

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 9,143,561

Case IPR2023-00429
Patent 10,289,607

Case IPR2023-00433
Patent 10,067,942

DECLARATION OF PRASHANT SHENOY, PH.D.

TABLE OF CONTENTS

I. BACKGROUND AND QUALIFICATIONS.....	3
II. LEGAL FRAMEWORK.....	6
A. Anticipation	6
B. Obviousness.....	7
III. OPINION	9
A. Claim Construction	10
B. Background of the Technology	10
C. '561 Ground 1: The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Sigurdsson and Shappell	14
D. '561 Ground 2: The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Brown and Hesselink	21
E. '942 Ground 1: The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Sigurdsson and Shappell	34
F. '942 Ground 2: The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Brown and Hesselink	43
G. '607 Ground 1: The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Sigurdsson, Shappell, and Rao.....	54
H. '607 Ground 2: The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Brown, Hesselink, Shappell, and Rao	63
IV. CONCLUSION.....	78

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 broadly 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 conferences, and those publications have been cited 24,000 times. My research has received several awards, including best paper awards and the ACM Sigmentrics 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 2025.

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 this matter, 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”)
1018	U.S. Patent No. 7,734,690 B2 to George P. Moromisato, et al. (“Moromisato”)
1019	U.S. Patent Application Publication No. 2006/0190506 A1 to Rajest M. Rao, et al. (“Rao”)
1020	U.S. Patent Application Publication No. 2005/0091289 A1 to Michael Shappell, et al. (“Shappell”)
1023	European Patent Application Publication No Ep 1,621,990 A2 to Kitamaru, et al. (“Kitamaru”)
1024	U.S. Patent No. 5,923,325 to Ronald Jason Barber and Edwin Joseph Selker (“Barber”)
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”)
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. 3	Petition for IPR of the ‘607 Patent (IPR2023-00429)
Paper No. 3	Petition for IPR of the ‘787 Patent (IPR2023-00431)
Paper No. 3	Petition for IPR of the ‘823 Patent (IPR2023-00432)
Paper No. 3	Petition for IPR of the ‘622 Patent (IPR2023-00430)

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 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.

13. 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.

14. 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.

15. It is my understanding that for purposes of this proceeding, Petitioners are

of the opinion that a POSA in November 2007 would have been: (1) a person with an undergraduate degree in computer science or closely related field and (2) two or more years of experience with systems and architecture for file management in a distributed network environment.

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, and 10-11 of 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 person of ordinary skill in the art ("POSA") would have had at the time of the invention and priority date of the '562 Patent, and the documents listed in the above-mentioned table.

18. In my opinion, claims 1, 3-4, 10, and 12-13 of 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 person of ordinary skill in the art ("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. In my opinion, claims 1, 3-5, 7, 12, 14, 17, 19, and 20 of the '607 Patent are patentable based on my review of the '607 Patent, the file history, and scope of the references discussed herein and in the petition, the knowledge and skill of a person of ordinary skill in the art ("POSA") would have had at the time of the invention and priority date of the '607 Patent, and the documents listed in the above-mentioned table.

20. I have also relied on my considerable academic and practical experience about the relevant field of technology in forming my opinions as expressed herein.

A. Claim Construction

21. In my analysis I have interpreted the claims in accordance with their plain and ordinary meaning, as instructed. In my opinion, these interpretations are consistent with the claims, specification, and knowledge of a person of ordinary skill in the art.

B. Background of the Technology

22. The '561 Patent¹, including the claims thereof, relate to architecture for management of digital files across distributed network. EX1001, Title, Abstract.

¹ While U.S. Patent No. 9,143,561 (the "'561 patent"), U.S. Patent No. 10,067,942 (the "'942 patent"), and U.S. Patent No. 10,289,607 (the "'607 patent") are discussed in this declaration, since these patents share a specification, references have been made to the specification of the '561 patent.

23. The ‘561 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. EX1001, Title, Abstract.

24. The background of the ‘561 Patent describes various problems in connection with computerized file management systems circa 2007/2008 timeframe. *Id.*, 1:22-3:39. As the ‘561 Patent explains, at least as late as 2007, users of modern computing systems were increasingly finding themselves in high-speed networked environments. *Id.*, 1:24-32. With the 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 handheld 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.*, 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.*

25. Although a user could copy files 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.*, 1:40-44.

This conventional process was tedious to resolve, error prone, and complex. *Id.*, 1:40-61.

26. Since copying of files across a user's devices was tedious and complex, another approach for accessing files from various devices was for users to store the file at a central location, such as a file server. *Id.*, 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.*, 2:19-26.

27. At the time, 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 inaccessible, 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.

28. These server-centric improvements for providing the most current version of a file to a user still had problems. *Id.*, 1:40-61. These problems frustrated the goal of providing customers with a seamless environment in which a user could use any one of a variety of the user's electronic devices to access the most current version of the user's file. *Id.*

29. Since server-centric systems stored files centrally on a server, alternative approaches were needed since the file became unavailable when the network was unavailable. Accordingly, users would store a copy of the file locally in addition to the version that resided on the server. However, this required that the local copy be synchronized with the server so that the server and the client had the up-to-date version of the file. Moreover, in view of communication network capabilities at the time, file synchronization techniques that made efficient use of network bandwidth were needed.

C. '561 Ground 1: The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Sigurdsson and Shappell

30. I have reviewed the Sigurdsson reference, and in my opinion, Sigurdsson reference teaches an HTTP server architecture for file synchronization which is fundamentally different from the teachings of the '561 patent.

31. Sigurdsson discloses a system, in which, a receiving device (i.e., a Client C 406) performs several operations before requesting a copy of missing content.

Much of Sigurdsson discusses client devices requesting such missing content from other client devices, in a peer-to-peer arrangement. See Abstract. Sigurdsson

describes an embodiment in which Client C obtains missing content from a server

106. But that embodiment requires Client C to expressly request the missing

content. Only in response to the server receiving the request from Client C does it transmit that content to Client C. (FIG. 4, 6, ¶¶71-75, 80). Therefore, in

Sigurdsson, the server does not automatically transfer a modified first electronic

file copy to the receiving device, such as Client C, since it only transfers the

missing content when the receiving device expressly requests it.

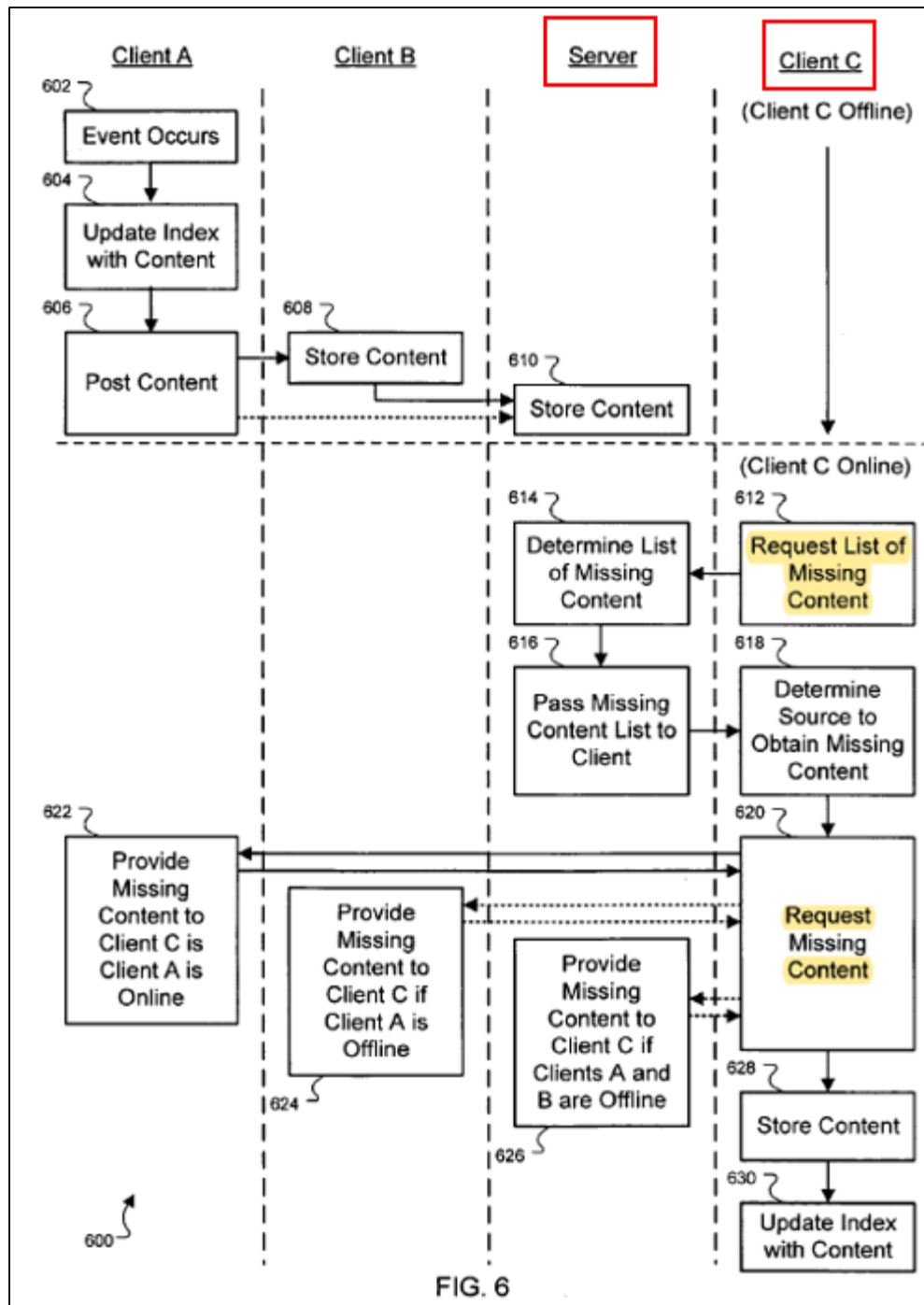
32. The automatic transfer in the claims of the '561 Patent refers to the first electronic device automatically transferring the copy of the first file.

33. Sigurdsson fails to teach or suggest that the first electronic device

“automatically transfer the modified first electronic file copy to the third electronic

device.” Sigurdsson’s transfer to Client C is the result of Client C requesting the transfer.

34. Sigurdsson, in FIG. 6 (reproduced below with annotations), discloses a sequence diagram of a request/response based mixed peer-to-peer and client/server architecture, including a server, and clients A, B, and C. As shown in FIG. 6, the server in Sigurdsson provides missing content only in response to a request from a client (e.g., “Client C” at steps 612 and 620). That is, in FIG. 6, client/server structure and operation of Sigurdsson requires at least two separate requests from the client device (“Client C”) to the server, before the server provides the missing content to the client device, neither of which are “receiving the copy of the first file from the first client device”.



Sigurdsson (EX 1016), FIG. 6

35. It is only after going through this sequence of Client C expressly requests the server for any missing content, Client C determining where it can obtain that

missing content, and then requesting the content from the server, before the server responds by transferring the content to Client C. As such, Sigurdsson does not teach or suggest that the server 106 automatically transfers the content 410 to the Client C 406 responsive to receiving the content 410 from Client A 402.

36. Sigurdsson at FIG. 2 and ¶¶ 37, 40 and 46 discloses a request for missing content from the second client. For instance, “**the second client polls** for updates” (¶37), the request is “**periodically issue[d]**”, and “**the request 310 includes a Time Range 312 parameter** identifying the range of time stamps for the missing content” ¶40 and “the Request 310 is issued **in order** for the Client B 104 to obtain the Client A Content 108 that the Server 106 has yet to send to the Client B104” (¶46).

37. FIG. 5 and the related text is general description without disclosing several of the claimed features including the “the first electronic device “automatically transfer the modified first electronic file copy to the third electronic device” as discussed above. In fact, the disclosure in FIG. 5 simply summarizes the request-based file transfer approach from FIG. 6 discussed above in detail.

38. When viewed under proper context, Sigurdsson discloses that only if the server 106 receives a request for the content from the Client C 406 (step 620), will the server provide the content to the client (step 626). ¶¶ 79-80.

