

**IN THE UNITED STATES DISTRICT COURT  
FOR THE DISTRICT OF DELAWARE**

TOPIA TECHNOLOGY, INC.,

Plaintiff,

v.

EGNYTE, INC.

Defendant.

C.A. No: 1:21-cv-01821-CJB

**DEFENDANT'S PRELIMINARY INVALIDITY CONTENTIONS**

*Of Counsel:*

Ryan T. Beard (*pro hac vice*)  
ryan.beard@fisherbroyles.com  
FISHER BROYLES, LLP  
501 Congress Avenue  
Suite 150  
Austin, TX 78701  
Phone: (512) 577-2673

Christopher R. Kinkade (*pro hac vice*)  
christopher.kinkade@fisherbroyles.com  
FISHER BROYLES, LLP  
100 Overlook Center, 2nd Floor  
Princeton, New Jersey 08540  
Phone: (609) 415-0101

Adam K. Yowell (*pro hac vice pending*)  
FisherBroyles, LLP  
5470 Kietzke Ln., Suite 300  
Reno, NV 89511  
Phone: (775) 230- 364  
Adam.Yowell@fisherbroyles.com

Carl D. Neff (No. 4895)  
CSC Station  
112 S. French Street  
Wilmington, Delaware 19801  
Telephone: (302) 482-4244  
carl.neff@fisherbroyles.com

*Attorneys for Defendant Egnyte, Inc.*

## TABLE OF CONTENTS

|                                                                                                                                      | Page |
|--------------------------------------------------------------------------------------------------------------------------------------|------|
| I. INTRODUCTION .....                                                                                                                | 2    |
| II. A CHART SETTING FORTH WHERE IN THE PRIOR ART REFERENCES<br>EACH ELEMENT OF THE ASSERTED CLAIM(S) ARE FOUND .....                 | 6    |
| A. Identity of Each Item of Prior Art.....                                                                                           | 8    |
| B. Identification of Anticipation and Obviousness Grounds .....                                                                      | 9    |
| 1. Obviousness and Motivations to Combine .....                                                                                      | 13   |
| a. Motivations Identified During Prosecution.....                                                                                    | 16   |
| b. Combinations of References Through Citations .....                                                                                | 16   |
| c. Key References and Combinations of Related References .....                                                                       | 17   |
| d. Obviousness Combination Groups and Further Motivations<br>to Combine .....                                                        | 18   |
| (1) “Synchronize File” References.....                                                                                               | 18   |
| (2) “Server-Based” References .....                                                                                                  | 37   |
| (3) “Device Application” References .....                                                                                            | 56   |
| (4) “Me-To-Me” References .....                                                                                                      | 67   |
| (5) “Transfer When Modified” References.....                                                                                         | 78   |
| (6) “Replace Older Version” References.....                                                                                          | 95   |
| (7) “Transfer Metadata” References.....                                                                                              | 105  |
| (8) “Transfer Metadata Before File” References.....                                                                                  | 124  |
| (9) “Graphical Availability Indication” References .....                                                                             | 142  |
| (10) “Transfer Upon Communication” References .....                                                                                  | 156  |
| e. Obviousness Combinations .....                                                                                                    | 167  |
| f. Secondary Considerations .....                                                                                                    | 169  |
| C. An Identification of Any Limitations the Defendant Contends are<br>Indefinite or Lack Written Description Under Section 112 ..... | 170  |
| 1. Section 112 – Preliminary Statements.....                                                                                         | 170  |
| 2. Section 112 – Contentions .....                                                                                                   | 174  |
| D. An Identification of Any Claims the Defendant Contends are Directed to<br>Ineligible Subject Matter Under Section 101 .....       | 176  |
| E. Other Prior Art.....                                                                                                              | 180  |

Defendant Egnyte, Inc. (“Defendant”) by and through counsel, hereby serves its Preliminary Invalidity Contentions on Plaintiff Topia Technology, Inc. (“Plaintiff”) regarding the following claims of U.S. Patent No. 9,143,561 (the “’561 Patent”), U.S. Patent No. 10,067,942 (the “’942 Patent”), U.S. Patent No. 10,289,607 (the “’607 Patent”), U.S. Patent No. 10,642,787 (the “’787 Patent”), U.S. Patent No. 10,754,823 (the “’823 Patent”), and U.S. Patent No. 11,003,622 (the “’622 Patent”) (collectively, the “Asserted Claims”):

- Claims 1, 3, 4, 8, 10 and 11 of the ’561 Patent
- Claims 1, 3, 4, 10, 12 and 13 of the ’942 Patent
- Claims 1, 3, 4, 5, 7, 12, 14, 17, 19, and 20 of the ’607 Patent
- Claims 1, 2, 3, 4, 8, 9, 10, 11, 13, 14, 15, 16 of the ’787 Patent
- Claims 1, 2, 3, 4, 8, 9, 10, 11, 13, 14, 15, 16 of the ’823 Patent; and
- Claims 1, 3, 4, 5, 7, 11, 12, 14, 16, 17 of the ’622 Patent.

These contentions set forth Defendant’s preliminary Invalidity Contentions with respect to the Asserted Claims currently asserted by Plaintiff. These contentions are only preliminary. Discovery is ongoing and Defendant reserves the right to amend and/or supplement these Invalidity Contentions as the case proceeds, including through the service of Final Invalidity Contentions.

## **I. INTRODUCTION**

The following contentions are served in view of Defendant’s current understanding of the Asserted Claims as applied in Plaintiff’s August 3, 2022 Infringement Contentions (“Infringement Contentions”). In many instances, Defendant’s contentions as stated may reflect or imply a certain claim scope or claim interpretation, which is set forth in view of positions or interpretations suggested by Plaintiff’s submissions to date, and in many instances Defendant’s contentions may

suggest different alternative claim interpretations. To the extent these contentions state, reflect, or suggest a particular interpretation or reading of any claim element, Defendant does not adopt, advocate, or acquiesce to such an interpretation or reading. Defendant's contentions therefore should not be relied upon as a statement of Defendant's claim interpretations and should not be relied upon as any admission regarding the proper scope of the claims. Defendant's claim construction positions will be provided at a later appropriate time in this case, as noted above. Nor do these Invalidity Contentions constitute any admission by Defendant that any accused products or services, including any current or past versions of those products or services, are covered by any Asserted Claim. Defendant does not take any position herein regarding the proper scope or construction of the Asserted Claims.

These Invalidity Contentions try to take into account and apply Plaintiff's apparent interpretations of the Asserted Claims. Accordingly, any assertion herein that a particular limitation is disclosed by a prior art reference or references may be based in part on Plaintiff's apparent interpretation and is not intended to be, and is not, an admission by Defendant that any such construction or application of the claim language is supportable or correct. To the extent the following contentions reflect constructions of claim limitations consistent with or implicit in Plaintiff's Infringement Contentions, no inference is intended, nor should any be drawn, that Defendant agrees with or concedes those claim constructions. Defendant expressly does not do so, and reserves its right to contest them.

To the extent that prior art cited for a particular limitation discloses functionality that is the same or similar in some respects to the alleged functionality in the accused products and/or services as set forth in Plaintiff's Infringement Contentions, Defendant does not concede that those limitations are in fact met by those accused functionalities. However, to the extent Plaintiff

accuses the same or similar functionality as that disclosed in the prior art, Defendant contends such functionality is evidence of invalidity.

Defendant further reserves the right to supplement and amend these disclosures and associated document production based on further investigation, analysis, and discovery, Defendant's consultation with experts and others, and contentions or court rulings on relevant issues such as claim construction and priority dates. For example, since generalized fact discovery has not opened and Plaintiff has produced limited documents to date, additional document production from Plaintiff and deposing the alleged inventors may reveal information that affects the disclosures and contentions herein. Furthermore, in its initial Infringement Contentions, and as part of the Court's requirement that Plaintiff "shall also identify the priority date (*i.e.* the earliest date of invention) for each asserted claim," Plaintiff asserts that "the Patents-in-Suit are entitled to a priority date of at least May 18, 2004," and separately asserts in its Infringement Contentions that it has provided for inspection software that "shows periodic commits related to Topia's software development efforts embodying features of the invention from at least January 6, 2006." Plaintiff has not demonstrated that any asserted claim is entitled to any claimed invention priority date or effective filing date prior to the November 10, 2008 filing of application no. 12/267,852, including not demonstrating any entitlement to support by provisional application no. 60/986,896 filed November 9, 2007. Accordingly, Defendant has included prior art references that constitute prior art prior to November 10, 2008, and reserves the right to further investigate and develop prior art dating prior to earlier dates in view of further discovery, information, and analysis. To the extent Plaintiff is permitted to provide additional evidence and/or disclosures relating to priority claims, Defendant reserves the right to amend these Invalidity Contentions and/or modify its selection of prior art references in view of such additional information. Defendant also reserves

the right to amend these Invalidity Contentions and/or to modify its selection of prior art references in the event that Plaintiff serves supplemental or modified infringement contentions.

Because Defendant is continuing its search for and analysis of relevant prior art, and has not had an opportunity to serve subpoenas on third parties, Defendant reserves the right to revise, amend, and/or supplement the information provided herein, including identifying, charting, and/or relying upon additional prior art references, relevant disclosures, and bases for Invalidity Contentions. Additional prior art, disclosures, and invalidity grounds, whether or not cited in this disclosure and whether known or not known to Defendant, may become relevant as investigation, analysis, and discovery continue, and following claim construction proceedings in this case. Defendant is currently unaware of the extent, if any, to which Plaintiff will contend that limitations of the Asserted Claims are not disclosed in the prior art identified by Defendant. To the extent that such an issue arises, Defendant reserves the right to identify and rely upon other references or portions of references regarding the allegedly missing limitation(s).

Additionally, because discovery has not yet completed and generalized discovery has not yet commenced in this action, Defendant reserves the right to present additional prior art references and/or disclosures under 35 U.S.C. §§ 102(a), (b), (e), (f), and/or (g), and/or § 103, located during the course of discovery or further investigation, and to assert invalidity under 35 U.S.C. §§ 102(c), (d), or (f), to the extent that such discovery or investigation yields information forming the basis for such invalidity.

Additionally, Defendant reserves the right to use these disclosures as well as further discovery and disclosures relating to the alleged utility and advantages of the Asserted Claims over old modes and devices for purposes of any damages issues in the case.

Furthermore, to the extent Plaintiff asserts the doctrine of equivalents, a “doctrine of

equivalents theory cannot be asserted if it will encompass or ‘ensnare’ the prior art.” *Jang v. Boston Sci.*, 872 F.3d 1275, 1285 (Fed. Cir. 2017). To the extent Plaintiff asserts the doctrine of equivalents for any of the claimed limitations, they would ensnare the prior art for the reasons and based on the evidence discussed below and in the accompanying charts. Defendant reserves the right to use these disclosures as well as further discovery and disclosures for purposes of noninfringement.

## **II. A CHART SETTING FORTH WHERE IN THE PRIOR ART REFERENCES EACH ELEMENT OF THE ASSERTED CLAIM(S) ARE FOUND**

Subject to Defendant’s reservation of rights, attached hereto as exhibits and incorporated by reference are claim charts providing an element-by-element analysis of where specifically, in each prior art item, each element of the Asserted Claims of the Asserted Patents is found. Any reference to a figure in cited text incorporates by reference the figure itself, and any citation to a figure incorporates by reference any description of that figure in a reference. Similarly, any reference to a cited part or portion of a document encompasses all figures and text within the cited page or portion.

