after such interception, the ghosting filter “obtain[s] the **ghosting information 24 from the metadata 22** for the ghosted file 14” (Ex-1005, ¶64) and that the “data 20 of the file 14 as stored at the sink 18 is **located** based on such identified ghosting information” (*id.*, ¶60). *See also id.*, ¶¶50, 51, 57. These operations are further shown in Figure 5, below, which are “performed with the aid of a **ghosting filter.**” *Id.*, ¶64.



Ex-1005, FIG.5.

133. Therefore, in my opinion, Havewala discloses “**determining by the software client** [e.g., ghosting filter 26] **if the file is physically located on the first device** [e.g., the non-ghosted file 14 on the source device is accessed by the user] **or if the file is a virtual file of a corresponding file physically stored on a second device by reviewing file metadata** [e.g., upon interception, the ghosting filter 26 determines from the ghosted file’s ghosting information 24 of the metadata 22 if the file is a ghosted file stored on the sink device].”

f. 1[E]

134. Limitation 1[E] requires: “wherein a visual representation of the singular file system on the first device is identical to a visual representation of the singular file system on the second device”.

135. In my opinion, Havewala discloses this limitation. Havewala’s system is designed to overcome a prior art “**replication architecture**” where multiple branch servers 10 (*i.e.*, a first computing device) are populated with “non-ghosted copies of all files 14 from a central hub server 18” (*i.e.*, the second computing device). Ex-1005, ¶62. At least to save bandwidth, Havewala’s “**replication engine** may decide to simply **create ghosted copies of the files 14 on the branch servers 10**” (*i.e.*, the first device). *Id.* As I discussed above at length, incorporated here, Havewala’s system as integrated into an operating system utilizing, e.g., Microsoft’s file system, would present a user with the same file system interface in which the user would view the ghosted and non-ghosted files in the same way, and in the same directory/manner (e.g., in Microsoft’s file system), on the first computing device or branch server/source 10 as on the second computing device or alternate location/sink 18. *See* Section XII(A)(1)(a); Ex-1005, ¶¶12-17. Indeed, Havewala explicitly discloses that the ghosted and non-ghosted files are “stored in the **same location**” and that “a user or the like can browse the **entire volume 12 of the source 10** by way of the file system 16 even if

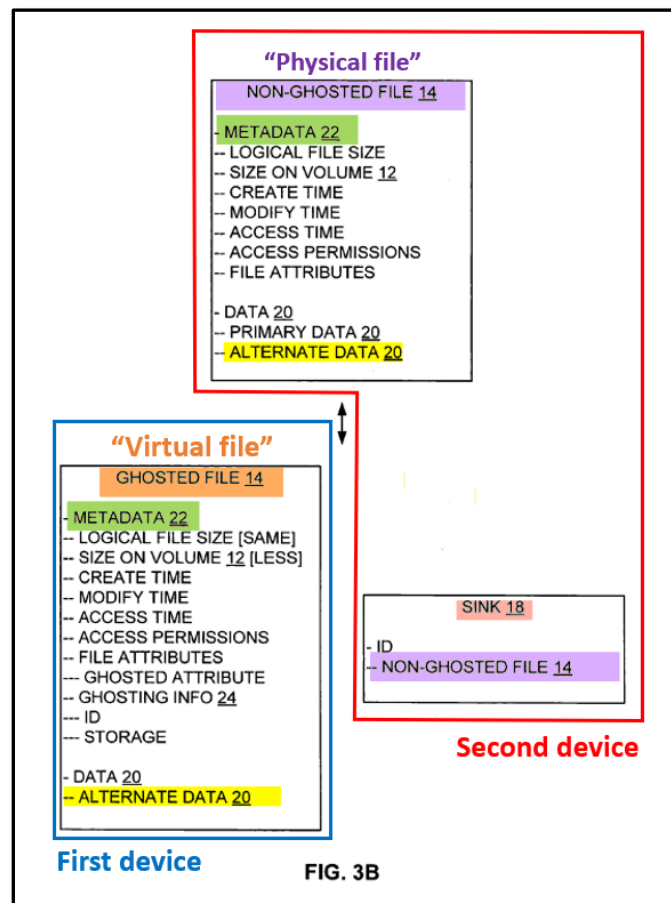
some of the files 14 on the volume 12 **are ghosted** and some of the files 14 on the volume 12 **are not ghosted.**” Ex-1005, ¶50-51.

136. Havewala further discloses that the ghosted file retains the metadata and alternate data (e.g., a thumbnail) of the corresponding non-ghosted file. *See id.*, ¶51-53. “As but one example, **such alternate data** may include a **graphic representation such as a ‘thumbnail’ that can be employed when displaying a representation** of the [ghosted] file 14, perhaps by...the file system 16 **at the source 10.**” *Id.*, ¶52. Therefore, Havewala discloses presenting a thumbnail of a ghosted file at the first computing device (branch server/source 10) in the same way as a thumbnail of a non-ghosted file is presented as they share the same metadata and alternate data. The same is true of the non-ghosted file on the second computing device or alternate location/sink 18. The user viewing files on the alternate location/sink 18 would find all ghosted and non-ghosted files in the same location and directory and would be able to open the file (which is local to the alternate location/sink 18) in the same way and with the same visual representation as it would be presented on the first device.

137. As shown by Figure 3B, below, the ghosted file maintains the same “alternate data” (e.g., including at least the graphic representation or thumbnail) as the corresponding non-ghosted file at the sink (second device). *Id.*, ¶52 This

confirms that a ghosted file and the file system at a branch 10 (first device) is visually represented in the same way as it would be at the hub (second device).

138. Thus, Havewala provides an identical visual representation of the singular file system across both devices in that not only are ghosted files on the first device are in the same locations as non-ghosted files on the second device, but also ghosted files visually look the same way as non-ghosted files. In other words, because users would browse through the file system in exactly the same way if they were on the first and second device and, the same visual information of files (*i.e.*, icons, thumbnails, and the like) is shown across both devices, the visual representation of the file systems on both devices is identical.



Ex-1005, FIG.3B.

139. A PHOSITA would have understood that Havewala’s replication engine creates identical visual representations of the singular file system across the first and second device because copies of ghosted files, containing the same alternate data (e.g., metadata) as its corresponding non-ghosted file, are replicated from the sink to the source.

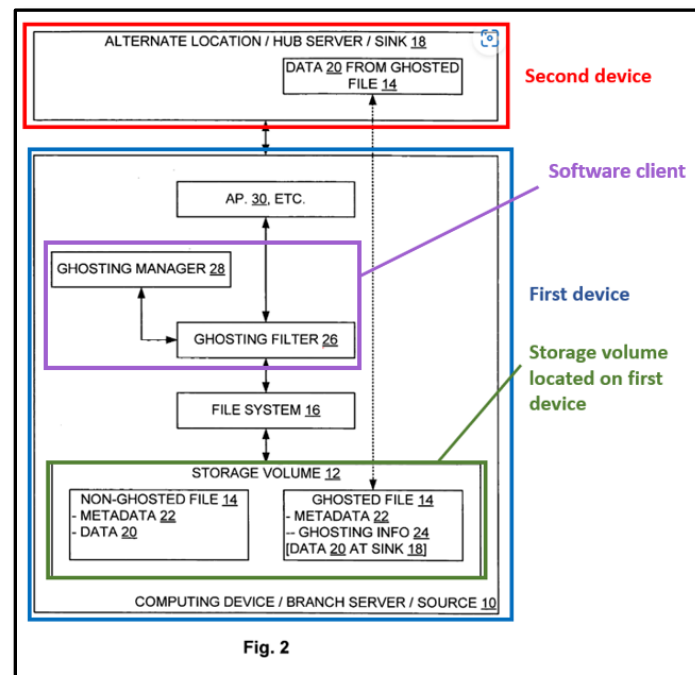
140. As such, Havewala discloses “wherein a visual representation of the singular file system on the first device [e.g., the file system including its files on

the branch/computing device 10] is identical to a visual representation of the singular file system on the second device [e.g., the same file system including its files on the computing device 18].”

g. 1[F]

141. Limitation 1[F] requires: “if the file is the virtual file of the corresponding file physically located on the second device, requesting by the software client on the first device that a peer-to-peer connection be brokered by a server-based web service between the first device and the second device.”

142. In my opinion, Havewala in view of Adams discloses this limitation. Havewala’s software client comprises a **ghosting filter 26** that works alongside a **ghosting manager 28** to request that the data be transferred from the **alternative sink 18 (second device)** to the **source 10 (first device)** to reconstitute the ghosted file. Ex-1005, FIG. 2; *id.* ¶73 (“[G]hosting filter 26 may interface with the ghosting manager 28 to perform higher-level ghosting functionality and **direct access to network resources such as the sink 18**, the ghosting filter 26 issues to the ghosting manager 28 **a request to obtain** some portion of the data 20 for the ghosted file 14 [or the entire non-ghosted file] **from the sink 18**”).