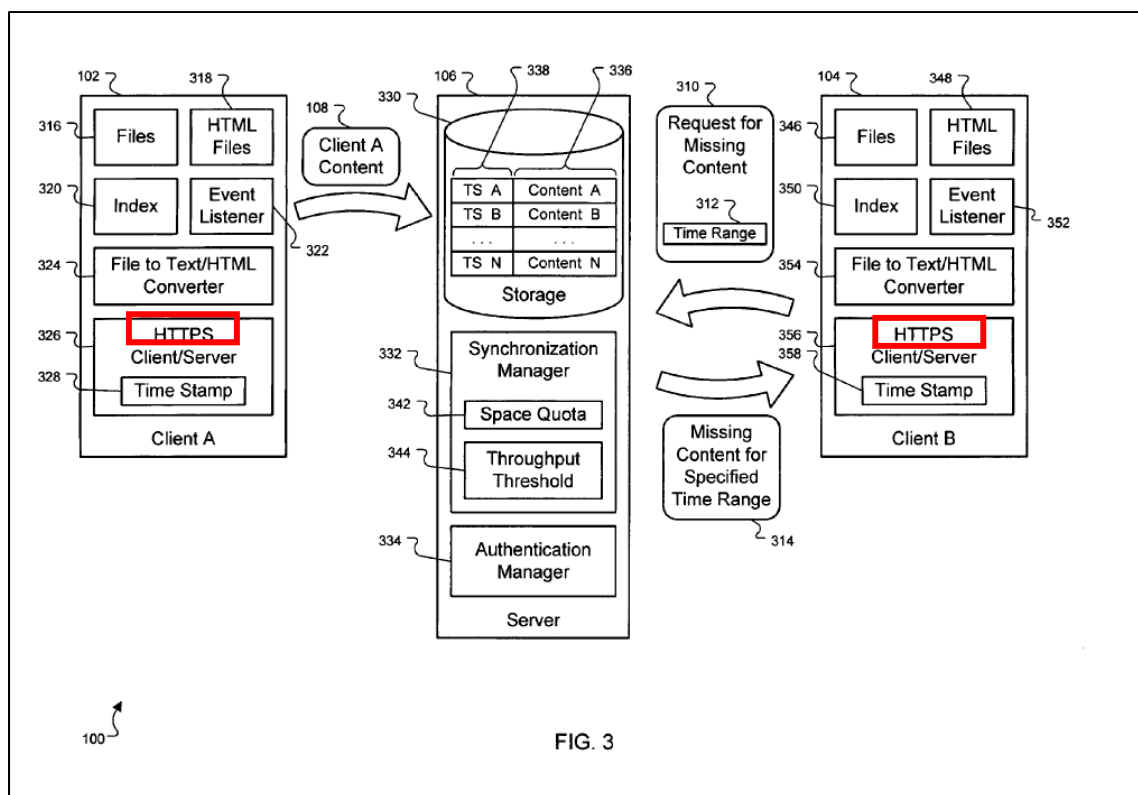
39. Sigurdsson discloses that the server 106 passes the list of content, and only if the server 106 receives a request for the content from the Client C 406 (step 620), the server will provide the content to the client (step 626). ¶¶ 79-80.

40. Sigurdsson does not teach or suggest that the server system the first electronic device “automatically transfer the modified first electronic file copy to the third electronic device”

41. A POSA would not have found it obvious to replace the request-response architecture taught by Sigurdsson with the claimed that automatic transfer of data without any client request.

42. Since Sigurdsson requires the server only transfers the missing content when requested by the second client device, the content is not automatically transferred by the first electronic device.

43. In Figure 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).



Sigurdsson (EX 1016), FIG. 3

44. HTTP (Hypertext Transfer Protocol) is a protocol used to carry requests from a browser to a Web server and to transport pages from Web servers back to the requesting browser. (Microsoft Computer Dictionary, p.259). Moreover, a HTTP server is server software that uses the HTTP protocol to serve up HTML documents and any associated files and scripts when requested by a client, such as a Web browser (Microsoft Computer Dictionary, p. 260). An HTTP/HTTPS server (e.g., the web server) plays the role of a server in a client–server model by implementing the server part of the HTTP/HTTPS network protocol. An important feature of an HTTP/HTTPS server is that the HTTP/HTTPS server waits for

incoming client requests and for each request the HTTP/HTTPS server responds by providing the requested information or performing a requested task. The HTTP/HTTPS server does not perform actions on its own without a request from a client. Since the HTTP protocol is a request-response protocol, where clients send HTTP requests to the server and the server sends HTTP responses after processing each request, the HTTP protocol does not support a functionality where the server can automatically send data to a client without the client making requests for data.

45. Figure 3 of Sigurdsson teaches a HTTPS server, which does not perform functions “automatically” without client requests for content. Thus, Sigurdsson requires a HTTPS server system based on an HTTP request/response process for data transfer between the client and the server as illustrated in FIG. 3. Therefore, Sigurdsson not only fails to satisfy the features of “**automatically transferring** the modified first electronic file copy to the third electronic”, it would not have been obvious to replace the principal structure (i.e., the server system based on an HTTP request/response process) taught by Sigurdsson with the automatically transfer of a copy of the file to a client without any client request to disclose the claim features. Specifically, a POSA would not have found it obvious to replace the request-response based HTTP server taught by Sigurdsson with the claimed features of “automatic transfer” of the copy of the file to a client without any client request for the missing data.

46. I have reviewed the secondary reference Shappell and in my opinion it discloses a peer-to-peer network environment which is fundamentally different from a Client-Server environment. Paragraph 0007 of Shappell. The Abstract of Shappell indicates that the disclosed system and methods disclosed are used to enable users to share files in a server-less shared space. By emphasizing a “server-less” environment, Shappell does not teach a Client-Server system that uses a server.

47. Shappell discloses “a computer implemented method and system enable users to share files in a **server-less** shared space. 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.” Abstract. Since Shappell *expressly* teaches away from using a server, Shappell cannot teach or suggest the first electronic device “automatically transfer the modified first electronic file copy to the third electronic device.”

**D. '561 Ground 2: The Petition Has Failed to Establish
Unpatentability for Challenged Claims Based on Brown and Hesselink**

48. I have reviewed the Brown reference and in my opinion, the process disclosed by Brown can be best understood within the context of a pull vs push architectural feature. In a “pull” based system, the initiating command for a file to

be copied from the server to the client arises from the client. On the other hand, in a “push” based system, the initiating command for a file to be copied from a server to the client arises from the server.

49. Brown discloses a “pull” based file transfer from the server to the client.

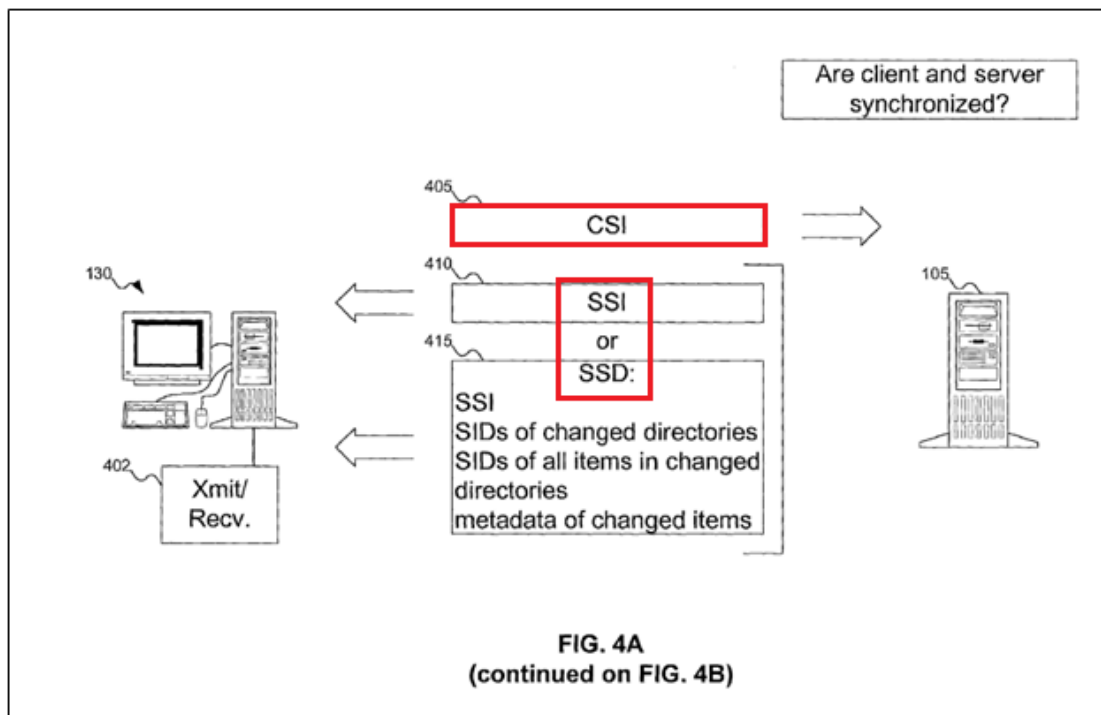
That is, the client or the user initiates a file synchronization that leads to transfer of the data/file from the server to the client. The Petition cites to Brown at 2:63-67, 3:1-8 and Balakrishnan, ¶¶896-898 and agrees with my analysis. The Petition argues “After execution of the functionality discussed 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.” Pet., 55. Consistent with this, Brown, 2:63-67, 3:1-8 discloses that “The client component is a client Synchronization Application (SA) that uses the server SFS to synchronize its local client data with the server. client machines synchronize with each other by synchronizing to a common server account.”

50. In other words, the client SA pulls changes from the server, and the modified file is not “pushed” or “automatically” transferred from the server to the client. Thus, Brown uses a protocol that uses a “pull” mechanism and not the “push” mechanism where the server automatically pushes 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).

51. Consistent with the above, at 5:14-24, Brown discloses that the synchronization process is initiated by the client SA when it makes a sync call to the server. Likewise, at 8:48-65 Brown discloses again that the client polls the server for changes.

52. In Brown, the client SA needs to poll the server to check if there are any changes to make to the client's directory and metadata to be synchronized to the server state. In detail, referring to FIG. 4A and 10:9-30 of Brown, *the client 130 first sends the client sync index (CSI) to server 105*, as shown in box 405, to see if server 105 and client 130 are synchronized. server 105 compares the received CSI with the server's current server sync index (SSI). If server 105 returns the SSI having the same value as the CSI to the client, as shown in box 410, client 130 knows that client 130 is synchronized with server 105. If the SSI received from server 105 does not match with the CSI of the client, the client's directory is not up to date and *the client then requests server synchronization data (SSD) to learn about the new changes that have been made at server 105.* ("If the CSI matches the server account's SSI value, then the client is up to date with the server. Otherwise the client SA requests server synchronization data (SSD)."; Brown 8:51-53).

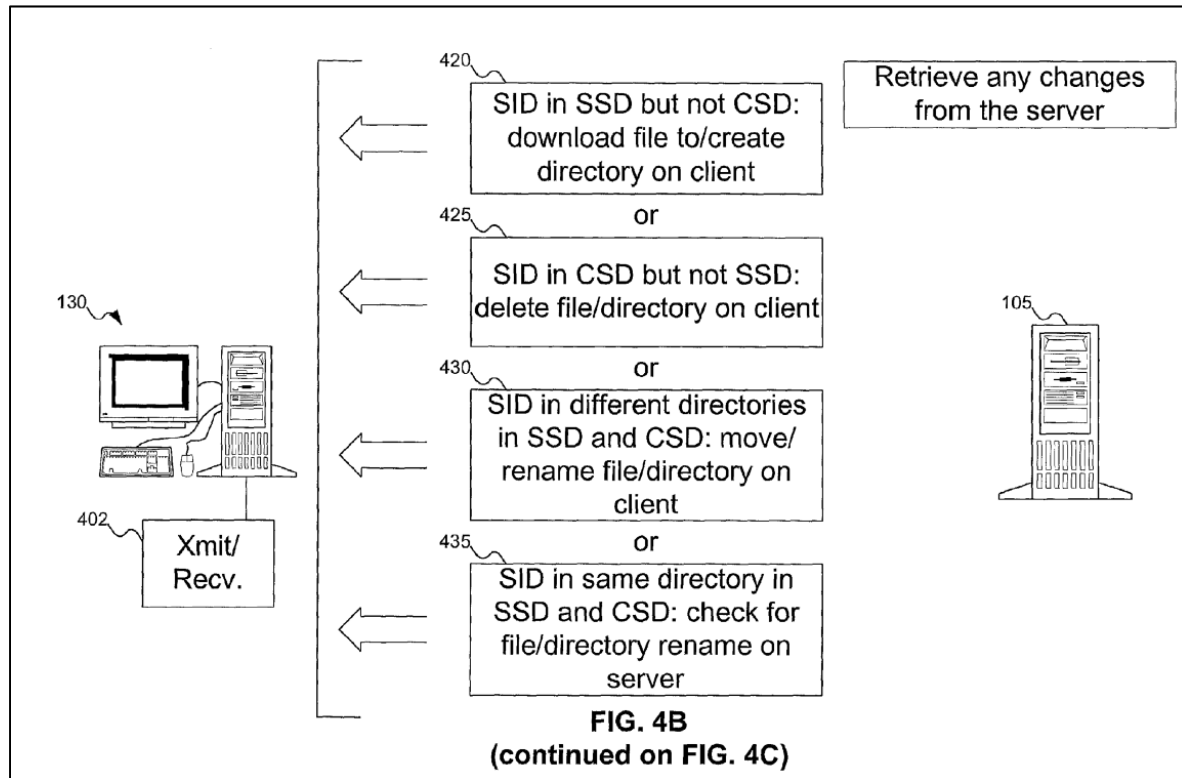


Brown (EX 1025), FIG. 4A

53. If server 105 has changes that client 130 lacks, server 105 then “sends the SSD to client 130 (*in response to a request for the SSD by the client*), informing the client of the pertinent changes, as shown in box 415.” Brown 10:19-21. Based on the SSD received from the server, “the *client SA updates the client's directory and metadata* to match the server state.” Brown 8:62-63.

54. Specifically, the client SA maintains client synchronization data (CSD) that it uses to track the state changes of the server, and by comparing the SSD with its CSD, client 130 can determine what changes have been made to the account on server 105. Brown 8:63-65. Referring to FIG. 4B, Brown illustrates 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 and requests the changes related to the file or directory from the server. Brown 10:45-48. That is, in Brown, the changes are retrieved from the server **upon the request received from the client**.