As noted above, these claim charts are based on Defendant’s understanding of the claim constructions applied by Plaintiff, even though Defendant does not necessarily agree with those constructions, and reserves the right to dispute them. To the extent certain claim elements are indefinite under 35 U.S.C. § 112(2), as identified and described herein, the claim charts attempt to apply the prior art to the extent Defendant can understand the claim constructions applied by Plaintiff as found in Plaintiff’s Infringement Contentions. By doing so, however, Defendant does not concede that these claim elements are definite or that one of skill in the art would have been informed with reasonable certainty about the scope of the claim element under *Nautilus Inc. v. Biosig Instrument, Inc.*, 134 S. Ct. 2120 (2014). To the extent any limitation is deemed not to be

met exactly by an item of prior art, Defendant contends that the difference would have been obvious to a person of ordinary skill in the art/or and within the knowledge of one skilled in the art at the time of the alleged invention, so that the claimed invention would have been obvious both in light of the single reference alone and/or in light of combined references. Defendant does not admit or concede that such element is not expressly or inherently disclosed by the reference at issue.

As a general matter, all portions of each prior art item are relied upon to support the disclosure of each patent claim limitation, as all portions provide general support and context. Supporting citations are nevertheless provided, but do not necessarily represent every location where a particular claim element may be found in the prior art item. Defendant reserves the right to rely on additional, or different, portions of the prior art items other than those specifically cited in these claims charts, and to supplement and/or amend these charts.

The following chart summarizes the prior art claim chart exhibits submitted herewith.

| <b>Exhibits</b> | <b>Reference(s)</b>                                                                                                  |
|-----------------|----------------------------------------------------------------------------------------------------------------------|
| 1.              | Sigurdsson '246 – U.S. Pat. Pub. 2007/0174246A1                                                                      |
| 2.              | Venkataraman '372 – U.S. Pat. Pub. 2002/0174372                                                                      |
| 3.              | Matsubara '098 – U.S. Pat. Pub. 2005/0076098                                                                         |
| 4.              | Salas '600 – U.S. Patent No. 6,233,600                                                                               |
| 5.              | Aboulhosn '042 – U.S. Patent No. 6,938,042                                                                           |
| 6.              | Moromisato '690 – U.S. Patent No. 7,734,690                                                                          |
| 7.              | Savage '050 – U.S. Pat. Pub. 2007/0283050 A1                                                                         |
| 8.              | Rao '890 – U.S. Patent No. 7,720,890                                                                                 |
| 9.              | Braginsky '414 – U.S. Pat. Pub. 2015/0199414 A1                                                                      |
| 10.             | Shappell '987 – U.S. Patent No. 7,567,987                                                                            |
| 11.             | Sagar '410 – U.S. Patent No. 7,941,410                                                                               |
| 12.             | Kaasten '022 – U.S. Patent No. 7,743,022                                                                             |
| 13.             | Brown '826 – U.S. Patent No. 7,734,826                                                                               |
| 14.             | Brown '847 – U.S. Patent No. 7,035,847                                                                               |
| 15.             | Hesselink '353 – U.S. Patent No. 7,546,353                                                                           |
| 16.             | Nurminen '601 – U.S. Pat. Pub. 2008/144601 A1                                                                        |
| 17.             | U.S. Pat. Pub. 2006/0101064, 2006/0242206, and 2004/0267697, and WO 2000/075781 (Strong, Brezak, Hamidi, and Ellman) |

| <b>Exhibits</b> | <b>Reference(s)</b>             |
|-----------------|---------------------------------|
| 18.             | SmartSync Pro system            |
| 19.             | Foldershare system              |
| 20.             | BeInSync system                 |
| 21.             | iFolder system                  |
| 22.             | Apple iDisk system              |
| 23.             | Apple iSync system              |
| 24.             | SharpCast system                |
| 25.             | GoodSync system                 |
| 26.             | Unison File Synchronizer system |
| 27.             | Subversion system               |
| 28.             | Intellisync system              |
| 29.             | TortoiseCVS system              |

**A. Identity of Each Item of Prior Art**

Subject to Defendant's reservation of rights, Defendant identifies each item of prior art that anticipates or renders obvious one or more of the Asserted Claims as cited in the claim charts submitted herewith. Where a prior art system or product is charted along with documents that describe the system or product, Defendant reserves the right to rely upon the documents as prior art to the extent such documents qualify as prior art. To the extent that references produced herewith are not identified as items of prior art that anticipate or render obvious an Asserted Claim, Defendant intends to rely on these references as background and as evidence of the state of the art at the time of Plaintiff's alleged invention.

Additionally, many of the prior art references are related patent applications and issued patents that contain substantially the same subject matter (e.g., published U.S. patent applications, and issued U.S. patents, foreign applications or issued patents). Any citation to or quotation from any of these patent applications or patents, therefore, should be understood as encompassing any parallel citation to the same subject matter in other related or corresponding applications or patents. Defendant also reserves the right to later rely upon all references or portions of references cited and submitted herewith to supplement or amend its disclosures contained herein. Also, to the

extent not expressly mentioned herein, Defendant incorporates by reference (1) any and all prior art contained or identified in documents produced thus far by Plaintiff to Defendant in this case; (2) any and all additional materials regarding invalidity that should have been produced to Defendant but have not been produced to date, to the extent that any exist; and (3) any prior art of which any named inventor of the Asserted Patents is aware and/or on which they contend the alleged invention of the Asserted Patents build upon or improve.

Each disclosed item of prior art is evidence of a prior invention and making of the invention in the United States by another under 35 U.S.C. § 102(g), as evidenced by the named inventor, authors, organizations, and publishers involved with each such reference, with the circumstances described and reflected in each reference including publications and system implementation references. Defendant further intends to rely on admissions of the named inventors concerning the prior art, including statements found in the Asserted Patents, the prosecution history, related patents and/or patent applications, any deposition testimony, and the papers filed and any evidence submitted by Plaintiff in conjunction with this litigation. For systems and products that are disclosed, Defendant also intends to rely on additional information describing those same systems and products, including information not yet received from third-parties in response to subpoenas.

**B. Identification of Anticipation and Obviousness Grounds**

Subject to Defendant's reservation of rights, Defendant identifies in the attached Prior Art Invalidity Charts prior art references that anticipate the Asserted Claims under at least 35 U.S.C. §§ 102(a), (b), (e), and/or (g), either expressly or inherently, and/or render obvious the Asserted Claims under 35 U.S.C. § 103 either alone or in combination with other references. Each Asserted Claim is anticipated by, and/or obvious in view of, one or more items of prior art identified in these disclosures, alone or in combination. Tables identifying ways in which the prior art references

cited herein anticipate and/or render obvious the Asserted Claims are provided below.

Much of the art identified in the attached exhibits/charts reflects common knowledge and the state of the art at the time of the filing date of the Asserted Patents. Defendant may rely on additional citations, references, expert testimony, fact testimony and other corroborating evidence, and other material to provide context and background illustrating the knowledge of a person of ordinary skill in the art at the time of the claimed inventions and/or to aid in understanding the cited portions of the references and/or cited features of the systems. Defendant may also rely on expert testimony explaining relevant portions of references, relevant hardware or software products or systems, and other discovery regarding these subject matters. Additionally, Defendant may rely on other portions of any prior art reference for purposes of explaining the background and general technical subject area of the reference.

Where an individual reference is cited with respect to all elements of an Asserted Claim, Defendant contends that the reference anticipates the claim under 35 U.S.C. §§ 102(a), (b), (e), and/or (g) and also renders obvious the claim under 35 U.S.C. § 103, both by itself in view of the knowledge of a person of ordinary skill in the art and in combination with the other cited references to the extent the reference is not found to disclose one or more claim elements. A single prior art reference, for example, can establish obviousness where the differences between the disclosures within the reference and the claimed invention would have been obvious to one of ordinary skill in the art. For example, “[c]ombining two embodiments disclosed adjacent to each other in a prior art patent does not require a leap of inventiveness.” *Boston Scientific Scimed, Inc. v. Cordis Corp.*, 554 F.3d 982, 991 (Fed. Cir. 2009). To the extent Plaintiff contends that an embodiment within a particular item of prior art does not fully disclose all limitations of a claim, Defendant accordingly reserves its right to rely on other embodiments in that prior art reference, or other information, to

show single reference obviousness under 35 U.S.C. § 103(a).

Where an individual reference is cited with respect to fewer than all elements of an Asserted Claim, Defendant contends that the reference renders obvious the claim under 35 U.S.C. § 103(a) in view of each other reference and combination of references that discloses the remaining claim element(s), as indicated in the claim charts submitted herewith and in the discussion below. “Under § 103, the scope and content of the prior art are to be determined; differences between the prior art and the claims at issue are to be ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background, the obviousness or nonobviousness of the subject matter is determined.” *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007), quoting *Graham v. John Deere Co. of Kansas City*, 383 U.S. 1, 17 (1966). Exemplary motivations to combine references are discussed below and in the accompanying charts. Defendant reserves the right to rely upon any references or assertions identified herein in connection with Defendant’s contention that each Asserted Claim is invalid under 35 U.S.C. § 103 and to rely upon expert testimony addressing such references and assertions. The fact that prior art is identified to anticipate the Asserted Claims presents no obstacle in also relying on that reference as a basis for invalidity based on obviousness. It is established that “a rejection for obviousness under § 103 can be based on a reference which happens to anticipate the claimed subject matter.” *In re application of Meyer*, 599 F.2d 1026, 1031 (C.C.P.A. 1979). To the extent any prior art item cited above may not fully disclose a limitation of an Asserted Claim or is alleged by Plaintiff to lack disclosure of the limitation, such limitation is present and identified in another prior art item as shown in the attached claim charts.

Many of the cited references cite or relate to additional references and/or products, services, or projects. Many of the cited references also cite software, hardware, or systems. Defendant may rely upon such cited additional references and copies or exemplars of such software, hardware, or

systems. Defendant will produce or make available for inspection any such cited references, software, hardware, or systems that it intends to rely upon. Defendant may also rely upon the disclosures of the references cited and/or discussed during the prosecution of the Asserted Patents and/or the assertions presented regarding those references.

Although Defendant has identified at least one disclosure of a limitation for each prior art reference, each and every disclosure of the same limitation in the same reference is not necessarily identified. In an effort to focus the issues, Defendant cites representative portions of an identified reference, even where a reference may contain additional support for a particular claim limitation. Persons of ordinary skill in the art generally read an item of prior art as a whole and in the context of other publications and literature. Thus, to understand and interpret any specific statement or disclosure within a prior art reference, such persons would rely on other information within the reference, along with other publications and their general scientific knowledge. As such, Defendant may rely upon uncited portions of the prior art references and on other publications and expert testimony to provide context, and as aids to understanding and interpreting the portions that are cited.