Ex-1005, FIG.2.

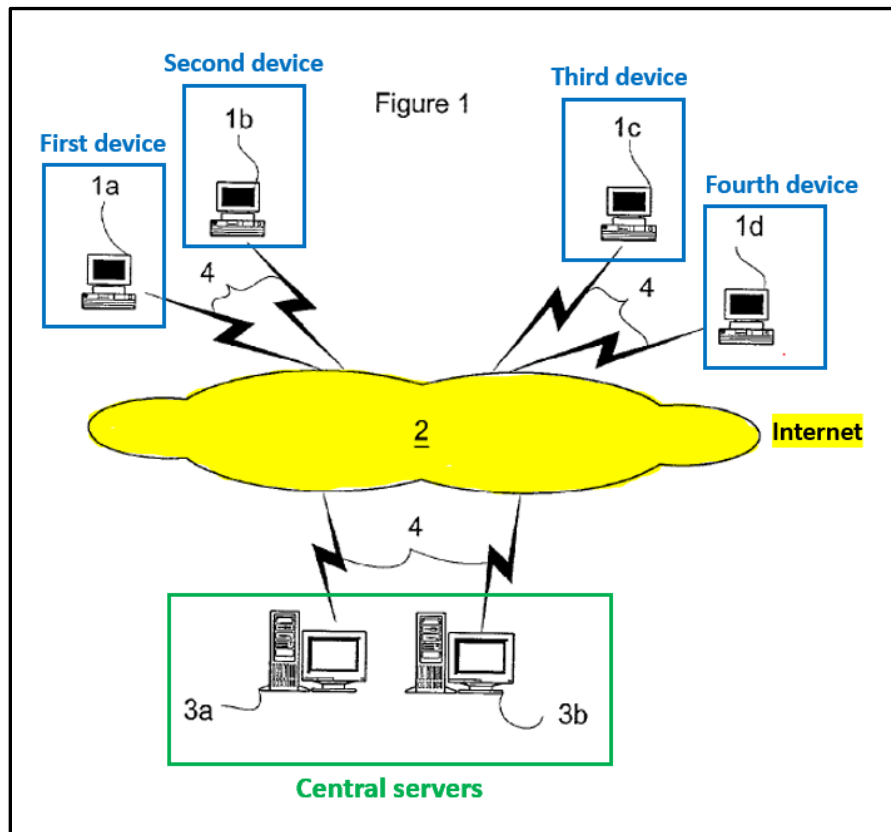
143. The '260 patent describes a “web service” as “a part of the system that **operates on the Internet** and is made up of several types of web technologies including **web servers**, databases, peer-to-peer services (e.g., jingle, BitTorrent, Skype, JXTA), an HTTP Relay and other systems that **interact with the clients.**” Ex-1001, 4:48-52. Havewala discloses that the ghosting manager 28 “includes all necessary functionality” to obtain the corresponding physical file stored on a second device (sink 18). Ex-1005, ¶73. For example, Havewala discloses that the data or file “can then be moved from the sink 18 to the source 10 **by way of an appropriate transport mechanism and conduit.**” *Id.*, ¶60; *see id.*, ¶40. The network described in Havewala is a “wide area network (WAN) 152” including

“networking environments [] commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.” *Id.*, ¶39. A PHOSITA would have understood that common transport mechanisms used for the Internet at the time included peer-to-peer connections brokered by a server-based web service, such as that disclosed by Adams.

144. In particular, Adams discloses a “second client terminal” that is “remote to the first client terminal” and that “the second client terminal **mak[es] a peer-to-peer network connection** to the first client terminal to obtain [a] selected file[,] and subsequent[ly]...download[s the file] to the second client terminal.” Ex-1006, ¶25. Adams explain that its “[p]eer-to-peer connections are made either via a direct TCP/IP connection, or via a relay server that passes information between nodes on the network.” *Id.*, ¶51. Per the 260 patent, a “web service” is described as “a part of the system that operates on the Internet and is made up of several types of web technologies including...peer-to-peer services....” Ex-1001, 4:48-52. In other words, Adams’ peer to peer services are a “web service” as the ’260 patent defines it.

145. As shown in Figure 1, Adams’s “client terminals 1” and “central index servers 3” are “connected to the internet 2.” *Id.*, ¶33. And Adams’s “peer-to-peer network connection” between the first and second terminals is brokered

through a “**central index server**” which connects devices through a TCP/IP connection. Ex-1006, ¶¶25, 51. Adams further discloses that the “software on User B’s computer [] makes a peer-to-peer network connection to User A’s computer to obtain the file.” *Id.*, ¶51.



Ex-1005, FIG.1

146. In my opinion, a PHOSITA would have understood that Adams discloses a server-based web service because Adam discloses explicitly a server that addresses limitations of search engines available at the time that attempted to locate Internet websites of interest based on search terms (Ex-1006, ¶10) and that

users could can manually search for or browse files via the central index server (id., ¶¶15, 16), whose results contain links to download files (id., ¶15). *Id.* ¶15 (Adams discussing “search results list contain[s] a link to at least one file which is located on the further client terminal....”). Adams also discusses links. A PHOSITA would have known what a link is and would have understood it to be web-based. Notably, “links” is shorthand for “hyperlinks” (such as a uniform resource locator or “URL”) and in the context of “searching for files” in a distributed file system, a link is a reference to an address on the web. *See* Ex.1028 (defining URL as an “address that defines the route to a file on the web or any other internet facility”). In other words, Adams not only discloses a peer-to-peer connection—which, per the ’260 patent is a web service—but it also discloses that the service uses links to remote files, which are web addresses, further confirming a web service as the ’260 defines it.

147. Moreover, a PHOSITA would have recognized that Adams’s central index server brokers a peer-to-peer connection between the first and second client terminal because a PHOSITA would understand that brokering a connection simply means assisting in establishing a peer-to-peer session (e.g., by identifying available end point client terminals). Adams discloses such brokering because “the location of the selected file [is] transmitted **from the central index** server to the

second client terminal.” Ex-1006, ¶25. As another example, Adams discloses that the “central index server 3 returns a list of files 6 and **file locations**” and, based on the server’s search results, client terminal 1 “can retrieve a file 6 from the search results by **requesting it from one of the client terminals 1 specified in the search results (a peer-to-peer transfer).**” *Id.*, ¶37.

148. Therefore, in my opinion, Havewala in view of Adams discloses “**if the file is the virtual file of the corresponding file physically located on the second device** [e.g., Havewala’s ghosted file on the sink device], **requesting by the software client on the first device that a peer-to-peer connection be brokered by a server-based web service between the first device and the second device** [e.g., Havewala’s source device modified by Adams to request a peer-to-peer connection be brokered by Adam’s central index server connected to the internet between the source and the sink in order to obtain the data from the non-ghosted file on the sink].”

h. 1[G]

149. Limitation 1[G] requires: “if the peer-to-peer connection is brokered, transferring the corresponding physical file from the second device to the first device.”

150. As I discussed in Section XII(A)(1)(g), which is fully incorporated herein, it would have been obvious to a PHOSITA to combine the teachings of

Havewala and Adams to broker a **peer-to-peer connection** by a server-based web service. In my opinion, Havewala in view of Adams further discloses **transferring the corresponding physical file from a second device to a first device** using a peer-to-peer connection.

151. *Transfer of physical file from the second to the first device.*

Havewala in view of Adams discloses transferring a physical file from the second computing device to the first computing device. *First*, while claim 1 requires transferring a physical file from the second device to the first device, it does **not** require sending a file containing both the data and the metadata. Havewala clearly discloses transferring the data from the second device to the first device, and a PHOSITA would understand that this data is transferred as a physical file. *Second*, to the extent it is argued that the claim somehow requires sending a physical file that includes both the data and the metadata (despite this not being required by the claim), Adams discloses transmitting such a file, and it would have been obvious for a PHOSITA to modify Havewala to transfer such a physical file. I address each of these in turn:

152. *First, Havewala Transmits Data 20.* As discussed in Section XII(A)(1)(e), Havewala discloses that, after a non-ghosted file corresponding to a ghosted file is located (e.g., through the ghosting information metadata associated

with the ghosted file), the physical file (e.g., containing the primary data 20) is **“moved from the sink 18 to the source 10,”** to “reconstitute” the ghosted file (*i.e.*, make the virtual ghosted file a physical non-ghosted file). Ex-1005 ¶¶60, 74, 75, FIG. 6. When Havewala transfers the primary data 20 from the sink to the source, Havewala transfers the data (e.g., Ex-1005, ¶52 (“primary data is the data 20 from the file 14 as may be employed by an application 30”)) but does not send the metadata and the alternate data (which Havewala states could include a thumbnail used by the file system), and updates the metadata information to reflect the transfer (e.g., *id.*, ¶61 (“the metadata 22 for the no-longer-ghosted file 14 is amended to remove the ghosting information 24”)).

