

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

BOX, INC., and DROPBOX, INC.,

Petitioner

v.

TOPIA TECHNOLOGY, INC.,

Patent Owner

Case IPR2023-00427
Patent No. 9,143,561

Case IPR2023-00433
Patent No. 10,067,942

DECLARATION OF PRASHANT SHENOY, PH.D.

I. BACKGROUND AND QUALIFICATIONS.....	3
II. LEGAL FRAMEWORK.....	7
A. Anticipation.....	7
B. Obviousness.....	8
C. Person of Skill in the Art	9
III. OPINION	10
A. Background of the Technology	11
B. Ground 1 of the ‘561 Patent – The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Sigurdsson and Shappell	23
C. Ground 2 of the ‘561 Patent – The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Brown and Hesselink.....	45
D. Ground 1 of the ‘942 Patent – The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Sigurdsson and Shappell	67
E. Ground 2 of the ‘942 Patent – The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Brown and Hesselink.....	90
IV. CONCLUSION.....	113

I. BACKGROUND AND QUALIFICATIONS

1. I am a Distinguished Professor and Associate Dean in the College of Information and Computer Sciences at University of Massachusetts, Amherst. I practice, research, and teach in the area of Distributed Systems and Networking.
2. I hold a PhD. (1998) and an M.S. (1994) in Computer Science from The University of Texas at Austin, where I was awarded the best dissertation award in Computer Science. I also hold a B. Tech (1993) in Computer Sciences from the Indian Institute of Technology (IIT) Bombay, where I was awarded the IIT Bombay Silver Medal for the top ranking student in Computer Science.
3. I have published over 300 technical papers in reputed scientific journals, and those publications have been cited 24,000 times. My research has received several awards, including best paper awards and the ACM Sigmetrics Test of Time Award. I am currently a Fellow at four professional computer societies – Association for Computing Machinery (ACM), Institute of Electrical and Electronics Engineers (IEEE), American Association of Advancement of Science (AAAS), and Asia-Pacific Artificial Intelligence Association (AAIA).
4. I have over 25 years of practical, research, and teaching experience in distributed systems, file systems and management systems, data storage, backup, and networking. I have led research projects and industry collaborations in storage and file systems, web and internet systems, content distribution, streaming media,

database systems, cloud computing, virtualization, RFID and sensor networks, and energy systems. In my 25 years of experience, I have consulted with both large companies and startups in a technical advisory role related to product architecture, IP issues, and applied research. My qualifications are further summarized in my *curriculum vitae*, which is submitted as EX.2014.

5. I have been retained by Plaintiff Topia Technology, Inc., (“Topia”) to provide my independent opinion on certain issues in the matter. I do not have any financial interest in Topia or the outcome of this matter. My compensation in this matter is independent of the substance of my opinions and analysis in this matter or in the outcome of this matter.

6. In connection with the present matters, I have studied and reviewed the following materials:

EXHIBIT	DESCRIPTION
1001	U.S. Patent 9,143,561 (“’561 Patent”)
1002	U.S. Patent 10,067,942 (“’942 Patent”)
1003	U.S. Patent 10,289,607 (“’607 Patent”)
1004	U.S. Patent 10,642,787 (“’787 Patent”)

1005	U.S. Patent 10,754,823 (“’823 Patent”)
1006	U.S. Patent 11,003,622 (“’622 Patent”)
1007	Prosecution History of the ’561 Patent
1008	Prosecution History of the ’942 Patent
1009	Prosecution History of the ’607 Patent
1010	Prosecution History of the ’787 Patent
1011	Prosecution History of the ’823 Patent
1012	Prosecution History of the ’622 Patent
1013	Case No. IPR2022-00782, Paper No. 1, Petition for IPR
1014	Case No. IPR2022-00782, Paper No. 17, Decision Granting Institution of IPR
1015	U.S. Provisional Patent Application No. 60/986,896 (“’896 Provisional”)
1016	U.S. Patent Application Publication No. 2007/0174246 A1 to Johann Tomas Sigurdsson, et al. (“Sigurdsson”)

1017	File History of U.S. Patent Application Publication No. 2007/0174246 A1 to Johann Tomas Sigurdsson, et al. (“Sigurdsson File History”)
1020	U.S. Patent Application Publication No. 2005/0091289 A1 to Michael Shappell, et al. (“Shappell”)
1025	U.S. Patent No. 7,035,847 B2 to David K. Brown, et al. (“Brown”)
1026	U.S. Patent Application Publication No. 2005/0120082 A1 to Lambertus Hesselink, et al. (“Hesselink”)
1028	Declaration of Dr. Ravin Balakrishnan, Ph.D.
Paper No. 3	Petition for IPR of the ‘561 Patent (IPR2023-00427)
Paper No. 3	Petition for IPR of the ‘942 Patent (IPR2023-00433)
Paper No. 17	Decision Granting Institution of IPR of the ‘561 Patent (IPR2023-00427)
Paper No. 17	Decision Granting Institution of IPR of the ‘942 Patent (IPR2023-00433)
2022	Selected Excerpts from “Computer Networks – A Systems Approach (3rd Ed.) (2003) by Larry L. Peterson and Bruce S. Davie”

2023	Selected Excerpts from “Computer Networks – A Systems Approach (4th Ed.) (2007) by Larry L. Peterson and Bruce S. Davie”
2024	Deposition Transcript of Dr. Ravin Balakrishnan, Ph.D. (dated October 27, 2023)

II. LEGAL FRAMEWORK

7. I am a technical expert and do not offer legal opinions. However, I have been informed about certain legal principles regarding patentability and related matters under United States patent law, which I have applied in performing my analysis and arriving at my technical opinions in this matter.

A. Anticipation

8. I have been informed and understand that a patent claim is invalid as “anticipated” if each and every feature of the claim is found in a single prior art reference. I also understand that anticipation requires the presence in a single prior art a description of all elements of the claim arranged as required in the claim. I have been informed that for a reference to be considered as anticipating, it must disclose the relevant technology in a manner such that a POSA (person of ordinary skill in the art) would be able to carry out or utilize the technology that the reference describes without having to undertake undue experimentation.

B. Obviousness

9. I have been informed and understand that a patent claim is not patentable if the claimed invention would have been obvious to someone of ordinary skill in the art at the time the patent application was filed. This means that even if all of the requirements of the claim cannot be found in a single prior art reference so as to anticipate the claim, the claim might still not be patentable if the subject matter in the claim would have been obvious. I also have been informed that a claim would have been obvious when the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to someone of ordinary skill in the art.

10. I have been informed and understand that to prove that a prior art reference or a combination of prior art references renders a patent obvious, it is necessary to: (1) identify the particular references that either alone, or in combination, render the patent obvious; (2) specifically identify which elements of the patent claim appear in each of the asserted references; and (3) explain how the prior art references could have been combined in order to create the inventions defined in the particular claim at issue.

11. I have been informed that, in order to establish that a claimed invention would have been obvious based on a combination of prior art documents, a clear

articulation of the reason(s) why the claimed invention would have been obvious, or why a certain prior art document would have been modified, must be provided.

12. I also have been informed and understand that it is improper to combine references where the references teach away from their combination. I have been informed that a prior art reference may be said to teach away when a person of ordinary skill in the art, upon reading the prior art reference, would have been led in a direction divergent from the path that was taken by the inventors.

13. I have been informed and understand that a final determination of obviousness should also consider secondary considerations, if any, such as commercial success of products covered by the patent. I understand that to date secondary considerations have not been raised in these proceedings.

C. Person of Skill in the Art

14. I have been informed and understand that for a patent claim to have been obvious, the claim must be obvious to someone of ordinary skill in the art at the time of the alleged invention. I have been informed that the factors to consider in determining the level of ordinary skill in the art include (1) the educational level and experience of people working in the field at the time the invention was made, (2) the types of problems faced in the art and the solutions found to those problems, and (3) the sophistication of the technology in the field.

15. It is my understanding that for purposes of this proceeding, the Petitioner is

of the opinion that a POSA in November 2007 would have possessed a bachelor's degree in computer science (or similar degree), and two years of work experience developing software, including at least some experience working with data storage and data synchronization. A person could also have qualified as a POSA with some combination of (1) more formal education (such as a master's of science degree) and less technical experience, or (2) less formal education and more technical or professional experience in the fields listed above.

16. Based on my education, training, and professional experience in the field of the claimed invention, I exceed the level and abilities of a POSA at the time of the claimed invention. Although my qualifications may exceed those of one skilled in the art, my analysis and opinions regarding the Patents have been rendered from the perspective of one skilled in the art at the time of the invention.

III. OPINION

17. In my opinion, claims 1, 3, 4, 8, 10, and 11 (the "challenged claims") of U.S. Patent No. 9,143,561 B2 (Ex. 1001, "the '561 patent") are patentable based on my review of the '561 Patent, the file history, and scope of the references discussed herein and in the petition, the knowledge and skill of a POSA would have had at the time of the invention (around 2007/2008) and priority date of the '561 Patent, and the documents listed in the above-mentioned table.

18. In my opinion, claims 1, 3, 4, 10, 12 and 13 (the “challenged claims”) of U.S. Patent No. 10,067,942 B2 (Ex. 1002, the “’942 patent”) are patentable based on my review of the ‘942 Patent, the file history, and scope of the references discussed herein and in the petition, the knowledge and skill of a POSA would have had at the time of the invention and priority date of the ‘942 Patent, and the documents listed in the above-mentioned table.

19. I have also relied on my considerable academic experience and extensive research projects with industry leaders in the relevant field of technology in forming my opinions as expressed herein.

A. Background of the Technology

20. The ‘942 Patent¹, including the claims thereof, relate to architecture for management of digital files across distributed network. EX1002, Title, Abstract.

21. The ‘942 Patent discloses, a system to perform full-file synchronization across multiple local devices associated with a user through a server, e.g., a user centric architecture.

22. The focus of the ‘942 Patent is to provide a user-centric computer network architecture for file synchronization, where a server automatically receives a modified file from one device associated with a user when the user modifies the

¹ The ‘561 Patent and the ‘942 Patent share a specification, therefore citations are made to the ‘942 Patent.

file and the server **automatically transfers** it to another device **associated with the same user**.

23. The background of the ‘942 Patent describes various problems in connection with computerized file management systems circa 2007/2008 timeframe. EX1002 at 1:22-4:2. As the ‘942 Patent explains, at least as late as 2007, users of modern computing systems were increasingly finding themselves in high-speed networked environments. *Id.*, at 1:24-32. With a recent proliferation of personal computing devices, these users would have several computing devices, such as a desktop computer at home, one at work, and a hand held device, such as a personal digital assistant (PDA), on which they would access their device files. *Id.* A user having multiple computing devices led to problems with managing the user’s files across those various devices. *Id.*, at 1:33-61. Users, or customers, needed and desired a seamless environment for accessing their files across their various electronic devices without having to worry about the details of getting the file, and any of its related components, transferred between those devices. *Id.*

24. Although a user could copy move to have them accessible on the user’s other devices, this often resulted in a proliferation of redundant file copies that created confusion about which was the latest or “official” version of the file. *Id.*, at 1:40-44. This conventional process was tedious to resolve, error prone, and complex. *Id.*, at 1:40-61.

25. Since copying of files across a user's devices was tedious and complex, another approach for users to access files at their devices, was for users to store the file at a central location, such as a file server. EX1002, 1: 63-2:46. At that time there were several automated solutions to allow a user to access the most current version of the user's file from any of their devices, but these too had problems. Id. Remote desktop software, distributed file systems, and file transfer protocol (FTP) systems purported to offer a way for a user to access the most current version of the user's files across their devices, but these methods relied on the files being stored in a single central location, such as the server, rather than locally on multiple devices of a user. Id. Because these solutions depended on the devices of the user being connected to a server through a network to access the user's file, the availability and performance of the network was a limiting factor for accessing and using the file. Id., at 2:19-26.

26. In the timeframe of 2007/2008, connections between a server and a user's local device (e.g., desktop computer, laptop, PDA, etc.) through a communication network, such as the Internet, often were intermittent and not as reliable as they are today. For a user to access the user's file residing on the server the user's device had to be connected to the server through the communication network. In this case, these computerized file management systems would store a document file on the server and not on the user's device. The user's device had to be connected through

the network to the server to access and edit the document. If the network connection was down or unavailable, the user would not be able to access the document file, or if the network was operating slowly, the user might not be able to edit the document file effectively because of the poor performance due to the slow communication connection.

27. There were many technologies for a client and a server to communicate with one another. Some common examples include the Network file system (NFS) protocol, File Transfer protocol (FTP), and HyperText Transfer Protocol (HTTP). The HyperText Transfer Protocol (HTTP) and HyperText Transfer Protocol Secure (HTTPS) are standard application-level protocol used by clients and servers to communicate over the World Wide Web. HTTP and HTTPS (referred interchangeably herein with “HTTP”) runs on top of the TCP/IP protocol (during the 2007/2008 timeframe) and (later) on the QUIC protocol. EX2022, Computer Networks – A Systems Approach, Third Edition (2003), at 651-656. *See also* EX2023, Computer Networks – A Systems Approach, Fourth Edition (2007), at 651- 655. A web browser is one common example of a HTTP client. Using HTTP, a web browser can send a request for a web page to a server, and the web server then processes this request by sending the requested web page back to the browser. EX2022, at 656. Using HTTP, web clients can use a GET request to request many types of data from a web server, including HTML pages and images, which are

different types of files. Id., at 652. Web clients can also use a HTTP PUT request to send or upload files to a server using HTTP. In this manner, client and server can use HTTP to exchange files with one another. EX2022, at 650-655.

28. Specifically, the HTTP protocol is a request-response protocol, in which HTTP servers use the client-pull architecture, with the client-pull architecture being the one where a client (e.g., web browser) sends HTTP requests to the server and the server sends HTTP responses after processing each request. See [RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1 \(ietf.org\), section 1.4; see also EX2022, at 655.](#)

29. An HTTP server plays the role of a server in a client–server model by implementing the server part of the HTTP protocol. An important feature of an HTTP server, as standardized during the 2007/2008 timeframe was, and is that the HTTP server waits for incoming client requests and for each request it responds by providing the requested information or performing a requested task. Id., at 652-653, 655.

30. To load a web page, a browser first requests an HTML page from the server. Once the HTML page is fetched from the server, it may contain one or more embedded images. Id., at 650-652. The browser then requests each image from the server via additional HTTP requests. The text content in the HTML page and the images are then displayed to the user. Id.

31. An advertisement (also referred to as “ads” herein) is simply an image that is embedded in an HTML page. Once a browser fetches the HTML page, it then requests all embedded images, including advertisements from the server via additional HTTP requests. In some cases, the embedded advertisement image is fetched from a different web server (e.g., a third-part ad server), but it still involves using HTTP to request and fetch the ad image from that server.

32. Therefore Dr. Balakrishnan’s testimony that “HTTP protocol does allow ... for [ads] to be done in response to a client request, but also to send [ads] directly to the client from a server” stating that ads are sent automatically from the server without any client request is incorrect. EX2024, at 63:20-24; see also *id.*, at 55:1-17. The web browser makes explicit requests for each advertisement embedded into a web page, just as it does for fetching other images contained in the web page. This process follows the standard operation of HTTP, where a client browser makes a request and the server responds to the request by sending the requested information. The server does not “automatically” send or push content such as ads to a browser since HTTP did not have such functionality and was a pull-based request-response protocol.

33. Dr. Balakrishnan incorrectly identifies ads or advertisements on webpages as an example of information a server may “automatically” send to the client without a request. EX2024, at 55:1-17.

34. A POSA would know that the standard HTTP protocol practiced during the 2007/2008 timeframe did not support push-based functionality where the server could “automatically” send data to a client without the client making requests for data.

35. Even when there is an already established connection or an open connection between an HTTP server and a client, a POSA would know that the HTTP server could not push data to the client prior to receiving an explicit HTTP request (e.g., an HTTP GET request) for that data.

36. Under the HTTP protocol in effect at the time of the invention, i.e., the 2007/2008 timeframe, an HTTP server (e.g., web server or an ad server) could not “automatically” push data or files without a request from a client.

37. Claim 1 of the ‘561 Patent Recites:

1. [preamble] A system, comprising:

[1a] a first electronic device configured to selectively execute a first application, the first electronic device being in communication with a second electronic device and a third electronic device, each associated with a user wherein the first electronic device is configured to:

[1b] device a copy of a first electronic file automatically transferred from the second application when the user modifies a content of the first electronic file; and

[1c] wherein the first electronic device is further configured to receive from a third application executable on the third electronic device a copy of a second electronic file automatically transferred from the third

application when the user modifies a content of the second electronic file; and

[1d] wherein the first application is further configured to automatically transfer the modified first electronic file copy to the third electronic device to replace an older version of the first electronic file stored on the third electronic device with the modified first electronic file copy having the content modified by the user; and

[1e] automatically transfer the modified second electronic file copy to the second electronic device to replace an older version of the second electronic file stored on the second electronic device with the modified second electronic file copy having the content modified by the user;

[1f] wherein the second application automatically transfers the copy of the modified first electronic file to the first electronic device upon determining that a save operation has been performed on the modified first electronic file.

Claim 8 of the '561 Patent Recites:

8. [preamble] A method implementable in an electronic system having a storage component, the electronic system being coupled to a first electronic device having stored thereon a first electronic file and a second electronic device having stored thereon a second electronic file, the first electronic device and the second electronic device each being associated with a user, the method comprising:

[8a] receiving from the first electronic device a copy of a first electronic file modified by the user, the copy of the modified first electronic file being automatically provided to the electronic system

by the first electronic device when a content of the first electronic file is modified by the user;

[8b] receiving from the second electronic device a copy of a second electronic file modified by the user, the copy of the modified second electronic file being automatically provided to the electronic system by the second electronic device when a content of the second electronic file is modified by the user;

[8c] automatically transferring the modified first electronic file copy to the second electronic device to replace an older version of the first electronic file stored on the second electronic device with the modified first electronic file copy having the content modified by the user; and

[8d] automatically transferring the modified second electronic file copy to the first electronic device to replace an older version of the second electronic file stored on the first electronic device with the modified second electronic file copy having the content modified by the user;

[8e] wherein the first electronic device automatically transfers the copy of the modified first electronic file to the electronic system upon determining that a save operation has been performed on the modified first electronic file.

38. Claim 1 of the '942 Patent recites:

1. [preamble] A system comprising:

[1a] a first electronic device, associated with a user, configured to:

[1b] receive, via a first application at the first electronic device, a copy of a modified first electronic file from a second application at a

second electronic device associated with the user, wherein the modified first electronic file copy is automatically received from the second application responsive to the user modifying a content of the first electronic file;

[1c] determine whether the first electronic device is in communication with a third electronic device;

[1d] automatically send, via the first application, the modified first electronic file copy to a third application at the third electronic device responsive to the determination that the first electronic device is in communication with the third electronic device and responsive to receiving the modified first electronic file copy from the second electronic device;

[1e] receive, via the first application, a copy of a modified second electronic file from the third application at the third electronic device associated with the user, wherein the modified second electronic file copy is automatically received from the third application responsive to the user modifying a content of the second electronic file;

[1f] determine whether the first electronic device is in communication with the second electronic device; and

[1g] automatically send, via the first application, the modified second electronic file copy to the second application at the second electronic device responsive to the determination that the first electronic device is in communication with the second electronic device and responsive to receiving the modified second electronic file copy from the third electronic device,

[1h] wherein, responsive to sending the modified first electronic file copy to the third electronic device, an older version of the first

electronic file stored on the third electronic device is automatically caused to be replaced with the modified first electronic file copy such that the modified first electronic file copy is stored on the third electronic device in lieu of the older version of the first electronic file, and

[1i] wherein, responsive to sending the modified second electronic file copy to the second electronic device, an older version of the second electronic file stored on the second electronic device is automatically caused to be replaced with the modified second electronic file copy such that the modified second electronic file copy is stored on the second electronic device in lieu of the older version of the second electronic file.