Defendant reserves the right to further streamline and reduce the number of anticipation or obviousness references relied upon with respect to a given Asserted Claim and to exchange or otherwise modify the specific references relied upon for anticipation and within each obviousness combination for each Asserted Claim. Generalized discovery has not yet begun, and Plaintiff has not provided any allegation as to any alleged pre-filing invention dates or with respect to claim limitations that are allegedly lacking or not obvious in the prior art. Each limitation of the Asserted Claims was well-known to those of ordinary skill in the art before the filing date of the non-provisional applications from which the Asserted Patents claim priority, as detailed below. As

explained in detail throughout these Contentions, the Asserted Claims of the Asserted Patents are anticipated by, and/or obvious in view of the prior art references identified herewith.

### **1. Obviousness and Motivations to Combine**

Each prior art reference may be combined with one or more other prior art references to render obvious the Asserted Claims in combination, as explained in more detail below. The disclosures of these references also may be combined with information known to persons skilled in the art at the time of the alleged invention, and understood and supplemented in view of the common sense of persons skilled in the art at the time of the alleged invention, including any statements in the intrinsic record of the Asserted Patents and related applications.

A person of ordinary skill would have been motivated to combine the identified prior art based on the nature of the problem to be solved, the teachings of the prior art, and the knowledge of persons of ordinary skill in the art. The identified prior art references, including portions cited in the Prior Art Invalidity Charts, address the same or similar technical issues and suggest the same or similar solutions to those issues as the Asserted Claims. On such basis, on an element-by-element basis, Defendant expressly intends to combine one or more prior art items identified in the claim charts with each other to address any further contentions from Plaintiff that a particular prior art item supposedly lacks one or more elements of an Asserted Claim. In other words, Defendant contends that each charted prior art item can be combined with other charted prior art items when a particular prior art item lacks or does not explicitly disclose an element or feature of an Asserted Claim. The suggested obviousness combinations described below are not to be construed to suggest that any reference included in the combinations is not anticipatory. Further, to the extent that Plaintiff contends that any of the anticipatory prior art fails to disclose one or more limitations of the Asserted Claims, Defendant reserves the right to identify other prior art

references that, when combined with the anticipatory prior art, would render the claims obvious despite an allegedly missing limitation. Defendant will further specify the motivations to combine the prior art, including through reliance on expert testimony, at the appropriate later stage of this lawsuit.

A person of skill in the art would have been motivated to combine the identified prior art items. As the United States Supreme Court held in *KSR International Company v. Teleflex, Inc.*, 550 U.S. 398, 416 (2007): “The combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results.” The Supreme Court further held that, “[w]hen a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill.” *Id.* at 417.

The Supreme Court has further held that “in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle.” *Id.* at 420. It is sufficient that a combination of elements was “obvious to try” holding that, “[w]hen there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions, a person of ordinary skill has good reason to pursue the known options within his or her technical grasp. If this leads to the anticipated success, it is likely the product not of innovation but of ordinary skill and common sense.” *Id.* at 421. “In that instance the fact that a combination was obvious to try might show that it was obvious under § 103.” *Id.* Finally, the Supreme Court recognized that “[g]ranting patent protection to advances that would occur in the

ordinary course without real innovation retards progress and may, in the case of patents combining previously known elements, deprive prior inventions of their value or utility.” *Id.* at 419. All of the following rationales recognized in *KSR* support a finding of obviousness:

- Combining prior art elements according to known methods to yield predictable results;
- Simple substitution of one known element for another to obtain predictable results;
- Use of known technique to improve similar devices (methods, or products) in the same way;
- Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;
- “Obvious to try”—choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;
- Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art; and
- Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

Certain of these rationales are discussed more specifically below. The fact that others are not discussed more specifically should not be interpreted as an admission or concession that it does not apply. To the contrary, the discussion below simply provides more explanation of these specific rationales.

Defendant further contends that the prior art identified in these Invalidity Contentions is

evidence of simultaneous or near-simultaneous independent invention by others of the alleged invention as recited in one or more of the Asserted Claims. Defendant reserves its right to rely on the simultaneous or near-simultaneous independent invention by others as further evidence of the obviousness of the Asserted Claims.

Each limitation of the Asserted Claims was well known to those of ordinary skill in the art before the filing dates of the respective non-provisional application to which each Asserted Patents claim priority, as detailed below and in the attached charts.

The elements recited in the Asserted Claims are mere combinations and modifications of these well-known elements. A person of ordinary skill in the art would be able, and motivated, to improve the existing technology in the same or similar manner by combining or modifying the individual elements that were already known in the art to yield predictable results.

Subject to the foregoing, Defendant identifies the following exemplary reasons that skilled artisans would have combined elements of the prior art to render obvious the Asserted Claims. The fact that others are not discussed more specifically should not be interpreted as an admission or concession that it does not apply. To the contrary, the discussion below simply provides more explanation of these specific rationales.

**a. Motivations Identified During Prosecution**

Defendant hereby expressly incorporates by reference any statements and reasons set forth by the Examiner during prosecution of the Asserted Patents and related patent applications as to why it would have been obvious to modify or combine references to achieve the limitations of the Asserted Claims.

**b. Combinations of References Through Citations**

Numerous prior art references cite to, discuss, or build upon other references. Where one

reference cites or discusses another reference, a person of ordinary skill in the art would have been motivated to consider their teachings in combination, as they would be understood to provide related teachings in a similar field.

**c. Key References and Combinations of Related References**

In some instances, multiple prior art publications and/or physical references discuss or address the same or substantially similar underlying system, software, or other project, such as commercial software products and successive versions thereof, or multiple publications discussing the same subject matter. Where multiple references discuss or relate to the same or related underlying projects, systems, or other subject matter, it was obvious to combine the discussions and disclosures of the references as they would be understood to describe features or potential features of the underlying project, system, technique, or other subject matter. Similarly, where one reference cites or discusses other references or their teachings, or references have one or more authors in common and a related area of subject matter, it was obvious to consider the teachings of the references in combination with each other due to the express relationships and commonalities between the references.

Further, for each set of references that together refer to or describe the same system and/or potential improvements to the same system, it would have been obvious to combine such references. In particular, to the extent the items are commonly owned, they would be within the knowledge of the same company, and the company would be motivated to use elements from one reference to improve another seeking to solve the same or similar problems.

Similarly, where more than one references discuss a system, it would have been obvious to combine any such references that refer to and/or discuss the same system.

**d. Obviousness Combination Groups and Further Motivations to Combine**

In addition to combinations of references and motivations to combine identified elsewhere herein, including within claim charts, Defendant identifies combinations and motivations to combine based on references grouped by subject matter, in the manner approved by courts applying the Northern District of California Patent Local Rules regarding invalidity contentions, which notably have additional requirements with respect to identifying obviousness combinations beyond those of the Scheduling Order in this matter. *See, e.g., Avago Techs. Gen IP PTE Ltd. v. Elan Micro. Corp.*, No. C04-05385 JW (HRL), 2007 U.S. Dist. LEXIS 97464, at \*10-11 (N.D. Cal. Mar. 28, 2007) (organizing prior art references into “groups” and identifying combinations as a set of references “and/or” another set of references); *Keithley v. Homestore.com, Inc.*, 553 F. Supp. 2d 1148, 1150 (N.D. Cal. 2008) (following *Avago* – “Apple’s grouping method is permissible under the Local Rules”); *see also Multimedia Patent Trust v. Apple Inc.*, Case No. 3:10-cv-02618-H (KSC), 2012 WL 12868250 (S.D. Cal. Nov. 5, 2012); *High Point Sarl v. Sprint Nextel Corp.*, 2011 U.S. Dist. LEXIS 111158, at \*29-30 (D. Kan. Sept. 28, 2011).

Defendant identifies references in groups below, based on common subject matter relevant to the Asserted Claims, for the purpose of disclosing obviousness combinations and further disclosing motivations to combine. The absence of a particular reference from a group is not an admission or suggestion that the reference does not disclose any particular claim limitations.

**(1) “Synchronize File” References.**

The Asserted Claims recite limitations regarding automatically transferring a copy of a file between electronic devices, such as when the user modifies the content of the file. Nothing about these claim limitations add anything novel or non-obvious over the prior art. To the contrary, this feature was well-known, routine, conventional, and widely available long before the alleged

invention. As the attached claim charts demonstrate, this subject matter is disclosed in numerous references. For purposes of these contentions, references whose disclosures are cited in these contentions with respect to this claimed subject matter as reflected in the claim charts (including without limitation the specific references cited below) are hereby labeled the “Synchronize File” references.

The implementation of the claimed features would have been obvious to a person of ordinary skill in the art least because doing so would have involved the combination of prior art elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the application of a known technique to a known device ready for improvement to yield predictable results, and/or would have been obvious to try. Moreover, a person of ordinary skill in the art would have readily appreciated the well-known and widespread use of these features and the associated benefits and advantages of their use. Persons of ordinary skill in the art would have been motivated to combine the teachings of these references with any system and/or disclosure that relates to the automatic synchronization of files or other data content, including server-based systems and peer to peer systems, as well as hybrid systems that implement both server-based features and peer-to-peer file transfer. It was well-known that synchronizing files such as by transferring copies of files was one optional design choice for maintaining consistency of files between user devices, among other known design choices such as transferring modified portions of files (e.g., modified blocks or difference files). Automatically synchronizing copies of files had known benefits in the prior art including simplicity and efficiency in system design by not needing to track portions of changed files, and permitting a source device (such as a server) to maintain a current complete copy of a given file that could be accessed by other connected devices for file synchronization purposes. For example,

references disclose the following motivations to combine, such as the following as well as the other disclosures that are cited in the prior art claim charts submitted herewith:

- Sigurdsson at, e.g., [0026] “The system **100** can permit a user using a second computer device to access copies of files (or copies of the content of the files) that were resident on a first computer device. Exchange of content between the computer devices can occur automatically without a user specifying that the content should be sent to another computer device. For example, a user may edit a word document file at his home computer. When the user saves the document, the documents content is automatically extracted and trans mitted to the user's computer at work. When the user searches for the modified document later at his work computer, the content is presented to the user even though the word document was not previously stored on the user's work computer. This can have the additional advantage that only documents that are recently accessed by the user are transferred between the user's various computers.”
- Salas at, e.g., 11:26-60 “For example, the object ID for an object is used to query the local database **20**. The object record may include a modification tag value (as described above), or each data object may be provided with one or more state bits which can be set to indicate the file or data is stale. If the modification tag value or state bits indicate that the object needs to be synchronized, the updated object may be requested from the server in the foreground or in the background. Alternatively, a client workstation **12** may periodically search its entire local database **20** for objects which need to be updated. This may take the form of a database query for objects having a modification tag value less than the current value, or a database query for objects having a particular value for state bit fields. Objects returned by the query are requested from the server as discussed above. Synchronization is enabled by storing all records in the

server database with an associated modification tag. The tag is a positive integer which is taken from an ever-increasing counter. The counter increments each time it is read, i.e., each time a new modification tag is assigned to a data object stored on the server **14**. When a client workstation **12** synchronizes its local databases and files, it also receives the current modification tag, i.e., it also receives the current value of the counter. Alternatively, the current modification tag value can be included as extra information in each "wrapper page." The client workstation **12** includes the last modification tag value it received when it makes a subsequent synchronization request. The server **14** transmits to the client workstation **12** any data objects to which the user has appropriate access rights that also have a modification tag value greater than the modification tag value sent with the synchronization request. The client workstation **12** stores the received data objects in its local database and stores the new received modification tag value." 13:7-26 "As noted above, users may perform work on files and objects locally and upload the modified files and objects to the server **14** for viewing, comment, or further modification by other project team members. The systems and method of the present invention allow users to upload newly created and modified files to a server **14** using an intuitive drag and-drop method. Referring now to FIG. 7, a user creates a new file or modifies a file downloaded from the server **14** (step **702**). It should be understood that this step includes actions such as creating a new version of the file locally so that other users may still check out the "original" copy of the file present on the server **14**. Once the user is finished editing the file, it may be uploaded to the server **14** to allow other users access to it. The user signals that the file should be transmitted to the server **14** by dragging the file onto an eRoom displayed by the browser (step **704**). Dropping the file into the displayed eRoom invokes an ActiveX control or a background daemon process which manages the upload of the file to the server **14**."