153. This is the same as what the '260 patent states is transferred—file content, not all associated metadata. *See* Ex-1001, 6:67-7:6 (discussing removing a “file from memory, so that it is stored only in the single location/device” and that the “file’s metadata is updated...and propagated to all other devices”); 6:9-11 (distinguishing “each identified data file” from “the meta-index [that] includes metadata” about the file) In my opinion, A PHOSITA would understand that the data is transferred as a physical file that is then reconstituted and opened.

154. *Second, to the extent that PO argues this limitation requires sending an entire file (including, e.g., the metadata) and that Havewala does not disclose*

this limitation, it is my opinion that Adams discloses it. Havewala discloses that in some embodiments “the entire file is stored at the sink,” (*i.e.*, Havewala sends the entire file from the source to the sink). Ex-1005, ¶24. As shown in Figure 3, the entire file (e.g., non-ghosted file) resides at the sink and a non-ghosted file contains metadata, primary data, and alternate data. *Id.*, FIG. 3B. Therefore, Havewala discloses transferring entire files (data and metadata) between computing devices.

155. Moreover, Adams discloses that a “client terminal 1a...can **retrieve a file**...by requesting it from the client terminal 1c.” Ex-1006 ¶39; *see also id.* ¶23 (“[T]he selected **file** then being accessed by or **downloaded to the first client terminal from the further client terminal.**”). In other words, just like in Havewala, Adams discloses transferring an entire file between computing devices. In Adams, only entire files are transferred, not in some instances only the data as in Havewala.

156. To the extent transferring a physical file requires transferring the entirety of the file, including the metadata (and that this is not already disclosed or rendered obvious by Havewala), a PHOSITA implementing Havewala would also be motivated to incorporate Adam’s disclosure of transferring entire files for at least the reasons discussed in Section XII(A)(1)(h) above. For example, a PHOSITA would understand that both: (1) there is value in transferring the

metadata in the file (discussed subsequently), and (2) metadata is on the order of 1024 bytes for NTFS (the standard filesystem used in Windows), and thus the overhead in transferring the metadata is unarguably inconsequential (consider that a single TCP packet would likely be sufficient to send the entire metadata in that payload data can be up to roughly 1500 bytes for both ethernet and WIFI communication links).

157. Moreover, a PHOSITA would see the value in transferring metadata along with the file for at least the following additional reasons: file metadata contains (1) timestamp information, which can be used as a straightforward conflict-resolution mechanism (which is disclosed by Havewala, Ex-1005, ¶127) to determine which data should be considered the “latest” in scenarios where the source and the sink both modify the file, (2) size information, which can be used to inform sources if the file size on the sink exceeds the current storage capacity of the source (a problem directly emphasized by Havewala), (3) file attributes, which inform the file system if the file is actually an executable program or should be read-only, and (4) permissions, which can be used to determine if the source would even have the necessary access rights to retrieve the data from the sink.

158. ***Transfer After Peer-to-peer connection brokered by server-based web service.*** Adams discloses that after the central index server transmits “the

location of the selected file...to the second client terminal,” the “second client terminal mak[es] a peer-to-peer network connection to the first client terminal to **obtain the selected file**” and the file is “**downloaded to the second client terminal.**” Ex-1006, ¶25. It is inherent, *i.e.*, necessarily required, that the peer-to-peer connection is brokered or established before the physical file is transferred.

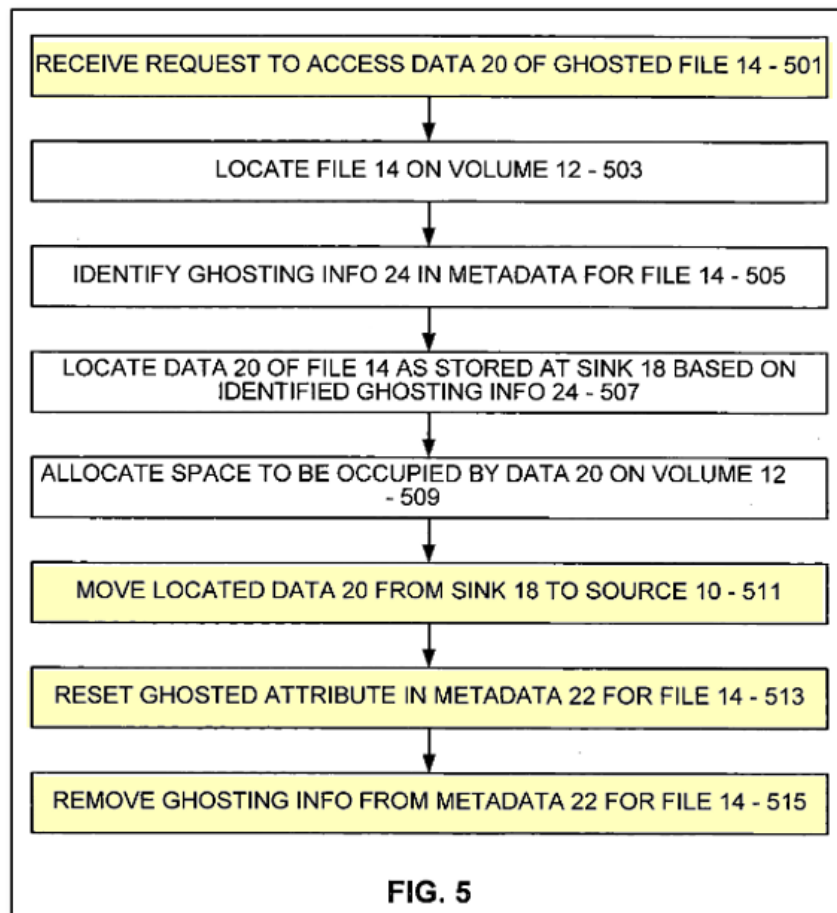
159. Therefore, in my opinion, Havewala in view of Adams discloses “if the peer-to-peer connection is brokered [e.g., once Adam’s peer-to-peer connection is brokered between Havewala’s source and sink], transferring the corresponding physical file from the second device to the first device [e.g., Havewala’s non-ghosted data is sent from the sink to the source to reconstitute the ghosted file, or modified by Adams to send the entire physical file from Havewala’s sink to its source].”

i. 1[H]

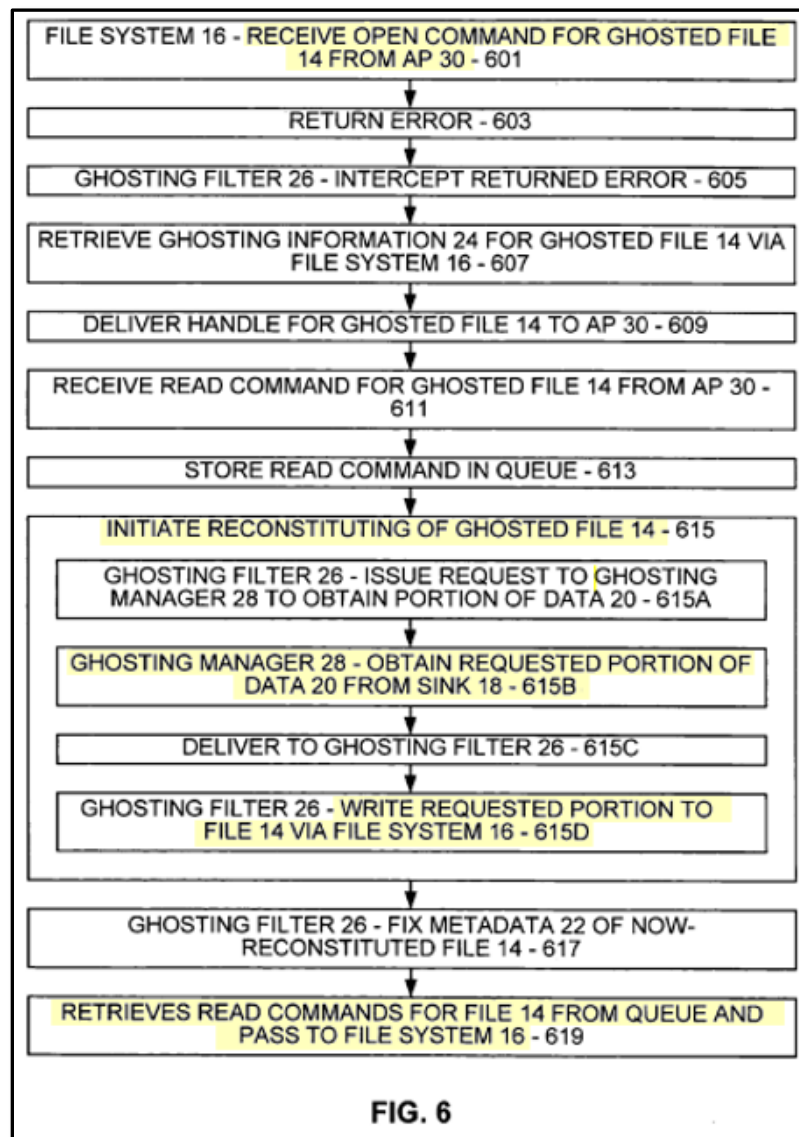
160. Limitation 1[H] requires: “performing the operation on the transferred corresponding physical file at the first device.”