Brown (EX 1025), FIG. 4B

55. The Petition admits that Brown requires the client to access the server to find out if there are any changes to download (“[I]t would have been obvious that Brown’s synchronization process on laptop client 135 would **begin by checking** whether the server has any updates, including updates from other client devices, such as desktop client 130.”). Pet., 55. This agrees with my analysis that Brown

does not disclose “automatic” transmission of modified file from the server to the client. Further, Brown requires the client to download the changes upon determining that changes have been made to the server account. See e.g., Brown 10:53-54 (“[C]lient 130 has **downloaded all the pertinent changes** from server 105.”), 11:19-21 (“[C]lient 130 ... returning to box 415 on FIG. 4A to **download the changes** made on the server by the other client.”), 14:2-3 (“At step 840, the client synchronizes with the server to **download any changes** on the server.”). As discussed above, the client downloading the requested changes is pull type, not the server pushing data to the client.

56. Brown at 4:2-5 discloses that communication between the clients and server 105 is tunneled within the hypertext transport protocol (HTTP). That is, Brown teaches a HTTPS server, which uses a client-pull architecture and does not perform functions “automatically” without client requests. Thus, Brown requires a HTTPS server system based on an HTTP request/response process for synchronization. Therefore, Brown fails to disclose the features related to “**the first application ... automatically transfer** the modified first electronic file copy to the third electronic device.”

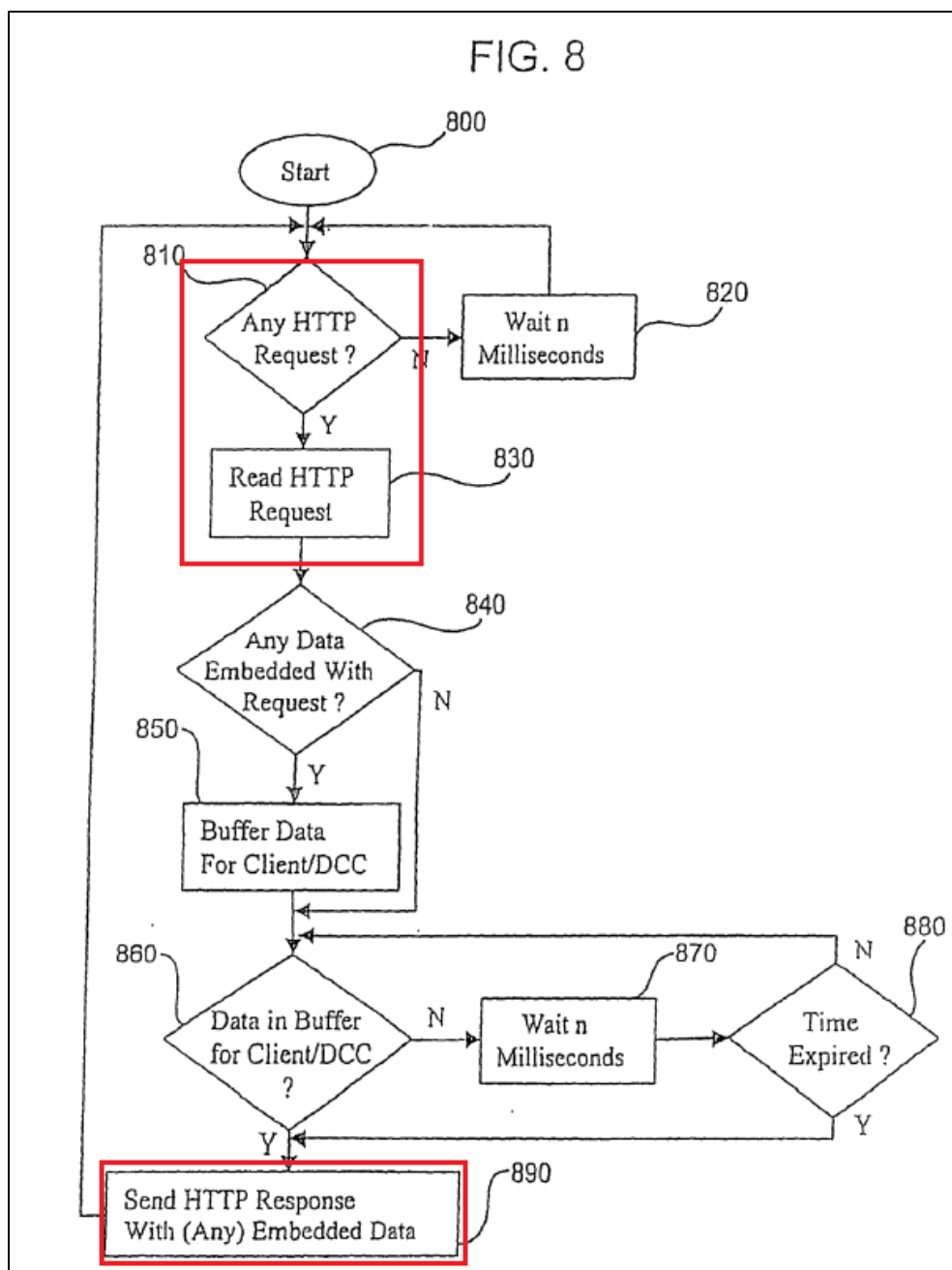
57. A POSA would not have found it obvious to replace the pull-based HTTP server taught by Brown with a push-based server that automatically pushes data to a client without any client request.

58. It should be noted that Brown intentionally and purposefully chose the “pull” based approach, and as such, a POSA would not have been motivated to modify Brown to apply to the “push” based file transfer because it would change the principle of operation of Brown.

59. 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.”

60. As noted above, 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. A POSA would not have been motivated to modify the teachings of Brown to use a server push mechanism.

61. I have reviewed Hesselink, and in my opinion, it describes synchronization based on an HTTP architecture which relies on the “pull” mechanism for pulling files/data from the server, just like Brown.
62. Fig. 8 of Hesselink illustrates such an HTTP request/response process for data transfer between the client and the connection server. At 810, it is determined “whether or not **any HTTP request has been received**.”
63. If an HTTP request has been received and there is no data is embedded within the HTTP request, in 860, it is determined if there is any Data present in the Buffer for the Client.
64. In 890, an HTTP Response is sent with the Data.



Hesslink (EX 1026), FIG. 8

65. Thus, Hesselink is 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.

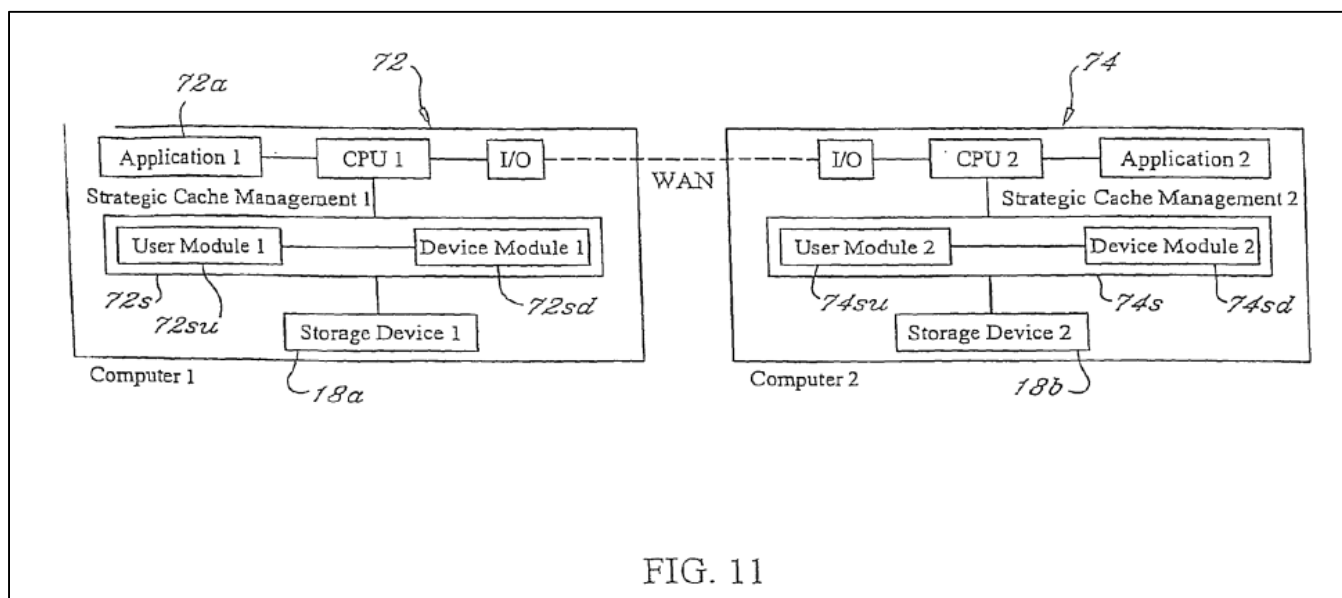
66. Hesselink is essentially related to file synchronization between multiple devices in a *peer-to-peer* environment where changes made in one or more remote storage devices are transmitted to the local device upon request from the local device. Hesselink, ¶27.

67. While Hesselink at FIG. 3 illustrates an embodiment in which connection servers 14a-14n are included in the peer-to-peer system, the purpose of the connection servers 14a-14n is only to provide authenticated and secure communications, via security server 58, between a plurality of client computers 12a-12 n and a plurality of device control computers 36a-36 d. Once a secure connection is established, data transfer is carried in the manner shown in FIGS. 7-8 between client computers 12a-12 n and device control computers 36a-36n. Specifically, 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, which merely serves as a pipeline or conduit in the proposed peer-to-peer system.

68. The Petition 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. Pet., 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-server architecture.

69. 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.



Hesselink (EX 1026), FIG. 11

70. 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.” Hesselink, ¶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. Hesselink, ¶¶137, 139.

71. Thus, 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 (Hesselink, ¶71), *upon receiving a request to access to its local storage from the second computer.*

72. While acknowledging the deficiency in Hesselink’s embodiment due to the peer-to-peer environment, the Petition still argues that a POSA would not have been dissuaded from combining Hesselink with Brown in view of Hesselink’s use of a peer-to-peer model as opposed to Brown’s client/server model. This is opposite of 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.”; Hesselink, ¶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. Hesselink, ¶10. Thus, Hesselink teaches away from implementing its method in a client/server model.

73. The Petition 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.” Pet., 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). Hesselink does not teach or suggest that the “file server” implements file synchronization between computers 72, 74, and 76 based on a client/server architecture.

74. Further, in my opinion, Client-Server architecture and Peer-to-Peer architecture are fundamentally different. Peer to Peer architectures provide a direct link between two devices and do not use a third device in between other than as a

pipeline or conduit. Therefore, a POSA would not take features from the peer-to-peer architecture to make changes to a client-server architecture.

75. A POSA would understand that Hesselink's file synchronization based on the peer-to-peer system cannot be modified with the technique based on the 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. A POSA would not have been motivated to combine the teachings of Brown and Hesselink because these two references teach fundamentally different architectures and are incompatible.

E. '942 Ground 1: The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Sigurdsson and Shappell

76. I have reviewed the Sigurdsson reference, and in my opinion, Sigurdsson reference teaches an HTTP server architecture for file synchronization which is fundamentally different from the teachings of the '942 patent.

77. Sigurdsson discloses a system, in which, a receiving device (i.e., a Client C 406) performs several operations before requesting a copy of missing content.