- Venkataraman at, e.g., [0028] “In a preferred embodiment, three different synchronization modes are possible. In the first, remote data having temporal priority (i.e., most recently created/modified) is preferred and overwrites data found in the central data. In the second, the opposite is true, and central data having temporal priority is preferred and overwrites data found the remote data. In the third mode, data having temporal priority is preferred regardless of its source. These three options are presented to the user at step **310**, along with an option to cancel the synchronization procedure. Upon receipt, the user's selection in response to the prompt is provided to the synchronization application. With the synchronization mode information, the synchronization application synchronizes the remote and central data accordingly at step **326**. In this manner the central data is now synchronized to reflect the changes necessitated, if at all, by the remote data. The synchronized data (i.e., the now-updated central data) is transmitted by the synchronization application at step **328** and received by the synchronization device at step **312**. The form of the synchronized data is a matter of design choice. For example, the synchronized data may comprise a copy of all of the relevant central data, or information indicative of the modifications that must be made to the remote data in order to synchronize it to the central data. Regardless, the synchronization application, at step **330**, subsequently updates (either periodically or on-demand) the local data on any local devices that subscribe to synchronization services for the central data.” [0017] “Note that the remote data **120** and central data **122** are of the same type as the local data, i.e., including, but not limited to, contact data (e.g., phone numbers, addresses, etc.), calendar data, email, shared files and the like.” [0026] “At any time prior to or substantially simultaneous with steps **302** through **306**, the central data may be updated by the synchronization application at step **320**. For example, while a user is away from the office, his or her assistant may enter local data on

a local device that requires synchronization with the central data. The changes to the local data may cause synchronization to be initiated automatically. Conversely, the synchronization application may periodically query the local device to determine if any updates to the central data are needed. Operation of a synchronization application in this manner is well known in the art. Regardless, the central data serves as the primary storage point for up to date data.”

- Matsubara at, e.g., [0030] “In a step **107**, the peer member can view the alert and can decide whether to access the information or not. If she decides she wants to access the information, her peer client software can access the information, in which case a copy **32** of File-X can be obtained. This can be done using typical procedures used in a P2P system. The peer client communicates with the management server **12** and with one of a number of peer clients to download (or otherwise) the information. The information may be:” [043] “Typically, the actual media and text files which are represented in the window **206** are stored among the machines/system of the peer members who participated in the conference session. Thus, when user accesses the contents of a file, the requesting peer client accesses the management server **12** to obtain information identifying which peer member has the file. The requesting peer client then communicates with the peer member to obtain the file.”
- Kaasten at, e.g., 4:37-53 “There are two distinct flavors of data synchronization. In the first, called here “full-copy synchronization,” complete copies of the synchronized files exist on each cooperating computing device. Note that in FIG. **1b**, files and folders on different computing devices are given different reference numbers. For example, file **A1 118** on computing device **A 102** is synchronized with file **C1 136** on the laptop **106**. Synchronization makes these two files act, in some respects, as if they were one file. However, they really are separate files and are stored in their entirety on the separate computing devices. These files

take up storage space on each computing device and may be accessed on one computing device even if the other computing devices are not accessible. Of course, in the latter scenario and until the other devices again become accessible, synchronizing changes to the files is not possible. (Setting up synchronization after Such a period of inaccessibility is discussed below with respect to step 308 of FIG.3a.)” 8:18-21 “The change notification itself can take several forms. A simple embodiment of a change notification, and perfectly adequate for small files, contains an identifier of the file and the entire contents of the file as changed.”

- Savage at, e.g., [0011] “FIG. 1 illustrates an example synchronization system 100. Multiple synchronization clients 102, 104, and 106 are coupled to a synchronization server 108 via a network 110 or another communications channel, such as FireWire (IEEE 1394), USB (Universal Serial Bus), RS232 (serial channel), etc. It should be understood that a synchronization system may include any number of synchronization clients (e.g., one, two, or more), and that three synchronization clients are shown merely as an example. In addition, a synchronization client may be coupled to multiple synchronization servers.” [0012] “A synchronization client 106 maintains a file system, which is configured to be synchronized with data stored on the synchronization server 108. In this way, for example, the synchronization server 108 can provide backup storage capabilities for the synchronization client 106, such that the synchronization client 106 can recover from a loss of data (e.g., a hard disk crash or an inadvertent file deletion) by restoring the data from the synchronization server 108. Likewise, if more than one synchronization client is coupled to the synchronization server 108, then data on any client can be shared with other clients via the synchronization server 108.” [0031] “In one implementation, the scanning module 112 detects that a file on the synchronization client has changed, generates and schedules a synchronization report for that

file, and transfers the synchronization report to the synchronization server 108 -at the appropriate time. In one implementation, the transfer may be accomplished by a transmission from the client to the server over a communications network. However, other implementations may transfer the synchronization report via an intermediary, such as a shared storage location, or by having the report read from the client by the server.” [0032] “Upon receipt of a synchronization report for a file, the synchronization server 108 checks to determine the state of the file within its own files system. If the synchronization server 108 needs to receive the file from the client in order to synchronize with the client, the synchronization server 108 request that the client upload the file to the server. An update module 118 uploads the requested file to the synchronization server 108 (see synchronizing files 120).”

- Rao at, e.g., 3:14-17 ““FIG. 1 illustrates an embodiment 100 of the present invention showing a synchronized database system. A server system 102 and a client system 104 are connected by a communications path 106.” 4:56-60 “The client system 104 is synchronized with the server system 102. When synchronized, the server's item repository 110 is synchronized with the client item repository 130. On the initial synchronization, the server system 102 may merely copy the entire item repository 110 to the client system 104.” 4:61-5:5 “Subsequent synchronization events may involve detecting which items have been updated or changed since the last synchronization event. If an item has been updated, but the corresponding item on the opposite system has not been changed since the last synchronization, the updated item may be copied to the opposite system. If both the item and its corresponding item on the opposite system have been updated, a decision must be made to select one or the other item. In some embodiments, rules may be used for determining which item to keep, while in other embodiments, a user may be queried. Synchronization of files may follow the same

methodology as synchronization of items.”

- Braginsky at, e.g., [0005] “A locally cached file system provides disconnected operation and access to data stored on a server system over a network. Changes to files made on a client system are automatically synchronized to the server system asynchronously. A client system operating in synchronized mode maintains copies of files stored on the server system. These files may be stored in folders of a virtual hard drive established on the server system. Asynchronous file upload and/or download operations are made transparent to the user through an automated background process. In some embodiments meta-data for files to be transferred is sent to the server system first, followed by the files. In some embodiments, file transfer order for a set of files to be transferred is governed by a user-defined priority policy based on one or more parameters, such as date and time of last file modification, folder or directory locations of the files (i.e., the locations of the files in a directory hierarchy), file size, and file type.” [0020] “FIG. 1 is a block diagram of one embodiment of a file management system 100 including a one or more client systems 102 (e.g., laptop computer, desktop computer, personal digital assistant (PDA), mobile phone, media player, etc.) coupled to one or more server systems 104 via a network 106. The network 106 can be one or more networks having one or more types of topologies, including but not limited to the Internet, intranets, local area networks (LANs), wireless networks, Ethernet, Storage Area Networks (SANs) and the like. The client system 102 generally includes a processor 118 coupled to a display device 114 for presenting a locally cached file system 116 to the user. In some embodiments, the file system 116 manages files stored in a cache data structure on one or more data storage devices 119 (e.g., random access memory (RAM), hard disk, optical disk, portable storage device, etc.). The term “cache” or “cache data structure” is not limited to cache memory but can include any

data structure on the computer-readable medium 119 that is accessible to the client system 102.”

- Shappell at, e.g., Abstract “A computer implemented method and system enable users to share files in a server-less shared space. By providing access to such 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.” 1:62-64 “The present invention is directed to a system and method for sharing files in a server-less, secured shared space that is presented to a user through a graphical interface.” 2:1-11 “A shared space is presented as an entry in the file system such that user selectable management tasks and items may be activated. Such action includes various operations that are performed with respect to shared files within the space. Upon creation of the shared space, other users can access and/or transmit files, and perform other file management operations such as drag and drop shared files, move, copy and other file system management tasks. In addition, shared files and directories may be linked into the shared space. This provides a clear sense of user operations that are available for shared files, to thereby create an ease of use of group spaces.” 2:12-15 “The invention provides a system and method for sharing files in the shared space. An owner of the shared space can invite other users into the space for the purpose of sharing files such as pictures, video or other content.”
- Sagar at, e.g., 2:65-3:5 “With reference to FIG. 1, an exemplary system for implementing the blocks of the claimed method and apparatus includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120.” 3:6-10 “The computer

110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180, via a local area network (LAN) 171 and/or a wide area network (WAN) 173 via a modem 172 or other network interface 170.” 3:46-59

“Generally, a mesh may be considered a set of nodes or devices that are associated with each other, intercommunicate with each other, and share resources. FIG. 2 illustrates a mesh 200 that may be a peer to peer network where nodes of the network 201-204 share common services. A peer to peer configuration may be one out of many possible implementations for the mesh. Generally, software and services that enable devices and applications to run in a mesh may be called a mesh operating environment (MOE). The software and services may include application portions that run on each node and application portions that run external to each node but are available to each node via the mesh. It should be noted that a mesh device or node may be a physical device or virtual object that can participate in the mesh.” 4:1-11

“A synchronized folder 210 may represent a set of folders that are maintained to be consistent across a plurality of nodes or devices 201-204. In other words, a local copy of a synchronized folder at each node may be consistent with every other local copy of the synchronized folder at every other node. Maintaining consistency of the synchronized folder among a plurality of nodes may be extremely difficult where each node may modify their local synchronized folder (e.g., a local copy of the synchronized folder) and when multiple nodes may each be required to provide a local view of the synchronized folder consistent with local node semantics.” 4:23-32

“Generally, updates or changes to content may be propagated using a broadcast model in which a change in the synchronized folder at one node initiates a broadcast to all nodes of the mesh of the synchronized folder to indicate the change. Synchronization conflicts may represent conflicts between different update data (e.g., originated by updates at different nodes)

for the same item or related item of the synchronized folder (where an item may be a folder or a file). Other conflicts may also arise due to the capabilities of the underlying file system.”