161. In my opinion, Havewala in view of Adams discloses this limitation. Havewala’s operations, for example an open or read command, can be performed on the physical file once back at the source 10 (first device). *See* Ex-1005 ¶64 (“**[O]perations of FIG. 5 [] are performed at the source 10**”); *see also id.* ¶66 (“[A]n application 30 wishing to read such data 20 from such file 14 typically

achieves a function by **issuing an open command to the file system 16** with regard to the file 14, then **issuing a read command to the file system 16** with regard to such opened file 14); *see also id.* FIG. 6. These operations (read and open) are part of the POSIX standard, released in 1988.



Ex-1005, FIG.5.



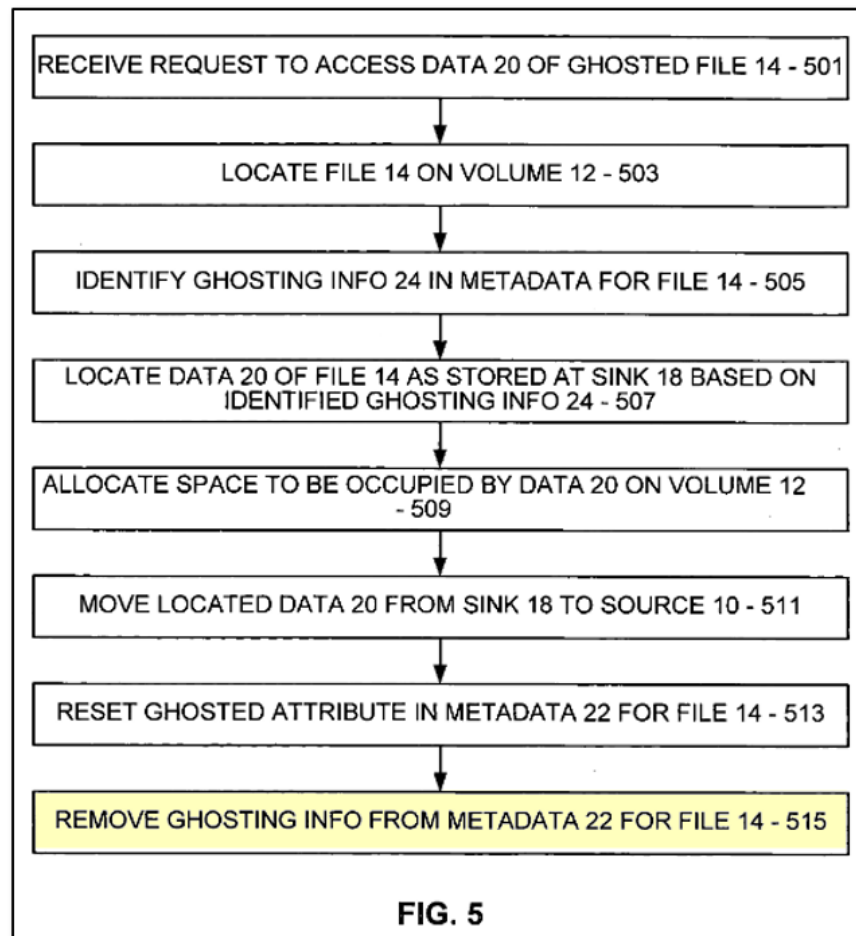
Id., FIG.6.

162. Therefore, in my opinion, Havewala discloses “**performing the operation on the transferred corresponding physical file** [*e.g.*, accessing, opening, reading the file] **at the first device** [*e.g.*, source 10].”

2. Claim 4

163. Claim 4 depends from independent claim 1, which in my opinion is rendered obvious by Havewala-Adams. *See* Section XII(A)(1). Claim 4 further recites “further comprising updating metadata of the singular file system on the first device and the second device to reflect a change in physical location of the file from the second device to the first device,” which, in my opinion, is disclosed by Havewala.

164. As I discussed in detail above in Section XII(A)(1)(c), a ghosted file is reconstituted at the first device when the non-ghosted file or its data is transferred to the first device from the second device. *See id.*, ¶¶51, 102. Havewala discloses updating the metadata across the first and second devices to reflect changes in physical locations of the ghosted and non-ghosted files, including that “the metadata 22 for the **no-longer-ghosted file 14** is **amended to remove the ghosting information 24** (step 515)” as shown in Figure 5. Ex-1005, ¶61, *see also id.*, Claim 17 (“[A]mending the metadata of the reconstituted file to remove the ghosting information”). Such ghosting information includes the location of a ghost file’s corresponding physical file because it “includes information that may be employed to retrieve the data 20 for the file 14 from the sink 18...[and] may include the ID stored with and **identifying the data 20 at the sink.**” Ex-1005, ¶58.



Ex-1005, FIG.5.

165. A PHOSITA would have understood that by removing the ghosting information from the no-longer ghosted (*i.e.*, reconstituted file), the metadata is updated to reflect that the physical file is no longer located on the second device but is now local on the first device. *See id.*, ¶¶51, 58, 61, 80. Specifically, metadata can be managed by the file system, and in this context, updating the metadata of the file system refers to updating (by the file system) the metadata associated with the (no-longer ghosted) file at the source. Ex-1010. For example,

in NTFS, the primary file system of the Windows operating system, all file metadata is stored in a Master File Table (MFT) and updated by the file system; Ex-1011. Moreover, the metadata of the file system at the sink (*i.e.*, second device) is also updated to reflect that the physical file is no longer located at the sink. *See* Ex-1005, ¶¶61, 82.

166. As such, in my opinion, Havewala in view of Adams renders obvious “updating metadata of the singular file system on the first device and the second device [*e.g.*, removing ghosting information in the metadata for the now non-ghosted file] to reflect a change in physical location of the file from the second device to the first device [*e.g.*, after the file state changes on the source from ghosted state to non-ghosted].”

3. Claim 5

167. Claim 5 depends from independent claim 1, which, in my opinion, is rendered obvious Havewala-Adams. *See* Section XII(A)(1). Claim 5 further recites “wherein an operating system of the first device is unaware of files being virtual or physical and believes that all files are physically stored on the first device within the singular file system,” which, in my opinion, is disclosed by Havewala.

168. Havewala discloses applications that are unaware of whether a file is virtual or physical. Ex-1005, ¶¶64, 67. When attempting to operate on a virtual

file, the process of retrieving the corresponding the physical file “**may be performed transparently**” to the application. *Id.* ¶64; *see also id.*, ¶68 (“[T]he ghosting filter 26 creates a buffer and a handle corresponding to the ghosted file 14, and **delivers the handle to the application 30 as the (normal) response to the open command therefrom** (step 609).”).

169. The ’260 patent describes that when an application attempts to operate on a virtual file at the first device, “the **operating system, and in turn the requesting application** (Adobe Reader) has no idea that the request file is not local to Device A.” Ex-1001, 8:54-60. Similarly, Havewala discloses that the application believes that all files are physically stored on the first device within the singular file system because “application 30 in issuing such open command **is not expected to have any knowledge that the particular file 14 is in fact ghosted.**” Ex-1005, ¶67; *see also id.*, ¶68. Further, as I discussed above, the ghosted and non-ghosted files are indistinguishable to the file system. Section XII(A)(1)(c); Ex-1005, ¶64. In my opinion, A PHOSITA would have understood that Havewala’s file system is the part of the operating system that deals with file organization and access, and the operating system is thus unaware of whether Havewala’s files are ghosted or non-ghosted. Specifically, the file system exposes both ghosted and non-ghosted files to the same operations (e.g., read, write, open,

and close) (Ex-1005, Fig. 6, Fig 7, Fig. 9, ¶66), and thus, the operating system is not aware that the requested file is not local, until it receives an error, which is intercepted by the ghosting filter. Ex-1005, ¶64.

170. These operations (read, write, open, and close) are part of the POSIX standard, released in 1988. Ex-1010. POSIX (Portable Operating System Interface) defines a set of standards to for how operating systems should operate to ensure compatibility between them. In other words, applications can request, for example, to open a file and the operating system can simply rely on the file system to request the data from the appropriate storage device. For example, NFS (network file system) is a file system from 1984 specifically designed to operate on files over a network. Ex-1011. It uses the same semantics to operate on files (open, read, write, close, etc.) as any other POSIX-compliant file system and yet obtains data over the network, as opposed to a more common storage medium, such as a hard drive. From the perspective of the operating system, it does not need to be aware at all that NFS operates on remote files; it trusts that the instructions given by NFS will return the appropriate data to the requesting application.