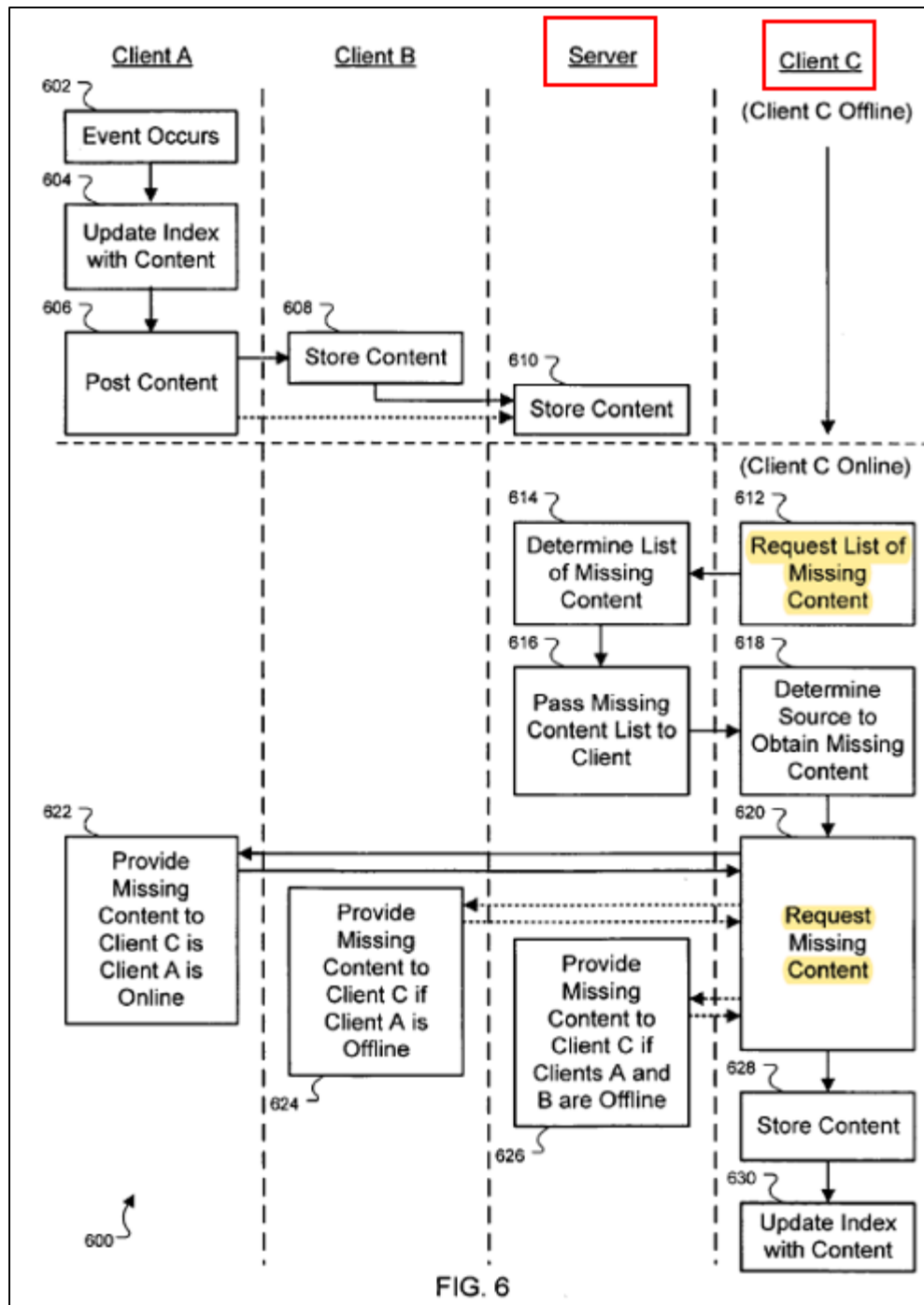
Much of Sigurdsson discusses client devices requesting such missing content from other client devices, in a peer-to-peer arrangement. See Abstract. Sigurdsson describes an embodiment in which Client C obtains missing content from a server 106. But that embodiment requires Client C to expressly request the missing content. Only in response to the server receiving the request from Client C does it transmit that content to Client C. (FIG. 4, 6, ¶¶71-75, 80). Therefore, in Sigurdsson, the server does not automatically transfer a modified first electronic file copy to the receiving device, such as Client C, since it only transfers the missing content when the receiving device expressly requests it.

78. The automatic send in the claims of the ‘942 Patent refers to the first electronic device automatically sends the modified first electronic file copy.

79. Not only does Sigurdsson fail to teach or suggest that a server “**automatically send[s]**, via the first application (at the first electronic device), the modified first electronic file copy to a third application at the third electronic device”, it also fails to disclose that such a transfer is “**responsive to receiving the modified first electronic file copy from the second electronic device.**”

Sigurdsson’s transfer to Client C is the result of Client C requesting the transfer.

80. Sigurdsson, in FIG. 6 (reproduced below with annotations), discloses a sequence diagram of a request/response based mixed peer-to-peer and client/server architecture, including a server, and clients A, B, and C. As shown in FIG. 6, the server in Sigurdsson provides missing content only in response to a request from a client (e.g., “Client C” at steps 612 and 620). That is, in FIG. 6, client/server structure and operation of Sigurdsson requires at least two separate requests from the client device (“Client C”) to the server, before the server provides the missing content to the client device, neither of which are “receiving the copy of the first file from the first client device”.



Sigurdsson (EX 1016), FIG. 6

81. It is only after going through this sequence of Client C expressly requests the server for any missing content, Client C determining where it can obtain that

missing content, and then requesting the content from the server, before the server responds by transferring the content to Client C. As such, Sigurdsson does not teach or suggest that the server 106 automatically transfers the content 410 to the Client C 406 responsive to receiving the content 410 from Client A 402.

82. Sigurdsson at FIG. 2 and ¶¶ 37, 40 and 46 discloses a request for missing content from the second client. For instance, “**the second client polls** for updates” (¶37), the request is “**periodically issue[d]**”, and “**the request 310 includes a Time Range 312 parameter** identifying the range of time stamps for the missing content” ¶40 and “the Request 310 is issued **in order** for the Client B 104 to obtain the Client A Content 108 that the Server 106 has yet to send to the Client B104” (¶46).

83. The disclosure in FIG. 5 simply summarizes the request-based file transfer approach from FIG. 6 discussed above in detail.

84. When viewed under proper context, Sigurdsson discloses that only if the server 106 receives a request for the content from the Client C 406 (step 620), will the server provide the content to the client (step 626). ¶¶ 79-80.

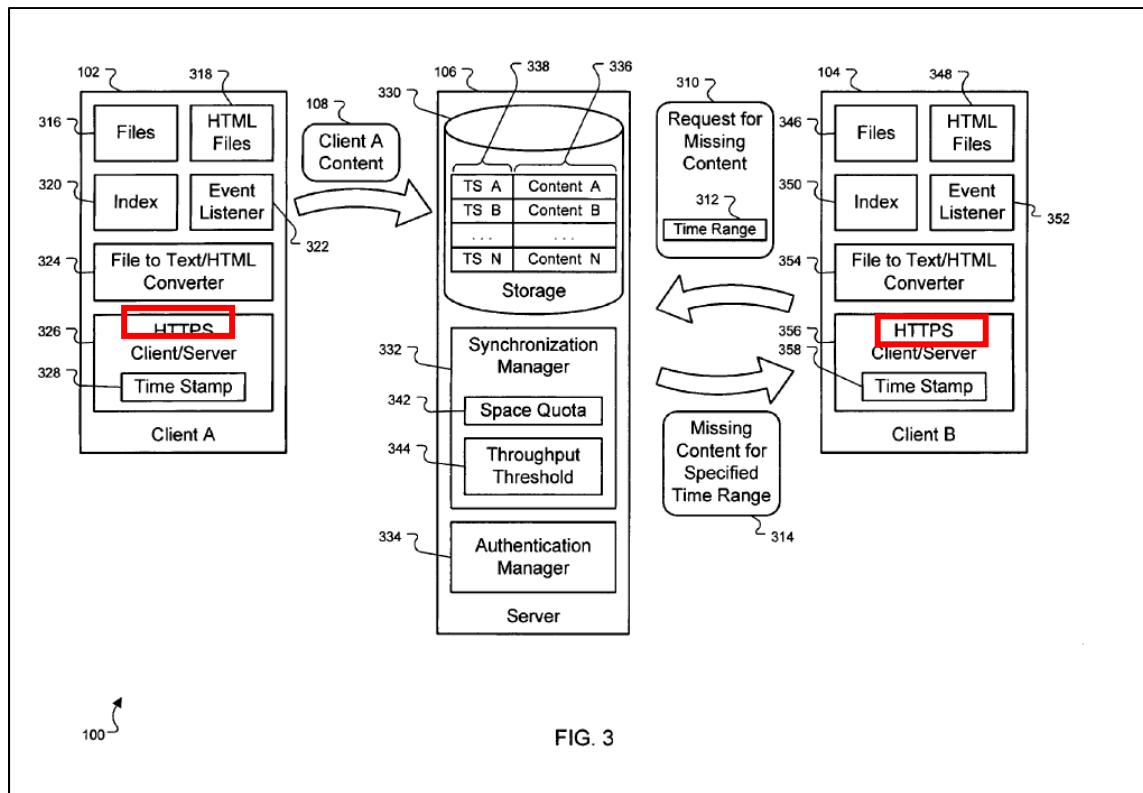
85. Sigurdsson discloses that the server 106 passes the list of content, and only if the server 106 receives a request for the content from the Client C 406 (step 620), the server will provide the content to the client (step 626). ¶¶ 79-80.

86. Sigurdsson does not teach or suggest that the server system “**automatically send**, via the first application (at the first electronic device), 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**”.

87. A POSA would not have found it obvious to replace the request-response architecture taught by Sigurdsson with the claimed that automatic transfer of data without any client request.

88. Since Sigurdsson requires the server only transfers the missing content when requested by the second client device, the transfer of the content to the second client device is responsive to a request from the second client device, and therefore, in Sigurdsson, the content is not automatically transferred by the server responsive to receiving the copy of the first file from the first client device.

89. In Figure 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).



Sigurdsson (EX 1016), FIG. 3

90. HTTP (Hypertext Transfer Protocol) is a protocol used to carry requests from a browser to a Web server and to transport pages from Web servers back to the requesting browser. (Microsoft Computer Dictionary, p.259). Moreover, a HTTP server is server software that uses the HTTP protocol to serve up HTML documents and any associated files and scripts when requested by a client, such as a Web browser (Microsoft Computer Dictionary, p. 260). An HTTP/HTTPS server (e.g., the web server) plays the role of a server in a client–server model by implementing the server part of the HTTP/HTTPS network protocol. An important feature of an HTTP/HTTPS server is that the HTTP/HTTPS server waits for

incoming client requests and for each request the HTTP/HTTPS server responds by providing the requested information or performing a requested task. The HTTP/HTTPS server does not perform actions on its own without a request from a client. Since the HTTP protocol is a request-response protocol, where clients send HTTP requests to the server and the server sends HTTP responses after processing each request, the HTTP protocol does not support a functionality where the server can automatically send data to a client without the client making requests for data.

91. Figure 3 of Sigurdsson teaches a HTTPS server, which does not perform functions “automatically” without client requests for content. Thus, Sigurdsson requires a HTTPS server system based on an HTTP request/response process for data transfer between the client and the server as illustrated in FIG. 3. Therefore, Sigurdsson not only fails to satisfy the features of “**automatically send[ing]** the modified first electronic file copy to the third electronic”, it would not have been obvious to replace the principal structure (i.e., the server system based on an HTTP request/response process) taught by Sigurdsson with the automatic transfer of a copy of the file to a client without any client request to disclose the claim features. Specifically, a POSA would not have found it obvious to replace the request-response based HTTP server taught by Sigurdsson with the claimed features of “automatic transfer” of the copy of the file to a client without any client request for the missing data.

92. I have reviewed the secondary reference Shappell and in my opinion it discloses a peer-to-peer network environment which is fundamentally different from a Client-Server environment. Paragraph 0007 of Shappell. The Abstract of Shappell indicates that the disclosed system and methods disclosed are used to enable users to share files in a server-less shared space. By emphasizing a “server-less” environment, Shappell does not teach a Client-Server system that uses a server.

93. Shappell discloses “a computer implemented method and system enable users to share files in a **server-less** shared space. 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.” Abstract. Since Shappell *expressly* teaches serverless systems, it does not teach or suggest a first electronic device (i.e., a server) configured to **automatically send**, via the first application (at the first electronic device), 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.**

F. '942 Ground 2: The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Brown and Hesselink

94. The automatic process described in Brown is from the user's perspective, meaning that the process may need little or no involvement of human operation from the user. In contrast, the automatic transfer in the claims of the '942 Patent refers to the automatic aspect from the server's point of view (that is, the first electronic device executing the first application), in view of the recitation "**a first electronic device ... automatically send, via the first application, the modified first electronic file copy to a third application at the third electronic device ...**" of claim 1, and similarly claim 10, of the '942 Patent. That is, the claims of the '942 Patent indicate that the server is to send the modified file to the client without a need for the client to explicitly request it.

95. Brown's disclosure is unambiguously directed to the opposite manner, that is, the file synchronization must be initiated by the client request to the server. Brown's approach therefore cannot satisfy the automatic transfer requirement of the claims of the '942 Patent – *i.e.*, the server transfers the file on its own without a request from a client.

96. That is, the claims of the '942 Patent adopt a ***push***-based file transfer from the server to the client, without the need for the client to explicitly request it. Brown's disclosure is directed to the opposite method, that is, client ***pull***. In Brown, the client SA or the user must initiate synchronization. Absent such

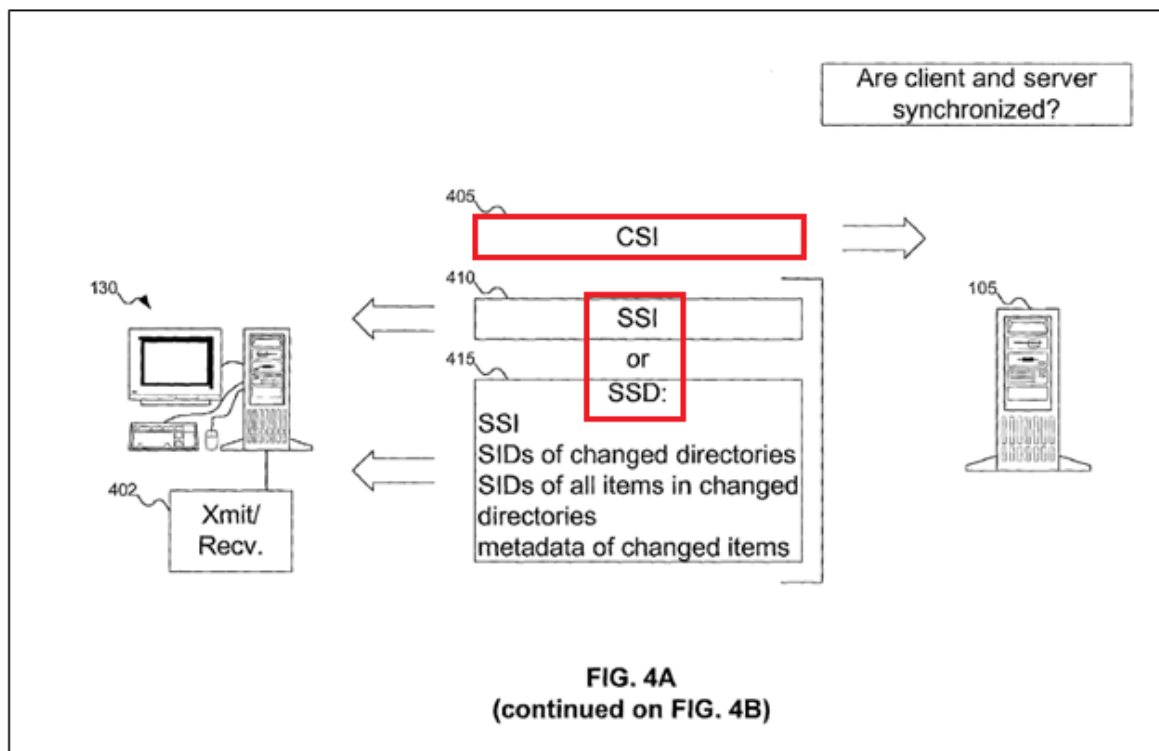
request, the server would not initiate the transfer of the version of the file generated from the user modifying the content of the file.