Claim 10 of the '942 Patent recites:

10. [preamble] A method comprising:

[10a] receiving, via a first application at a first electronic device, a copy of a modified first electronic file from a second application at a second electronic device associated with a user, wherein the modified first electronic file copy is automatically received from the second application responsive to the user modifying a content of the first electronic file;

[10b] determining whether the first electronic device is in communication with a third electronic device;

[10c] automatically sending, via the first application, the modified first electronic file copy to a third application at the third electronic device responsive to the determination that the first electronic device

is in communication with the third electronic device and responsive to receiving the modified first electronic file copy from the second electronic device;

[10d] receiving, via the first application, a copy of a modified second electronic file from the third application at the third electronic device associated with the user, wherein the modified second electronic file copy is automatically received from the third application responsive to the user modifying a content of the second electronic file;

[10e] determining whether the first electronic device is in communication with the second electronic device; and

[10f] automatically sending, via the first application, the modified second electronic file copy to the second application at the second electronic device responsive to the determination that the first electronic device is in communication with the second electronic device and responsive to receiving the modified second electronic file copy from the third electronic device,

[10g] wherein, responsive to sending the modified first electronic file copy to the third electronic device, an older version of the first electronic file stored on the third electronic device is automatically caused to be replaced with the modified first electronic file copy such that the modified first electronic file copy is stored on the third electronic device in lieu of the older version of the first electronic file, and

[10h] wherein, responsive to sending the modified second electronic file copy to the second electronic device, an older version of the second electronic file stored on the second electronic device is automatically caused to be replaced with the modified second

electronic file copy such that the modified second electronic file copy is stored on the second electronic device in lieu of the older version of the second electronic file.

B. Ground 1 of the ‘561 Patent – The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Sigurdsson and Shappell

39. In my opinion, Sigurdsson does not disclose that “the first application [executed on the first electronic device] is further configured to **automatically transfer** the modified first electronic file copy to the third electronic device” where a server automatically receives a modified file from one device associated with a user when the user modifies the file and the server **automatically transfers** the modified file (i.e., without a request from the client for the modified file) to another device **associated with the same user** because the disclosures in Sigurdsson explicitly requires a request from the client to initiate data transfer to another device **associated with the same user**.

40. Specifically, considering the disclosure of Sigurdsson as a whole, Sigurdsson requires a server to receive a request from a client device before the server transmits missing content to the client device. Throughout its disclosure, Sigurdsson describes that its system is implemented as a HTTPS client/server operating based on request/response process. As stated earlier, the standard HTTP protocol practiced during the 2007/2008 timeframe did not support push-based

functionality where the server could “automatically” send data to a client without the client making requests for data. Therefore a POSA would not read Sigurdsson as disclosing “automatically transferring the modified first electronic file copy to the third electronic device.”

41. Petitioner argues that Sigurdsson’s two “alternative embodiments” of steps in the sequence diagram of Fig. 6 “at least suggests the automatic send of a modified file from the server to Client C.” Paper No. 3, at 23. In making these arguments, the Petitioner relies on statements Sigurdsson makes in ¶ 68 and ¶ 80, in isolation and without considering the reference as a whole, to support this assertion. *Id.*

42. A POSA, considering the reference as a whole, would not have understood the discussion of the alternate embodiment of step 606 described in ¶68, and the alternate embodiment of step 620 described in ¶80 to be divorced from the accompanying steps in Fig. 6.

43. These alternative embodiments are part of the sequence of the “method for sharing and synchronizing content using a peer-to-peer and server architecture of Fig. 4 when a client that is offline comes online.” EX1016, at ¶ 21. The reference describes this as enabling “peer-to-peer content sharing and synchronization using a combination of peer-to-peer and ‘drop-box’ techniques.” *Id.*, at ¶ 62. The reference further explains that the “peer-to-peer techniques can be used by clients

that currently are online to share the content directly between clients,” and “the server can be used as a drop box for the content” to supply that content to an offline client once it comes online again. Id.

44. A POSA would understand that paragraphs 61 through 82 of Sigurdsson describes the sequence shown in Fig. 6 for performing the method of using the peer-to-peer and client/server architecture of Fig. 4. Both ¶68 and ¶80 are part of this sequence. To extract only these two paragraphs, or even a single sentence from ¶68, from the sequence fails to consider the teachings of the reference as a whole. EX1016, at ¶¶ 61-82.

45. Sigurdsson explains that in step 606 Client A posts content to another client (Client B) to initiate the peer-to-peer content flow. Id., at ¶ 65. Client B, in step 608, stores the content and could propagate the content directly to Client C, if that client was online. Id. In the example shown in Fig. 6, however, Client C is offline so Client B propagates the content to Server 106 instead. Id. Sigurdsson goes on to explain how Server 106 transmits the content to Client C, when that client comes online, through steps 612-626. Id., at ¶¶68-80.

46. The alternate embodiment of step 606 described in ¶68, shown by the dashed line between Client A in step 606 and the Server in step 610, merely concerns how the content can be transferred from Client A directly to the Server without first going through Client B. While ¶68 states that the “Sever 106 can then

transmit the content to the remaining client peers, such as Clients B and C” it in no way teaches or suggests that Server 106 *automatically* transfers the content to a client. On the contrary, the very next paragraph begins the discussion of *how* Server 106 transfers the content, through steps 612 to 626, when Client C comes back online.

47. A POSA would understand that Sigurdsson at ¶ 68 discloses that the alternative embodiment of step 606 is part of the overall sequence for handling synchronization *when a client is offline* when a transfer of content is to take place. As Sigurdsson explains, with the peer-to-peer and server architecture of Fig. 4 “[t]he peer-to-peer techniques can be used by clients that are currently online to share the content directly between clients” and “the server can be used as a drop box for the content until the offline client comes online again.” EX1016, at ¶61.

48. Considering the Sigurdsson reference as a whole, a POSA would have considered Sigurdsson’s alternative embodiment of step 606 as limiting the transfer of the content to using step 620 and its associated steps 612-620 and 626-630. This is because the alternative embodiment of step 606 is for use with the sequence of Fig. 6 which is for the case of Client C being offline at the time of the transfer. If Client C was online, Sigurdsson teaches that the transfer would be performed using the peer-to-peer technique that does not involve transferring

content from the server. In this case where both clients are online the transfer would not be performed by the server using the “drop box” technique.

49. Similarly, step 626, discussed in ¶80, also is part of the sequence of Fig. 6 which concerns an offline client coming online. EX1016, at ¶21. Step 626 is performed to “Provide Missing Content to Client C if Clients A and B are Offline.” Id., at Fig. 6 Step 626. This concerns when *both* the first and second clients are offline “at the time of the third client’s request for missing content.” Id., at ¶79. Petitioners and the Board rely on the statement in ¶80 that “the Server 106 sends the Client X Content 410 to Client C 406,” for teaching or suggesting that Server 106 automatically transfers the content to Client C. But this statement cannot be considered in isolation without considering the teachings of the whole reference. A person of ordinary skill in the art would not understand the statement in ¶80 to teach or suggest Server 106 automatically transferring the modified content to Client C.

50. While it is true that Sigurdsson discloses that the Server 106 can transmit the content to the remaining client peers, such as the Clients B and C, Sigurdsson is clear that this transmission of the content by the server 106 is performed *only* after receiving a request for the content from the Clients B and C. Id., at ¶¶61-82.

51. Fig. 6 of Sigurdsson is reproduced below:

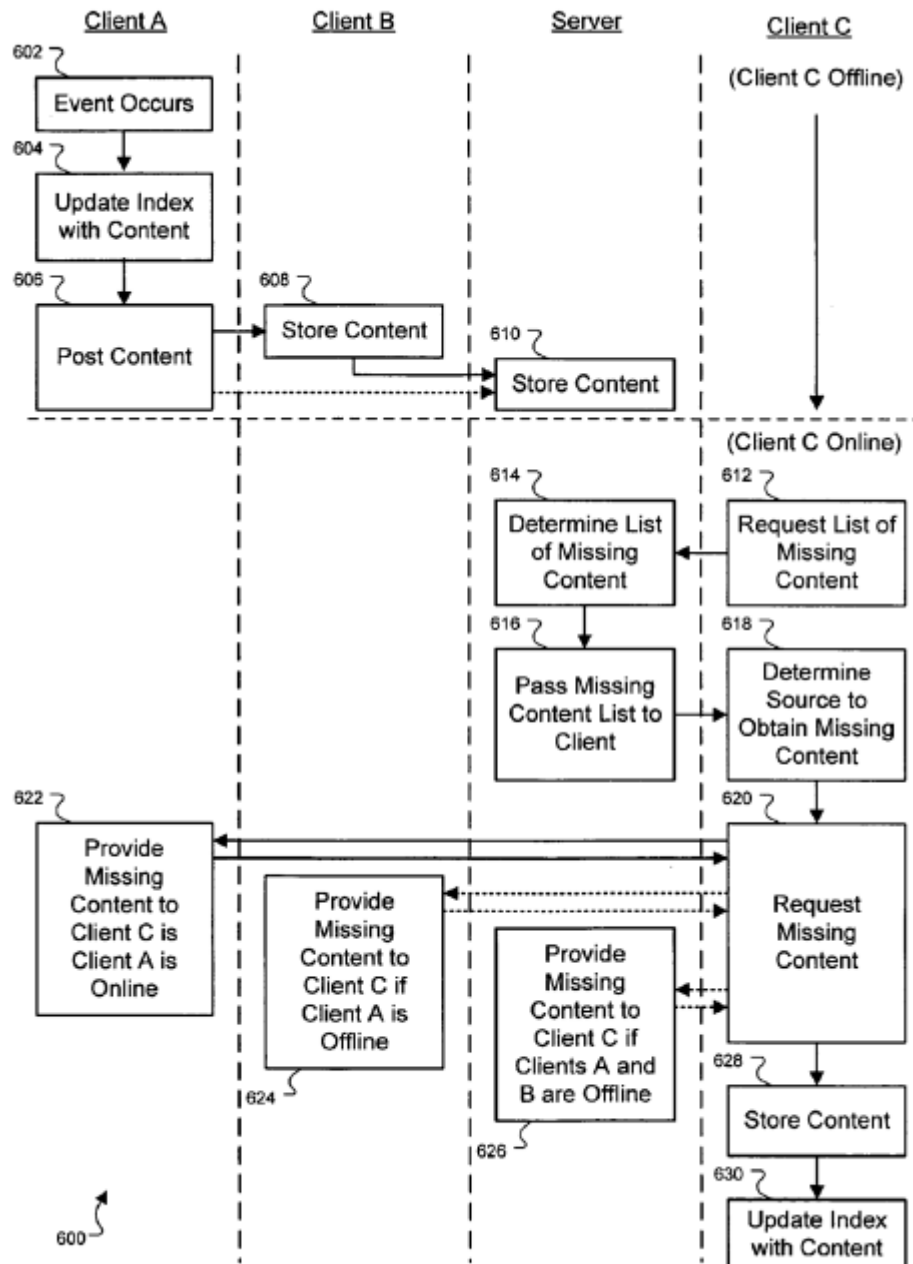


FIG. 6

52. In connection with Fig. 6, Sigurdsson explains the following at ¶¶ 61-62 (emphasis added):

FIG. 6 is another sequence diagram showing an illustrative method for sharing and synchronizing content using a mixed peer-to-peer and server architecture of

FIG. 4 **when a client that is offline comes online.** In this case, the system 400 automatically detects that a particular client is offline. In one implementation, determination of whether a client is online or offline can be made by monitoring polling by the client. **If no request is received from a client, then the client is assumed to be offline.** Similarly, the Server 106 can maintain a table of client IDs and whether each client is online or offline. **When a client comes online, it can notify the Server 106 that it is online,** and the Server 106 can update its table of clients with the online status.

The system 400 can enable peer-to-peer content sharing and synchronization using **a combination of peer to-peer and “drop box” techniques.** The peer-to-peer techniques can be used by **clients that are currently online** to share the content directly between the clients. However, in order for the system 400 to supply the content to an offline client, the server can be used as a drop box for the content until **the offline client comes online again.**

EX1016, at ¶¶ 61-62 (emphasis added).

53. Here, Sigurdsson establishes that Fig. 6 illustrates two scenarios, *i.e.*, a first scenario when **clients are currently online** and a second scenario when the **offline client comes online again.** According to Sigurdsson, when the **clients are**

currently online, a peer-to-peer technique is used by the clients to share the content directly between the clients (step 606 according to a first embodiment described in ¶65). Alternatively, when a **client is offline, and then comes online**, a “drop box” technique is used by the client *to request* missing content stored in the server 106 (step 606 according to an alternative embodiment described in ¶ 68).

54. Specifically, according to one embodiment in which **clients are currently online** (*i.e.*, first scenario), Sigurdsson at ¶65 discloses that “In step 606, the content originating on the first client is posted to a second client, thus initiating a peer-to-peer content flow.” EX1016, at ¶ 65. In another embodiment in which **client is offline, and then comes online** (*i.e.*, second scenario), Sigurdsson at ¶68 discloses that “An alternative embodiment of step 606 bypasses the second client and sends the data directly to the server, as shown by the dashed line pointing from the step 606 to the step 610.” *Id.*, at ¶ 68.

55. From the above description, it is clear that Sigurdsson at ¶65 already contemplates the scenario suggested by the Board, that is, the first scenario in which the client is already online and Sigurdsson at ¶ 65 unambiguously states that the peer-to-peer technique should apply (“In step 606, the content originating on the first client is posted to a second client, thus initiating a peer-to-peer content flow.”)

56. Therefore nothing in the alternative embodiment of step 606 discloses nor suggests to a POSA that the server 106 automatically transfers the content 410 to the client C 406 without receiving a request from the client C 406.

57. In fact, in the second scenario where the content is transmitted from a server (i.e., client-server scenario), Fig. 6 of Sigurdsson requires the following steps in order:

- “In step 612, the third client **requests** a list of missing content from the server. For example, the Client C 406 **requests** a list of missing content from the Server 106 when it connects to network that links the server and other clients, such as the Internet”. (§69).
- In response, “In step 616, the server passes the list of missing content to the third client (e.g., Client C), which had been offline.” (§73).
- In step 618, the third client uses the list it received from the server to **determine where it can obtain the missing content**. The missing content may exist on other peer clients, the content may be stored at the server, or the content may be stored at both the client and the server. For example, referring to FIG. 4, the Client C 406 may determine that it can synchronize its content by requesting the Client X Content 410 (which is the Client A's content in this case) from the Client A102. In the current example for FIG. 6, the content happened to originate on the Client A 102, but a client can request content from any client that has a copy of the missing content, regardless of where the content originated. (§74).
- In step 620, the third client **requests** the missing content from the first client, provided the first client is online at the time. For example, referring to FIG. 4, the Client C 406 **requests** the Client X Content 410

from the Client A102 if the Client A is currently networked to the Client C. The content requested is based on the time range corresponding to the time stamps of content resident on the Client A102 but not resident on the Client C 406. (¶75).

- In step 626, the server sends the missing content to the third client. For example, referring to FIG. 4, the Server 106 sends Content 410 to the Client C 406.

EX1016, at ¶¶ 69-80 (emphasis added.).

58. This focus on a client request for missing content in the context of client-server scenario is further evident throughout the disclosure of Sigurdsson. For example, at ¶9 discloses (emphasis added):

[0009] In a second aspect, a computer system having one or more servers is described. The system includes a table manager module to receive an indication from a first client that a user has performed an action on a file that is unrelated to a transfer of the file. The indication including content extracted from the file and a metadata value assigned to the content. The system also includes a data table to store the content extracted from the file on the first client and the metadata value, an interface to receive from a second client a request for content that is associated with one or more metadata values within a specified metadata value range, and a selection module to initiate transmission of the content to the second client if the metadata value associated with the content is within the specified metadata value range.

EX1016, at ¶ 9 (emphasis added).

59. Additionally, Sigurdsson at ¶ 37 discloses (emphasis added):

[0037] In step 212, **the second client polls** for updates, representing content generated at other clients. For example, referring to FIG. 1, the Client B104 can poll the Server 106 for new content. Here, the new content can be the Client A Content

108 (e.g., the resume text) stored on the Server 106 but not yet received by the Client B 104.

[0040] In step 218, the client **requests** the updated content that it needs. For example, in FIG. 1, the Client B104 sends a request to the Server 106 for the missing Client A Content 108. The request may be based on a time range as discussed above.

EX1016, at ¶¶ 37 and 40; *see also* Fig. 2.

60. Therefore, in every case involving a server 106 transmitting content to the Client C, Sigurdsson requires the server 106 to receive a request from the Client C before the server 106 transmits the content 106 to the Client C.

61. A POSA would consider Sigurdsson's disclosure to limit the transfer described for the alternative embodiment of step 606 to step 620 and its associated steps 612-618 and 626-630. Specifically, Sigurdsson teaches that in a situation where the client (Client C) is not offline (e.g., is online) when the transfer is to occur, the "peer-to-peer techniques can be used by clients that that are currently online to share the content directly between the clients," rather than using the "drop-box" technique implemented using a server. Ex. 1016 ¶62. But even if Sigurdsson were read to have one client send the modified content to the server to synchronize with another client that was online at the time, this hypothetical example would fall outside the scope of claims 4 and 11. Claim 4 expressly recites that "the first application is configured to store the modified first electronic file

copy to the memory device associated with the first electronic device when the third electronic device is not in communication with the first electronic device.”

Similarly, claim 11 recites that “the second electronic device is not in communication with the electronic system.” As such, the hypothetical scenario in which the client is already online does not render features of claims 4 and 11 obvious.

62. Furthermore, as discussed above in detail, Sigurdsson in Fig. 6, and at ¶79 unambiguously states that a client who was offline must request for the missing content from the server (“Client C 406 requests the Client X Content 410 from the Server 106.”). As such, Sigurdsson cannot and does not disclose “automatically transferring the modified first electronic file copy to the third electronic device.”

63. In my opinion, Sigurdsson is directed to a pull mechanism, and fails to automatically transfer the modified first electronic file copy to the third electronic device associated with the user.

64. The requirement of **automatically transferring** the modified first electronic file copy to the third electronic device requires the claimed method to not *merely* have the file end up at the third electronic device, but **automatically transferring** the modified first electronic file copy to the third electronic. That is, in Sigurdsson, the client 406 (e.g., the third electronic device) first requests a list (of missing content), determines a source content in the list, and then requests for the content

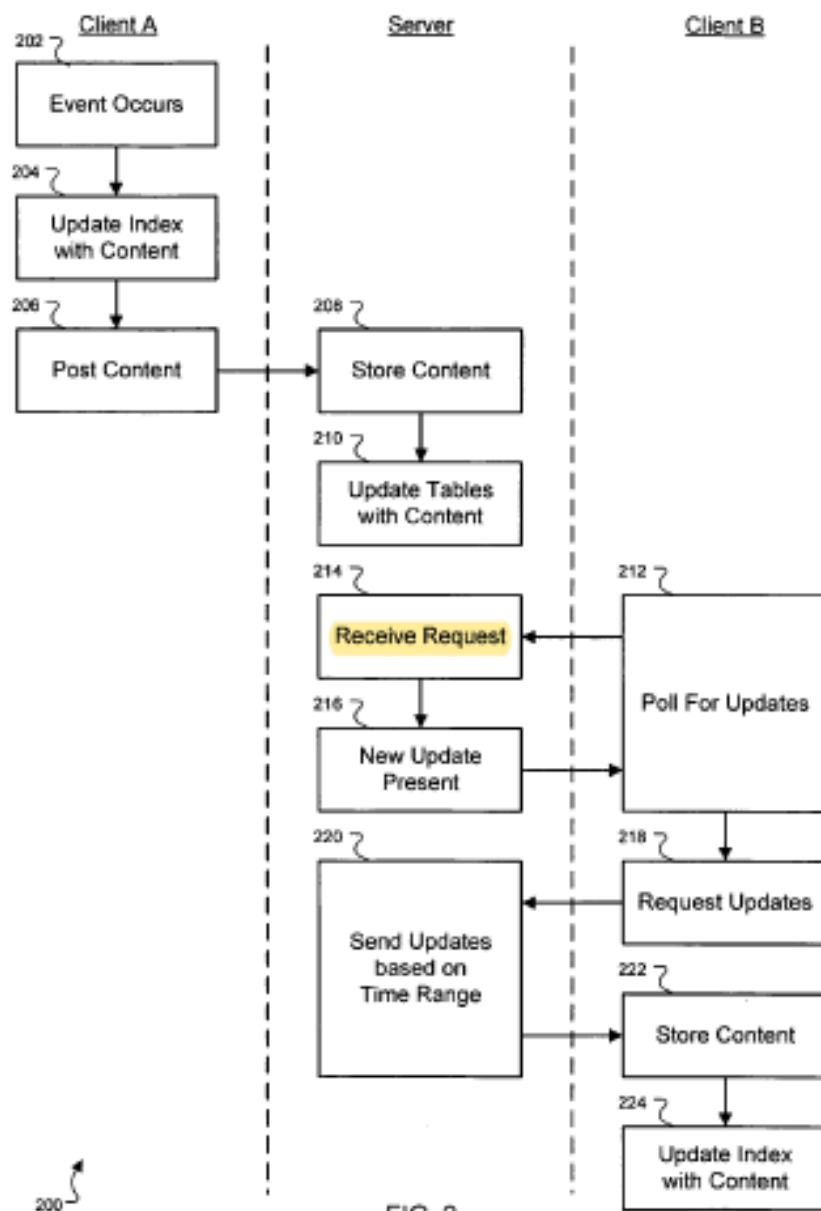
from the server only if the client determines that the content should be received from the server 106.