171. As such, in my opinion, Havewala in view of Adams renders obvious **“operating system of the first device [e.g., Havewala’s operating system including its file system] is unaware of files being virtual or physical and**

believes that all files are physically stored on the first device within the singular file system [e.g., file access is transparent to the operating system, file system, and applications and any file access errors are intercepted and dealt with].”

4. Claim 6

172. Claim 6 depends from independent claim 1, which, in my opinion, is rendered obvious by Havewala-Adams. Claim 6 further recites “further comprising: upon completing the operation, pushing the physical file back to the second device; and retaining the virtual file at the first device,” which, in my opinion, is disclosed by Havewala.

173. Havewala discloses that “**a ghosted file 14 that has been reconstituted** or partially reconstituted **can at some point be re-ghosted.**” Ex-1005, ¶101. As discussed above with respect to claim 1, a ghosted file is “reconstituted” when its corresponding physical file is transferred from the second device (e.g., alternative location/sink/branch hub 18) to the first device (source/branch server 10) containing the ghosted file. By “re-ghosting” the reconstituted file the virtual file is retained at the first device, Havewala discloses pushing the entire physical file or data file back to the second device (*i.e.*, sink 18). *Id.*, ¶80, 101, 102. Havewala teaches that the first device (source 10) is set to re-ghost reconstituted files and offload them from the first device in certain situations. *See id.*, ¶102 (“re-ghosting is **triggered by some event**”). For instance, “possible

selection factors for determining whether to **re-ghost** a particular file includes...whether the file 14 has been **modified**,” (*i.e.*, once a file modification is complete) at the source. *Id.*, ¶¶102-129. Havewala also discloses that a triggering event (e.g., lapse of time, file is too large for the storage space, or manual trigger by user) will re-ghost the selected modified file at the source. *Id.*

174. A PHOSITA would further recognize that Havewala’s re-ghosting by pushing the physical file back to the sink would not be necessary if the physical file remained at the sink (*i.e.*, the sink does not keep the physical file after transferring it to the source).

175. As such, in my opinion, Havewala in view of Adams renders obvious “**upon completing the operation** [e.g., after completing the access to the file], **pushing the physical file back to the second device** [e.g., re ghosting the file by sending it back to the sink]; **and retaining the virtual file at the first device** [e.g., and retaining the ghosted file at the source].”

5. Motivation to Combine Havewala and Adams

176. In my opinion, Havewala and Adams are **analogous art**, at least because they are in the **same field of endeavor** as the ’260 patent, e.g., file storage across multiple devices. The ’260 patent and both references are in CPC class G06F and U.S. class 707 (I understand that CPC class and U.S. class are used by the US Patent Office to classify patents based on technology areas, and such

classification information appears on the face of patents and patent publications).

All three references disclose distributed file systems for storing files across multiple computing devices. *See e.g.*, Ex-1001, Abstract (“[V]irtual representation of data means that the files are not actually stored on all devices and thus, with the exception of a single physical location, they do not take up any space (*i.e.*, storage) on any additional device’s hard drive.”); Ex-1005, ¶36, (“[F]ile 14 is maintained on the volume 12 of the branch server 10 in a reduced format without at least some data 20 thereof. Here such data is stored at an alternate location 18 such as a centralized hub server”); Ex-1006, ¶15 (“the present invention provides a system for allowing a user to search for and locate a file in a distributed file sharing computer network”). Moreover, all address similar problems associated with accessing files that are stored across multiple devices in distributed file systems. Ex-1001, 2:7-12 (“[T]here is a need in the art for a system and process that facilitates access to content that physically resides on any one of multiple user devices, *e.g.*, laptop, desktop, personal digital assistant (PDA), cellphone or the like, from any one of the multiple user devices without the need to physically store all content on every device.”); Ex-1005, ¶12 (“[I]t would be useful to be able to ‘ghost’ the non-relevant files at the branch server of the particular branch, such that each non-relevant file remains on the branch server in a smaller, ‘ghosted’ form.

Ex-1006, ¶¶13-14 (related to “selectively facilitate the locating of any specific type of file...relevant to a user’s request” and identifying “a need for a new type of system for and method of distributed file sharing over a computer network.”).

177. Havewala discloses that any suitable transport mechanism known in the art may be used to transfer the physical file from the sink to the source. In my opinion, A PHOSITA looking to implement Havewala’s system would have been motivated to look to other references suitable transport mechanisms for transferring files within distributed file systems, including Adams’ peer-to-peer connections that are also directed to a distributed file system focused on efficiently locating a particular file on a remote device and quickly and securely transferring the file from the remote device to a local device. *See* Ex-1006, ¶¶10-13, 26.

178. In my opinion, A PHOSITA would have had a **reasonable expectation of success** in incorporating Adams’s peer-to-peer connections into Havewala’s system. Prior to the Critical Date, a PHOSITA would have understood that there was a finite number of transport mechanisms for transferring files, and that a peer-to-peer connection was one of the most widely used methods for connecting computing devices. It further would have been simple for the PHOSITA to implement Adam’s central index server to broker connections between devices and establish “[p]eer-to-peer connections [that are] made...via a

relay server that passes information between nodes on the network.” Ex-1005,

¶51. It would have also been readily apparent to a PHOSITA that Adam’s “central index servers 3 are also connected to the Internet 2” (Ex-1006, ¶33), confirming and leading a PHOSITA to the simple implementation of brokering peer-to-peer connections through a server-based web service. Peer-to-peer networks were increasingly popular with the advent of broadband in the early 2000s in that they substantially reduced the high load placed on a central server by tcp directly between peers. Ex-1012.

179. *To the extent transferring a physical file requires transferring the entirety of the file, including the secondary data* (and that this is not already disclosed or rendered obvious by Havewala), a PHOSITA implementing Havewala would also have been motivated to incorporate Adams’s disclosure of transferring entire files for at least the reasons discussed in Section XII(A)(1)(h) above. For example, a PHOSITA would understand that receiving the entire file including the metadata would be advantageous for detecting changes in the file or its access permissions. In my opinion, a PHOSITA would have had a reasonable expectation of success because this would have been a simple modification.

6. Claim 7

180. In my opinion, claim 7 is unpatentable because it is rendered obvious by Havewala in view of Adams. Claim 7 of the '260 patent recites the following.

I address each of these limitations in my analysis below.

7[P] A non-transitory computer-readable storage medium storing a set of instructions that, when executed by a processor, cause the processor to perform operations, comprising:

7[A] accepting a request to operate on a file at a first device, wherein the file is selected from the singular file system on the first device;

7[B] modifying the singular file system on the first device to make local files and virtual files appear indistinguishable to the singular file system, the local files and virtual files sharing a same location on the first device;

7[C] intercepting the request by a software client on the first device;

7[D] determining by the software client if the file is physically located on the first device or if the file is a virtual file of a corresponding file physically stored on a second device by reviewing file metadata,

7[E] wherein a visual representation of the singular file system on the first device is identical to a visual representation of the singular file system on the second device; and

7[F] if the file is the virtual file of the corresponding file physically located on the second device, requesting by the software client on the

first device that a peer-to-peer connection be brokered by a server-based web service between the first device and the second device;

7[G] if the peer-to-peer connection is brokered, transferring the corresponding physical file from the second device to the first device; and

7[H] performing the operation on the transferred corresponding physical file at the first device.

a. 7[P]

181. To the extent that 7[P] is limiting, it is my opinion that Havewala discloses it.

182. Preamble 7[P] requires: “[a] non-transitory computer-readable storage medium storing a set of instructions that, when executed by a processor, cause the processor to perform operations, comprising.”

183. Havewala discloses that the invention is practiced on a “general purpose computing system include[ing] a conventional personal computer 120” (Ex-1005, ¶35) which “may further include...**drives** and their associated **computer-readable media** provid[ing] non-volatile storage of **computer readable instructions**, data structures, program modules and other data for the personal computer 120” (*id.*, ¶36); *see also id.*, ¶34 (“[T]he invention described in the general context of **computer-executable instructions**, such as **program modules**, being executed by a computer.”).

184. Therefore, Havewala discloses “a non-transitory computer-readable storage medium storing a set of instructions that, when executed by a processor, cause the processor to perform operations.”

b. 7[A]-[H]

185. As I show below, 7[A]-[H] collectively are identical to 1[A]-H collectively, and disclosed for the same reasons I discuss above and incorporate herein by reference. *See* Section XII(A)(1)(a)-(i).