97. In Brown, the client SA needs to poll the server to check if there are any changes to make to the client's directory and metadata to be synchronized to the server state. Brown, 5:14-27 ("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.") In detail, referring to FIG. 4A and 10:9-30 of Brown, *the client 130 first sends the client sync index (CSI) to server 105*, as shown in box 405, to see if server 105 and client 130 are synchronized. Server 105 compares the received CSI with the server's current server sync index (SSI). If server 105 returns the SSI having the same value as the CSI to the client, as shown in box 410, client 130 knows that client 130 is synchronized with server 105. If the SSI received from server 105 does not match with the CSI of the client, the client's directory is not up to date and *the client then requests server synchronization data (SSD)* to learn about the new changes that have been made at server 105. ("If the CSI matches the server account's SSI value, then the client is up to date with the server. Otherwise **the client SA requests server synchronization data (SSD)**."; Brown, 8:51-53).

98. In Brown, to synchronize with the server, the client must pull changes from the server, as opposed to the modified file being "automatically" transferred from

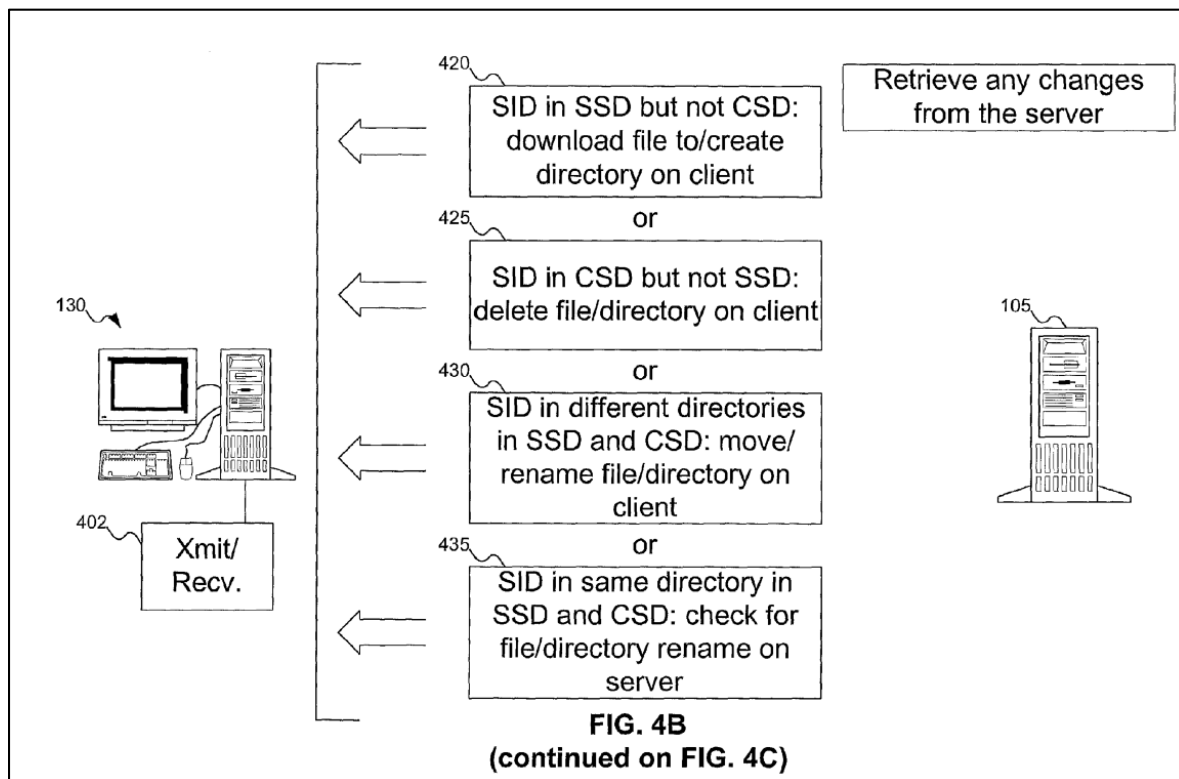
the server to the client. Clearly, Brown uses a protocol that uses a “pull” mechanism and not the server automatically pushing the modified file to the client. In the “pull” mechanism, the client does not know when new data is available and the client must poll the server to see if there is new data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

99. If server 105 has changes that client 130 lacks, server 105 then “sends the SSD to client 130 (*in response to a request for the SSD by the client*), informing the client of the pertinent changes, as shown in box 415.” Brown, 10:19-21. Based on the SSD received from the server, “the *client SA updates the client's directory and metadata* to match the server state.” Brown, 8:62-63.



Brown (EX 1025), FIG. 4A

100. Specifically, the client SA maintains client synchronization data (CSD) that it uses to track the state changes of the server, and by comparing the SSD with its CSD, client 130 can determine what changes have been made to the account on server 105. Brown, 8:63-65. Referring to FIG. 4B, Brown illustrates 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 and requests the changes related to the file or directory from the server. Brown, 10:45-48. That is, in Brown, the changes are retrieved from the server **upon the request received from the client.**



Brown (EX 1025), FIG. 4B

101. Brown does not disclose “automatic” transmission of modified file from the server to the client. Further, Brown requires the client to download the changes upon determining that changes have been made to the server account. See e.g., Brown, 10:53-54 (“[C]lient 130 has **downloaded all the pertinent changes** from server 105.”), 11:19-21 (“[C]lient 130 ... returning to box 415 on FIG. 4A to **download the changes** made on the server by the other client.”), 14:2-3 (“At step 840, the client synchronizes with the server to **download any changes** on the server.”). As discussed above, the client downloading the requested changes is pull type, not the server pushing data to the client.

102. Brown at 4:2-5 discloses that communication between the clients and server 105 is tunneled within the hypertext transport protocol (HTTP).

103. The HTTP protocol is a request-response protocol, the HTTP servers use the client-pull architecture, where clients send HTTP requests to the server and the server sends HTTP responses after processing each request. The HTTP protocol does not support push-based functionality where the server can automatically send data to a client without the client making requests for data.

104. Brown teaches a HTTPS server, which uses a client-pull architecture and does not perform functions “automatically” without client requests. Thus, Brown requires a HTTPS server system based on an HTTP request/response process for

synchronization. Therefore, Brown fails to disclose the features related to “**a first electronic device ... automatically send**, via the first application, **the modified first electronic file copy** to a third application **at the third electronic device.**”

Furthermore, it would not have been obvious to replace the principal structure (i.e., “the pull-based HTTP server”) of Brown with a push-based server that automatically pushes data to a client without any client request. Specifically, a POSA would not have found it obvious to replace the pull-based HTTP server taught by Brown with a push-based server that automatically pushes data to a client without any client request. Neither Petition provides any rationale for making such replacement. In fact, a fundamental feature of an HTTP architecture, namely, providing a service in response to a request from a client teaches away from an “automatic” provision of a service without any request.

105. 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.”

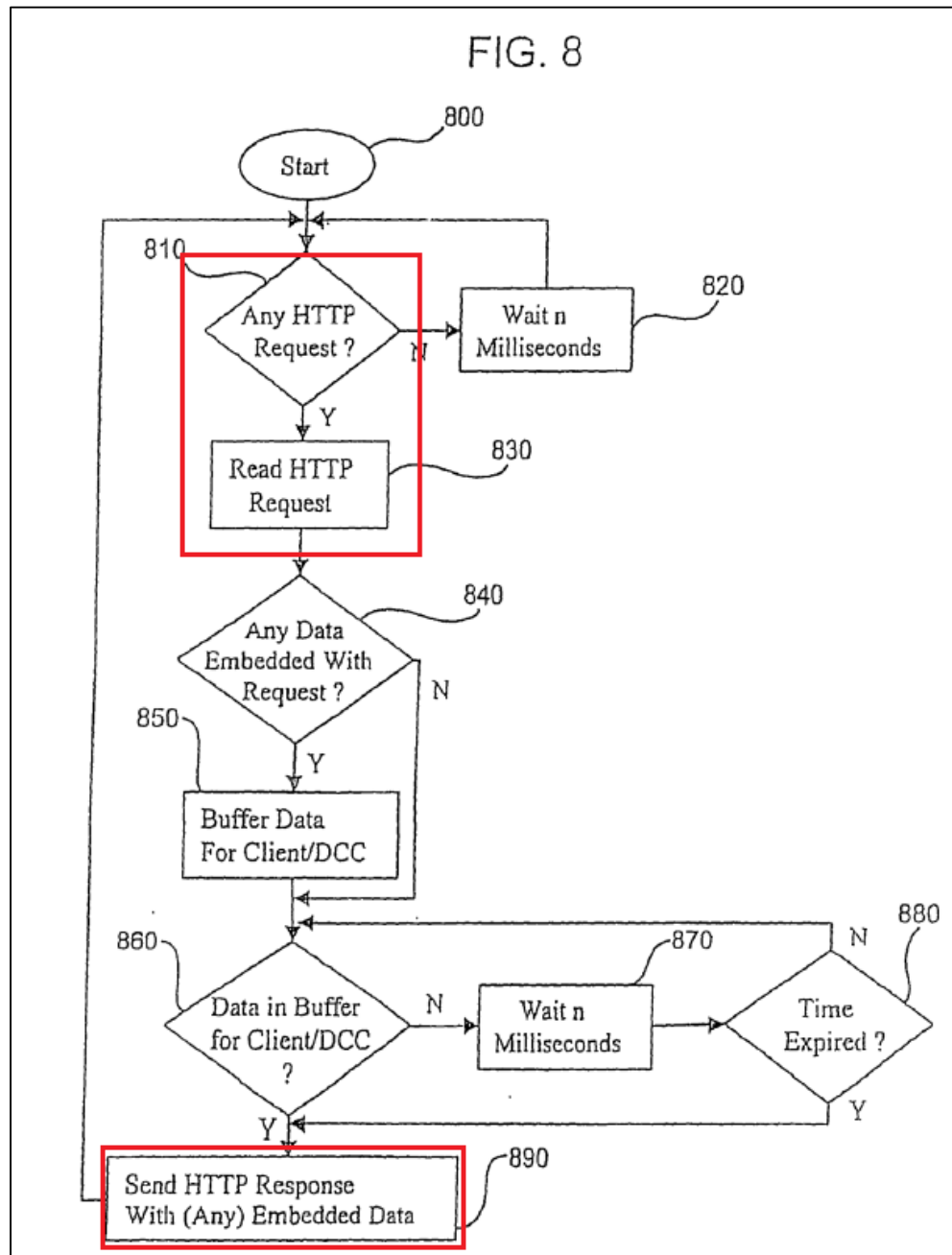
106. As noted above, Brown aims to minimize the load on the server processor, and to that end, as many synchronization operations as possible are performed by the client, including the operation to make a sync poll call to the server.

107. A POSA would understand that Brown cannot be modified to disclose push-based file transfer because it would change the principle of operation.

108. 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.