- Moromisato at, e.g., 20:8–16 “Once the file synchronizer 308 determines that the file system has been changed it passes the changes to the document share engine in the shared workspace which thereupon disseminates the changes. If a file has been edited, then the document share engine may either disseminate the entire updated file contents or determine the changes made to the file by comparing the current file contents against a copy of the contents that it maintains and disseminate only the differences.” 7:25–28 “The file synchronizer 308 is a software program that insures that any changes made to a file in the Windows shell file system 302 are also made to copies of that file that are stored in the collaborative system and vice versa.”
- Brown ‘847 at, e.g., 13:11–23 “The client also computes a MDA from the file. The client SA then requests the MDA from the file to be downloaded and compares the two MDAs. If the two arrays are sufficiently similar, the client SA performs a special download where it requests the specific 4K blocks that have differing message digest values. It creates the download file by modifying the copy of the file with the requested downloaded 4K blocks. On the other hand, if less than a threshold number of message digests in the MDAs match, then the entire file is downloaded from the server. Once the download file is completely constructed, the client inserts the download file into its final location, replacing an older version of the file if it exists.” 12:43–50 “MDA 620 can then be used to determine if the file can be only partially uploaded. If at least a threshold number of message digests in the MDAs on the client and server match, then only the blocks corresponding to message digests that differ between the client and server need to be transmitted. On the other hand, if less than a threshold number of message digests in the MDAs match, the entire file is transmitted.” 3:40–53 “From the user’s perspective, the

synchronization process is automatic, runs in the background and requires minimal monitoring or intervention. In an embodiment of the invention, the client SA initiates synchronization on a fixed time interval. But a person skilled in the art will recognize that the client SA can use any scheduling algorithm to initiate synchronization. In addition, the user can also initiate synchronization at any time manually. The client SA monitors local file system activity within the directory (sometimes called a folder) on the client so that it can efficiently locate changes to send to the server SFS during synchronization. The client SA also monitors client file system activity to prevent synchronization from interfering with running applications on the client machine.”

- Brown ‘826 at, e.g., 13:1–14 “The client also computes a MDA from the file. The client SA then requests the MDA from the file to be downloaded and compares the two MDAs. If the two arrays are sufficiently similar, the client SA performs a special download where it requests the specific 4K blocks that have differing message digest values. It creates the download file by modifying the copy of the file with the requested downloaded 4K blocks. On the other hand, if less than a threshold number of message digests in the MDAs match, then the entire file is downloaded from the server. Once the download file is completely constructed, the client inserts the download file into its final location, replacing an older version of the file if it exists.” 12:37–44 “MDA 620 can then be used to determine if the file can be only partially uploaded. If at least a threshold number of message digests in the MDAs on the client and server match, then only the blocks corresponding to message digests that differ between the client and server need to be transmitted. On the other hand, if less than a threshold number of message digests in the MDAs match, the entire file is transmitted.” 3:42–55 “From the user’s perspective, the synchronization process is automatic, runs in the background and requires minimal monitoring

or intervention. In an embodiment of the invention, the client SA initiates synchronization on a fixed time interval. But a person skilled in the art will recognize that the client SA can use any scheduling algorithm to initiate synchronization. In addition, the user can also initiate synchronization at any time manually. The client SA monitors local file system activity within the directory (sometimes called a folder) on the client so that it can efficiently locate changes to send to the server SFS during synchronization. The client SA also monitors client file system activity to prevent synchronization from interfering with running applications on the client machine.”

- Hesselink at, e.g., 41:05–14 “To work on a selected file, the user may choose to transfer the entire file over from the remote device to the local device, and this transfer can be done automatically upon receiving the file data access requests made by local applications, or by simply selecting a transfer command from a drop down menu, selection box, or other visual indicator on the display associated with the user module. Thus, the file may be accessed and worked on in exactly the same manner that the user would work if accessing the file directly from the computer that stores that file locally.”
- iFolder at, e.g., <https://www.novell.com/documentation/ifolder21/pdfdoc/admin/admin.pdf> at 150--51. “Automatic Sync: ... Allows the iFolder client to automatically synchronize the user's iFolder files between the local iFolder directory and the iFolder server. Sync to Server Delay: ... If Automatic Sync is allowed, sets the default time (in seconds) that the iFolder client waits after a file in the local iFolder directory changes until it automatically uploads the file to the iFolder server. Also sets the minimum and maximum values allowed. Sync from Server Interval: ... If Automatic Sync is allowed, sets the default time (in seconds) after a synchronization occurs that the iFolder client waits to check with the iFolder server to

determine if there are changed files it needs to automatically download to the local iFolder directory. Also sets the minimum and maximum values allowed.” <https://www.novell.com/documentation/ifolder21/pdfdoc/admin/admin.pdf> at 194. “There are some applications that rewrite the complete file regardless of how minor the change. Microsoft Word, for example, behaves like this. Thus, if the application that you are using completely rewrites the file, iFolder will recognize it as 100% new content and synchronize the whole file.” <https://www.novell.com/documentation/ifolder21/pdfdoc/enduser/enduser.pdf> at 10–11. “5. The iFolder server downloads any new files from the iFolder server to the local iFolder directory. ... It might transfer the entire file, depending on how the application in use saved file changes. ... 6. The iFolder client uploads any new files or changes to files from the local iFolder directory to the iFolder server. ... It might transfer the entire file, depending on how the application in use saved file changes.”

- BeInSync at, e.g., [https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 5. “Each shared folder can now be set to Auto or Manual synchronization. Auto synchronization will constantly synchronize your files, while manual synchronization requires the user to specify when to transfer the updated files” [https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 40. “BeInSync constantly synchronizes all of your information while BeInSync is online and connected on all of your computers. Whenever a file on your local computer is edited, a new file is added, or a file is deleted, the change will automatically be synchronized to your remote computers.” [https://web.archive.org/web/20060315125029/http://www.beinsync.com/notes/release\\_notes\\_1\\_5\\_12.php](https://web.archive.org/web/20060315125029/http://www.beinsync.com/notes/release_notes_1_5_12.php) “Sync Now/Manual Sync: Define priority for a particular file or folder. BeInSync will fetch the file from the best available route on your

computer network.”

- FolderShare at, e.g., <https://web.archive.org/web/20060313171107/https://www.foldershare.com/download/files/FolderShareGuide.pdf> at 3. “By default each device is set to automatically sync – this means that all the files in these folders will be copied to each computer and any changes made to files in this folder will be made on all devices that you’ve setup to be included in this library (a library is the collection of folders that you’ve selected to keep in sync or share with other people). ... Once the file is downloaded – any changes made to that file will be automatically synced. ... FolderShare runs silently in the background constantly monitoring the files in the folders you specified to keep in sync for any changes. Once changes are detected, they are replicated to all other computers. You can continue to access and work with the files in this synced folder as you normally do.” <https://www.cnet.com/reviews/bytetaxi-foldershare-review> “FolderShare automatically updates files, so once a file is changed on one system, all the other linked computers automatically use a direct, peer-to-peer SSL-encrypted connection to download the updated file.” <https://web.archive.org/web/20060313171026/https://www.foldershare.com/info/faq/AboutFolderShare.php> “FolderShare works in the background to automatically recognize modified files and auto-upload them to other computers connected to your library.”
- SharpCast discloses automatic file synchronization. “All your desktop photos automatically synchronized to the web, and vice versa.” <https://web.archive.org/web/20060829025503/http://www.sharpcast.com/products/photos/>
- GoodSync discloses automatic synchronization of copies of files. “Most synchronization software claim to synchronize your files, but many simply copy from place to place. GoodSync offers true bi-directional synchronization, which prevents deletion of files and data loss.” <https://web.archive.org/web/20060417232608/http://www.goodsync.com/index.html>

- Subversion at, e.g.,  
[https://web.archive.org/web/20071124042709/http://www.eclipse.org/subversive/documentation/gettingStarted/subversion/svn\\_intro.php](https://web.archive.org/web/20071124042709/http://www.eclipse.org/subversive/documentation/gettingStarted/subversion/svn_intro.php) “Subversion® is an open-source version control system. Subversion® allows developers to share there projects on the repositories, where they are stored afterwards. Repository is much similar to a file server, except the fact, that it not only stores the copy of the file system, but its previous states and changing history. Subversion® access its repositories using network, so it provides a probability for a person to work over some shared files and watch for every possible changes made by other developers.”  
<https://web.archive.org/web/20071116172448/http://svnbook.red-bean.com/en/1.4/svn-book.pdf> at xviii-xix. “Subversion is a free/open-source version control system. That is, Subversion manages files and directories, and the changes made to them, over time. This allows you to recover older versions of your data, or examine the history of how your data changed. In this regard, many people think of a version control system as a sort of ‘time machine’. Subversion can operate across networks, which allows it to be used by people on different computers. At some level, the ability for various people to modify and manage the same set of data from their respective locations fosters collaboration. Progress can occur more quickly without a single conduit through which all modifications must occur. And because the work is versioned, you need not fear that quality is the trade-off for losing that conduit—if some incorrect change is made to the data, just undo that change. Some version control systems are also software configuration management (SCM) systems. These systems are specifically tailored to manage trees of source code, and have many features that are specific to software development—such as natively understanding programming languages, or supplying tools for building software. Subversion, however, is not one of these systems. It is a general system that

can be used to manage any collection of files. For you, those files might be source code—for others, anything from grocery shopping lists to digital video mixdowns and beyond.”

- SmartSync Pro at

<http://web.archive.org/web/20070204114427/http://www.smartsync.com:80/smartsyncpro/key-features.html>. “Common features:

- Attractive and easy-to-use interface.
- Synchronizes your data only when it needs synchronizing.
- Flexible filtering system to include only the files you want.
- Tree-like structure of profiles and profile groups.
- Drag and drop support for folders, profiles and profile groups.
- Full support for unicode file names (Japanese, Korean, Chinese, etc.).
- Activity log.
- Multi-threaded background processing.
- Easy-to-use New Profile Wizard for creating new profiles.
- Color highlighting for new, modified and deleted files and file conflicts.
- Detailed synchronization progress for all operations.
- Resolves file conflicts when the same files have been modified on both sides.
- Adjusts for different time zones, Daylight Saving Time.
- Scheduling support to run profiles whenever you want.
- Can dial-up before synchronization and hang up afterward.
- Can run external programs before and after synchronization.
- Profile access can be restricted to selected users only.
- Detects locked files and asks to retry to overwrite them.
- Command line support.
- Can be run as service under NT4/2000/XP.”

- Nokia Intellisync at

[https://www.novell.com/documentation/groupwise\\_mobile\\_3/pdfdoc/admin\\_gde/admin\\_gde.pdf](https://www.novell.com/documentation/groupwise_mobile_3/pdfdoc/admin_gde/admin_gde.pdf). “Nokia Intellisync Mobile Suite is a collection of products designed to help you support

your mobile workforce through a single-source solution to meet your mobile computing needs.

Nokia Intellisync Mobile Suite consists of both server and client components that facilitate interaction with the server. The server is the data synchronization host to which mobile devices connect. The server Admin Console controls functions such as security, user authentication, communication, logging, and reporting, among others.”

- Unison File Synchronizer at

<https://web.archive.org/web/20071026022954/https://www.cis.upenn.edu/~bcpierce/unison/index.html>

. Unison is a file-synchronization tool for Unix and Windows. It allows two replicas of a collection of files and directories to be stored on different hosts (or different disks on the same host), modified separately, and then brought up to date by propagating the changes in each replica to the other.