Claim 7 Limitation		Claim 1 Limitation	
7[A]	accepting a request to operate on a file at a first device, wherein the file is selected from the singular file system on the first device;	1[A]	accepting a request to operate on a file at a first device, wherein the file is selected from the singular file system on the first device;
7[B]	modifying the singular file system on the first device to make local files and virtual files appear indistinguishable to the singular file system, the local files and virtual files sharing a same location on the first device;	1[B]	modifying the singular file system on the first device to make local files and virtual files appear indistinguishable to the singular file system, the local files and virtual files sharing a same location on the first device;
7[C]	intercepting the request by a software client on the first device;	1[C]	intercepting the request by a software client on the first device;
7[D]	determining by the software client if the file is physically located on the first device or if	1[D]	determining by the software client if the file is physically located on the first device or if

	the file is a virtual file of a corresponding file physically stored on a second device by reviewing file metadata,		the file is a virtual file of a corresponding file physically stored on a second device by reviewing file metadata,
7[E]	wherein a visual representation of the singular file system on the first device is identical to a visual representation of the singular file system on the second device; and	1[E]	wherein a visual representation of the singular file system on the first device is identical to a visual representation of the singular file system on the second device; and
7[F]	if the file is the virtual file of the corresponding file physically located on the second device, requesting by the software client on the first device that a peer-to-peer connection be brokered by a server-based web service between the first device and the second device;	1[F]	if the file is the virtual file of the corresponding file physically located on the second device, requesting by the software client on the first device that a peer-to-peer connection be brokered by a server-based web service between the first device and the second device;
7[G]	if the peer-to-peer connection is brokered, transferring the corresponding physical file from the second device to the first device; and	1[G]	if the peer-to-peer connection is brokered, transferring the corresponding physical file from the second device to the first device; and
7[H]	performing the operation on the transferred corresponding physical file at the first device.	1[H]	performing the operation on the transferred corresponding physical file at the first device.

7. Claim 8

186. In my opinion, claim 8 is unpatentable because it is rendered obvious by Havewala in view of Adams. Claim 8 of the '260 patent recites the following.

I address each of these in my analysis below.

8[P] A client comprising:

8[A] a memory;

8[B] at least one processor configured to:

8[C] accepting a request to operate on a file at a first device, wherein the file is selected from the singular file system on the first device;

8[D] modifying the singular file system on the first device to make local files and virtual files appear indistinguishable to the singular file system, the local files and virtual files sharing a same location on the first device;

8[E] intercepting the request by a software client on the first device;

8[F] determining by the software client if the file is physically located on the first device or if the file is a virtual file of a corresponding file physically stored on a second device by reviewing file metadata,

8[G] wherein a visual representation of the singular file system on the first device is identical to a visual representation of the singular file system on the second device; and

8[H] if the file is the virtual file of the corresponding file physically located on the second device, requesting by the software client on the

first device that a peer-to-peer connection be brokered by a server-based web service between the first device and the second device;

8[I] if the peer-to-peer connection is brokered, transferring the corresponding physical file from the second device to the first device; and

8[J] performing the operation on the transferred corresponding physical file at the first device.

a. 8[P]

187. To the extent the preamble is limiting, it is my opinion that Havewala discloses it.

188. Preamble 8[P] requires: “[a] client comprising.”

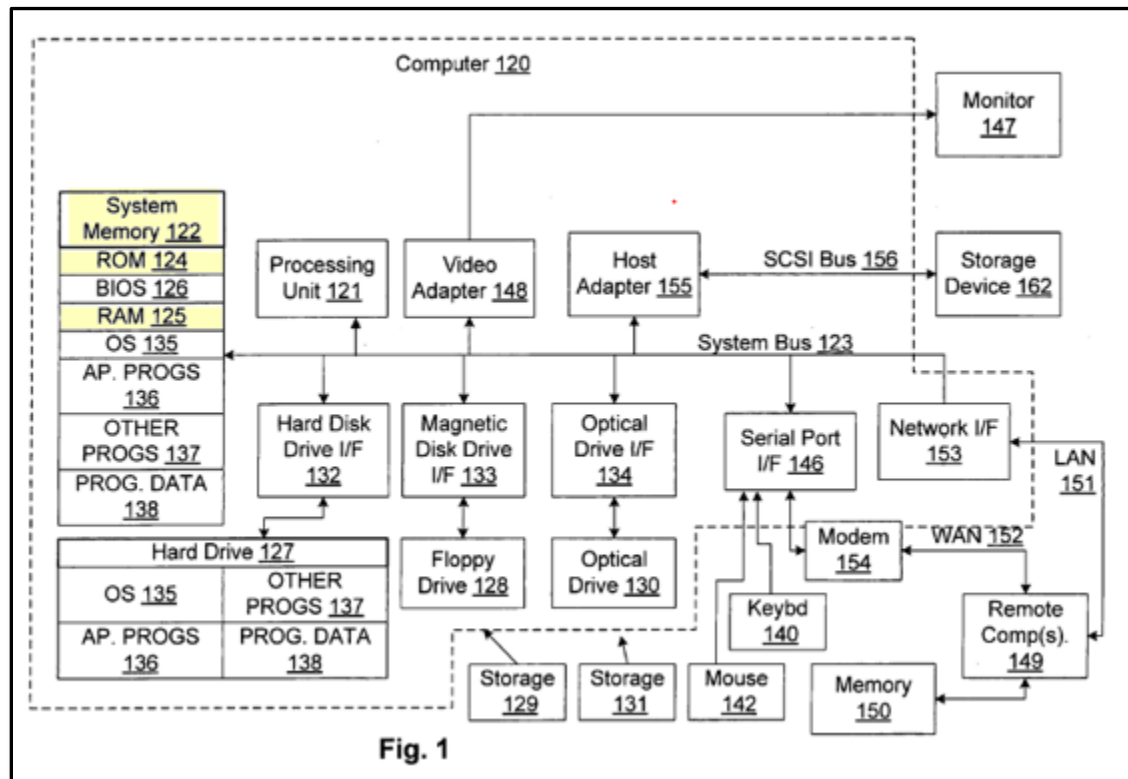
189. Havewala discloses that its invention “is described in the general context of computer-executable instructions, such as program modules, being executed by a computer, such as a **client workstation** or a server.” Ex-1005, ¶34. As discussed above, Havewala’s system is installed on a first and second computing (or client) device.

190. Therefore, in my opinion, Havewala discloses a client for practicing the invention.

b. 8[A]

191. Limitation 8[A] requires: “a memory.”

192. Havewala discloses that its invention is practiced on a “general computing system include[ing]...a **system memory** 122” with the system memory “include[ing] **read-only memory (ROM)** 124 and **random access memory (RAM)** 125.” Ex-1005, ¶35. This memory is further illustrated in FIG. 1 below.



Ex-1005, FIG.1.

193. Therefore, in my opinion, Havewala discloses “a memory.”

c. $\frac{8[B]}{[A]}$

194. Limitation 8[B] requires: “at least one processor configured to:”

195. Havewala’s general purpose computing system “includ[es] a **processing** unit 121.” Ex-1005, ¶35. The general computing system provides the mechanism for implementing the Havewala system. *Id.*

196. Therefore, in my opinion, Havewala discloses “at least one processor (e.g., processing unit) configured to” practice the ’260 claims.

d. 8[C]-[J]

197. As I show below, 8[C]-[J] collectively are substantively identical to 1[A]-[H], respectively, and disclosed for the same reasons I discussed above and incorporated herein by reference. *See* Section XII(A)(1)(a)-(i).

Claim 8 Limitation		Claim 1 Limitation	
8[C]	accept a request to operate on a file at a first device, wherein the file is selected from the singular file system on the first device;	1[A]	accepting a request to operate on a file at a first device, wherein the file is selected from the singular file system on the first device;
8[D]	modifying the singular file system on the first device to make local files and virtual files appear indistinguishable to the singular file system, the local files and virtual files sharing a same location on the first device;	1[B]	modifying the singular file system on the first device to make local files and virtual files appear indistinguishable to the singular file system, the local files and virtual files sharing a same location on the first device;
8[E]	intercepting the request by a software client on the first device;	1[C]	intercepting the request by a software client on the first device;

8[F]	determining by the software client if the file is physically located on the first device or if the file is a virtual file of a corresponding file physically stored on a second device by reviewing file metadata,	1[D]	determining by the software client if the file is physically located on the first device or if the file is a virtual file of a corresponding file physically stored on a second device by reviewing file metadata,
8[G]	wherein a visual representation of the singular file system on the first device is identical to a visual representation of the singular file system on the second device; and	1[E]	wherein a visual representation of the singular file system on the first device is identical to a visual representation of the singular file system on the second device; and
8[H]	if the file is the virtual file of the corresponding file physically located on the second device, requesting by the software client on the first device that a peer-to-peer connection be brokered by a server-based web service between the first device and the second device;	1[F]	if the file is the virtual file of the corresponding file physically located on the second device, requesting by the software client on the first device that a peer-to-peer connection be brokered by a server-based web service between the first device and the second device;
8[I]	if the peer-to-peer connection is brokered, transferring the corresponding physical file from the second device to the first device; and	1[G]	if the peer-to-peer connection is brokered, transferring the corresponding physical file from the second device to the first device; and
8[J]	performing the operation on the transferred corresponding physical file at the first device.	1[H]	performing the operation on the transferred corresponding physical file at the first device.