65. A POSA would read Sigurdsson as teaching a standard client-server architecture using HTTP. As explained above, HTTP is a pull-based request response protocol. Sigurdsson does not teach or suggest the use of push functionality for its server and nor did the HTTP protocol support such a feature during the 2007/2008 timeframe. I disagree with the petition that it would have been obvious to modify Sigurdsson to incorporate push into its HTTP server.

66. Sigurdsson at ¶ 31 does not correspond to the “automatic send” limitation (*i.e.*, without a client request). Rather, it merely states a result from interactions between Client A and Client C for synchronizing files. The details of the process and structure of the system are omitted in Fig. 1 and ¶ 31 of Sigurdsson for brevity, but lays groundwork for the detailed disclosure in the ensuing passages related to a request from C consistent with a “pull” approach as illustrated in Fig. 2.

67. For instance, Sigurdsson describes Fig. 2 (reproduced below) as a sequence diagram showing an illustrative method 200 for sharing and synchronizing content across multiple client devices **according to the implementation shown in Fig. 1.**

EX1016 at ¶ 32.



EX1016, FIG. 2 (annotations added.).

68. Specifically, Sigurdsson discloses the following steps performed by the system:

- In step 212, the second client polls for updates, representing content generated at other clients. For example, referring to FIG. 1, **the Client B104 can poll the Server 106 for new content.** (§37)
- In step 214, the polling request is received by the server. For example, in FIG. 1, the **Server 106 can receive polling requests** from the Client B 104. (§38)
- In step 216, **the server indicates to the polling client** that content updates are available. (§39)
- In step 218, **the client requests the updated content** that it needs. For example, in FIG. 1, **the Client B104 sends a request to the Server 106** for the missing Client A Content 108. (§40)
- **The request may be based on a time range**
- In step 220, the server sends the updated content **corresponding to the time range requested by the server.** (§41)

EX1016, at §§ 37-41 (emphasis added.).

69. Therefore, in Sigurdsson, several intermediate steps are performed by the Server 106 and the Client B104, including the *pull* steps performed by the Client B104: “**the Client B104 sends a request to the Server 106** for the missing Client A Content 108.”

70. As I explained earlier, a person of skill would not understand HTTP to support push, since HTTP did not include such a functionality. Further, neither paragraph 47, nor other teachings of Sigurdsson, indicate that the server had push

abilities. As a standard HTTP server, it would operate a pull-based server where the client would need to make explicit HTTP requests in order to receive files.

71. At ¶ 47, Sigurdsson *merely* discloses:

- The Client A102 can include both data and applications
- The applications include an Event Listener 322, a File to Text/HTML Converter 324, and a hypertext transfer protocol secure (HTTPS) Client/server 326.
- The HTTPS Client/server 326 can be used as a client that tracks web access and can contain a Time Stamp 328 which associates viewed web pages with a time that they were accessed by the user. The HTTPS Client/server can also be used as a server to provide content to peer clients.
- EX1016, at ¶ 47.

72. Contrary to Dr. Balakrishnan's assertions, paragraph 47 Sigurdsson does not teach "pushing" data "directly to the client from a server." See [RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1 \(ietf.org\)](#).

73. Moreover, Sigurdsson describes Fig. 4 as "a schematic showing a system 400 for sharing and synchronizing content using **a peer-to peer and client/server architecture**. In the system 400, content still can be shared among the Client A 102, the Client B 104 and a Client C 406 using the Server 106 in a method **similar to that of method 200 of FIG. 2.**" EX1016, at ¶ 51.

74. Since Sigurdsson *expressly* states that Fig. 4 implements a method similar to Fig. 2, the method of Fig. 4 also includes steps are performed by the Client B104 to receive content. That is, **the Client B104 sends a request to the Server 106** for the missing Client A Content 108.”

75. Therefore, Sigurdsson, when read as a whole by a POSA, discloses that the server 106 sends content to Client B104 **only after receiving a polling request and subsequently receiving a request for the updated content**. Accordingly, a POSA would understand that Sigurdsson is directed to a pull mechanism, not a push mechanism, and thus fails to disclose “automatically transferring the modified first electronic file copy to the third electronic device”.

76. In Fig. 3, Sigurdsson discloses that the client devices 102 and 104 are implemented as a hypertext transfer protocol secure (HTTPS) client/server structure (i.e., HTTPS Client/Server 256 and HTTPS Client/Server 356).

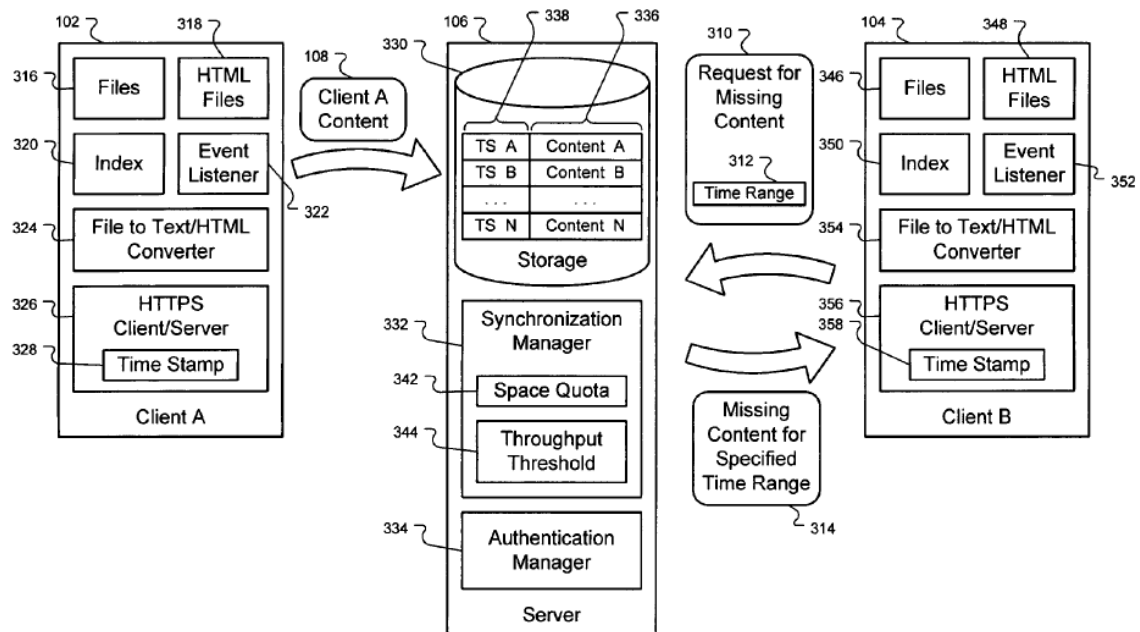


FIG. 3

EX1016, at Fig. 3.

77. During the deposition, Dr. Balakrishnan relied on paragraph 47 of Sigurdsson and stated the following:

I think it's all over the place. 12 So, for example, in Paragraph 47, the last sentence says, "The HTTPS Client/server can be used as a server to provide content to peer clients, which is described in more detail below." It doesn't limit it to being only in response to a request, and that would be a person of skill understanding of how HTTP works, would understand that to be included, pulling and pushing, for example.

EX2024, at 64:11-20.

78. As I explained earlier, a POSA would not understand HTTP to support push, since HTTP did not include such a functionality. Further, neither paragraph 47, nor other teachings of Sigurdsson, indicate that the server had push abilities. As a standard HTTP server, it would operate a pull-based server where the client would need to make explicit HTTP requests in order to receive files.

79. Further, there is no suggestion or teaching in Sigurdsson about any particular limitations or drawbacks of its pull-based HTTP server or any discussion of a push-based server. Consequently, a POSA would not be motivated or find it obvious to replace the pull-based HTTP client/server taught by Sigurdsson with a push-based server that automatically transfers data to a client without any client request.

80. As explained above, Sigurdsson illustrates two scenarios, i.e., a first scenario when **clients are currently online** where peer-to-peer architecture is used and a second scenario when the **offline client comes online again** where client-server or “drop box” architecture is used.

81. Sigurdsson aims to use the server as “drop box” for storing content for a client that is offline, and to that end, when the client is back online, the client can request to server to provide the missing content (i.e., access the “drop box” to

retrieve data) as shown at least in Fig. 2 (step 218), Fig. 3 (step 310), Fig. 6 (step 610) and Fig. 7 (step 710).

82. As I explained earlier, the peer to peer architecture and client server architecture are two fundamentally different methods for file access and are typically used in different situations. Sigurdsson also uses them for two different situations, namely offline and online devices. Given their differences, it would not be obvious to a POSA to use features from one approach and incorporate them into the other.

83. In my opinion, it would not have been obvious to a POSA to combine Sigurdsson and Shappell because Shappell teaches away from using a server.

84. Shappell discloses that “a computer implemented method and system enable users to share files in a **server-less** shared space. EX1020, at ¶ 1. By providing access to such **[server-less]** spaces via a visual presentation, the system renders content available for access by other group members. Access is sometimes provided through propagation of metadata or other uniquely identifying indicia associated with the shared space to all group members. EX1020, at Abstract.

85. Shappell does not merely provide a general preference for an alternative invention. Instead, Shappell expressly discounts the use of a server by specifically requiring a **server-less** shared space. EX1020, at ¶ 6. Shappell, when considered as

a whole, overwhelmingly teaches that peer-to-peer system without using a server. *See*, for example, EX1026, at ¶¶ 1, 7, 37, and 46.

86. Shappell repeatedly teaches using a shared space without a server. And for good reason. Shappell was concerned about facilitating group interaction through which a desired group of users can interact with each other in a secured environment, and through which users can access, and maintain control with respect to files shared in their shared relationships with one another. EX1020, at ¶ 6.

87. Specifically, Shappell was trying to address problems with existing network interaction mechanisms that do not provide a mechanism for readily manipulating shared data and information among a desired group of users or an user interface to simplify management of such shared data. EX1020, at ¶ 5.

88. On the other hand, Sigurdsson proposes a system including a client/server mechanism. Therefore, it would not have been obvious to a POSA to have modified the teachings of Sigurdsson to provide a server-less shared space (i.e., replace the client/server mechanism) as disclosed in Shappell, because doing so would change the principle of operation of Sigurdsson.

89. A central feature of Sigurdsson network architecture is using the client/server mechanism for providing access to client that are offline, and then come online. Modifying Sigurdsson's architecture to a server-less shared space

would change the principle of operation of Sigurdsson and as such a POSA would not have modified the teachings of Sigurdsson with the teachings of Shappell.

90. Claim 1 recites the first application executed on the first electronic device **automatically transferring** the modified first electronic file copy **to the third electronic device** to replace an older version of the first electronic file stored on the third electronic device with the modified first electronic file copy having the content modified by the user, where the second electronic device and the third electronic device are each associated with a **same** user.

91. Thus the second electronic device and a third electronic device are **associated with the same user**.

92. The **associated with the same user** limitation requires the claimed method to send the file not only a specific user but a device that is associated with the very same user that sent the file.

93. A POSA would not find that Shappell's sharing files **among group members** in the shared space teaches the **associated with the same user** limitation. While Shappell discloses sharing a file with members of a group, Shappell simply does not send the file to a specific device "associated with the user."

C. Ground 2 of the '561 Patent – The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Brown and Hesselink

94. As stated earlier, the focus of the '561 Patent is to provide a user-centric computer network architecture for file synchronization, where a server automatically receives a modified file from one device associated with a user when the user modifies the file and the server **automatically transfers** it (i.e., without a request from the client for the modified file). This is in contrast with the disclosures in Brown and Hesselink that explicitly require a request from the client to initiate data transfer to another device **associated with the same user** (as opposed to any devices connectable over a network as in Hesselink).

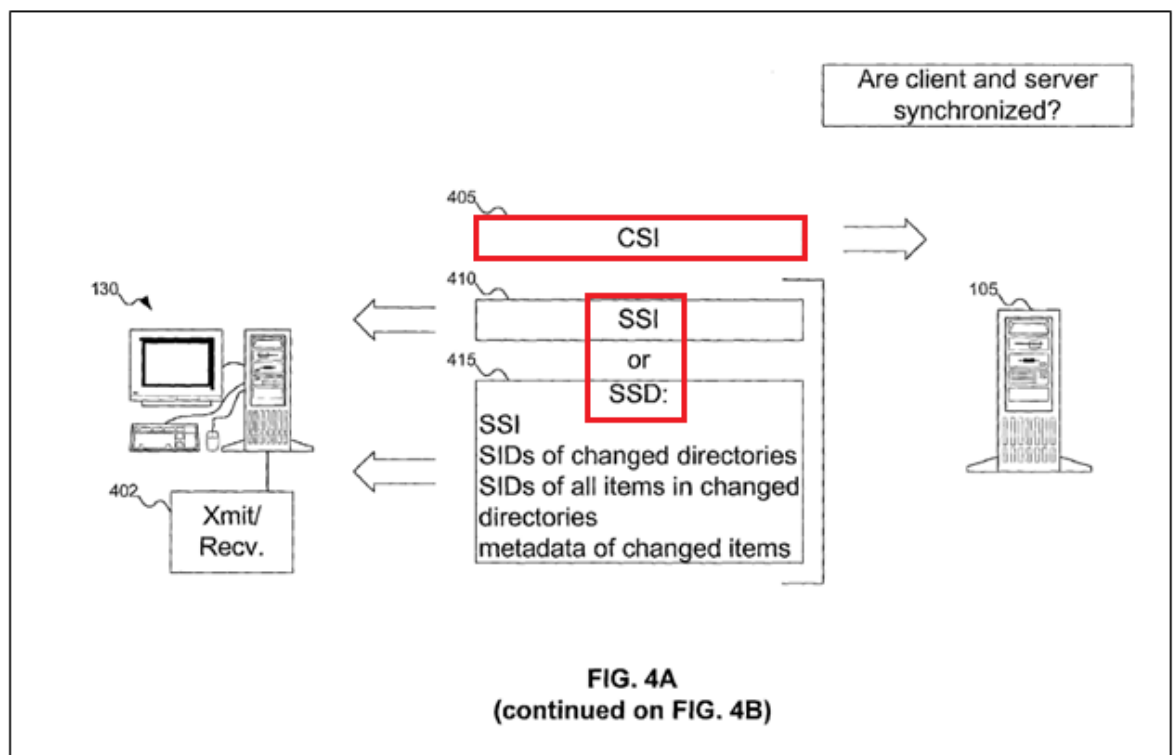
95. Like Sigurdsson, in Brown and Hesselink, the client must initiate a request for any update to the connection server. Brown discloses that communication between the clients and server uses the hypertext transport protocol (HTTP). EX1025, at 4:2-5. Also Hesselink consistently states that its data transfer process is a HTTP request/response process. See e.g., EX1026, at Fig. 8, steps 810, 830. As discussed above, the HTTP protocol does not support push-based functionality at the time of the invention. Therefore, to a POSA, the combined teachings of Brown and Hesselink, do not disclose that a modified file would be automatically transferred to a client without a request from the client.

96. Brown discloses a transfer of changes to a file from a server to a client is initiated by the client. Paper No. 3, at 55 (“After execution of the functionality

discussed for claim [1b], above, client 135 would initiate communication with the server and receive a copy of the modified electronic file sent to server 105 by client 130, as described above for claim [1b]. EX1025, at 2:63-67, 3:1-8; EX1028, at ¶¶896-898).”)

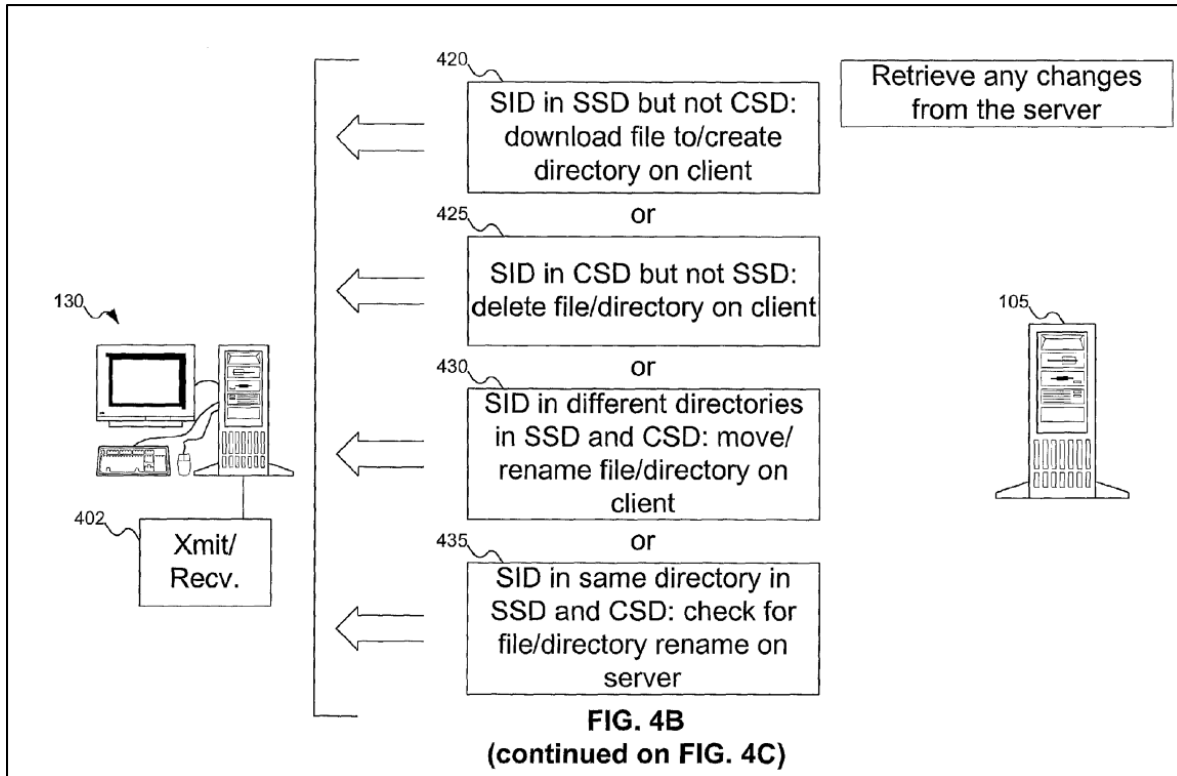
97. Petitioners rely upon Fig. 1 of Brown depicting client 130 as an exemplary “second electronic device” and client 135 as an exemplary “third electronic device.” Paper No. 3, at 40. However, Brown only discloses that client machines synchronize with each other by synchronizing to a common server account. EX1025, at 1:59-63. In Brown, the synchronization process is only between the initiating client synchronization application (SA) and the server. Referring to the embodiment of Fig. 4A of Brown, the synchronization process is only between the client 130 and the server 105, and this synchronization process does not synchronize another client 135 with the server 105. In order for the other client 135 to synchronize with the server 105, the other client 135 must initiate synchronization with the server 105 on its own schedule. Id., at 3:42-46. Unless the other client 135 initiates the synchronization process by making a sync poll call to the server 105, the server 105 would not send the modified file received from the client 130. Id., at 5:14-24 (“The synchronization process is **initiated by a client SA when it makes a sync poll call to the server**, passing it the client sync index (CSI) identifying its current known state of the account.”)