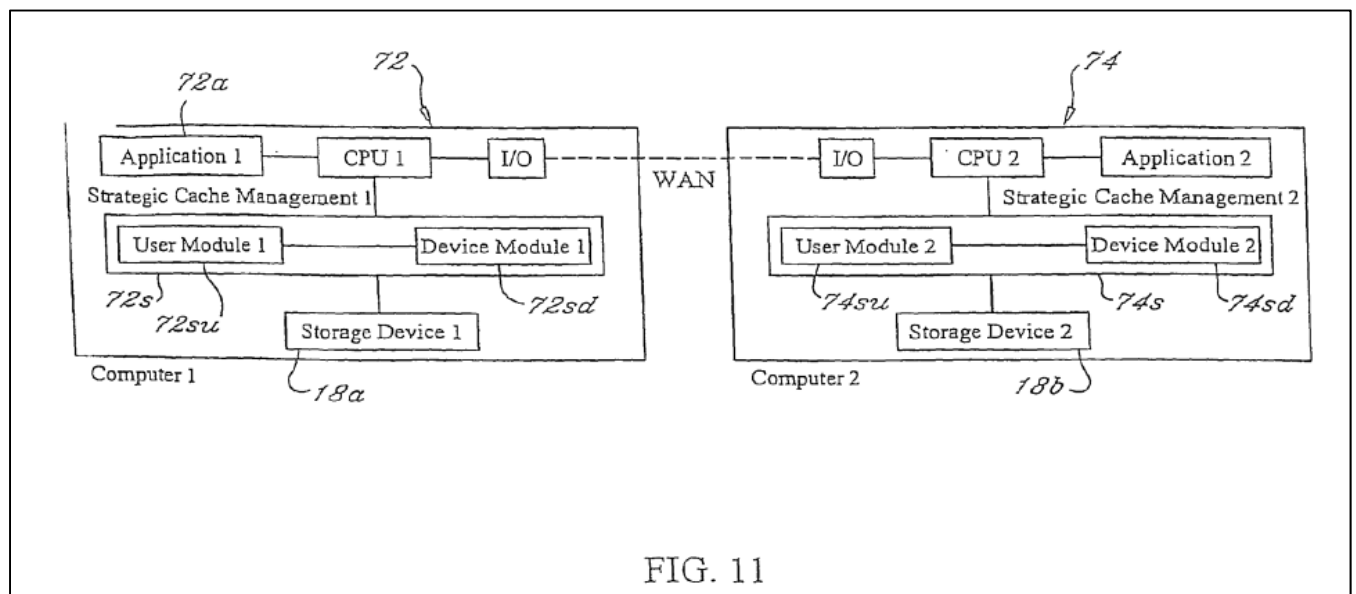
109. The HTTP request/response process for data transfer between the client and the connection server is illustrated in FIG. 8 of Hesselink. The process includes event 810 in which it is determined “whether or not **any HTTP request has been received** from client computer 12 a-12 n (or device control 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 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.



Hesselink (EX 1026), FIG. 8

110. The Petition 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. Pet., 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. Specifically, 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.



Hesselink (EX 1026), FIG. 11

111. 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.” Hesselink, ¶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. Hesselink, ¶¶137, 139. Clearly, 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 (Hesselink, ¶71), upon receiving a request to access to its local storage from the second computer.

112. While acknowledging the deficiency in Hesselink’s embodiment due to the peer-to-peer environment, the Petition still argues that a POSA would not have been dissuaded from combining Hesselink with Brown in view of Hesselink’s use of a peer-to-peer model as opposed to Brown’s client/server model. This is opposite of 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.”; Hesselink, ¶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. Hesselink, ¶10. Thus, Hesselink teaches away from implementing its method in a client/server model.

113. While Hesselink mentions about 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). The mere mention of a “file server” that may implement any of computers 72, 74 and 76 in FIG. 11 of Hesselink, ¶122 by no means implicates, let alone render obvious, Hesselink’s applicability to a client/server model.

114. It would have been apparent to a POSA that Hesselink’s file synchronization based on the peer-to-peer system cannot be modified with the technique based on the 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. A POSA would have been prevented from deriving the claimed invention from Brown and Hesselink because these two references teach fundamentally different architectures and are incompatible.

G. '607 Ground 1: The Petition Has Failed to Establish Unpatentability for *Challenged Claims* Based on Sigurdsson, Shappell, and Rao

115. Sigurdsson discloses a system, in which, a receiving device (i.e., a Client C 406) performs several operations before requesting a copy of missing content.

Much of Sigurdsson discusses client devices requesting such missing content from other client devices, in a peer-to-peer arrangement. See Abstract. Sigurdsson describes an embodiment in which Client C obtains missing content from a server

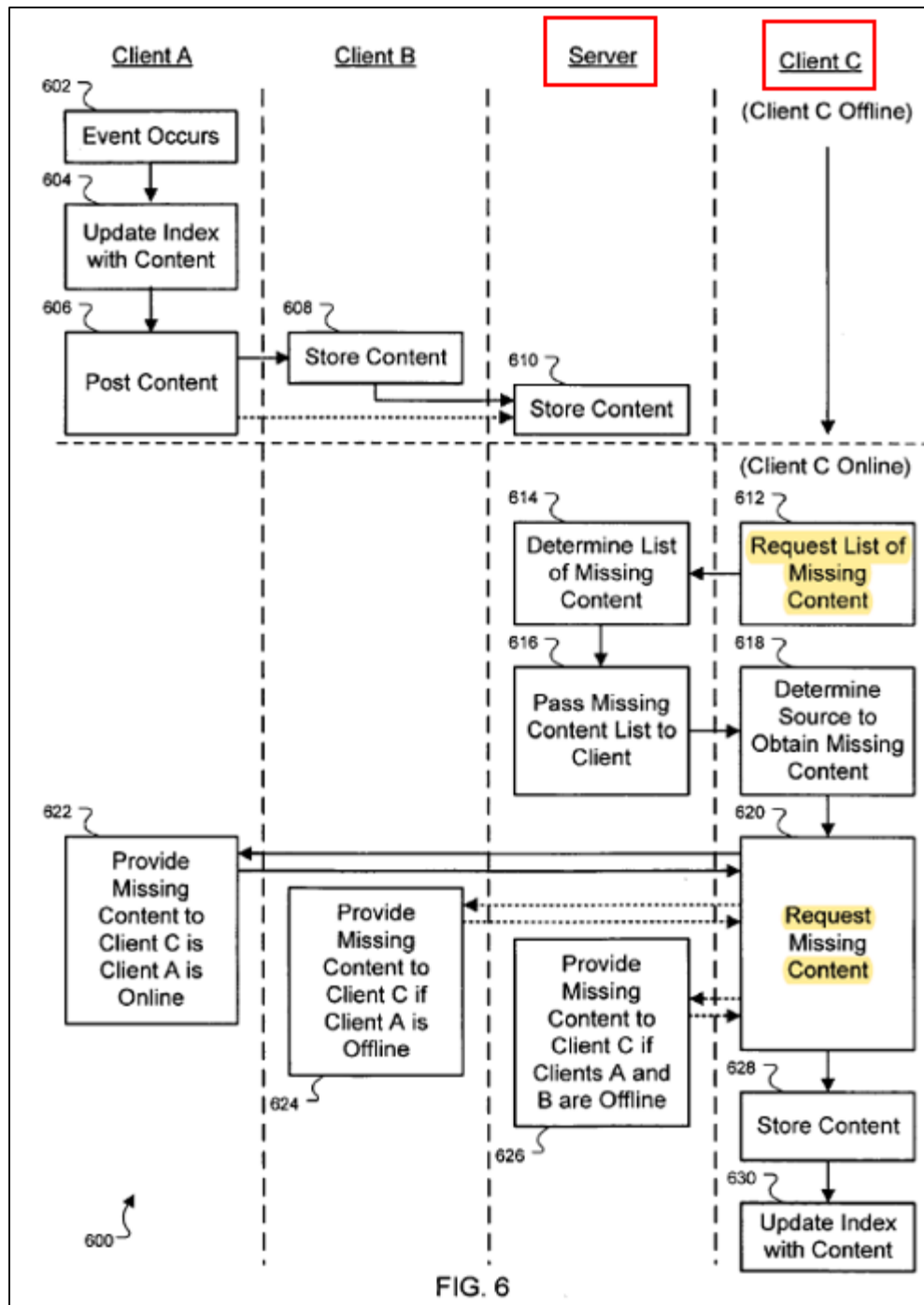
106. But that embodiment requires Client C to expressly request the missing content. Only in response to the server receiving the request from Client C does it transmit that content to Client C. (FIG. 4, 6, ¶¶71-75, 80). Therefore, in

Sigurdsson, the server does not automatically transfer a modified first electronic file copy to the receiving device, such as Client C, since it only transfers the missing content when the receiving device expressly requests it.

116. The automatic transfer in the claims of the ‘607 Patent refers to the server system automatically transferring the copy of the first file.

117. Not only does Sigurdsson fail to teach or suggest that a server “**automatically transfer[s]**, over a network, the copy of the first file to the second client device”, it also fails to disclose that such a transfer is “**responsive to ... receiving the copy of the first file from the first client device.**” Sigurdsson’s transfer to Client C is the result of Client C requesting the transfer.

118. Sigurdsson, in FIG. 6 (reproduced below with annotations), discloses a sequence diagram of a request/response based mixed peer-to-peer and client/server architecture, including a server, and clients A, B, and C. As shown in FIG. 6, the server in Sigurdsson provides missing content only in response to a request from a client (e.g., “Client C” at steps 612 and 620). That is, in FIG. 6, client/server structure and operation of Sigurdsson requires at least two separate requests from the client device (“Client C”) to the server, before the server provides the missing content to the client device, neither of which are “receiving the copy of the first file from the first client device”.



Sigurdsson (EX 1016), FIG. 6

119. It is only after going through this sequence of Client C expressly requests the server for any missing content, Client C determining where it can obtain that

missing content, and then requesting the content from the server, before the server responds by transferring the content to Client C. As such, Sigurdsson does not teach or suggest that the server 106 automatically transfers the content 410 to the Client C 406 responsive to receiving the content 410 from Client A 402.

120. Sigurdsson at FIG. 2 and ¶¶ 37, 40 and 46 discloses a request for missing content from the second client. For instance, “**the second client polls** for updates” (¶37), the request is “**periodically issue[d]**”, and “**the request 310 includes a Time Range 312 parameter** identifying the range of time stamps for the missing content” ¶40 and “the Request 310 is issued **in order** for the Client B 104 to obtain the Client A Content 108 that the Server 106 has yet to send to the Client B104” (¶46).

121. FIG. 5 and the related text does not disclose several of the claimed features including the “**automatically transfer**, over a network, the copy of the first file to the second client device associated with the user to replace an older version of the first file stored on the second client device, **responsive to ... receiving the copy of the first file from the first client device**” as discussed above. In fact, the disclosure in FIG. 5 simply summarizes the request-based file transfer approach from FIG. 6 discussed above in detail.

122. When viewed under proper context, Sigurdsson discloses that only if the server 106 receives a request for the content from the Client C 406 (step 620), will the server provide the content to the client (step 626). ¶¶ 79-80.

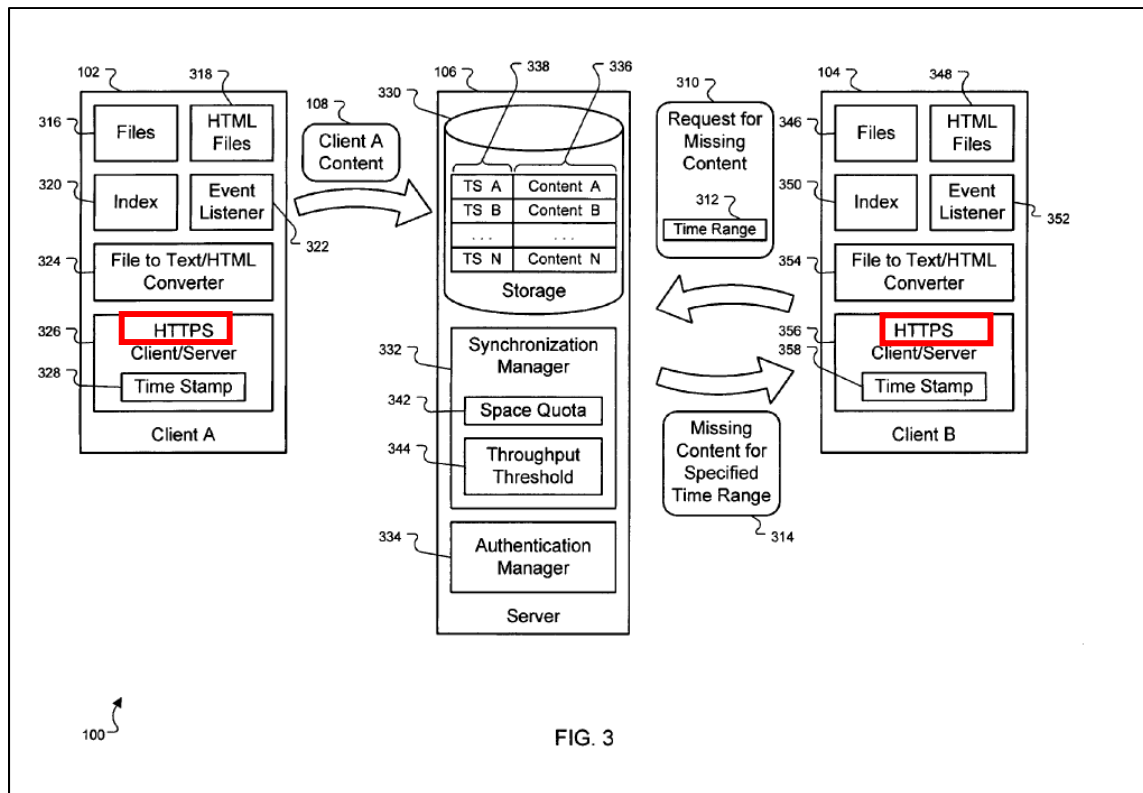
123. Sigurdsson discloses that the server 106 passes the list of content, and only if the server 106 receives a request for the content from the Client C 406 (step 620), the server will provide the content to the client (step 626). ¶¶ 79-80.