B. My Opinions as to Claims 2 and 3 based on Havewala in view of Adams and Saridakis.

1. Claim 2

198. Claim 2 depends from claim 1, which in my opinion, is rendered obvious by Havewala-Adams. *See* Section XII(A)(1). Claim 2 further recites “**wherein if the peer-to-peer connection cannot be brokered between the first device and the second device, an alternative connection is established by the server-based web service,**” which, in my opinion, is disclosed by Havewala-Adams in view of Saridakis.

199. As I discussed in Sections XII(A)(1)(d) and XII(A)(1)(c), which are fully incorporated herein, Havewala discloses maintaining a physical file on a second device (e.g., alternate location/sink/branch hub 18) that corresponding to a ghosted file on a first device (e.g., source/branch server 18) that can be transferred to the first device “by way of an appropriate transport mechanism and conduit.” Ex-1005 ¶60; *see also id.* 40, 73. In my opinion, a PHOSITA would have understood appropriate transportation mechanisms at the time included alternative connections where a peer-to-peer connection could not be brokered. Such alternative connections are disclosed in Saridakis.

200. Saridakis discloses a “**peer-to-peer communication channel** between peers” that “permits the **peer device and the another peer device** to send and

receive data.” Ex-1007, Abstract. Saridakis recognizes that in some instances, a peer-to-peer connection cannot be successfully established. For instance, with “strict firewall policies, there is **no straightforward way to establish a P2P connection** between two peers that lie on different side of such a firewall.” *Id.*, 1:64-67. To combat this issue, Saridakis discloses a known “method of **circumventing network obstacles** to provide a peer-to-peer communication channel between peers.” *Id.*, 3:7-9. Saridakis achieves this circumvention (*i.e.*, alternative connection) by “a first peer device communicating with a relay via a network **including an obstacle**, a second peer device communicating with the same relay via a network including another device, and a **server** coupled to the first and second peer devices and **including programmed instructions to carrying out functions** of relaying the communication from the first peer device to the second and vice versa.” *Id.*, 7-17. Moreover, Saridakis’s server is the claimed server-based web service because it is “coupled to the network and communicates using HTTP (hypertext transfer protocol) messages” where the “network 18 can be the Internet.” Ex-1007, 3:62-65. HTTP is a common protocol used to communicate with the World-Wide-Web (often abbreviated as, “the Web”) to retrieve documents, and thus, by definition, the HTTP Server used in Saridakis is “web-based.” Ex-1013.

201. Therefore, Havewala in view of Adams and Saridakis renders obvious **“if the peer-to-peer connection [e.g., Adams’s peer-to-connection between clients via central index server] cannot be brokered between the first device and the second device, an alternative connection is established by the server-based web service [e.g., Saridakis’s HTTP communication].”**

2. Claim 3

202. Claim 3 depends from claim 2, which, in my opinion, is rendered obvious by Havewala in view of Adams and Saridakis. Claim 3 further recites **“wherein the alternative connection is an HTTP relay,”** which, in my opinion, is rendered obvious by Havewala in view of Adams and Saridakis. As discussed I Section XII(B)(1), incorporated herein, Saridakis discloses an alternative connection to broker peer-to-peer connections as an appropriate transportation mechanism for the Havewala system.

203. For example, Saridakis discloses a system for establishing **“peer to peer connections** between personal computers (PCs) and smart phone **despite the obstacles imposed by firewalls”** by way of **“utiliz[ing] an HTTP-based protocol that does message relaying.”** Ex-1007, 2:48-55. The ’260 patent identifies this same reason as a reason to establish an alternative HTTP connection. *See* Ex-1001, 8:52-54 (“In instances where p-to-p connection is not possible, e.g., due to firewall restriction, the web service acts as an HTTP relay server.”). As discussed

in Section IX(A), Saridakis further discloses a server that corresponds to the '260 patent's server-based web service. Moreover, the “**server receives a HTTP request** from the first peer devices. This HTTP request includes data intended for another peer device. The server further **relays** the aforementioned data to the intended peer device **establishing thus a virtual communication channel between the first peer device and the second peer device** to enable sending and receiving of data.” Ex-1007, 3:17-23.

204. Therefore, in my opinion, Havewala in view of Adams and Saridakis renders obvious an “alternative connection [of] an HTTP relay.”

3. Motivation to Combine Havewala with Adams and Saridakis

205. In my opinion, Havewala, Adams and Saridakis are **analogous art** at least because they are in the **same field of endeavor** as the '260 patent, *i.e.*, sharing data across multiple devices. Like Havewala, Adams, and the '260 patent, Saridakis is also in CPC class G06F and discloses “send[ing] and receiv[ing] data” between devices through some form of “communication channel.” Ex-1007, Abstract. A PHOSITA would have recognized that Adams discloses that, “[p]eer-to-peer connections are made via a direct TCP/IP connection, or via a relay server that passes information between nodes on a network.” Ex-1006, ¶51. However, Adams does not provide details on how this relay server would work to support

“passing information between nodes.” Notably, P2P networks became common alongside the popularity of the World-Wide-Web in the early 2000s, and thus, a PHOSITA would look to Saridakis in leveraging the Web-based services (which all clients with Internet service can connect to) to serve as the relay server in the event a direct TCP/IP connection would not be possible. Ex-1012. In my opinion, A PHOSITA would have had a **reasonable expectation of success** in implementing Saridakis’ system to the Havewala-Adams combination. Each reference discloses well known components for web-traffic and modifying the Havewala-Adams combination to include Saridakis is routine and easy for a PHOSITA (or any network engineer).

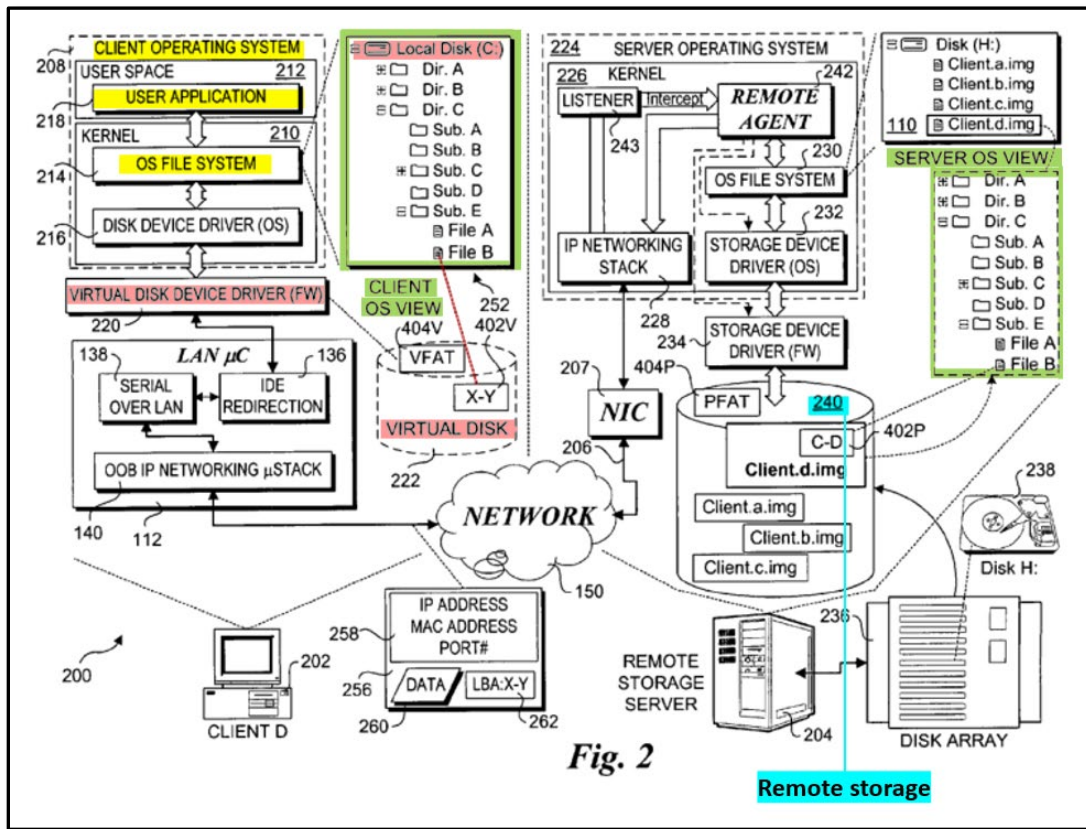
C. My Opinions as to Claims 2 and 3 based on Havewala in view of Adams, Saridakis, and Rothman.

1. Claim 1-8

206. To the extent it is argued by PO that Havewala in view of Adams and Saridakis does not explicitly disclose (i) the “singular file system” recited in 1[pre], (ii) “local files and virtual files appear[ing] indistinguishable to the singular file system” recited in 1[B], and/or (iii) “an operating system of the first device is unaware of files being virtual or physical and believes that all files are physically stored on the first device within the singular file system” as recited by claim 5, it is my opinion that Rothman discloses all.