Unison shares a number of features with tools such as configuration management packages (CVS, PRCS, Subversion, BitKeeper, etc.), distributed filesystems (Coda, etc.), uni-directional mirroring utilities (rsync, etc.), and other synchronizers (Intellisync, Reconcile, etc). However, there are several points where it differs:

- Unison runs on both Windows and many flavors of Unix (Solaris, Linux, OS X, etc.) systems. Moreover, Unison works across platforms, allowing you to synchronize a Windows laptop with a Unix server, for example.
- Unlike simple mirroring or backup utilities, Unison can deal with updates to both replicas of a distributed directory structure. Updates that do not conflict are propagated automatically. Conflicting updates are detected and displayed.
- Unlike a distributed filesystem, Unison is a user-level program: there is no need to modify the kernel or to have superuser privileges on either host.
- Unison works between any pair of machines connected to the internet, communicating over either a direct socket link or tunneling over an encrypted ssh connection. It is careful with network bandwidth, and runs well over slow links such as PPP connections. Transfers of small updates to large files are optimized using a compression protocol similar to rsync.
- Unison is resilient to failure. It is careful to leave the replicas and its own private structures in a sensible state at all times, even in case of abnormal termination or communication failures.
- Unison has a clear and precise specification.
- Unison is free; full source code is available under the GNU Public License.

- TortoiseCVS at, e.g.,

[https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf)

at 3. “The Concurrent Versioning System (CVS) enables developers to work concurrently by using the copy-modify-merge model. Developers download a working

copy of the source code from a CVS server, make changes, and subsequently upload those changes into the CVS repository. The source code is usually in a directory tree that contains the files that make up the software project. The main reasons for using a source code version control system like CVS are:

- The CVS repository serves as backup
- The CVS repository maintains a complete version history of every file
- The CVS repository facilitates multiple developers working concurrently on the same project.”

TortoiseCVS at, e.g., [https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 3. “It is assumed that the TortoiseCVS software has already been installed on the client machine and that the user has an account on CropForge.”

- Apple iDisk at, e.g., <https://flylib.com/books/en/4.411.1/> at 22. “A local copy of your iDisk is created on your desktop, and Mac OS X attempts to synchronize the local copy with your online copy by copying files from the remote iDisk to the local iDisk”
- Apple iSync at, e.g., <https://flylib.com/books/en/3.338.1/> at 153. “When you iSync your contacts, calendars, and bookmarks, etc., copies of your files are put on the .Mac synchronization server. You can also choose to put copies of your Keychain passwords, Mail accounts, Mail Rules, Mail Signatures, and Smart Mailboxes on the .Mac server. The .Mac copy of your information can then be copied (synced) to another Mac. If you have more than one computer, this is a great way to keep duplicate copies of your information on each computer.”

## **(2) “Server-Based” References**

The Asserted Claims recite features including system implementations with at least three devices, where copies of files may be transferred from one device to a second device through a third device, and/or system configurations where a device implemented for file transfer may be a server system. Nothing about these claim limitations add anything novel or non-obvious over the prior art. To the contrary, this feature was well-known, routine, conventional, and widely available long before the alleged invention. As the attached claim charts demonstrate, this subject matter is disclosed in numerous references. For purposes of these contentions, references whose disclosures are cited in these contentions with respect to this claimed subject matter as reflected in the claim charts (including without limitation the specific references cited below) are hereby labeled the “Server-Based” references.

The implementation of the claimed features would have been obvious to a person of ordinary skill in the art least because doing so would have involved the combination of prior art elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the application of a known technique to a known device ready for improvement to yield predictable results, and/or would have been obvious to try. Moreover, a person of ordinary skill in the art would have readily appreciated the well-known and widespread use of these features and the associated benefits and advantages of their use. Persons of ordinary skill in the art would have been motivated to combine the teachings of these references with any system and/or disclosure that relates to the synchronization of files or other data content, where using a centralized repository such as a server was one well-known design option. Implementing a centralized file repository such as a server for file synchronization between user devices had multiple known advantages and benefits, including simplicity and efficiency in maintaining files in one centralized location for access by multiple client devices,

storage backups and redundancy provided by servers, and accessibility and availability of files to be accessed from a server that could be perpetually accessible as opposed to access only from user devices that would be inaccessible when not active and operational such as in a purely peer-to-peer model. For example, references disclose the following motivations to combine, such as the following as well as the other disclosures that are cited in the prior art claim charts submitted herewith:

- Sigurdsson at, e.g., [0027] “The system **100** includes a Client A **102**, a Client B **104**, and a Server **106**. The Client A **102** and the Client B **104** represent computers which a user may employ to manage digital information, for example by saving or opening files, or accessing web pages. A user may access many computers including, for example, computers at home, computers at work, and mobile computers, such as personal digital assistants (PDAs) and cell phones. The Server **106** can provide a temporary storage location, or “drop box”, for a user’s files, documents, web pages, or the contents of these documents. The drop-box can facilitate sharing this information between the user’s computers by storing the shared content if the target computer is offline. Later, when the target computer comes online, the Server **106** may transmit the content to the target computer.” [0028] “In one implementation, when a user at the Client A **102** performs a file save or an open operation, Client A Content **108** is extracted from the file and is sent to the Server **106**, as indicated by arrow **110**. For example, the user may save a text file on his home computer. The Client A Content **108**, which can include a copy of the user’s file, is sent automatically to the Server **106**. Similarly, when the user employs the Client B **104** to save or open a file stored on the Client B **104**, Client B Content **112** is extracted from the file stored on the Client B **104** and is sent to the Server **106**, as indicated by arrow **114**. For example, the Client B **104** may be the user’s work computer.”

- Salas at, e.g., 3:16-29 “Client workstations **12** are connected to one or more servers **14**. The client workstations **12** may be connected in any physical arrangement such as a star, loop, ring, or bus. The network connecting client workstations **12** and the server **14** may use any physical media, including wireless, provided that the physical media supports the HyperText Transfer Protocol (HTTP). The server **14** stores information relating to a project or a set of projects, referred to as a facility, in a database **20** which may be a flat database, relational database, multidimensional database, or object-oriented database.” 3:44-51 “The server **14** responds to requests for portions of the database **20** made by the client workstations **12** and transfers the requested data objects over the network to the requesting client workstation **12**. The server database **20** stores various tables which contain information about eRooms, members, access controls, and other data objects.” 13:7-26 “As noted above, users may perform work on files and objects locally and upload the modified files and objects to the server **14** for viewing, comment, or further modification by other project team members. The systems and method of the present invention allow users to upload newly created and modified files to a server **14** using an intuitive drag and-drop method. Referring now to FIG. 7, a user creates a new file or modifies a file downloaded from the server **14** (step **702**). It should be understood that this step includes actions such as creating a new version of the file locally so that other users may still check out the “original” copy of the file present on the server **14**. Once the user is finished editing the file, it may be uploaded to the server **14** to allow other users access to it. The user signals that the file should be transmitted to the server **14** by dragging the file onto an eRoom displayed by the browser (step **704**). Dropping the file into the displayed eRoom invokes an ActiveX control or a background daemon process which manages the upload of the file to the server **14**.”

- Venkataraman at, e.g., [0015] “The central device **110** preferably comprises one or more servers, well known to those having ordinary skill in the art, coupled to the network **102**. In particular, the central device **110** is uniquely addressable on the network **102** through the use, for example, of a so-called Uniform Resource Locator (URL) in the case where the network **102** comprises the World Wide Web. As such, direct communication between the central device **110** and, for example, the synchronization device **104** is possible. A synchronization application **112** preferably resides on the central device **110**. Examples of suitable synchronization applications are the services provided by various on-line sites, such as [www.lexacom.com](http://www.lexacom.com), [www.anyday.com](http://www.anyday.com) and [www.fusionone.com](http://www.fusionone.com). In general, each of these synchronization applications provides a subscriber the ability to maintain central data **122** against which local data **124** or remote data **120** is synchronized on a periodic or on-demand basis. Stated another way, a single subscriber account to the synchronization application **112** allows a variety of device access to the central data **122** for that subscriber account. After the synchronization application **112** has synchronized data, and therefore updated the central data **122**, the central data **122** can be sent automatically (or on demand) to the devices (other than the one that initiated the synchronization) that subscribe to maintenance of the central data **122**.”
- Matsubara at, e.g., [0019] “FIG. 1 shows a peer-to-peer (P2P) communications **10** as an illustrative embodiment of the present invention. A management server **12** manages information relating to files that are being shared among peer clients (members) in the P2P network. For example, shared files that are located on the system of a peer client A are listed in the management server. Likewise, shared files that are stored on the system of another peer client B are listed in the management server. In this way, the management server provides

directory services to the members in a P2P network of files to be shared among the peer members. The particular P2P architecture comprising this illustrative embodiment of the invention is more fully described in one or more of the above-referenced related applications.”

[0024] “A peer client (e.g., peer client **22a**) communicates with the notification server **14** to “subscribe” to an information source, in a step **101**. In accordance with the present invention, the term “information source” includes all forms of electronic information (e.g., documents, multimedia files, images, audio files, video files, folders, directories, and so on). An information source can be a virtual shared space where a BBS (bulletin board service) is shared by peer members. An information source can also refer to a virtual shared space where a conference can take place among two or more communicating peers. In particular, the information source can correspond to events occurring during the conference. For example, suppose a group of peer members are participating in a conference over the network (e.g., voice over IP conferencing, chat room type conference, and so on). Events relating to the conference might include the act of someone joining the conference, or the act of someone leaving the conference. Arbitrary actions can be defined as notable events, such as a vote that is taken among the participants in a conference. In accordance with the present invention, a peer client can “subscribe” to such events.” [0030] “In a step **107**, the peer member can view the alert and can decide whether to access the information or not. If she decides she wants to access the information, her peer client software can access the information, in which case a copy **32** of File-X can be obtained. This can be done using typical procedures used in a P2P system. The peer client communicates with the management server **12** and with one of a number of peer clients to download (or otherwise) the information. The information may be:”

- Savage et al., e.g., [0011] “FIG. 1 illustrates an example synchronization system 100. Multiple

synchronization clients 102, 104, and 106 are coupled to a synchronization server 108 via a network 110 or another communications channel, such as FireWire (IEEE 1394), USB (Universal Serial Bus), RS232 (serial channel), etc. It should be understood that a synchronization system may include any number of synchronization clients (e.g., one, two, or more), and that three synchronization clients are shown merely as an example. In addition, a synchronization client may be coupled to multiple synchronization servers.” [0012] “A synchronization client 106 maintains a file system, which is configured to be synchronized with data stored on the synchronization server 108. In this way, for example, the synchronization server 108 can provide backup storage capabilities for the synchronization client 106, such that the synchronization client 106 can recover from a loss of data (e.g., a hard disk crash or an inadvertent file deletion) by restoring the data from the synchronization server 108. Likewise, if more than one synchronization client is coupled to the synchronization server 108, then data on any client can be shared with other clients via the synchronization server 108.”