124. Sigurdsson does not teach or suggest that the server system “automatically transfer[s], over a network, the copy of the first file to the second client device ... responsive to receiving the copy of the first file from the first client device”.

125. A POSA would not have found it obvious to replace the request-response architecture taught by Sigurdsson with the claimed that automatic transfer of data without any client request.

126. Since Sigurdsson requires the server only transfers the missing content when requested by the second client device, the transfer of the content to the second client device is responsive to a request from the second client device, and therefore, in Sigurdsson, the content is not automatically transferred by the server responsive to receiving the copy of the first file from the first client device.

127. In Figure 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).



Sigurdsson (EX 1016), FIG. 3

128. HTTP (Hypertext Transfer Protocol) is a protocol used to carry requests from a browser to a Web server and to transport pages from Web servers back to the requesting browser. (Microsoft Computer Dictionary, p.259). Moreover, a HTTP server is server software that uses the HTTP protocol to serve up HTML documents and any associated files and scripts when requested by a client, such as a Web browser (Microsoft Computer Dictionary, p. 260). An HTTP/HTTPS server (e.g., the web server) plays the role of a server in a client–server model by implementing the server part of the HTTP/HTTPS network protocol. An important feature of an HTTP/HTTPS server is that the HTTP/HTTPS server waits for

incoming client requests and for each request the HTTP/HTTPS server responds by providing the requested information or performing a requested task. The HTTP/HTTPS server does not perform actions on its own without a request from a client. Since the HTTP protocol is a request-response protocol, where clients send HTTP requests to the server and the server sends HTTP responses after processing each request, the HTTP protocol does not support a functionality where the server can automatically send data to a client without the client making requests for data.

129. Figure 3 of Sigurdsson teaches a HTTPS server, which does not perform functions “automatically” without client requests for content. Thus, Sigurdsson requires a HTTPS server system based on an HTTP request/response process for data transfer between the client and the server as illustrated in FIG. 3. Therefore, Sigurdsson not only fails to satisfy the features of “**automatically transferring** the modified first electronic file copy to the third electronic”, it would not have been obvious to replace the principal structure (i.e., the server system based on an HTTP request/response process) taught by Sigurdsson with the automatically transfer of a copy of the file to a client without any client request to disclose the claim features. Specifically, a POSA would not have found it obvious to replace the request-response based HTTP server taught by Sigurdsson with the claimed features of “automatic transfer” of the copy of the file to a client without any client request for the missing data.

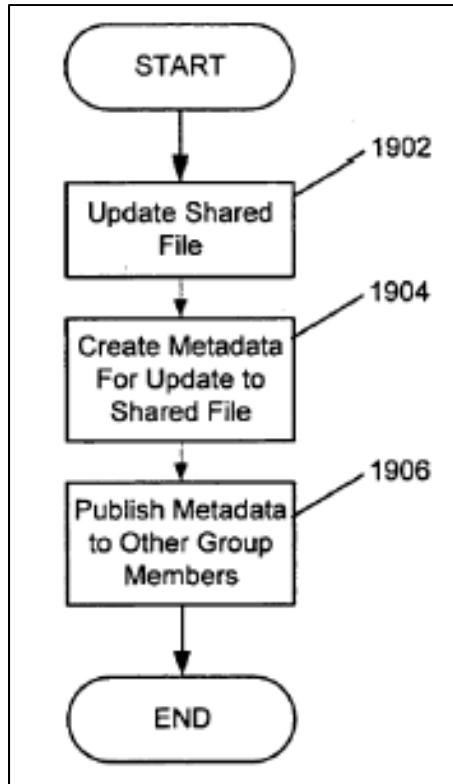
130. Sigurdsson does not teach or suggest that “**the first metadata being assigned a first priority greater than a second priority assigned to the copy of the first file**”, because the “priority algorithm” in Sigurdsson is for prioritizing between the content that is to be synchronized and not for prioritizing between metadata and the content. For example, in ¶58, Sigurdsson discloses three data items include 1. very large document such as a maintenance manual, 2. data from digital camera, and 3. a small office memo. Moreover, Sigurdsson discloses that, although the server 106 can receive content for all three data items in the order that the user accessed them, the server 106 can also deal with the data items by assigning it priorities. As such, the server 106 can handle the higher priority data items first, and save the lower priority data items for later.

131. I have reviewed the secondary reference Shappell and in my opinion it discloses a peer-to-peer network environment which is fundamentally different from a Client-Server environment. Paragraph 0007 of Shappell. The Abstract of Shappell indicates that the disclosed system and methods disclosed are used to enable users to share files in a server-less shared space. By emphasizing a “server-less” environment, Shappell does not teach a Client-Server system that uses a server.

132. Shappell discloses “a computer implemented method and system enable users to share files in a **server-less** shared space. 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.” Abstract. Since Shappell *expressly* teaches away from using a server, Shappell cannot teach or suggest a server system **automatically transferring** the copy of the first file to the second client device associated with the user to replace an older version of the first file stored on the second client device, **responsive to ... receiving the copy of the first file from the first client device.**

133. Shappell is silent about a priority being assigned to the first metadata or the copy of the first file. In particular, in cited FIG. 19 (reproduced below), Shappell at most discloses “creating metadata for update to shared file” and “publishing metadata to other group members.” However, there is no mention of a priority being assigned to the metadata. No other portion in Shappell discloses a priority being assigned to the metadata. Therefore, Shappell cannot and does not teach or suggest “the first metadata being assigned a first priority greater than a second priority assigned to the copy of the first file”.



Shappell (EX 1020), FIG. 19

134. In FIG. 19, step 1906, Shappell discloses the metadata to all other group members, which is different from the claimed features of automatically transferring the first metadata **to the second client device associated with the same user.**

H. '607 Ground 2: The Petition Has Failed to Establish Unpatentability for Challenged Claims Based on Brown, Hesselink, Shappell, and Rao

135. I have reviewed the Brown reference and in my opinion, the process disclosed by Brown can be best understood within the context of a pull vs push architectural feature. In a “pull” based system, the initiating command for a file to be copied from the server to the client arises from the client. On the other hand, in

a “push” based system, the initiating command for a file to be copied from a server to the client arises from the server.

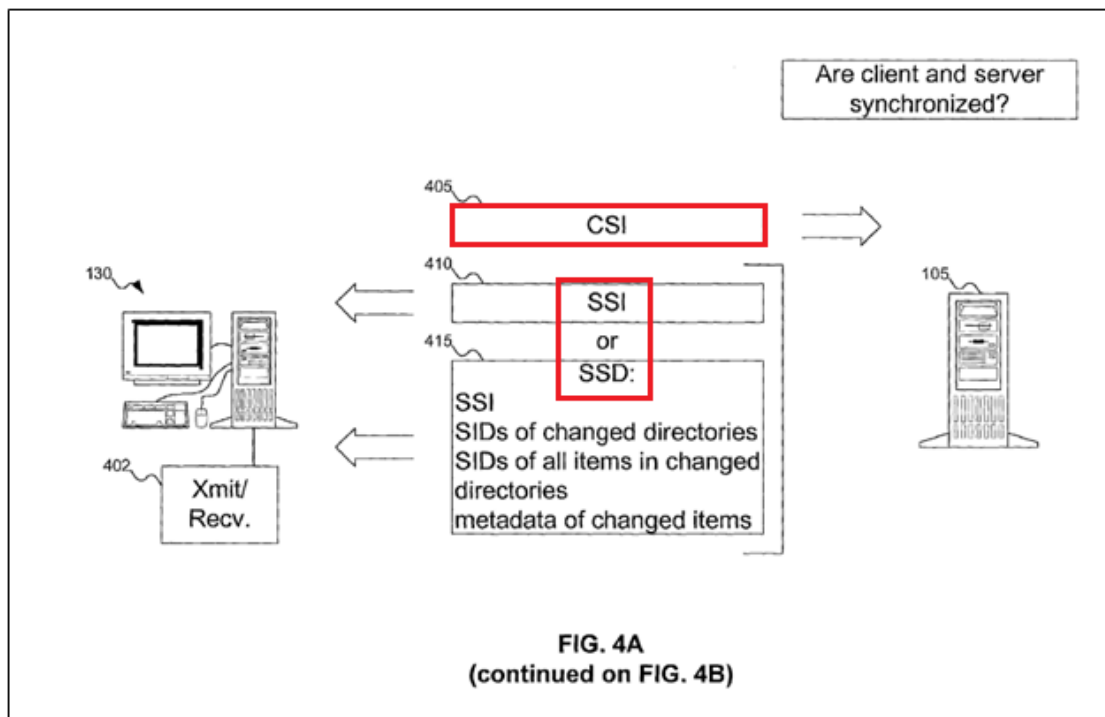
136. Brown discloses a “pull” based file transfer from the server to the client.

That is, the client or the user initiates a file synchronization that leads to transfer of the data/file from the server to the client. The Petition cites to Brown at 2:63-67, 3:1-8 and Balakrishnan, ¶¶896-898 and agrees with my analysis. The Petition 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.” Pet., 55. Consistent with this, Brown, 2:63-67, 3:1-8 discloses that “The client component is a client Synchronization Application (SA) that uses the server SFS to synchronize its local client data with the server. client machines synchronize with each other by synchronizing to a common server account.”

137. In other words, the client SA pulls changes from the server, and the modified file is not “pushed” or “automatically” transferred from the server to the client. Clearly, Brown uses a protocol that uses a “pull” mechanism and not the “push” mechanism where the server automatically pushes 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).

138. Consistent with the above, at 5:14-24, Brown discloses that the synchronization process is initiated by the client SA when it makes a sync call to the server. Likewise, at 8:48-65 Brown discloses again that the client polls the server for changes.

139. In Brown, the client SA needs to poll the server to check if there are any changes to make to the client's directory and metadata to be synchronized to the server state. In detail, referring to FIG. 4A and 10:9-30 of Brown, *the client 130 first sends the client sync index (CSI) to server 105*, as shown in box 405, to see if server 105 and client 130 are synchronized. Server 105 compares the received CSI with the server's current server sync index (SSI). If server 105 returns the SSI having the same value as the CSI to the client, as shown in box 410, client 130 knows that client 130 is synchronized with server 105. If the SSI received from server 105 does not match with the CSI of the client, the client's directory is not up to date and *the client then requests server synchronization data (SSD) to learn about the new changes that have been made at server 105.* ("If the CSI matches the server account's SSI value, then the client is up to date with the server. Otherwise the client SA requests server synchronization data (SSD)."; Brown 8:51-53).

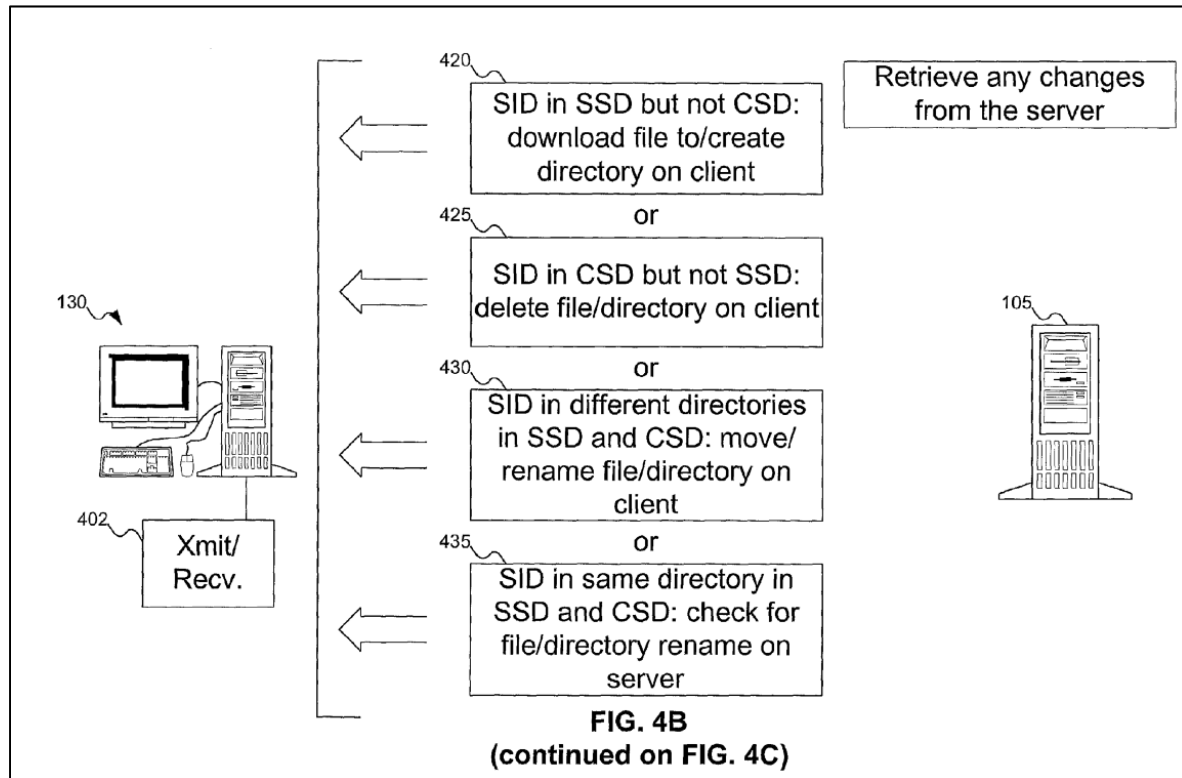


Brown (EX 1025), FIG. 4A

140. If server 105 has changes that client 130 lacks, server 105 then “sends the SSD to client 130 (*in response to a request for the SSD by the client*), informing the client of the pertinent changes, as shown in box 415.” Brown 10:19-21. Based on the SSD received from the server, “the *client SA updates the client's directory and metadata* to match the server state.” Brown 8:62-63.