98. Thus, Brown requires each client to maintain its client sync index (CSI), which identifies its current known state of the user's server account. Id., at 5:14-17. The server needs to receive the CSI from the other client 135 and compare it with a server sync index (SSI) to determine if there is any change client 135 lacks, before proceeding with the synchronization process with client 135. In the absence of a CSI received from client 135, Brown's server 105 cannot automatically transmit the modified file received from the client 130 to the client 135. EX1025, at 10:45-48.



EX1025, Fig. 4A (annotations added.).

99. Brown further illustrates in Fig. 4B the update process **performed by the client** in which the client determines what updates need to be retrieved from the server based on any of four possible results of the comparison between the SSD and the CSD. The client then requests the server to send the changes related to the file or directory. EX1025, at 10:45-48. That is, in Brown, the client retrieves the changes from the server **only upon the server receiving the request from the client**.



EX1025, Fig. 4B

100. Thus, in Brown, to synchronize with the server, **the client must actively pull down changes from the server**, as opposed to the modified file being

“automatically” transferred from the server to the client. Clearly, Brown uses a protocol that uses a “pull” mechanism and not the server automatically pushing the modified file to the client. In the “pull” mechanism, the client does not know when new data is available and the client must poll the server to see if there is new data available to the client and request that data from the server (i.e., pull the data).

101. For the alleged teaching of claim [1d], the Petition relies on EX1025, at 2:63-67, 3:1-8 and EX1028, at ¶¶896-898 and argues “After execution of the functionality discussed for claim [1b], above, client 135 would initiate communication with the server and receive a copy of the modified first electronic file sent to server 105 by client 130.” Paper No. 3, at 55. However, Petitioners ignores the requisite steps between the client 135 initiating communication and receiving a copy of the modified file from the server 105. As discussed above, in order for the client 135 to begin the synchronization process, the client 135 (i) first should poll the server for changes by other clients by passing its current CSI to the server in a sync polling call and (ii) request the server to send server synchronization data (SSD) if the CSI does not match the server account's SSI value. EX1025, at 8:49-53. Petitioners also argue that Brown discloses automatic file synchronization, citing Brown at 3:40-42. (“From the user's perspective, the synchronization process is automatic, runs in the background and requires minimal monitoring or intervention.”).

102. The process described in Brown may be automatic from the user's perspective, meaning that the process may need little or no involvement of human operation from the user but it is not an automatic send from the server to the client

103. While Brown's file synchronization requires the client to execute prescribed steps, Brown expressly teaches its "automatic synchronization" as being automatic from the *user's* perspective as long as the client does not necessitate any action from the user in order to initiate the synchronization. EX1025, at 3:40-42. To a POSA, this use of the phrase "from the user's perspective" in connection with file synchronization, limits Brown's disclosure and is different from the automatic transfer in the '561 patent.

104. In contrast, the automatic transfer in the claims of the '561 Patent is automatic from the *server's* perspective (that is, the first electronic device executing the first application). See claim 1 of the '561 Patent, "**the first application** is further configured to **automatically transfer** the modified first electronic file copy to the **third electronic device** ... (associated with a user)" and similarly claim 8 of the '561 Patent.

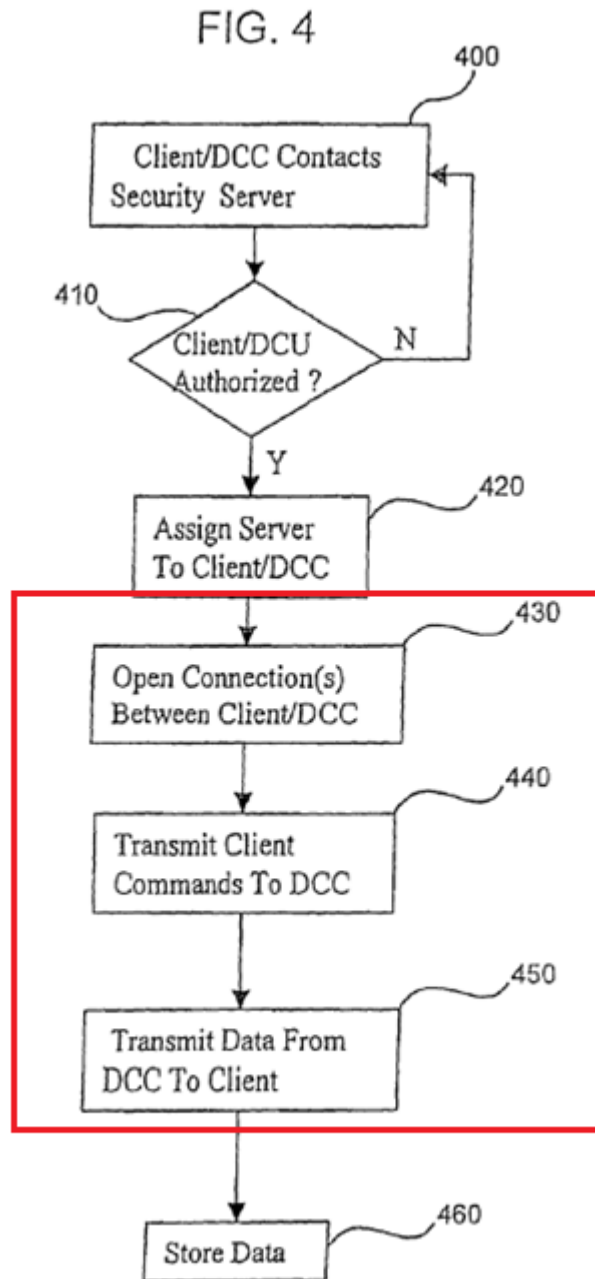
105. That is, the automatic file transfer of the '561 Patent teaches a push-based file transfer from the server to the client, thus allowing the server to send the modified file to the client without a need for the client to explicitly request it. Brown's disclosure is directed to the opposite method, that is, client pull.

106. In Brown, the client SA must initiate synchronization by sending a request to the server. EX1025, at 8:49-55 (“The client polls the server for changes by other clients by passing its current CSI to the server in a sync polling call ... The client SA requests server synchronization data (SSD).”) Absent such request, the server would not transfer the modified file to the other client.

107. A POSA would not rely on Hesselink at ¶ 183 as disclosing teaching automatic push from the server. Hesselink at ¶183 states: “As noted above, when data is updated, such as when a file is edited, for example, the **system** may be configured to automatically update all or some designated subset of computers on the network as soon as the update has been performed at one location.” However, this paragraph merely states that all or some computers are automatically updated but does not discuss how the automatic update is done. Specifically, Hesselink at the cited portion or anywhere else does not contain any disclosure of the **server** automatically pushing the modified file.

108. It should be noted that Hesselink at ¶183 refers to the previous discussion of file synchronization that requires a client request, because the paragraph begins with “As noted above.” There is no discussion whatsoever of a push-based file transfer from the server in the preceding paragraphs in Hesselink. To the contrary, Hesselink only discusses the pull-based synchronization throughout its disclosure, for example, as illustrated in detail with reference to FIGS. 4 and 8.

109. Referring to Fig. 4, Hesselink describes the operation of the access and control system 54. If the user of client computer 12a-12n is authorized in event 410, security server 58 assigns one of the connection servers 14a-14n to the user of client computer 12a-12n, in even 420. In event 430, a connection is made between the connection server 14a-14n assigned in event 420, and the network-enabled device 18a-18n selected by the user authorized in event 410. In event 440, client computer 12a-12n transmits instructions or commands to a network-enabled device 18a-18n, and in event 450, network-enabled device 18a-18n transmits data to client computer 12a-12n. EX1026, at ¶¶86-87. Hesselink at ¶87 further discloses, referring to events 440 and 450: “Client process 32 on client computers 12a-12n periodically sends HTTP requests to connection server 14 a-14 n that include embedded command data. Device control process 38 on device control computer 36 a-36 n also sends periodic HTTP requests to connection server 14 a-14 n with embedded data from network-enabled device 18 a-18 n.”

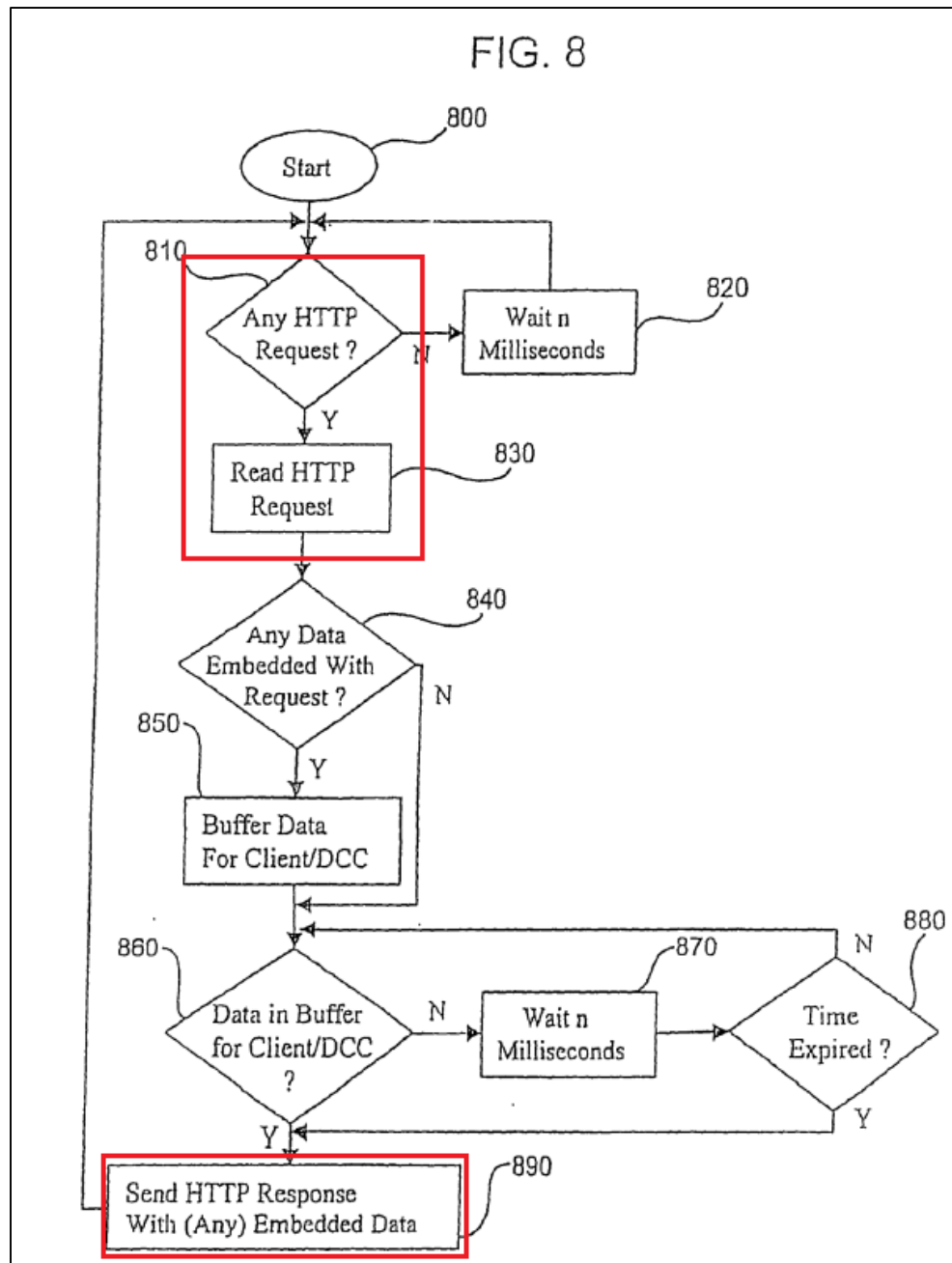


EX 1026, Fig. 4 (annotations added.).

110. Thus, just like in Brown, Hesselink uses a pull mechanism in its file synchronization process. That is, in Hesselink, the client **must initiate a request** for any update to the connection server. The HTTP request/response process for

data transfer between the client and the connection server is also illustrated in Fig. 8 of Hesselink. EX1026, at FIG. 8. The process includes event 810 in which it is determined “whether or not **any HTTP request has been received** from client computer 12 a-12 n (or device control computer 36 a-36 n)”; if an HTTP request has been received and there is no data is embedded within the HTTP request, event 860 in which it is determined “if there is any data present for client computer 12 a-12 n from network-enabled device 18 a-18 n”; and event 890 in which “**an HTTP response** is sent to client computer 12 a-12 n by connection server 14 a-14 n,” and “[i]f any data was present in the connection server sending buffer for client computer 12 a-12 n, this data is **embedded within the HTTP response.**” Thus, Hesselink is precisely focused on a data transfer process based on client pull, which is implemented by clients sending HTTP requests to the server to request the data. This is the opposite to the server initiating data push to a client.

111. As explained earlier, the HTTP protocol is a request-response protocol and the HTTP servers use the client-pull architecture. The standard HTTP protocol practiced at during the 2007/2008 timeframe did not support push-based functionality where the server can automatically send data to a client without the client making requests for data.



EX1026, at Fig. 8

Indeed, Hesselink’s synchronization process only begins when a request for access to data is received. *See* EX1026, at Abstract (“**Upon receiving a request for access to data** originally stored in a remote storage device, determining whether the

data is already available and valid in a local storage device.”); ¶ 78 (“client computers 12 a, 12 b may each comprise programming 32 stored in memory that allow client computers 12 a, 12 b to **send instructions** to network-enabled devices 18 a, 18 b **as requests** through connection server 14, and receive data from network-enabled devices 18 a, 18 b **via responses** through connection server 14.”) Hesselink does not disclose or suggest any alternative embodiment in which a server automatically transfers the modified file without a client request.

112. Hesselink describes an embodiment where a file transfer can be based on an active subscription of the local device to the remote device. EX1026, at ¶141 (“Further, device module 74 sd may also listen to file access events from its local operating system to detect any changes to files and folders that are actively subscribed by connected user modules. Device module 74 sd transmits any locally detected file overhead modifications (detected through the described listening) to all connected and actively subscribed user modules.”). However, even with the active subscription, Hesselink clearly states that a user should still make the initial network request to a remote directory in remote storage device and **continuously and repeatedly makes data access requests** to that remote storage device to locally store information. EX1026, at ¶140 (“[E]ven if the user uses local applications to **continuously or repeatedly makes data access requests** to a remote directory in remote storage device 18 b, user module 72 only needs to make

a network request once during the period of the active subscription, since after the initial request, the information is stored locally at the site of the user.”).

113. For at least the reasons above a POSA reading the description in Hesselink at ¶183 would not consider that description as disclosing the “automatic transfer” feature in [1d] and [1e] of claim 1 and similarly in features [8c] and [8d] of claim 8.

114. In my opinion, to a POSA, Hesselink fails to disclose the first application associated with a user automatically transferring a modified first electronic file copy to the third electronic associated with a same user.

115. As explicitly recited in the preamble of claim 1, each of the first electronic device, the second electronic device, and the third electronic device are all associated with a user, i.e., the same user.

116. Specifically, claims of the ’561 Patent recite a system of specifically arranged applications (that is, “a first application”, “a second application” and “a third application”) that are executable on multiple electronic devices, each associated with the same user, to provide a user-centric computer network architecture. The claims describe how this arrangement of applications operates to automatically transfer a copy of an electronic file that is modified at one electronic device to another electronic device based on the triggering event (i.e., the user modifying the electronic file). Thus, file sharing in the ’561 Patent is essentially

integrated with these applications and is implemented on a client-server system, as illustrated in Fig. 3 of the '561 Patent (reproduced below). Referring to Fig. 3 of the '561 Patent, the server 230 transfers a file-transfer and/or synchronization application (which is “a second application” and “a third application”) to the user devices 210, 280 for installation thereon, which then provides file-transfer clients 310, 320 executable by the user devices 210, 280, respectively. Based on the file-transfer application, a user can manage and synchronize files across multiple devices.

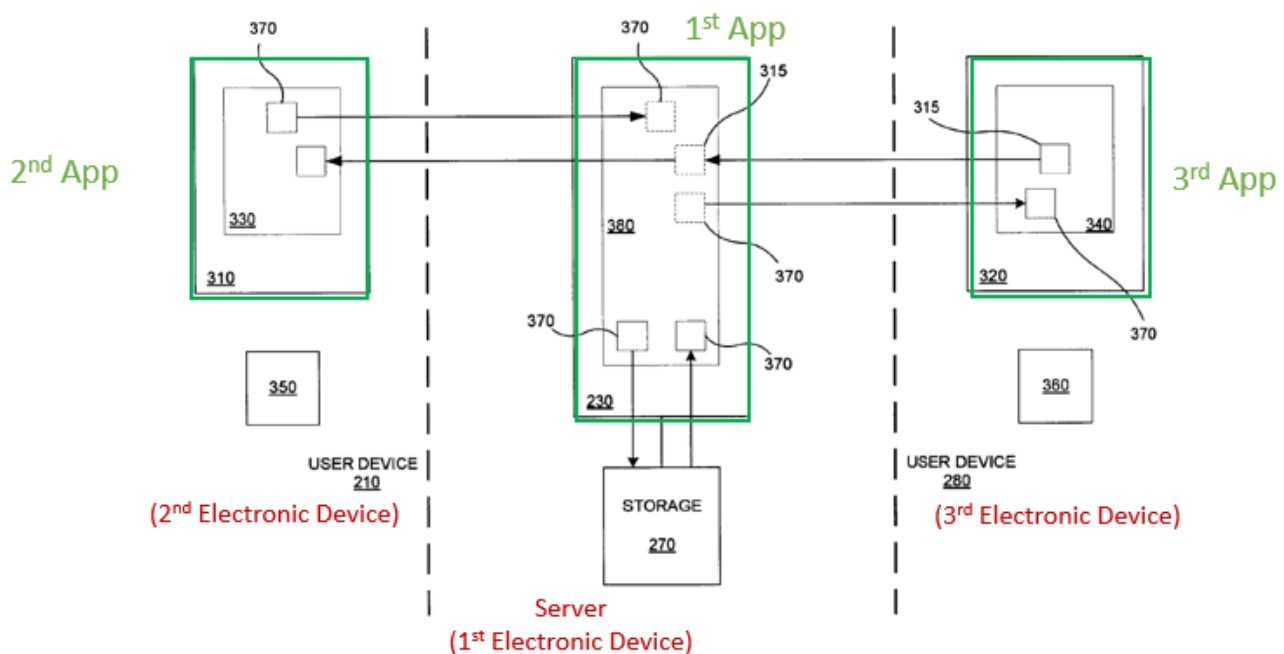


FIG. 3

EX1002, Fig. 3 (annotations added.).