- Rao at, e.g., 3:14-17 “FIG. 1 illustrates an embodiment 100 of the present invention showing a synchronized database system. A server system 102 and a client system 104 are connected by a communications path 106.” 3:28-32 “The server system 102 and client system 104 may be computer systems or any other device that may store and manipulate data. In one embodiment, the server system 102 may be a server computer and the client system 104 may be a client computer, such as a portable laptop computer.”
- Braginsky at, e.g., [0020] “FIG. 1 is a block diagram of one embodiment of a file management system 100 including a one or more client systems 102 (e.g., laptop computer, desktop computer, personal digital assistant (PDA), mobile phone, media player, etc.) coupled to one

or more server systems 104 via a network 106. The network 106 can be one or more networks having one or more types of topologies, including but not limited to the Internet, intranets, local area networks (LANs), wireless networks, Ethernet, Storage Area Networks (SANs) and the like. The client system 102 generally includes a processor 118 coupled to a display device 114 for presenting a locally cached file system 116 to the user. In some embodiments, the file system 116 manages files stored in a cache data structure on one or more data storage devices 119 (e.g., random access memory (RAM), hard disk, optical disk, portable storage device, etc.). The term “cache” or “cache data structure” is not limited to cache memory but can include any data structure on the computer-readable medium 119 that is accessible to the client system 102.” [0023] “A user can manage files on the client system 102 using the locally cached file system 116. File management includes all the various operations typically associated with files, including but not limited to creating, deleting, opening, editing, moving, copying, renaming, saving, searching and the like. Files can include any known data structures or formats, including but not limited to text files, documents, digital images, video files, web pages, emails, applications, instant messages, audio files, video files, calendar events, music files, or any other data or applications that may reside on one or more computer systems. The file system 116 can have any topology or configuration, including but not limited to the ubiquitous hierarchal directory/folder/file architectures used by WINDOWS, LINUX, MAC OS and UNIX operating systems. The file system 116 is capable of periodically synchronizing with the server system 104 asynchronously to receive updates and other information, as described with respect to FIGS. 2-9.”

- Shappell at, e.g., 3:50-65 “The invention may be implemented in various computing device configurations. For example, the invention may be realized in hand-held devices, mobile

phones, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers and the like, wearable computing or communication devices, and any other device capable of both visual display and direct or indirect communication with another device. The invention may also be practiced in distributed computing environments, where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices. Thus it will be understood that the invention is preferably incorporated into many types of computing environments as suggested above.” 4:53-67 “The device 20 is operable in a networked environment using fixed or transient logical connections to one or more remote computing devices, such as a remote computer 49. The remote computer 49 may be another similar computing device, a server, a router, a network PC, a peer device or other common network node, or any other device type such as any of those mentioned elsewhere herein, and typically includes many or all of the elements described above relative to the computing device 20, although there is no such requirement, and only a memory storage device 50 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.” 5:46-52 “Furthermore, while the type of network 209 is not critical, it is preferably a peer-to-peer network that allows an exchange of information, directly or indirectly, between two or more user machines. Via the network 209, a communication from one user may be transmitted to another user or users (via their respective user devices).”

- Sagar at, e.g., 5:55-63 “Devices 301-303 may be nodes of a mesh that are interconnected to

each other. In some embodiments, the nodes 301-303 may be implemented by a peer-to-peer network as known by those skilled in the art. System 300 illustrates a storage service and enclosure lookup service component 305 that may provide a set of feeds to the nodes of the mesh as well as an enclosure locator service. System 300 may also include a notification service 307 that provides indications of updates or changes in a feed.” 5:64-6:3 “In some embodiments, enclosures reference by items may be synchronized from the storage service. In some embodiments, enclosures referenced by items may be synchronized peer-to-peer. Devices generally synchronize their feeds (and thus the items within) only with the Storage Service. The Storage Service may store all feeds of all items of the synchronized folder.”

- Moromisato at, e.g., 1:34-54 "One of these models is a client-server model in which all collaborators are connected, via the network, to a central server. Information generated by each collaborator is sent over the network to the server that then transmits the information back over the network to each other collaborator. In one version of this system data that is being collaboratively modified may be stored centrally on the server. Then, each collaborator that wants to modify the data sends information to the server to effect a change in the server data. The server modifies its copy of the data and then sends information, synchronously or asynchronously, representing a 'view' of the modified data to all collaborators, so that each collaborator can display the data locally. Alternatively, in a server based system, in addition to the data copy maintained in the server, additional local data copies may be maintained in each collaborating computer. Each collaborator that wants to modify the data sends information to the server to effect a change in the server data. The server modifies its copy of the data and then transmits the command to all collaborators. Each collaborator uses the command to update its local data copy. This local data copy is then displayed locally."

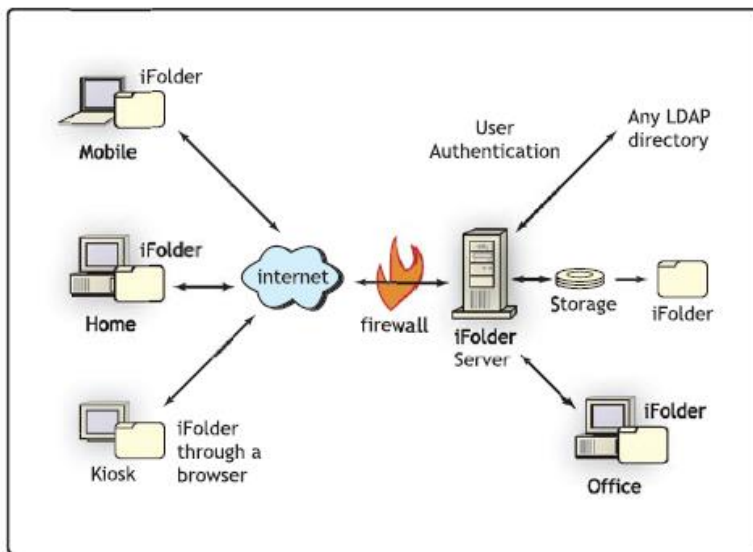
- Aboulhosn at, e.g., 1:59–62 "A method and system for sharing files between a group of computer systems is provided. In one embodiment, the file sharing system allows a group of computer systems to be defined. One computer system of the group may be designated as the "group owner." 2:42–48 "In one embodiment, the group owner maintains a list of the group members. All messages related to metadata updates are routed through the group owner. For example, when a shared file is updated, the file owner sends a message to the group owner containing new metadata for the shared file. The group owner identifies the other members of the group and sends the updated metadata to each member."
- Brown '847 at, e.g., 1:55–63 "The structure of the invention is a client/server application where the server component is a Synchronization File System (SFS) with a functional interface and metadata organization that enables efficient and reliable synchronization by the clients. The client component is a Synchronization Application (SA) that uses the server SFS to synchronize the local client data with the server. Client machines synchronize with each other by synchronizing to a common server account." 2:56–67 "An embodiment of the invention is a client/server application that allows users to synchronize data on multiple machines. The server component is a server Synchronization File System (SFS) with a functional interface and metadata organization that enables efficient and reliable synchronization by the clients. (A glossary of acronyms can be found at the end of this document.) 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."
- Brown '826 at, e.g., 1:57–65 "The structure of the invention is a client/server application where the server component is a Synchronization File System (SFS) with a functional interface and

metadata organization that enables efficient and reliable synchronization by the clients. The client component is a Synchronization Application (SA) that uses the server SFS to synchronize the local client data with the server. Client machines synchronize with each other by synchronizing to a common server account.” 2:60–3:3 “An embodiment of the invention is a client/server application that allows users to synchronize data on multiple machines. The server component is a server Synchronization File System (SFS) with a functional interface and metadata organization that enables efficient and reliable synchronization by the clients. (A glossary of acronyms can be found at the end of this document.) 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.”

- Hesselink at, e.g., 19:46–57 “The connection server 14 a-14 n may serve as a bridging medium between client computer(s) 12 a-12 n and network-enabled device(s) 18 a-18 n (via device control computer(s) 36 a-36 n), between client computer(s) 12 a-12 n and client computer(s) 12 a-12 n, and between device control computer(s) 36 a-36 n and device control computer(s) 36 a-36 n. Data from the device(s) 18 a-18 n that are destined for client computer(s) 12 a-12 n are posted to the connection server 14 a-14 n using an HTTP request as previously described. Receiving the HTTP request, the connection server 14 a-14 n then temporarily stores or buffers any data from the request in its memory (i.e., the sending buffer).”
- iFolder at, e.g., <https://www.novell.com/documentation/ifolder21/pdfdoc/enduser/enduser.pdf> at 7. “Novell iFolder 2.1 is a Net services software solution that allows your files to automatically follow you everywhere—online, offline, all the time—across multiple workstations and the Net. You simply store your files in the iFolder directory on your

workstation. As you work, the iFolder client intelligently tracks and logs updates to your files. It can automatically and transparently synchronize those changes through Internet or network connections with your files on the iFolder server and the various workstations that you use.”

<http://gwise.itwelzel.biz/Novellpdf/!Appnotes/eGuide%C2%B4s/October2001.pdf> at 40.



**Figure 1:** Typical Novell iFolder environment.

- BeInSync at, e.g., [https://web.archive.org/web/20050510121847/http://www.beinsync.com:80/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20050510121847/http://www.beinsync.com:80/help/Beinsync_User_guide.pdf) at 5. “BeInSync can now traverse most firewalls and NAT routers automatically allowing both data synchronization and remote Web access. When a Peer-to-Peer connection is not available, BeInSync can revert to communicating via a relay server” [https://web.archive.org/web/20060315125029/http://www.beinsync.com/notes/release\\_notes\\_1\\_5\\_12.php](https://web.archive.org/web/20060315125029/http://www.beinsync.com/notes/release_notes_1_5_12.php) “I have a firewall running. Will BeInSync work? Yes. BeInSync supports automatic detection of firewalls and NAT devices and will use a relay server in these cases. If you wish to maximize or you are using very restrictive firewalls, you should to configure your firewall or NAT router accordingly.”
- FolderShare at, e.g., <https://www.networkworld.com/article/2312749/syncing-with-folder-share.html> “FolderShare is a folder synchronization product that uses a server-based directory

to manage computers that are to share a "library," ByteTaxi's term for a synchronization connection. The subdirectories to be synchronized at either end will most likely have different names and be located in different subdirectories, but the library name will be the same for all computers. ... The FolderShare server is what makes FolderShare powerful. Once logged on from any PC (even one that isn't running the FolderShare client - what ByteTaxi calls the "satellite" software) you can create, edit or delete libraries and then assign two or more PCs to each library.” <https://web.archive.org/web/20060313171040/https://www.foldershare.com/info/faq/Firewalls.php> “How do I configure the Windows XP and Service Pack 2 firewall and other software firewalls (Zone Alarm & Norton) to work with FolderShare? When FolderShare runs for the first time, Windows might ask you if you want to allow this program to connect to the Internet and act as a server, you need to choose yes for FolderShare to continue to work. If you don't, you can adjust the XP firewall settings under the control panel and manually add FolderShare to the list of allowed programs.”