141. Specifically, the client SA maintains client synchronization data (CSD) that it uses to track the state changes of the server, and by comparing the SSD with its CSD, client 130 can determine what changes have been made to the account on server 105. Brown 8:63-65. Referring to FIG. 4B, Brown illustrates 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 and requests the changes related to the file or directory from the server. Brown 10:45-48. That is, in Brown, the changes are retrieved from the server **upon the request received from the client**.



Brown (EX 1025), FIG. 4B

142. The Petition admits that Brown requires the client to access the server to find out if there are any changes to download (“[I]t would have been obvious that Brown’s synchronization process on laptop client 135 would **begin by checking** whether the server has any updates, including updates from other client devices, such as desktop client 130.”). Pet., 55. Thus, Brown does not disclose

“automatic” transmission of modified file from the server to the client. Further, Brown requires the client to download the changes upon determining that changes have been made to the server account. See e.g., Brown 10:53-54 (“[C]lient 130 has **downloaded all the pertinent changes** from server 105.”), 11:19-21 (“[C]lient 130 ... returning to box 415 on FIG. 4A to **download the changes** made on the server by the other client.”), 14:2-3 (“At step 840, the client synchronizes with the server to **download any changes** on the server.”). As discussed above, the client downloading the requested changes is pull type, not the server pushing data to the client.

143. Brown at 4:2-5 discloses that communication between the clients and server 105 is tunneled within the hypertext transport protocol (HTTP). That is, Brown teaches a HTTPS server, which uses a client-pull architecture and does not perform functions “automatically” without client requests. Thus, Brown requires a HTTPS server system based on an HTTP request/response process for synchronization. Therefore, Brown fails to disclose the features related to automatically transferring the first metadata or the copy of the first file that is modified. Furthermore, it would not have been obvious to replace the principal structure (i.e., “the pull-based HTTP server”) of Brown with a push-based server that automatically pushes data, such as file or metadata, to a client without any client request.

144. A POSA would not have found it obvious to replace the pull-based HTTP server taught by Brown with a push-based server that automatically pushes data to a client without any client request.

145. 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.”

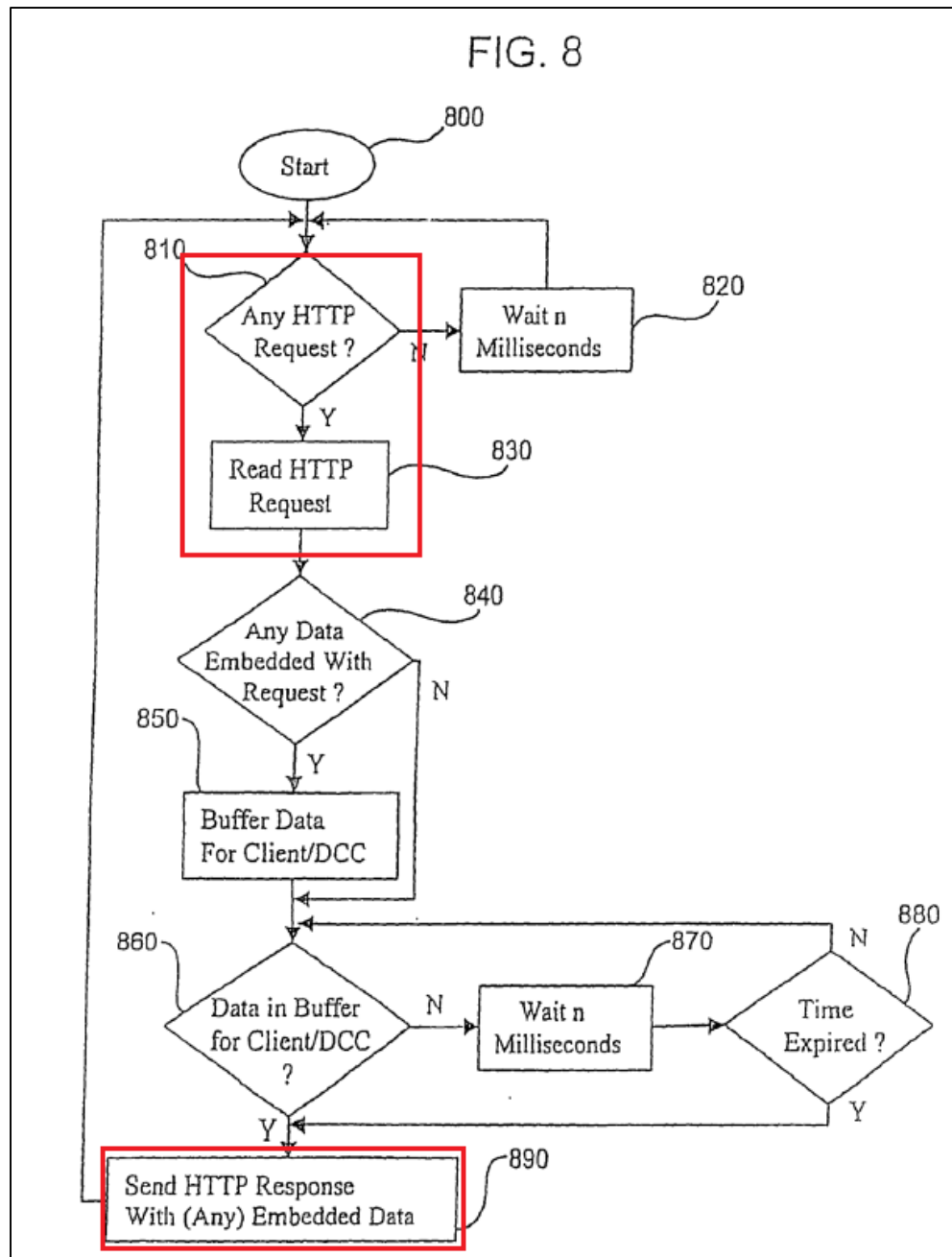
146. As noted above, Brown aims to minimize the load on the server processor, and to that end, as many synchronization operations as possible are performed by the client, including the operation to make a sync poll call to the server. In view of this, a POSA would understand that Brown cannot be modified to use a server-push mechanism.

147. I have reviewed Hesselink, and in my opinion, it describes synchronization based on an HTTP architecture which relies on the “pull” mechanism for pulling files/data from the server, just like Brown.

148. Fig. 8 of Hesselink illustrates such an HTTP request/response process for data transfer between the client and the connection server. At 810, it is determined “whether or not **any HTTP request has been received**.”

149. If an HTTP request has been received and there is no data is embedded within the HTTP request, in 860, it is determined if there is any Data present in the Buffer for the Client.

150. In 890, an HTTP Response is sent with the Data.



Hesslink (EX 1026), FIG. 8

151. Thus, Hesslink is 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.

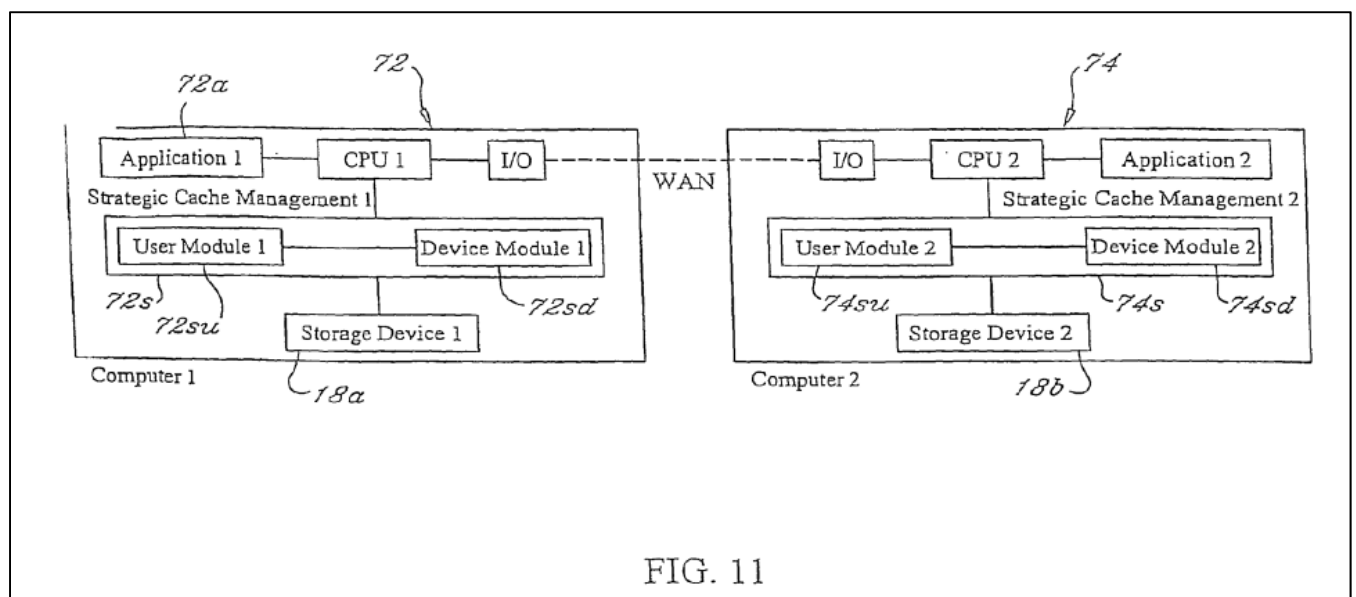
152. Hesselink is essentially related to file synchronization between multiple devices in a *peer-to-peer* environment where changes made in one or more remote storage devices are transmitted to the local device upon request from the local device. Hesselink, ¶27.

153. While Hesselink at FIG. 3 illustrates an embodiment in which connection servers 14a-14n are included in the peer-to-peer system, the purpose of the connection servers 14a-14n is only to provide authenticated and secure communications, via security server 58, between a plurality of client computers 12a-12 n and a plurality of device control computers 36a-36 d. Once a secure connection is established, data transfer is carried in the manner shown in FIGS. 7-8 between client computers 12a-12 n and device control computers 36a-36n. Specifically, 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, which merely serves as a pipeline or conduit in the proposed peer-to-peer system.

154. The Petition 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. Pet., 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.

155. 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.



Hesselink (EX 1026), FIG. 11

156. 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.” Hesselink, ¶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. Hesselink, ¶¶137, 139.

157. Clearly, 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 (Hesselink, ¶71), upon receiving a request to access to its local storage from the second computer.

158. While acknowledging the deficiency in Hesselink's embodiment due to the peer-to-peer environment, the Petition still argues that a POSA would not have been dissuaded from combining Hesselink with Brown in view of Hesselink's use of a peer-to-peer model as opposed to Brown's client/server model. This is opposite of 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."; Hesselink, ¶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. Hesselink, ¶10. Thus, Hesselink teaches away from implementing its method in a client/server model.

159. The Petition 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.” Pet., 52. However, as discussed above, while Hesselink mentions about 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). Thus, Hesselink does not teach or suggest that the “file server” implements file synchronization between computers 72, 74, and 76 based on a client/server architecture.

160. Further, in my opinion, Client-Server architecture and Peer-to-Peer architecture are fundamentally different. Peer-to-Peer architectures provide a direct link between two devices and peer do not use an intermediary such as server for communication. . Therefore, a POSA would not be motivated to use features from a peer-to-peer architecture to make changes to a client-server architecture.

161. A POSA would understand that Hesselink’s file synchronization based on the peer-to-peer system cannot be modified with the technique based on the 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. A POSA would not have been motivated to combine the teachings of Brown and Hesselink because these two references teach fundamentally different architectures and are incompatible.

162. I have reviewed the secondary reference Rao and, in my opinion, it discloses the synchronization of the metadata and raw data in databases in a manner which is fundamentally different from that for which the Petition cite it. Paragraph 0002 of Rao.

163. Rao discloses the two-phase synchronization of a database, whereby, in the first phase, metadata is synchronized between a server and a client while raw data remains on the server and, in the second phase, selected raw data is synchronized to the client. Paragraph 0063 of Rao.

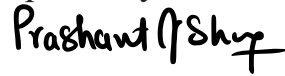
164. Rao does not teach or suggest that **“the first metadata being assigned a first priority greater than a second priority assigned to the copy of the first**

file” or the “automatic[] transfer [of] the first metadata to the second client device based on the first priority being greater than the second priority”, because Rao instead focuses on the wholesale synchronization of a database in phases – first metadata transfer and then raw data transfer. Rao does not disclose the assignment of a greater priority to the metadata than to the file, and Rao instead transfers metadata before selected raw data irrespective of any such priority. Rather, Rao categorically transfers metadata before its associated raw data.

IV. CONCLUSION

161. 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 Unites States Code.

Respectfully submitted,



Prashant Shenoy, PhD.

Date: May 9, 2023