117. The **automatically transferring** limitation requires the claimed method to not merely have the first electronic file copy modified by a user at the second

electronic device end up at any third electronic device, but the limitation requires the first application executed on the first electronic device (i.e., the server), associated with that user, to find the third electronic device that is associated with the same user and automatically transfer the modified first electronic file copy to the third electronic device via the third application executed on the third electronic device. The “automatic transfer” feature of the asserted claims should not be considered in isolation but must be considered in connection with the specific arrangement of first, second and third applications running on three different electronic devices, including the server between the multiple devices to be synchronized. It is the use of these application programs and their arrangement that made possible the “automatic transfer” to another device (i.e., without the another device requesting it) associated with the same user.

118. ¶ 183 of Hesselink states:

As noted above, when data is updated, such as when a file is edited, for example, the system may be configured to automatically update **all or some designated subset of computers on the network** as soon as the update has been performed at one location. Computers that are offline at such a time will be automatically updated as soon as those computers go online again. The persistence of the system ensures that **updates are completed for all online computers**, as if there is some failure or problem

during the update, the connection restores itself to ensure that the update is completed....

Ex1026, at ¶ 183 (emphasis added.).

119. The “all or some designated subset of computers on the network” and “online computers” discussed in this paragraph refer to client or user computers 12a, 12b, that may be operatively coupled to a connection server 14 via a public or global network 16 and one or more network-enabled devices 18a, 18b that are also operatively coupled to connection server 14 via global network 16. *See* EX1026, at Fig. 1 and ¶¶ 76, 120-121.

120. That is, Hesselink only discloses updating all or some of computers online that the user has access to. The file-access permission is stored in the database 52 and permission management information can be downloaded and stored in the user module's local cache to determine what data access functionalities that the particular user module can perform and to which files or directories may be accessed on which remote device modules. *Id.*, at Fig. 10 and ¶132. Therefore, Hesselink does not make a determination as to whether the third electronic device is associated with the same user by use of the first, second, and third applications recited in the asserted claims, and hence Hesselink cannot make an automatic transfer of a modified file based on such determination. While Hesselink at Fig. 10 mentions application 72a (e.g., Windows Explorer®), this application is only for accessing remote storage device 18b at computer 74 over a wide area network.

EX1026, at ¶127. Thus, Hesselink does not teach a dedicated file transfer and/or synchronization application as recited in the asserted claims.

121. As Hesselink discloses, updating all or some of online computers the user has access to, cannot be construed as teaching “automatically transfer the modified first electronic file copy to the third electronic device” that is associated with the same user, because such construction would render “associated with a user” language meaningless. In the ’561 Patent, based on the “association” with a user established via the first, second, and third applications executable on different devices, the server can find the third electronic device to which the “automatic transfer” should be performed. In contrast, Hesselink only looks at whether the user is making a data access request to a remote directory in the remote device, in order to determine whether to update the user’s local device. Hesselink’s automatic update is not based on the determination of the third electronic device being associated with a user via the third application.

122. In my opinion, a POSA would not be motivated to modify Brown’s disclosure to that of a push-based file transfer due to the differences between the pull and push-based methods.

123. Brown intentionally and purposefully chose the “pull” based approach, and as such, a POSA would not be motivated to modify Brown to apply to the “push” based file transfer.

124. Brown specifically states at 3:17-28:

The client/server architecture is designed to **minimize the load on the server processor** in order to maximize server scalability. To that end, as many processor-intensive operations as possible, such as message digest computation and comparison, data encryption and decryption, and **synchronization itself, are performed by the client SA**. Also, the **polling mechanism used by the client SA** to determine if the client is synchronized with the server is designed to **require minimal processing on the server** when the client and server are in sync. This is significant because as a rule only a small percentage of clients require synchronization activity at any point in time.

EX1025, at 3:17-28 (emphasis added.).

125. Brown aims to minimize the load on the server processor, and to that end, as many synchronization operations as possible are performed by the client, including the operation to make a sync poll call to the server. Thus, the intended principle of operation and purpose of Brown teaches away from using the server-push mechanism for file synchronization.

126. For at least the reasons mentioned above, there is lack of motivation for a POSA to modify Brown to incorporate “automatically transfer” from a server to a client since it would fundamentally change the operation of Brown’s system.

127. In my opinion, Hesselink's peer-to-peer embodiment cannot be modified to teach "automatic transfer" of the modified file from the server to a client device as required by claim 1 of the 561 patent.

128. The Petitioners argues that: "[I]t would have been obvious in view of Hesselink to automatically forward updated files from server 105 to synchronizing clients as soon as file updates are received at server 105," citing Hesselink, ¶183. Paper No. 3, at 56. However, updating designated subset of computers when data is updated as described in Hesselink, ¶183 is from the context of a file subscription discussed in Hesselink, FIGS. 11-13, which is based on a peer-to-peer connection between two computers, as opposed to a client-network architecture.

129. Referring to Fig. 11 of Hesselink, two computers 72, 74 are connectable across a wide area network and a user module 72su of the computer 72 accesses data from remote storage device 18 b and user module 74su of the computer 74 accesses data from remote storage device 18 a.

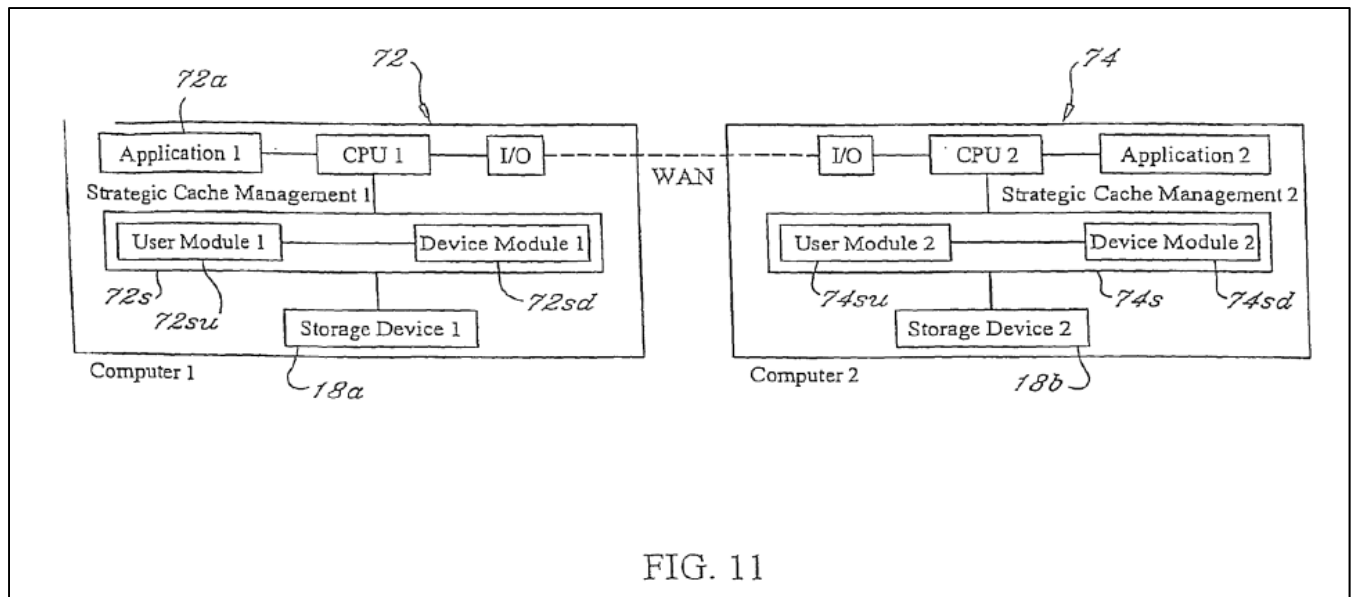


FIG. 11

EX 1026, at Fig. 11

130. Referring now to FIGS. 11 and 12 of Hesselink, where the user module 72su of the computer 72 receives a request from the user to browse through files and directory of the remote file systems of storage device 18 b, “it may first check its local cache to find whether such file overhead information is available and valid at event 1202.” EX1026, at ¶137. If the user module 72su is actively subscribed to the device module 74d of the second computer 74 and an active subscription period is running, it is determined that there is file overhead information that is available and valid at event 1204, and the device module 74sd sends any updates to user module 74su regarding the file overhead information of the associated directory over the prescribed active subscription period, at event 1214. EX1026, at ¶¶137, 139.

131. In this case, Hesselink merely discloses updating, by a first computer, a second computer with the file overhead information such as file name, file size, folder status, last modified date, created date, etc. regarding its directory (Ex1026, at ¶71), upon receiving a request to access to its local storage from the second computer.

132. I disagree with the petition that a POSA would not have been dissuaded from combining the peer-to-peer teachings of Hesselink with Brown's client/server model. This is opposite to what Hesselink actually teaches. Hesselink addresses the security concerns in using server/client architecture for file synchronization.

("[M]any users may be uncomfortable storing photos of their families on some photo service server in some unknown location where it is also often unknown what level of security (if any) such server is provided with."; EX1026, at ¶6). To solve this problem, Hesselink proposes a peer-to-peer method that allows users to access their files on any network accessible communication device, from any location accessible over that network. EX1026, at ¶10. Thus, contrary to the Petitioner's assertion, Hesselink clearly teaches away from implementing its method in a client/server model.

133. The Petitioner argues that "Hesselink does not teach that the devices in its synchronization network must not include any servers" and cites Hesselink, ¶122 as teaching any of computers 72, 74 and 76 as being a computer device including

“file server.” Paper No. 3, 52. However, as discussed above, while Hesselink mentions including a server within its peer-to-peer system, such server only serves as a pipeline or conduit between two peers. Also, the “file server” mentioned in Hesselink, ¶122 is only an example of a type of computing device of computers 72, 74 and 76, each of which are peers in the peer-to-peer architecture (i.e., nodes between which a direct link is provided). There is nothing that suggests in Hesselink that the “file server” implements file synchronization between computers 72, 74, and 76 based on a client-server architecture.

134. As explained earlier, Client Server architecture and Peer-to-Peer architecture are fundamentally different and are used in different situations. Therefore, a POSA would not be motivated take features from one of the architecture to make changes to the other architecture.

135. Since Hesselink’s file synchronization is based on the peer-to-peer system, a POSA would not be motivated to combine it with file synchronization in Brown’s client-server system as these systems employ two fundamentally different architectures. A peer-to-peer network allows computer hardware and software to communicate between peers. Unlike a client-server architecture, there is no central server for processing requests in a peer-to-peer architecture. A peer-to-peer based file synchronization is entirely different from the client-server system because each node is the one that processes the query from another node to decide whether to

grant or deny it and thus each node should have the capability to access and manage information of other nodes. Hence, A POSA would not be motivated to combine the teachings of Brown and Hesselink because these two references teach fundamentally different architectures.

D. Ground 1 of the ‘942 Patent – The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Sigurdsson and Shappell

136. In my opinion, Sigurdsson does not disclose that “**automatically send[ing]**, via the first application, the modified first electronic file copy **to a third application at the third electronic device** ... responsive to receiving the modified first electronic file copy from the second electronic device” where a server automatically receives a modified file from one device associated with a user when the user modifies the file and the server **automatically sends** it to another device **associated with the same user**. Consistent with this focus, claim feature [1d] (and similarly claim [10c]) recites “a **first electronic device**, associated with a user, configured to ... **automatically send, via the first application**, the modified first electronic file copy to **a third application at the third electronic device** [associated with a user] ...”

137. Specifically, throughout its disclosure, Sigurdsson requires a server to receive a request from a client device before the server transmits missing content to

the client device. EX1016, Figs. 2, 3, 6; ¶ 36, 40-47, 69, 61-79. Moreover, Sigurdsson describes that its system is implemented as an HTTP client/server operating based on request/response process. EX1016, ¶ 47. Likewise, in Brown and Hesselink, the client must initiate a request for any update to the connection server. Brown discloses that synchronization process is initiated by a client, not the server. EX1025, 5:14-17. Also Hesselink consistently states that its data transfer process is initiated by the client computer requesting for data. *See, e.g.*, EX1026, FIG. 4, step 440 and FIG. 8, steps 810, 830. An HTTP protocol did not support push-based functionality at the time of the invention. Therefore a POSA would not read Sigurdsson as disclosing “automatically sending the modified first electronic file copy to the third electronic device.”

138. Petitioner argues that Sigurdsson’s two “alternative embodiments” of steps in the sequence diagram of Fig. 6 “at least suggests the automatic transfer of a modified file from the server to Client C.” Paper No. 3, at 22. In making these arguments, the Petitioner relies on statements Sigurdsson makes in ¶ 68 and ¶ 80, in isolation and without considering the reference as a whole, to support this assertion. *Id.*

139. A POSA, considering the reference as a whole, would not have understood the discussion of the alternate embodiment of step 606 described in ¶68, and the

alternate embodiment of step 620 described in ¶80 to be divorced from the accompanying steps in Fig. 6.

140. These alternative embodiments are part of the sequence of the “method for sharing and synchronizing content using a peer-to-peer and server architecture of Fig. 4 when a client that is offline comes online.” EX1016, at ¶ 21. The reference describes this as enabling “peer-to-peer content sharing and synchronization using a combination of peer-to-peer and ‘drop-box’ techniques.” Id., at ¶ 62. The reference further explains that the “peer-to-peer techniques can be used by clients that currently are online to share the content directly between clients,” and “the server can be used as a drop box for the content” to supply that content to an offline client once it comes online again. Id.

141. A POSA would understand that paragraphs 61 through 82 of Sigurdsson describes the sequence shown in Fig. 6 for performing the method of using the peer-to-peer and client/server architecture of Fig. 4. Both ¶68 and ¶80 are part of this sequence. To extract only these two paragraphs, or even a single sentence from ¶68, from the sequence fails to consider the teachings of the reference as a whole. EX1016, at ¶¶ 61-82.

142. Sigurdsson explains that in step 606 Client A posts content to another client (Client B) to initiate the peer-to-peer content flow. Id., at ¶ 65. Client B, in step 608, stores the content and could propagate the content directly to Client C, if that

client was online. Id. In the example shown in Fig. 6, however, Client C is offline so Client B propagates the content to Server 106 instead. Id. Sigurdsson goes on to explain how Server 106 transmits the content to Client C, when that client comes online, through steps 612-626. Id., at ¶¶68-80.

143. The alternate embodiment of step 606 described in ¶68, shown by the dashed line between Client A in step 606 and the Server in step 610, merely concerns how the content can be transferred from Client A directly to the Server without first going through Client B. While ¶68 states that the “Sever 106 can then transmit the content to the remaining client peers, such as Clients B and C” it in no way teaches or suggests that Server 106 *automatically* sends the content to a client. On the contrary, the very next paragraph begins the discussion of *how* Server 106 transfers the content, through steps 612 to 626, when Client C comes back online.

144. A POSA would understand that Sigurdsson at ¶ 68 discloses that the alternative embodiment of step 606 is part of the overall sequence for handling synchronization *when a client is offline* when a transfer of content is to take place. As Sigurdsson explains, with the peer-to-peer and server architecture of Fig. 4 “[t]he peer-to-peer techniques can be used by clients that are currently online to share the content directly between clients” and “the server can be used as a drop box for the content until the offline client comes online again.” EX1016, at ¶61.

145. Considering the Sigurdsson reference as a whole, a POSA would have considered Sigurdsson's alternative embodiment of step 606 as limiting the transfer of the content to using step 620 and its associated steps 612-620 and 626-630. This is because the alternative embodiment of step 606 is for use with the sequence of Fig. 6 which is for the case of Client C being offline at the time of the transfer. If Client C was online, Sigurdsson teaches that the transfer would be performed using the peer-to-peer technique that does not involve sending content from the server. In this case where both clients are online the transfer would not be performed by the server using the "drop box" technique.

146. Similarly, step 626, discussed in ¶80, also is part of the sequence of Fig. 6 which concerns an offline client coming online. EX1016, at ¶21. Step 626 is performed to "Provide Missing Content to Client C if Clients A and B are Offline." Id., at Fig. 6 Step 626. This concerns when *both* the first and second clients are offline "at the time of the third client's request for missing content." Id., at ¶79. Petitioners and the Board rely on the statement in ¶80 that "the Server 106 sends the Client X Content 410 to Client C 406," for teaching or suggesting that Server 106 automatically sends the content to Client C. But this statement cannot be considered in isolation without considering the teachings of the whole reference. A person of ordinary skill in the art would not understand the statement in ¶80 to

teach or suggest Server 106 automatically sending the modified content to Client C.

147. While it is true that Sigurdsson discloses that the Server 106 can transmit the content to the remaining client peers, such as the Clients B and C, Sigurdsson is clear that this transmission of the content by the server 106 is performed *only* after receiving a request for the content from the Clients B and C. Id., at ¶¶61-82.

148. Fig. 6 of Sigurdsson is reproduced below:

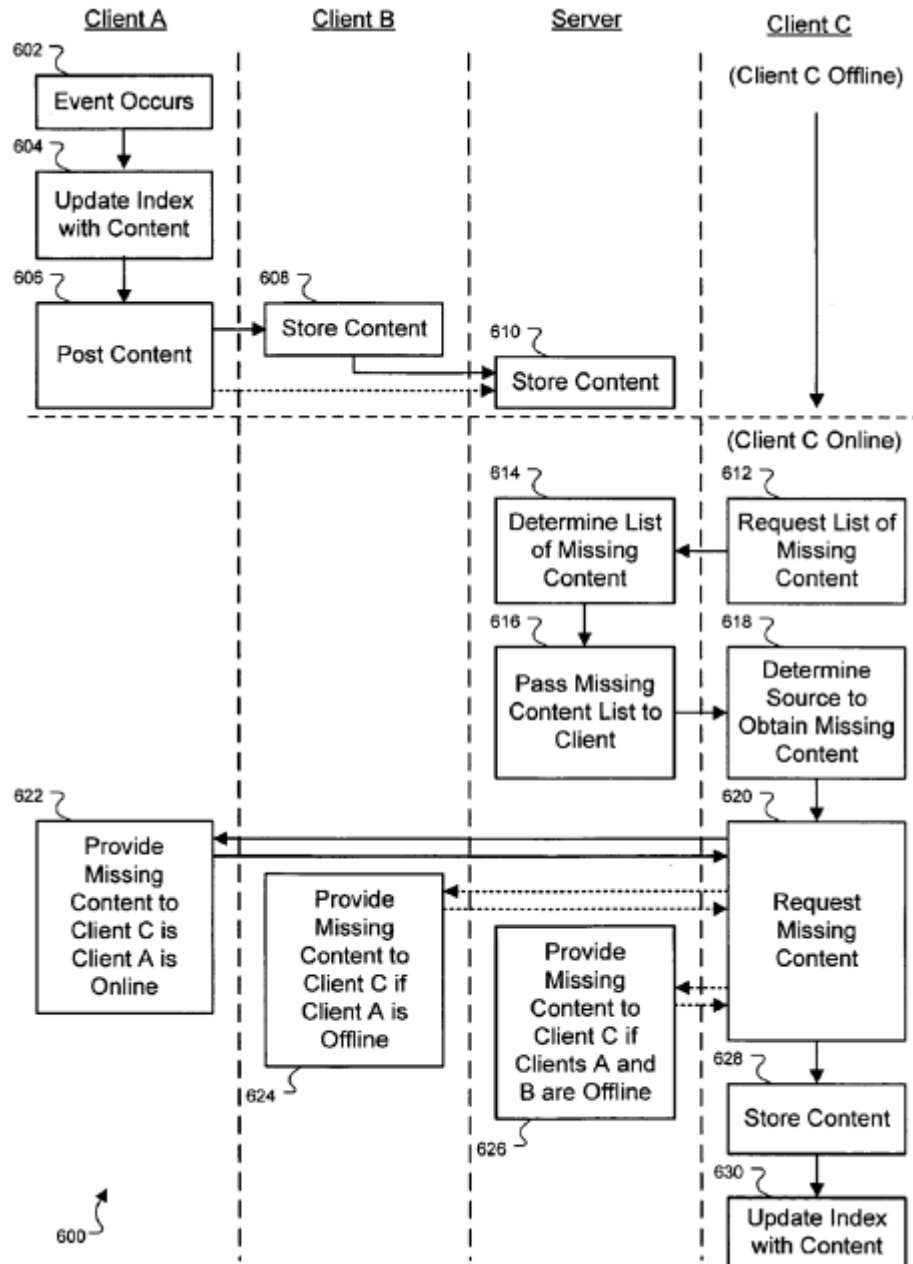


FIG. 6

149. In connection with Fig. 6, Sigurdsson explains the following at ¶¶ 61-62 (emphasis added):

FIG. 6 is another sequence diagram showing an illustrative method for sharing and synchronizing content using a mixed peer-to-peer and server architecture of

FIG. 4 **when a client that is offline comes online.** In this case, the system 400 automatically detects that a particular client is offline. In one implementation, determination of whether a client is online or offline can be made by monitoring polling by the client. **If no request is received from a client, then the client is assumed to be offline.** Similarly, the Server 106 can maintain a table of client IDs and whether each client is online or offline. **When a client comes online, it can notify the Server 106 that it is online,** and the Server 106 can update its table of clients with the online status.