207. *Claim 1[pre] – Singular file system.* As discussed in Section IX(A)(1) (discussing limitation 1[pre]), Havewala discloses (or at least renders obvious) that its system would have been installed on multiple devices—e.g., branch servers 10 and branch hubs 18—that include the same operating systems and file systems (e.g., Windows file system as modified by Havewala). Once Havewala’s ghosting software is installed, the representations of the file system and files stored therein on both the first and second devices would appear the same. Ex-1005, ¶52.

208. Rothman similarly discloses a system that includes virtual and local files, “wherein the client runs an operating system that is a member of the Microsoft Windows family of operating systems, and the remote storage server runs an operating system that is also a member of the Microsoft Windows family of operating systems”). Ex-1008, claim 12. Rothman discloses that such operating system includes, for example “a conventional Windows file system.” *Id.*, ¶43. As shown in Figure 2, with the same family of operating systema on a client (first device) and remote storage server (second device), the **file system** across both devices is the same (*i.e.*, singular file system).



Ex-1008, FIG.2.

209. To the extent that Patent Owner argues that Havewala does not disclose or render obvious a singular file system or how to implement or structure the singular file system across multiple computers (it does), it is my opinion that a PHOSITA would have been motivated to look to other references concerning how to implement Havewala's ghosting system across a family of computers in, for example, the branch server/branch hub environment as discussed in Havewala.

Ex-1005, ¶¶13-17.

210. A PHOSITA would have been led and motivated to apply the teachings of Rothman because it confirms that Havewala's system would be installed across the related computers utilizing the same family of standard and widely used operating and file system (e.g., Windows and Windows file system as modified by Havewala). A PHOSITA would have also been led to Rothman because Rothman discloses an exemplary display of a file system hierarchy—e.g., Client OS View and Server OS View as shown in Figure 2—demonstrating that the file hierarchies on both devices would be the same. In my opinion, Rothman would further confirm to a PHOSITA implementing Havewala's system that the hierarchal structure of the file system across a family of computers should appear the same on those devices.

211. **Claim 5 – virtual files that appear as local files to an operating system.** Rothman discloses that a “**virtual disk drive** comprising an emulation of a non-existent local (to a client) disk drive is facilitated via out-of-band (OOB) communications with a remote storage server in a manner that **is transparent to an operating system (OS) running on the client.**” Ex-1008, Abstract. Moreover, the “**virtual disk appears to client operating system 208** (and thus to all of user applications 212) **as a local disk drive.**” *Id.* ¶39 *see also, id.*, ¶76 (“In fact, from the operating system's standpoint, the virtual disk actually exists as a local disk.”).

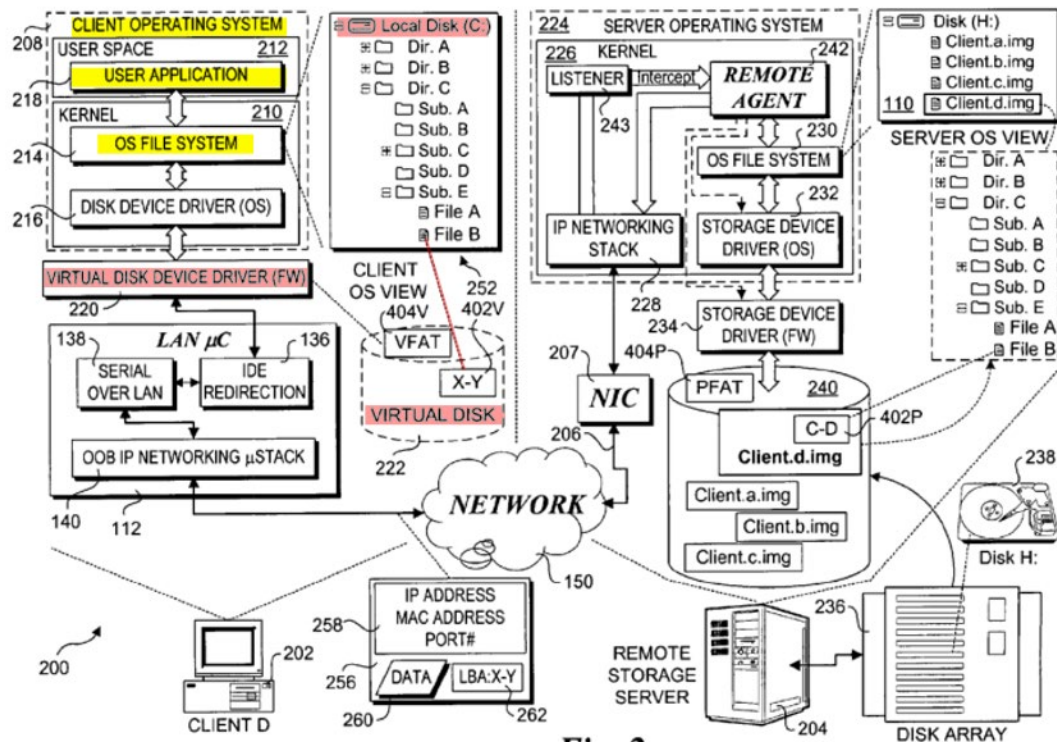


Fig. 2

212. **Claim 1[B] – Virtual and physical files indistinguishable to the file system.** As I discussed above, Rothman discloses virtual files that appear as local files to an operating system. Rothman further discloses that the file system similarly does not distinguish between virtual and local files by explaining that “among the[] core **OS components is an OS file system 214.**” Rothman’s virtual disk drives have “**virtual file[s] ‘stored’**” on the disk. Therefore, in my opinion, PHOSITA would have readily understood that these **virtual files** necessarily **appear local** to the file system and operating system because they **appear as if they reside on a local disk drive** [e.g., Files A and B are appear on the client, but

are actually located on the server]. Specifically, the operations made by the operating system that are applied to the virtual files that reside on the virtual disk are identical to conventional local disks. Ex-1008, ¶40. That is, traditional file system operations, such as reads and writes, can be applied to both local files and virtual files, as shown in Fig. 3a. and Fig 3b.

2. Motivation to Combine Havewala, Adams, Saridakis, and Rothman

213. In my opinion, Havewala, Adams and Rothman qualify as **analogous art** at least because they are in the **same field of endeavor** as the '260 patent, *i.e.*, accessing remotely stored files. Like Havewala, Adams, and the '260 patent, Rothman is in CPC class G06F. Also, like Havewala, Adams, and the '260 patent, Rothman discloses “send[ing] and receiv[ing] data” between devices through some form of “communication channel.” Ex-1007, Abstract.

214. Havewala discloses that its “application 30 in issuing such open command **is not expected to have any knowledge that the particular file 14 is in fact ghosted.**” Ex-1005, ¶67. Havewala further discloses that this “**transparency** is likely preferable to such user or application.” *Id.*, ¶64. In my opinion, A PHOSITA implementing Havewala would look to other references utilizing virtual files for improvements to this preferred transparency, including Rothman. Specifically, a PHOSITA would recognize that providing transparency is preferred

because applications and the operating system do not require additional functionality to, for example, open files. Thus, this transparency increases the compatibility of the Havewala file system across a broad range of applications and is more reliable in that a software developer implementing Havewala would only have to focus on supporting a single file system, thereby reducing the complexity of the software.

215. Rothman discloses that its “virtual disk appears to client operating system 208 (and **thus to all of user applications 212**) as a **local disk drive**.” Ex-1008, ¶39. Specifically, virtual disks appear to applications as a local disk drive because they are often implemented as “lower-level firmware device drivers which provide a layer of abstraction” (*id.*, ¶32), wherein a consistent interface is provided across varying types of disks (*i.e.*, including local and virtual disks). Thus, this layer of abstraction means that applications (and users) can use the same familiar interface to interact with, in this context, both local and virtual disks. Thus, a PHOSITA would look to Rothman to provide a uniform interface such that whether a file is local or remote is transparent to the user, application, and operating system.

216. In my opinion, A PHOSITA would have a **reasonable expectation of success** as modifying Havewala-Adams-Saridakis to include Rothman’s added

transparency is fully compatible with the Havewala-Adams-Saridakis system and easily implemented by a competent OS software engineer familiar with Microsoft Windows.

XIII. CONCLUDING STATEMENTS

217. In my opinion, the challenged claims 1, 4-8 are obvious by Havewala in view of Adams. As such, in my opinion, these claims should be found unpatentable and cancelled.

218. In my opinion, claims 2 and 3 are obvious by Havewala in view of Adams and further in view of Saridakis. As such, in my opinion, these claims should be found unpatentable and cancelled.

219. In my opinion, the challenged claims 1-8 are obvious by Havewala in view of Adams and Saridakis and further in view of Rothman. As such, in my opinion, these claims should be found unpatentable and cancelled.

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

Date: December 14, 2023

A handwritten signature in black ink, appearing to read "Patrick D. McDaniel", written in a cursive style.

Patrick D. McDaniel, Ph.D.