The system 400 can enable peer-to-peer content sharing and synchronization using **a combination of peer to-peer and “drop box” techniques.** The peer-to-peer techniques can be used by **clients that are currently online** to share the content directly between the clients. However, in order for the system 400 to supply the content to an offline client, the server can be used as a drop box for the content until **the offline client comes online again.**

EX1016, at ¶¶ 61-62 (emphasis added).

150. Here, Sigurdsson establishes that Fig. 6 illustrates two scenarios, *i.e.*, a first scenario when **clients are currently online** and a second scenario when the **offline client comes online again.** According to Sigurdsson, when the **clients are**

currently online, a peer-to-peer technique is used by the clients to share the content directly between the clients (step 606 according to a first embodiment described in ¶65). Alternatively, when a **client is offline, and then comes online**, a “drop box” technique is used by the client *to request* missing content stored in the server 106 (step 606 according to an alternative embodiment described in ¶ 68).

151. Specifically, according to one embodiment in which **clients are currently online** (*i.e.*, first scenario), Sigurdsson at ¶65 discloses that “In step 606, the content originating on the first client is posted to a second client, thus initiating a peer-to-peer content flow.” EX1016, at ¶ 65. In another embodiment in which **client is offline, and then comes online** (*i.e.*, second scenario), Sigurdsson at ¶68 discloses that “An alternative embodiment of step 606 bypasses the second client and sends the data directly to the server, as shown by the dashed line pointing from the step 606 to the step 610.” *Id.*, at ¶ 68.

152. From the above description, it is clear that Sigurdsson at ¶65 already contemplates the scenario suggested by the Board, that is, the first scenario in which the client is already online and Sigurdsson at ¶ 65 unambiguously states that the peer-to-peer technique should apply (“In step 606, the content originating on the first client is posted to a second client, thus initiating a peer-to-peer content flow.”)

153. Therefore nothing in the alternative embodiment of step 606 discloses nor suggests to a POSA that the server 106 automatically transfers the content 410 to the client C 406 without receiving a request from the client C 406.

154. In fact, in the second scenario where the content is transmitted from a server (i.e., client-server scenario), Fig. 6 of Sigurdsson requires the following steps in order:

- “In step 612, the third client **requests** a list of missing content from the server. For example, the Client C 406 **requests** a list of missing content from the Server 106 when it connects to network that links the server and other clients, such as the Internet”. (§69).
- In response, “In step 616, the server passes the list of missing content to the third client (e.g., Client C), which had been offline.” (§73).
- In step 618, the third client uses the list it received from the server to **determine where it can obtain the missing content**. The missing content may exist on other peer clients, the content may be stored at the server, or the content may be stored at both the client and the server. For example, referring to FIG. 4, the Client C 406 may determine that it can synchronize its content by requesting the Client X Content 410 (which is the Client A's content in this case) from the Client A102. In the current example for FIG. 6, the content happened to originate on the Client A 102, but a client can request content from any client that has a copy of the missing content, regardless of where the content originated. (§74).
- In step 620, the third client **requests** the missing content from the first client, provided the first client is online at the time. For example, referring to FIG. 4, the Client C 406 **requests** the Client X Content 410

from the Client A102 if the Client A is currently networked to the Client C. The content requested is based on the time range corresponding to the time stamps of content resident on the Client A102 but not resident on the Client C 406. (¶75).

- In step 626, the server sends the missing content to the third client. For example, referring to FIG. 4, the Server 106 sends Content 410 to the Client C 406.

EX1016, at ¶¶ 69-80 (emphasis added.).

155. This focus on a client request for missing content in the context of client-server scenario is further evident throughout the disclosure of Sigurdsson. For example, at ¶9 discloses (emphasis added):

[0009] In a second aspect, a computer system having one or more servers is described. The system includes a table manager module to receive an indication from a first client that a user has performed an action on a file that is unrelated to a transfer of the file. The indication including content extracted from the file and a metadata value assigned to the content. The system also includes a data table to store the content extracted from the file on the first client and the metadata value, an interface to receive from a second client a request for content that is associated with one or more metadata values within a specified metadata value range, and a selection module to initiate transmission of the content to the second client if the metadata value associated with the content is within the specified metadata value range.

EX1016, at ¶ 9 (emphasis added).

156. Additionally, Sigurdsson at ¶ 37 discloses (emphasis added):

[0037] In step 212, **the second client polls** for updates, representing content generated at other clients. For example, referring to FIG. 1, the Client B104 can poll the Server 106 for new content. Here, the new content can be the Client A Content

108 (e.g., the resume text) stored on the Server 106 but not yet received by the Client B 104.

[0040] In step 218, the client **requests** the updated content that it needs. For example, in FIG. 1, the Client B104 sends a request to the Server 106 for the missing Client A Content 108. The request may be based on a time range as discussed above.

EX1016, at ¶¶ 37 and 40; *see also* Fig. 2.

157. Therefore, in every case involving a server 106 transmitting content to the Client C, Sigurdsson requires the server 106 to receive a request from the Client C before the server 106 transmits the content 106 to the Client C.

158. A POSA would consider Sigurdsson's disclosure to limit the transfer described for the alternative embodiment of step 606 to step 620 and its associated steps 612-618 and 626-630. Specifically, Sigurdsson teaches that in a situation where the client (Client C) is not offline (e.g., is online) when the transfer is to occur, the "peer-to-peer techniques can be used by clients that that are currently online to share the content directly between the clients," rather than using the "drop-box" technique implemented using a server. Ex. 1016 ¶62. But even if Sigurdsson were read to have one client send the modified content to the server to synchronize with another client that was online at the time, this hypothetical example would fall outside the scope of claims 4 and 11. Claim 4 expressly recites that "the first application is configured to store the modified first electronic file

copy to the memory device associated with the first electronic device when the third electronic device is not in communication with the first electronic device.”

Similarly, claim 11 recites that “the second electronic device is not in communication with the electronic system.” As such, the hypothetical scenario in which the client is already online does not render features of claims 4 and 11 obvious.

159. Furthermore, as discussed above in detail, Sigurdsson in Fig. 6, and at ¶79 unambiguously states that a client who was offline must request for the missing content from the server (“Client C 406 requests the Client X Content 410 from the Server 106.”). As such, Sigurdsson cannot and does not disclose “automatically sending the modified first electronic file copy to the third electronic device.”

160. In my opinion, Sigurdsson is directed to a pull mechanism, and fails to automatically send the modified first electronic file copy to the third electronic device associated with the user.

161. The requirement of **automatically sending** the modified first electronic file copy to the third electronic device requires the claimed method to not *merely* have the file end up at the third electronic device, but **automatically sending** the modified first electronic file copy to the third electronic. That is, in Sigurdsson, the client 406 (e.g., the third electronic device) first requests a list (of missing content), determines a source content in the list, and then requests for the content from the

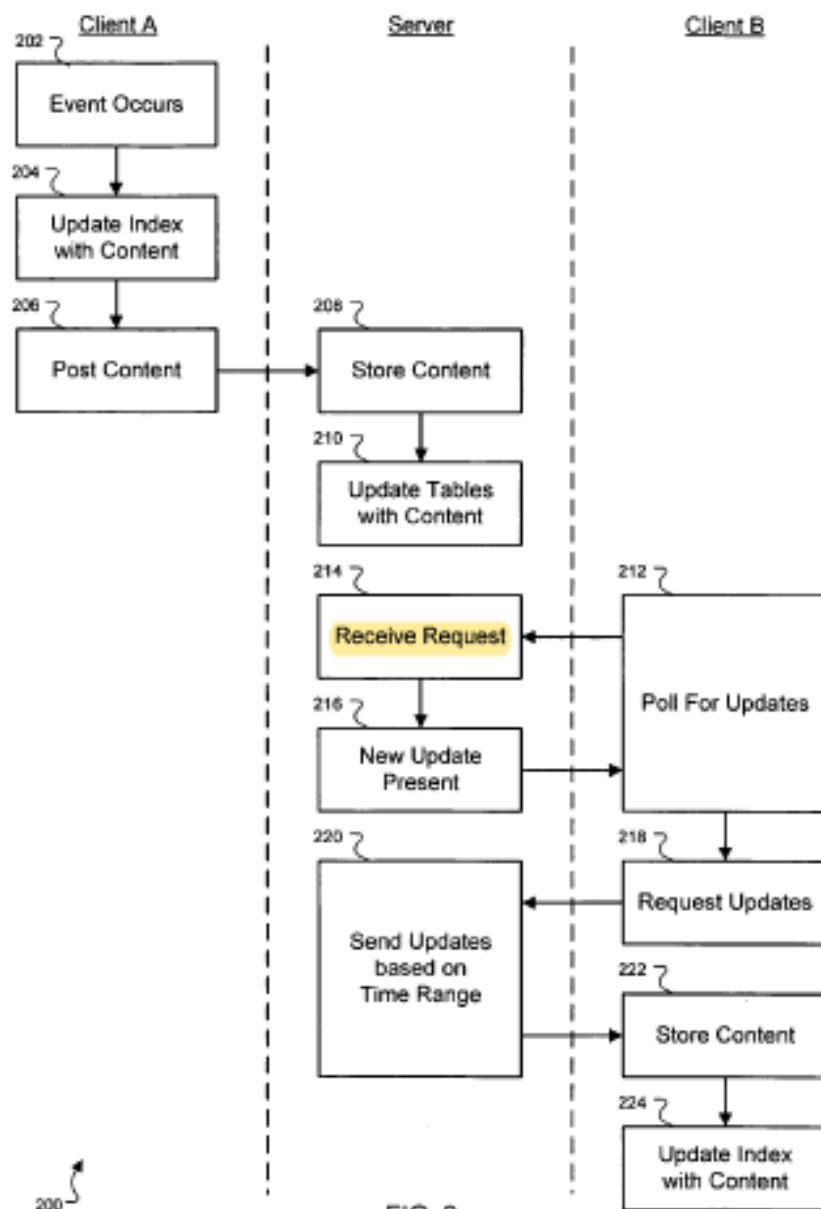
server only if the client determines that the content should be received from the server 106.

162. A POSA would read Sigurdsson as teaching a standard client-server architecture using HTTP. As explained above, HTTP is a pull-based request response protocol. Sigurdsson does not teach or suggest the use of push functionality for its server and nor did the HTTP protocol support such a feature during the 2007/2008 timeframe. I disagree with the petition that it would have been obvious to modify Sigurdsson to incorporate push into its HTTP server.

163. Sigurdsson at ¶ 31 does not correspond to the “automatic send” limitation (*i.e.*, without a client request). Rather, it merely states a result from interactions between Client A and Client C for synchronizing files. The details of the process and structure of the system are omitted in Fig. 1 and ¶ 31 of Sigurdsson for brevity, but lays groundwork for the detailed disclosure in the ensuing passages related to a request from C consistent with a “pull” approach as illustrated in Fig. 2.

164. For instance, Sigurdsson describes Fig. 2 (reproduced below) as a sequence diagram showing an illustrative method 200 for sharing and synchronizing content across multiple client devices **according to the implementation shown in Fig. 1.**

EX1016 at ¶ 32.



EX1016, FIG. 2 (annotations added.).

165. Specifically, Sigurdsson discloses the following steps performed by the system:

- In step 212, the second client polls for updates, representing content generated at other clients. For example, referring to FIG. 1, **the Client B104 can poll the Server 106 for new content.** (§37)
- In step 214, the polling request is received by the server. For example, in FIG. 1, the **Server 106 can receive polling requests** from the Client B 104. (§38)
- In step 216, **the server indicates to the polling client** that content updates are available. (§39)
- In step 218, **the client requests the updated content** that it needs. For example, in FIG. 1, **the Client B104 sends a request to the Server 106** for the missing Client A Content 108. (§40)
- **The request may be based on a time range**
- In step 220, the server sends the updated content **corresponding to the time range requested by the server.** (§41)

EX1016, at §§ 37-41 (emphasis added.).

166. Therefore, in Sigurdsson, several intermediate steps are performed by the Server 106 and the Client B104, including the *pull* steps performed by the Client B104: “**the Client B104 sends a request to the Server 106** for the missing Client A Content 108.”

167. As I explained earlier, a person of skill would not understand HTTP to support push, since HTTP did not include such a functionality. Further, neither paragraph 47, nor other teachings of Sigurdsson, indicate that the server had push

abilities. As a standard HTTP server, it would operate a pull-based server where the client would need to make explicit HTTP requests in order to receive files.

168. At ¶ 47, Sigurdsson *merely* discloses:

- The Client A102 can include both data and applications
- The applications include an Event Listener 322, a File to Text/HTML Converter 324, and a hypertext transfer protocol secure (HTTPS) Client/server 326.
- The HTTPS Client/server 326 can be used as a client that tracks web access and can contain a Time Stamp 328 which associates viewed web pages with a time that they were accessed by the user. The HTTPS Client/server can also be used as a server to provide content to peer clients.
- EX1016, at ¶ 47.

169. Contrary to Dr. Balakrishnan's assertions, paragraph 47 Sigurdsson does not teach "pushing" data "directly to the client from a server." See [RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1 \(ietf.org\)](#).

170. Moreover, Sigurdsson describes Fig. 4 as "a schematic showing a system 400 for sharing and synchronizing content using **a peer-to peer and client/server architecture**. In the system 400, content still can be shared among the Client A 102, the Client B 104 and a Client C 406 using the Server 106 in a method **similar to that of method 200 of FIG. 2.**" EX1016, at ¶ 51.

171. Since Sigurdsson *expressly* states that Fig. 4 implements a method similar to Fig. 2, the method of Fig. 4 also includes steps are performed by the Client B104 to receive content. That is, **the Client B104 sends a request to the Server 106** for the missing Client A Content 108.”

172. Therefore, Sigurdsson, when read as a whole by a POSA, discloses that the server 106 sends content to Client B104 **only after receiving a polling request and subsequently receiving a request for the updated content**. Accordingly, a POSA would understand that Sigurdsson is directed to a pull mechanism, not a push mechanism, and thus fails to disclose “automatically sending the modified first electronic file copy to the third electronic device”.

173. In Fig. 3, Sigurdsson discloses that the client devices 102 and 104 are implemented as a hypertext transfer protocol secure (HTTPS) client/server structure (i.e., HTTPS Client/Server 256 and HTTPS Client/Server 356).

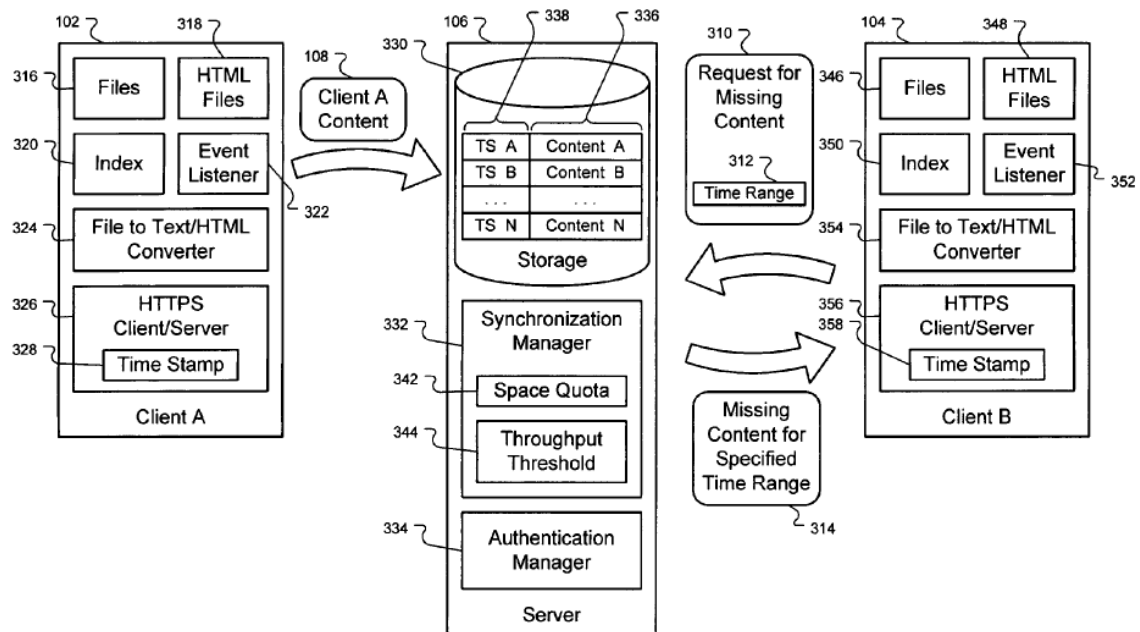


FIG. 3

EX1016, at Fig. 3.

174. During the deposition, Dr. Balakrishnan relied on paragraph 47 of Sigurdsson and stated the following:

I think it's all over the place. 12 So, for example, in Paragraph 47, the last sentence says, "The HTTPS Client/server can be used as a server to provide content to peer clients, which is described in more detail below." It doesn't limit it to being only in response to a request, and that would be a person of skill understanding of how HTTP works, would understand that to be included, pulling and pushing, for example.

EX2024, at 64:11-20.

175. As I explained earlier, a POSA would not understand HTTP to support push, since HTTP did not include such a functionality. Further, neither paragraph 47, nor other teachings of Sigurdsson, indicate that the server had push abilities. As a standard HTTP server, it would operate a pull-based server where the client would need to make explicit HTTP requests in order to receive files.

176. Further, there is no suggestion or teaching in Sigurdsson about any particular limitations or drawbacks of its pull-based HTTP server or any discussion of a push-based server. Consequently, a POSA would not be motivated or find it obvious to replace the pull-based HTTP client/server taught by Sigurdsson with a push-based server that automatically transfers data to a client without any client request.

177. As explained above, Sigurdsson illustrates two scenarios, i.e., a first scenario when **clients are currently online** where peer-to-peer architecture is used and a second scenario when the **offline client comes online again** where client-server or “drop box” architecture is used.

178. Sigurdsson aims to use the server as “drop box” for storing content for a client that is offline, and to that end, when the client is back online, the client can request to server to provide the missing content (i.e., access the “drop box” to

retrieve data) as shown at least in Fig. 2 (step 218), Fig. 3 (step 310), Fig. 6 (step 610) and Fig. 7 (step 710).

179. As I explained earlier, the peer to peer architecture and client server architecture are two fundamentally different methods for file access and are typically used in different situations. Sigurdsson also uses them for two different situations, namely offline and online devices. Given their differences, it would not be obvious to a POSA to use features from one approach and incorporate them into the other.

180. In my opinion, it would not have been obvious to a POSA to combine Sigurdsson and Shappell because Shappell teaches away from using a server.

181. Shappell discloses that “a computer implemented method and system enable users to share files in a **server-less** shared space. EX1020, at ¶ 1. By providing access to such **[server-less]** spaces via a visual presentation, the system renders content available for access by other group members. Access is sometimes provided through propagation of metadata or other uniquely identifying indicia associated with the shared space to all group members. EX1020, at Abstract.

182. Shappell does not merely provide a general preference for an alternative invention. Instead, Shappell expressly discounts the use of a server by specifically requiring a **server-less** shared space. EX1020, at ¶ 6. Shappell, when considered as

a whole, overwhelmingly teaches that peer-to-peer system without using a server. *See*, for example, EX1026, at ¶¶ 1, 7, 37, and 46.

183. Shappell repeatedly teaches using a shared space without a server. And for good reason. Shappell was concerned about facilitating group interaction through which a desired group of users can interact with each other in a secured environment, and through which users can access, and maintain control with respect to files shared in their shared relationships with one another. EX1020, at ¶ 6.

184. Specifically, Shappell was trying to address problems with existing network interaction mechanisms that do not provide a mechanism for readily manipulating shared data and information among a desired group of users or an user interface to simplify management of such shared data. EX1020, at ¶ 5.

185. On the other hand, Sigurdsson proposes a system including a client/server mechanism. Therefore, it would not have been obvious to a POSA to have modified the teachings of Sigurdsson to provide a server-less shared space (i.e., replace the client/server mechanism) as disclosed in Shappell, because doing so would change the principle of operation of Sigurdsson.

186. A central feature of Sigurdsson network architecture is using the client/server mechanism for providing access to client that are offline, and then come online. Modifying Sigurdsson's architecture to a server-less shared space

would change the principle of operation of Sigurdsson and as such a POSA would not have modified the teachings of Sigurdsson with the teachings of Shappell.

187. Claim 1 recites the first application executed on the first electronic device **automatically sending** the modified first electronic file copy **to the third electronic device** to replace an older version of the first electronic file stored on the third electronic device with the modified first electronic file copy having the content modified by the user, where the second electronic device and the third electronic device are each associated with a **same** user.

188. Thus the second electronic device and a third electronic device are **associated with the same user**.

189. The **associated with the same user** limitation requires the claimed method to send the file not only a specific user but a device that is associated with the very same user that sent the file.

190. A POSA would not find that Shappell's sharing files **among group members** in the shared space teaches the **associated with the same user** limitation. While Shappell discloses sharing a file with members of a group, Shappell simply does not send the file to a specific device "associated with the user."

E. Ground 2 of the '942 Patent – The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Brown and Hesselink

191. As stated earlier, the focus of the '942 Patent is to provide a user-centric computer network architecture for file synchronization, where a server automatically receives a modified file from one device associated with a user when the user modifies the file and the server **automatically transfers** it (i.e., without a request from the client for the modified file). This is in contrast with the disclosures in Brown and Hesselink that explicitly require a request from the client to initiate data transfer to another device **associated with the same user** (as opposed to any devices connectable over a network as in Hesselink).

192. Like Sigurdsson, in Brown and Hesselink, the client must initiate a request for any update to the connection server. Brown discloses that communication between the clients and server uses the hypertext transport protocol (HTTP).

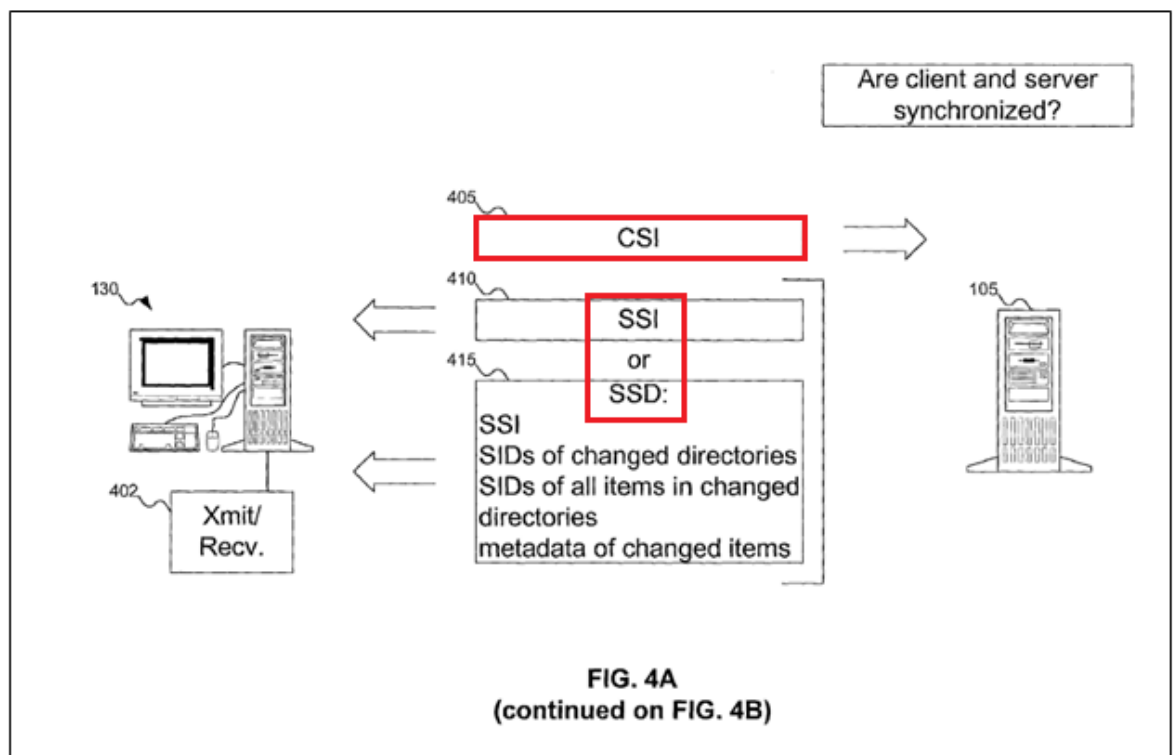
EX1025, at 4:2-5. Also Hesselink consistently states that its data transfer process is a HTTP request/response process. See e.g., EX1026, at Fig. 8, steps 810, 830. As discussed above, the HTTP protocol does not support push-based functionality at the time of the invention. Therefore, to a POSA, the combined teachings of Brown and Hesselink, do not disclose that a modified file would be automatically transferred to a client without a request from the client.

193. Brown discloses a transfer of changes to a file from a server to a client is initiated by the client. Paper No. 3, at 55 (“After execution of the functionality

discussed for claim [1d], above, client 135 would initiate communication with the server and receive a copy of the modified electronic file sent to server 105 by client 130, as described above for claim [1g]. EX1025, at 2:63-67, 3:1-8; EX1028, at ¶¶896-898).”)

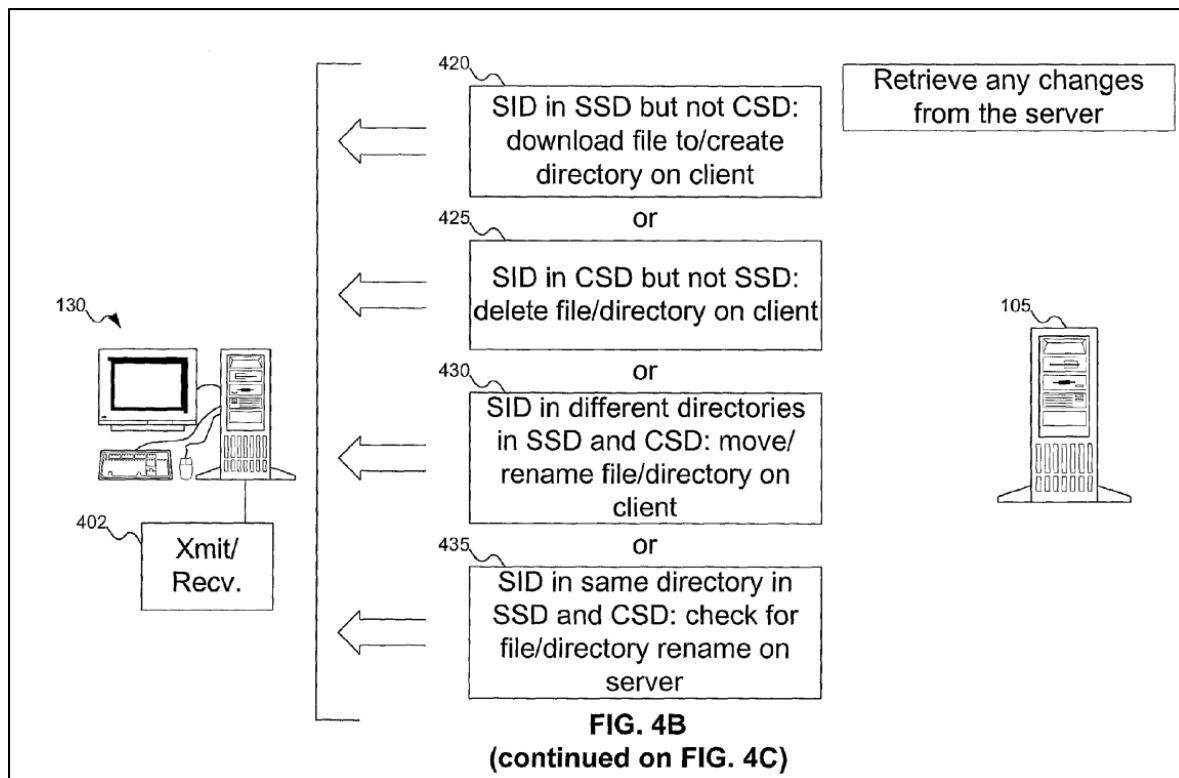
194. Petitioners rely upon Fig. 1 of Brown depicting client 130 as an exemplary “second electronic device” and client 135 as an exemplary “third electronic device.” Paper No. 3, at 40. However, Brown only discloses that client machines synchronize with each other by synchronizing to a common server account. EX1025, at 1:59-63. In Brown, the synchronization process is only between the initiating client synchronization application (SA) and the server. Referring to the embodiment of Fig. 4A of Brown, the synchronization process is only between the client 130 and the server 105, and this synchronization process does not synchronize another client 135 with the server 105. In order for the other client 135 to synchronize with the server 105, the other client 135 must initiate synchronization with the server 105 on its own schedule. Id., at 3:42-46. Unless the other client 135 initiates the synchronization process by making a sync poll call to the server 105, the server 105 would not send the modified file received from the client 130. Id., at 5:14-24 (“The synchronization process is **initiated by a client SA when it makes a sync poll call to the server**, passing it the client sync index (CSI) identifying its current known state of the account.”)

195. Thus, Brown requires each client to maintain its client sync index (CSI), which identifies its current known state of the user's server account. Id., at 5:14-17. The server needs to receive the CSI from the other client 135 and compare it with a server sync index (SSI) to determine if there is any change client 135 lacks, before proceeding with the synchronization process with client 135. In the absence of a CSI received from client 135, Brown's server 105 cannot automatically transmit the modified file received from the client 130 to the client 135. EX1025, at 10:45-48.



EX1025, Fig. 4A (annotations added.).

196. Brown further illustrates in Fig. 4B the update process **performed by the client** in which the client determines what updates need to be retrieved from the server based on any of four possible results of the comparison between the SSD and the CSD. The client then requests the server to send the changes related to the file or directory. EX1025, at 10:45-48. That is, in Brown, the client retrieves the changes from the server **only upon the server receiving the request from the client**.



EX1025, Fig. 4B

197. Thus, in Brown, to synchronize with the server, **the client must actively pull down changes from the server**, as opposed to the modified file being

“automatically” transferred from the server to the client. Clearly, Brown uses a protocol that uses a “pull” mechanism and not the server automatically pushing the modified file to the client. In the “pull” mechanism, the client does not know when new data is available and the client must poll the server to see if there is new data available to the client and request that data from the server (i.e., pull the data).

198. For the alleged teaching of claim [1g], the Petition relies on EX1025, at 2:63-67, 3:1-8 and EX1028, at ¶¶896-898 and argues “After execution of the functionality discussed for claim [1d], above, client 135 would initiate communication with the server and receive a copy of the modified first electronic file sent to server 105 by client 130.” Paper No. 3, at 55. However, Petitioners ignores the requisite steps between the client 135 initiating communication and receiving a copy of the modified file from the server 105. As discussed above, in order for the client 135 to begin the synchronization process, the client 135 (i) first should poll the server for changes by other clients by passing its current CSI to the server in a sync polling call and (ii) request the server to send server synchronization data (SSD) if the CSI does not match the server account's SSI value. EX1025, at 8:49-53. Petitioners also argue that Brown discloses automatic file synchronization, citing Brown at 3:40-42. (“From the user's perspective, the synchronization process is automatic, runs in the background and requires minimal monitoring or intervention.”).

199. The process described in Brown may be automatic from the user's perspective, meaning that the process may need little or no involvement of human operation from the user but it is not an automatic send from the server to the client

200. While Brown's file synchronization requires the client to execute prescribed steps, Brown expressly teaches its "automatic synchronization" as being automatic from the *user's* perspective as long as the client does not necessitate any action from the user in order to initiate the synchronization. EX1025, at 3:40-42. To a POSA, this use of the phrase "from the user's perspective" in connection with file synchronization, limits Brown's disclosure and is different from the automatic send in the '942 patent.

201. In contrast, the automatic sending in the claims of the '942 Patent is automatic from the *server's* perspective (that is, the first electronic device executing the first application). See claim 1 of the '942 Patent, "**a first electronic device ... automatically send, via the first application**, the modified first electronic file copy to a third application" and similarly claim 10 of the '942 Patent.

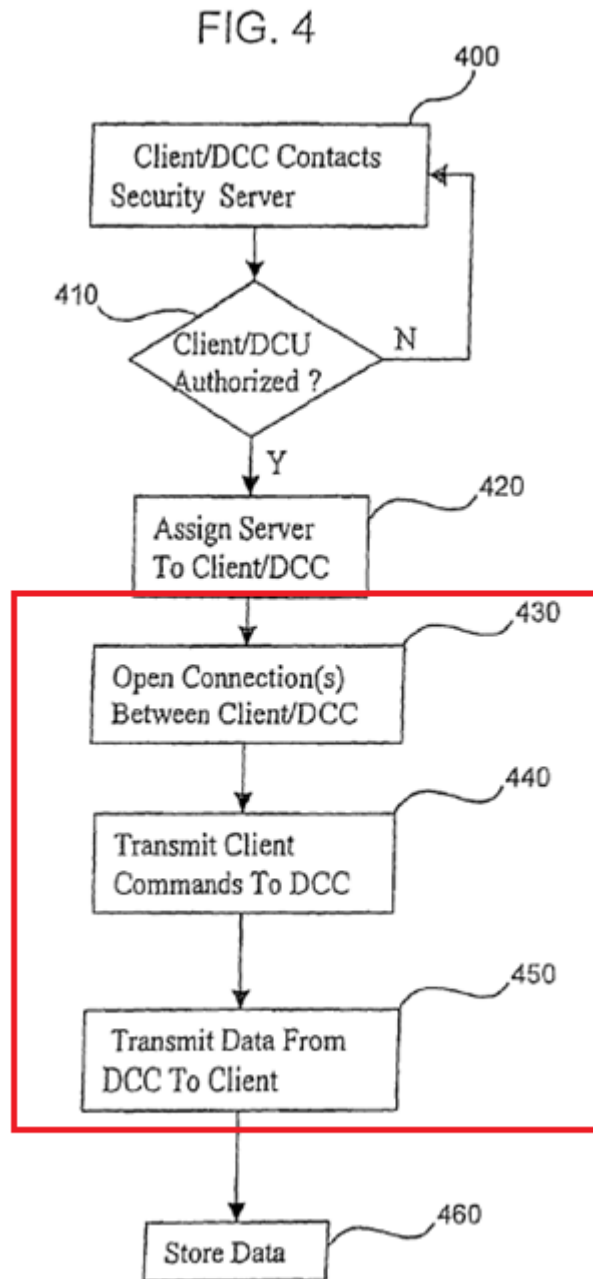
202. That is, the automatic file transfer of the '942 Patent teaches a push-based file transfer from the server to the client, thus allowing the server to send the modified file to the client without a need for the client to explicitly request it. Brown's disclosure is directed to the opposite method, that is, client pull.

203. In Brown, the client SA must initiate synchronization by sending a request to the server. EX1025, at 8:49-55 (“The client polls the server for changes by other clients by passing its current CSI to the server in a sync polling call ... The client SA requests server synchronization data (SSD).”) Absent such request, the server would not transfer the modified file to the other client.

204. A POSA would not rely on Hesselink at ¶ 183 as disclosing teaching automatic push from the server. Hesselink at ¶183 states: “As noted above, when data is updated, such as when a file is edited, for example, the **system** may be configured to automatically update all or some designated subset of computers on the network as soon as the update has been performed at one location.” However, this paragraph merely states that all or some computers are automatically updated but does not discuss how the automatic update is done. Specifically, Hesselink at the cited portion or anywhere else does not contain any disclosure of the **server** automatically pushing the modified file.

205. It should be noted that Hesselink at ¶183 refers to the previous discussion of file synchronization that requires a client request, because the paragraph begins with “As noted above.” There is no discussion whatsoever of a push-based file transfer from the server in the preceding paragraphs in Hesselink. To the contrary, Hesselink only discusses the pull-based synchronization throughout its disclosure, for example, as illustrated in detail with reference to FIGS. 4 and 8.

206. Referring to Fig. 4, Hesselink describes the operation of the access and control system 54. If the user of client computer 12a-12n is authorized in event 410, security server 58 assigns one of the connection servers 14a-14n to the user of client computer 12a-12n, in even 420. In event 430, a connection is made between the connection server 14a-14n assigned in event 420, and the network-enabled device 18a-18n selected by the user authorized in event 410. In event 440, client computer 12a-12n transmits instructions or commands to a network-enabled device 18a-18n, and in event 450, network-enabled device 18a-18n transmits data to client computer 12a-12n. EX1026, at ¶¶86-87. Hesselink at ¶87 further discloses, referring to events 440 and 450: “Client process 32 on client computers 12a-12n periodically sends HTTP requests to connection server 14 a-14 n that include embedded command data. Device control process 38 on device control computer 36 a-36 n also sends periodic HTTP requests to connection server 14 a-14 n with embedded data from network-enabled device 18 a-18 n.”

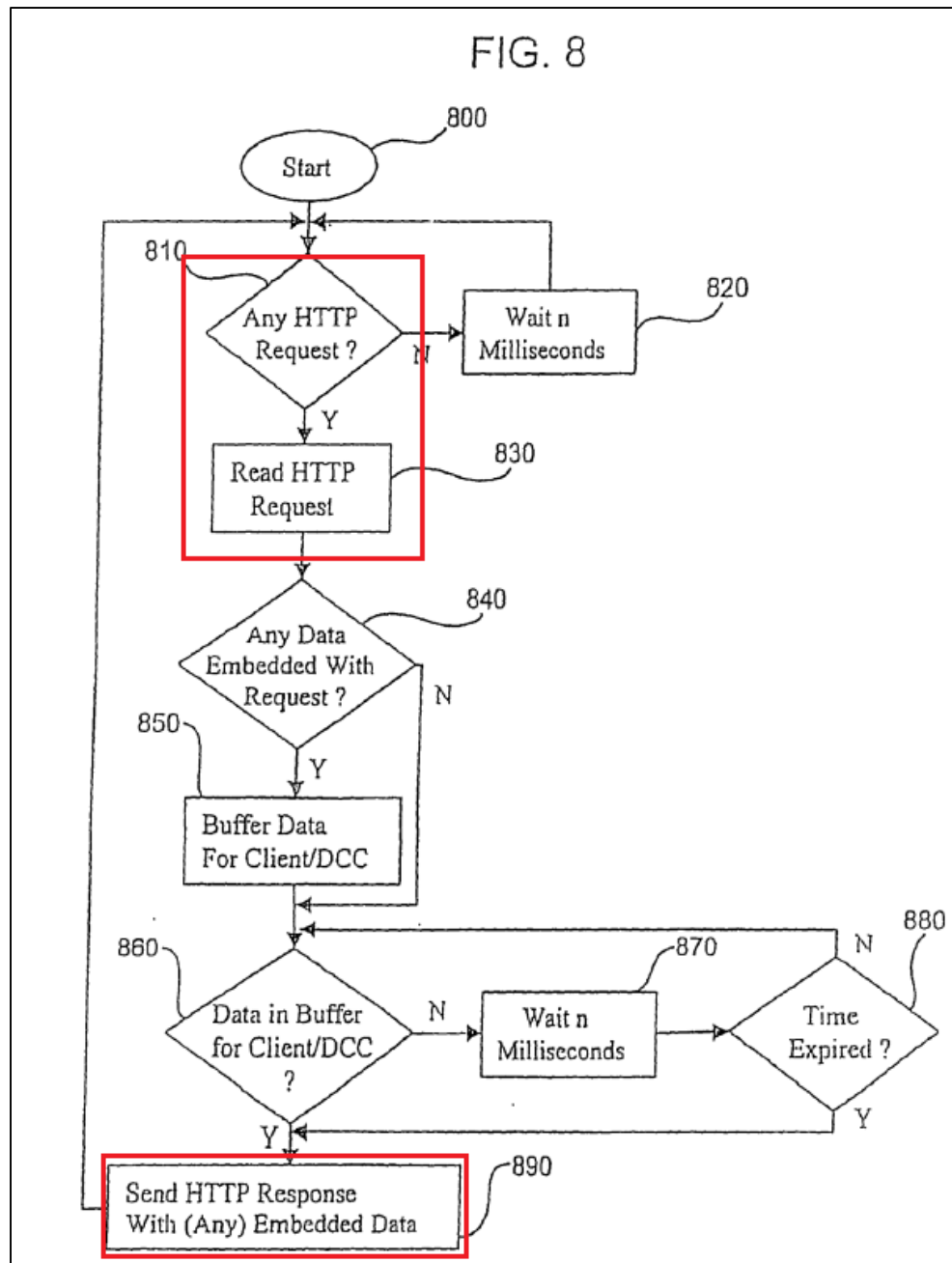


EX 1026, Fig. 4 (annotations added.).

207. Thus, just like in Brown, Hesselink uses a pull mechanism in its file synchronization process. That is, in Hesselink, the client **must initiate a request** for any update to the connection server. The HTTP request/response process for

data transfer between the client and the connection server is also illustrated in Fig. 8 of Hesselink. EX1026, at FIG. 8. The process includes event 810 in which it is determined “whether or not **any HTTP request has been received** from client computer 12 a-12 n (or device control computer 36 a-36 n)”; if an HTTP request has been received and there is no data is embedded within the HTTP request, event 860 in which it is determined “if there is any data present for client computer 12 a-12 n from network-enabled device 18 a-18 n”; and event 890 in which “**an HTTP response** is sent to client computer 12 a-12 n by connection server 14 a-14 n,” and “[i]f any data was present in the connection server sending buffer for client computer 12 a-12 n, this data is **embedded within the HTTP response.**” Thus, Hesselink is precisely focused on a data transfer process based on client pull, which is implemented by clients sending HTTP requests to the server to request the data. This is the opposite to the server initiating data push to a client.

208. As explained earlier, the HTTP protocol is a request-response protocol and the HTTP servers use the client-pull architecture. The standard HTTP protocol practiced at during the 2007/2008 timeframe did not support push-based functionality where the server can automatically send data to a client without the client making requests for data.



EX1026, at Fig. 8

Indeed, Hesselink’s synchronization process only begins when a request for access to data is received. *See* EX1026, at Abstract (“**Upon receiving a request for access to data** originally stored in a remote storage device, determining whether the

data is already available and valid in a local storage device.”); ¶ 78 (“client computers 12 a, 12 b may each comprise programming 32 stored in memory that allow client computers 12 a, 12 b to **send instructions** to network-enabled devices 18 a, 18 b **as requests** through connection server 14, and receive data from network-enabled devices 18 a, 18 b **via responses** through connection server 14.”) Hesselink does not disclose or suggest any alternative embodiment in which a server automatically transfers the modified file without a client request.

209. Hesselink describes an embodiment where a file transfer can be based on an active subscription of the local device to the remote device. EX1026, at ¶141 (“Further, device module 74 sd may also listen to file access events from its local operating system to detect any changes to files and folders that are actively subscribed by connected user modules. Device module 74 sd transmits any locally detected file overhead modifications (detected through the described listening) to all connected and actively subscribed user modules.”). However, even with the active subscription, Hesselink clearly states that a user should still make the initial network request to a remote directory in remote storage device and **continuously and repeatedly makes data access requests** to that remote storage device to locally store information. EX1026, at ¶140 (“[E]ven if the user uses local applications to **continuously or repeatedly makes data access requests** to a remote directory in remote storage device 18 b, user module 72 only needs to make

a network request once during the period of the active subscription, since after the initial request, the information is stored locally at the site of the user.”).

210. For at least the reasons above a POSA reading the description in Hesselink at ¶183 would not consider that description as disclosing the “automatic sending” feature in [1d] and [1e] of claim 1 and similarly in features [10c] and [10f] of claim 10.

211. In my opinion, to a POSA, Hesselink fails to disclose the first application associated with a user automatically sending a modified first electronic file copy to the third electronic associated with a same user.

212. As explicitly recited in the preamble of claim 1, each of the first electronic device, the second electronic device, and the third electronic device are all associated with a user, i.e., the same user.

213. Specifically, claims of the '942 Patent recite a system of specifically arranged applications (that is, “a first application”, “a second application” and “a third application”) that are executable on multiple electronic devices, each associated with the same user, to provide a user-centric computer network architecture. The claims describe how this arrangement of applications operates to automatically send a copy of an electronic file that is modified at one electronic device to another electronic device based on the triggering event (i.e., the user modifying the electronic file). Thus, file sharing in the '942 Patent is essentially

integrated with these applications and is implemented on a client-server system, as illustrated in Fig. 3 of the '942 Patent (reproduced below). Referring to Fig. 3 of the '942 Patent, the server 230 transfers a file-transfer and/or synchronization application (which is “a second application” and “a third application”) to the user devices 210, 280 for installation thereon, which then provides file-transfer clients 310, 320 executable by the user devices 210, 280, respectively. Based on the file-transfer application, a user can manage and synchronize files across multiple devices.

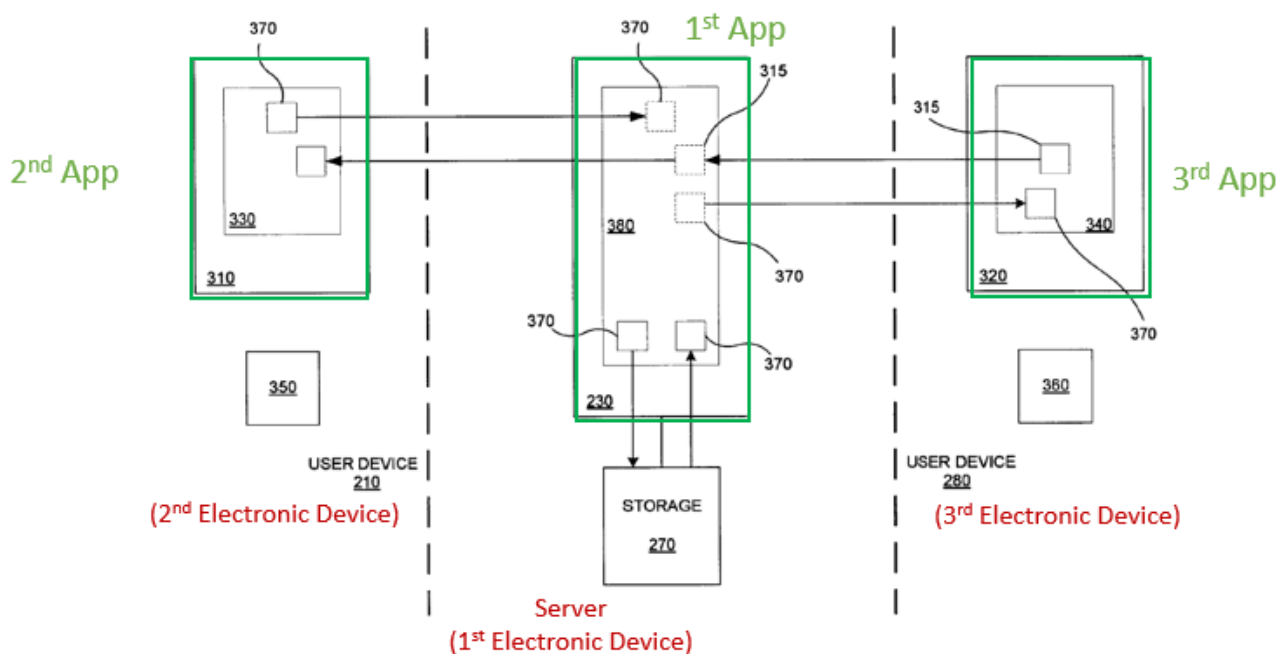


FIG. 3

EX1002, Fig. 3 (annotations added.).

214. The **automatically sending** limitation requires the claimed method to not merely have the first electronic file copy modified by a user at the second

electronic device end up at any third electronic device, but the limitation requires the first application executed on the first electronic device (i.e., the server), associated with that user, to find the third electronic device that is associated with the same user and automatically transfer the modified first electronic file copy to the third electronic device via the third application executed on the third electronic device. The “automatic sending” feature of the asserted claims should not be considered in isolation but must be considered in connection with the specific arrangement of first, second and third applications running on three different electronic devices, including the server between the multiple devices to be synchronized. It is the use of these application programs and their arrangement that made possible the “automatic sending” to another device (i.e., without the another device requesting it) associated with the same user.

215. ¶ 183 of Hesselink states:

As noted above, when data is updated, such as when a file is edited, for example, the system may be configured to automatically update **all or some designated subset of computers on the network** as soon as the update has been performed at one location. Computers that are offline at such a time will be automatically updated as soon as those computers go online again. The persistence of the system ensures that **updates are completed for all online computers**, as if there is some failure or problem

during the update, the connection restores itself to ensure that the update is completed....

Ex1026, at ¶ 183 (emphasis added.).

216. The “all or some designated subset of computers on the network” and “online computers” discussed in this paragraph refer to client or user computers 12a, 12b, that may be operatively coupled to a connection server 14 via a public or global network 16 and one or more network-enabled devices 18a, 18b that are also operatively coupled to connection server 14 via global network 16. *See* EX1026, at Fig. 1 and ¶¶ 76, 120-121.

217. That is, Hesselink only discloses updating all or some of computers online that the user has access to. The file-access permission is stored in the database 52 and permission management information can be downloaded and stored in the user module's local cache to determine what data access functionalities that the particular user module can perform and to which files or directories may be accessed on which remote device modules. *Id.*, at Fig. 10 and ¶132. Therefore, Hesselink does not make a determination as to whether the third electronic device is associated with the same user by use of the first, second, and third applications recited in the asserted claims, and hence Hesselink cannot automatically send a modified file based on such determination. While Hesselink at Fig. 10 mentions application 72a (e.g., Windows Explorer®), this application is only for accessing remote storage device 18b at computer 74 over a wide area network. EX1026, at

¶127. Thus, Hesselink does not teach a dedicated file transfer and/or synchronization application as recited in the asserted claims.

218. As Hesselink discloses, updating all or some of online computers the user has access to, cannot be construed as teaching “automatically sending” the modified first electronic file copy to the third electronic device that is associated with the same user, because such construction would render “associated with a user” language meaningless. In the ’942 Patent, based on the “association” with a user established via the first, second, and third applications executable on different devices, the server can find the third electronic device to which the “automatic sending” should be performed. In contrast, Hesselink only looks at whether the user is making a data access request to a remote directory in the remote device, in order to determine whether to update the user’s local device. Hesselink’s automatic update is not based on the determination of the third electronic device being associated with a user via the third application.

219. In my opinion, a POSA would not be motivated to modify Brown’s disclosure to that of a push-based file transfer due to the differences between the pull and push-based methods.

220. Brown intentionally and purposefully chose the “pull” based approach, and as such, a POSA would not be motivated to modify Brown to apply to the “push” based file transfer.

221. Brown specifically states at 3:17-28:

The client/server architecture is designed to **minimize the load on the server processor** in order to maximize server scalability. To that end, as many processor-intensive operations as possible, such as message digest computation and comparison, data encryption and decryption, and **synchronization itself, are performed by the client SA**. Also, the **polling mechanism used by the client SA** to determine if the client is synchronized with the server is designed to **require minimal processing on the server** when the client and server are in sync. This is significant because as a rule only a small percentage of clients require synchronization activity at any point in time.

EX1025, at 3:17-28 (emphasis added.).

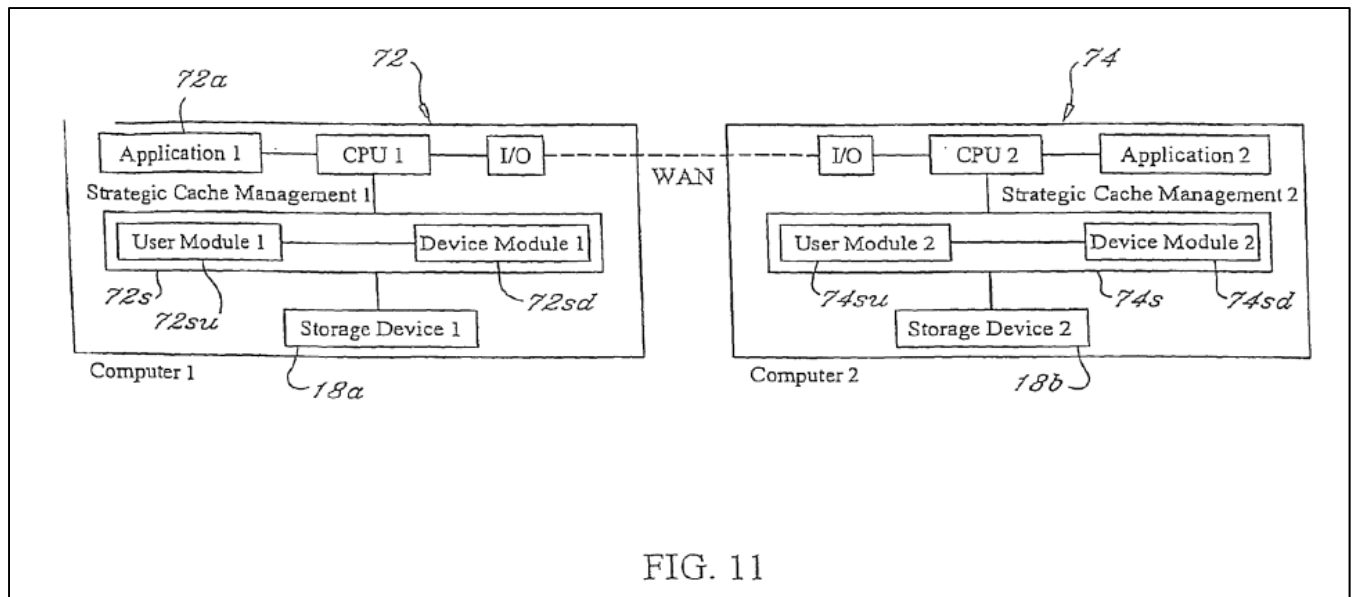
222. Brown aims to minimize the load on the server processor, and to that end, as many synchronization operations as possible are performed by the client, including the operation to make a sync poll call to the server. Thus, the intended principle of operation and purpose of Brown teaches away from using the server-push mechanism for file synchronization.

223. For at least the reasons mentioned above, there is lack of motivation for a POSA to modify Brown to incorporate “automatically sending” from a server to a client since it would fundamentally change the operation of Brown’s system.

224. In my opinion, Hesselink's peer-to-peer embodiment cannot be modified to teach "automatic sending" of the modified file from the server to a client device as required by claim 1 of the 942 patent.

225. The Petitioners argues that: "[I]t would have been obvious in view of Hesselink to automatically forward updated files from server 105 to synchronizing clients as soon as file updates are received at server 105," citing Hesselink, ¶183. Paper No. 3, at 56. However, updating designated subset of computers when data is updated as described in Hesselink, ¶183 is from the context of a file subscription discussed in Hesselink, FIGS. 11-13, which is based on a peer-to-peer connection between two computers, as opposed to a client-network architecture.

226. Referring to Fig. 11 of Hesselink, two computers 72, 74 are connectable across a wide area network and a user module 72su of the computer 72 accesses data from remote storage device 18 b and user module 74su of the computer 74 accesses data from remote storage device 18 a.



EX 1026, at Fig. 11

227. Referring now to FIGS. 11 and 12 of Hesselink, where the user module 72su of the computer 72 receives a request from the user to browse through files and directory of the remote file systems of storage device 18 b, “it may first check its local cache to find whether such file overhead information is available and valid at event 1202.” EX1026, at ¶137. If the user module 72su is actively subscribed to the device module 74d of the second computer 74 and an active subscription period is running, it is determined that there is file overhead information that is available and valid at event 1204, and the device module 74sd sends any updates to user module 74su regarding the file overhead information of the associated directory over the prescribed active subscription period, at event 1214. EX1026, at ¶¶137, 139.

228. In this case, Hesselink merely discloses updating, by a first computer, a second computer with the file overhead information such as file name, file size, folder status, last modified date, created date, etc. regarding its directory (Ex1026, at ¶71), upon receiving a request to access to its local storage from the second computer.

229. I disagree with the petition that a POSA would not have been dissuaded from combining the peer-to-peer teachings of Hesselink with Brown's client/server model. This is opposite to what Hesselink actually teaches. Hesselink addresses the security concerns in using server/client architecture for file synchronization.

("[M]any users may be uncomfortable storing photos of their families on some photo service server in some unknown location where it is also often unknown what level of security (if any) such server is provided with."; EX1026, at ¶6). To solve this problem, Hesselink proposes a peer-to-peer method that allows users to access their files on any network accessible communication device, from any location accessible over that network. EX1026, at ¶10. Thus, contrary to the Petitioner's assertion, Hesselink clearly teaches away from implementing its method in a client/server model.

230. The Petitioner argues that "Hesselink does not teach that the devices in its synchronization network must not include any servers" and cites Hesselink, ¶122 as teaching any of computers 72, 74 and 76 as being a computer device including

“file server.” Paper No. 3, 52. However, as discussed above, while Hesselink mentions including a server within its peer-to-peer system, such server only serves as a pipeline or conduit between two peers. Also, the “file server” mentioned in Hesselink, ¶122 is only an example of a type of computing device of computers 72, 74 and 76, each of which are peers in the peer-to-peer architecture (i.e., nodes between which a direct link is provided). There is nothing that suggests in Hesselink that the “file server” implements file synchronization between computers 72, 74, and 76 based on a client-server architecture.

231. As explained earlier, Client Server architecture and Peer-to-Peer architecture are fundamentally different and are used in different situations. Therefore, a POSA would not be motivated take features from one of the architecture to make changes to the other architecture.

232. Since Hesselink’s file synchronization is based on the peer-to-peer system, a POSA would not be motivated to combine it with file synchronization in Brown’s client-server system as these systems employ two fundamentally different architectures. A peer-to-peer network allows computer hardware and software to communicate between peers. Unlike a client-server architecture, there is no central server for processing requests in a peer-to-peer architecture. A peer-to-peer based file synchronization is entirely different from the client-server system because each node is the one that processes the query from another node to decide whether to

grant or deny it and thus each node should have the capability to access and manage information of other nodes. Hence, A POSA would not be motivated to combine the teachings of Brown and Hesselink because these two references teach fundamentally different architectures.

IV. CONCLUSION

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.

Respectfully submitted,

A handwritten signature in dark ink, appearing to read 'Prashant Shenoy', is written over a horizontal line.

Prashant Shenoy, PhD.

Date: November 21, 2023