- SmartCast discloses server-based synchronization including files stored and synchronized through web servers. “All your desktop photos automatically synchronized to the web, and vice versa.” <https://web.archive.org/web/20060829025503/http://www.sharpcast.com/products/photos/>
- GoodSync discloses implementations on a server as one of the synchronized devices. “To synchronize over the Internet:
  - share a folder on a Server computer: in Windows Explorer right-click folder or drive, select Properties, go to Sharing tab, check Share This Folder on the Network
  - make Server computer accept connections on port 139
  - if you have firewalls, you have to allow connections on port 139
  - on a Client computer type IP address of server computer into Address bar of Windows

Explorer to make sure that server computer file system is visible from the client computer

- In GoodSync use the path to server computer that was showing remote files in Windows Explorer

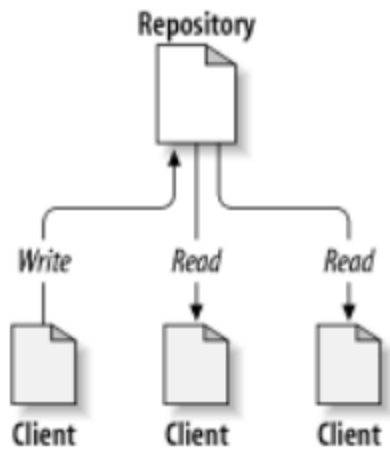
<https://web.archive.org/web/20060427031435/http://www.goodsync.com/networksync.html>

At the workplace, IT staff can use GoodSync to synchronize multiple workstations with a corporate file server. Regardless of the moves/additions/changes made within the company's system, GoodSync ensures that all devices are kept completely up to date.

[https://www.goodsync.com/press-releases/20060206\\_GoodSync-Product-Release.pdf](https://www.goodsync.com/press-releases/20060206_GoodSync-Product-Release.pdf)

- Subversion at, e.g., <https://web.archive.org/web/20071116172448/http://svnbook.red-bean.com/en/1.4/svn-book.pdf> at xxi-xxii. “On one end is a Subversion repository that holds all of your versioned data. On the other end is your Subversion client program, which manages local reflections of portions of that versioned data (called “working copies”). Between these extremes are multiple routes through various Repository Access (RA) layers. Some of these routes go across computer networks and through network servers which then access the repository. Others bypass the network altogether and access the repository directly.” <https://web.archive.org/web/20071116172448/http://svnbook.red-bean.com/en/1.4/svn-book.pdf> at 1. “Subversion is a centralized system for sharing information. At its core is a repository, which is a central store of data. The repository stores information in the form of a filesystem tree—a typical hierarchy of files and directories. Any number of clients connect to the repository, and then read or write to these files. By writing data, a client makes the information available to others; by reading data, the client receives information from others. Figure 1.1, “A typical client/ server system” illustrates this.”

**Figure 1.1. A typical client/server system**



- SmartSync Pro at

<http://web.archive.org/web/20070204114427/http://www.smartsync.com:80/smartsyncpro/key-features.html>. “Common features:

- Attractive and easy-to-use interface.
- Synchronizes your data only when it needs synchronizing.
- Flexible filtering system to include only the files you want.
- Tree-like structure of profiles and profile groups.
- Drag and drop support for folders, profiles and profile groups.
- Full support for unicode file names (Japanese, Korean, Chinese, etc.).
- Activity log.
- Multi-threaded background processing.
- Easy-to-use New Profile Wizard for creating new profiles.
- Color highlighting for new, modified and deleted files and file conflicts.
- Detailed synchronization progress for all operations.
- Resolves file conflicts when the same files have been modified on both sides.
- Adjusts for different time zones, Daylight Saving Time.

- Scheduling support to run profiles whenever you want.
  - Can dial-up before synchronization and hang up afterward.
  - Can run external programs before and after synchronization.
  - Profile access can be restricted to selected users only.
  - Detects locked files and asks to retry to overwrite them.
  - Command line support.
  - Can be run as service under NT4/2000/XP.”
- Nokia Intellisync at [https://www.novell.com/documentation/groupwise\\_mobile\\_3/pdfdoc/admin\\_gde/admin\\_gde.pdf](https://www.novell.com/documentation/groupwise_mobile_3/pdfdoc/admin_gde/admin_gde.pdf). “Nokia Intellisync Mobile Suite is a collection of products designed to help you support your mobile workforce through a single-source solution to meet your mobile computing needs. Nokia Intellisync Mobile Suite consists of both server and client components that facilitate interaction with the server. The server is the data synchronization host to which mobile devices connect. The server Admin Console controls functions such as security, user authentication, communication, logging, and reporting, among others.”
  - Unison File Synchronizer at <https://web.archive.org/web/20071026022954/https://www.cis.upenn.edu/~bcpierce/unison/index.html>. Unison is a file-synchronization tool for Unix and Windows. It allows two replicas of a collection of files and directories to be stored on different hosts (or different disks on the same host), modified separately, and then brought up to date by propagating the changes in each replica to the other.

Unison shares a number of features with tools such as configuration management packages (CVS, PRCS, Subversion, BitKeeper, etc.), distributed filesystems (Coda, etc.), uni-directional mirroring utilities (rsync, etc.), and other synchronizers (Intellisync, Reconcile, etc). However, there are several

points where it differs:

- Unison runs on both Windows and many flavors of Unix (Solaris, Linux, OS X, etc.) systems. Moreover, Unison works across platforms, allowing you to synchronize a Windows laptop with a Unix server, for example.
  - Unlike simple mirroring or backup utilities, Unison can deal with updates to both replicas of a distributed directory structure. Updates that do not conflict are propagated automatically. Conflicting updates are detected and displayed.
  - Unlike a distributed filesystem, Unison is a user-level program: there is no need to modify the kernel or to have superuser privileges on either host.
  - Unison works between any pair of machines connected to the internet, communicating over either a direct socket link or tunneling over an encrypted ssh connection. It is careful with network bandwidth, and runs well over slow links such as PPP connections. Transfers of small updates to large files are optimized using a compression protocol similar to rsync.
  - Unison is resilient to failure. It is careful to leave the replicas and its own private structures in a sensible state at all times, even in case of abnormal termination or communication failures.
  - Unison has a clear and precise specification.
  - Unison is free; full source code is available under the GNU Public License.
- TortoiseCVS at, e.g., [https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 3. “The Concurrent Versioning System (CVS) enables developers to work concurrently by using the copy-modify-merge model. Developers download a working

copy of the source code from a CVS server, make changes, and subsequently upload those changes into the CVS repository. The source code is usually in a directory tree that contains the files that make up the software project. The main reasons for using a source code version control system like CVS are:

- The CVS repository serves as backup
- The CVS repository maintains a complete version history of every file
- The CVS repository facilitates multiple developers working concurrently on the same project.”
- Apple iDisk at, e.g., [https://appleinsider.com/articles/03/06/23/idisk\\_sync\\_in\\_panther](https://appleinsider.com/articles/03/06/23/idisk_sync_in_panther) “The all-new iDisk architecture keeps a copy of your files on your local hard disk, then regularly and automatically synchronizes any changes you make to Apple's servers. Your latest changes are accessible from anywhere, so if you use multiple Macs you'll have the same up-to-date files available directly from the Finder on any of them.”  
<https://www.informit.com/articles/article.aspx?p=401141&seqNum=5> “From the perspective of the end user, iDisk synchronization is entirely transparent—your iDisk appears like any other disk whether working with the contents online or using the local copy. As you make changes to the files, they are noted and automatically uploaded to the .Mac server in the background.”
- Apple iSync at, e.g., <https://flylib.com/books/en/3.338.1/> at 153. “When you iSync your contacts, calendars, and bookmarks, etc., copies of your files are put on the .Mac synchronization server. You can also choose to put copies of your Keychain passwords, Mail accounts, Mail Rules, Mail Signatures, and Smart Mailboxes on the .Mac server. The .Mac copy of your information can then be copied (synced) to another Mac. If you have more than

one computer, this is a great way to keep duplicate copies of your information on each computer. ... Information that you choose to sync (see the following page) gets copied to the .Mac server. Then other Macs can sign in and sync to the same server, a procedure called Mac-to-Mac syncing”

### **(3) “Device Application” References**

The Asserted Claims recite features including an “application” executed at each of multiple devices in a system, such as for example a “first application” executed at a first device, a “second application” at a second device, and a “third application” at a third device. Nothing about these claim limitations add anything novel or non-obvious over the prior art. To the contrary, this feature was well-known, routine, conventional, and widely available long before the alleged invention. As the attached claim charts demonstrate, this subject matter is disclosed in numerous references. For purposes of these contentions, references whose disclosures are cited in these contentions with respect to this claimed subject matter as reflected in the claim charts (including without limitation the specific references cited below) are hereby labeled the “Device Application” references.

The implementation of the claimed features would have been obvious to a person of ordinary skill in the art least because doing so would have involved the combination of prior art elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the application of a known technique to a known device ready for improvement to yield predictable results, and/or would have been obvious to try. Moreover, a person of ordinary skill in the art would have readily appreciated the well-known and widespread use of these features and the associated benefits and advantages of their use. Persons of ordinary skill in the art would have been motivated to combine the teachings of these references with any system and/or disclosure that relates to applications for the

synchronization of files or other data content because implementation in an application, as opposed to firmware or hardcoded, had multiple known advantages and benefits, including the ability to more quickly modify and update the application and ease in porting the application to different hardware architectures. For example, references disclose the following motivations to combine, such as the following as well as the other disclosures that are cited in the prior art claim charts submitted herewith:

- Salas at, e.g., 12:47-59 “Once the file has been downloaded, or if the file was already present in local mass storage, the watcher launches the application used to edit the file (step **604**). The indicated application may be determined using the Object Linking Embedding standard (OLE), the file suffix (three characters in a DOS-based system), or the server **14** may store file metadata which is transmitted together with the file and indicates which application should be used to open and edit the file. If the server **14** stores file metadata, a list of alternate applications may be stored, either on the server **14** or the client workstation **12**, So that if a client workstation does not have access to a first application, other applications are specified which may be used.”
- Kaasten at, e.g., 9:29-32 “In the user mode of the computing device **102** runs one or more application programs **400**. Under the direction of a user, these application programs **400** change synchronized data objects.” 9:38-47 “When a change is directed to a full-copy synchronized data object, that change is implemented on the local data object. This is shown by the dataflow connecting the application program **400**, the input/output manager **408**, the file system drivers **410**, the local storage drivers **412**, the file system A **112**, and finally the target data object within the file system A **112**. Via dataflow **406**, the data synchronization service **402** notices the change. As appropriate, the service **402** creates a change notification to send to the computing devices that host a counterpart to the changed data object.”

- Savage at, e.g., [0013] “The synchronization client 106 is shown as including a scanning module 112, which maintains a synchronization state datastore 114. The synchronization state datastore 114 records synchronization state of files of the synchronization client 106 and the synchronization server 108. Using the synchronization state datastore 114, the synchronization client 106 can determine which files to upload from the synchronization server 118 in an updating stage of synchronization.” [0031] “In one implementation, the scanning module 112 detects that a file on the synchronization client has changed, generates and schedules a synchronization report for that file, and transfers the synchronization report to the synchronization server 108 -at the appropriate time. In one implementation, the transfer may be accomplished by a transmission from the client to the server over a communications network. However, other implementations may transfer the synchronization report via an intermediary, such as a shared storage location, or by having the report read from the client by the server.” [0032] “Upon receipt of a synchronization report for a file, the synchronization server 108 checks to determine the state of the file within its own files system. If the synchronization server 108 needs to receive the file from the client in order to synchronize with the client, the synchronization server 108 request that the client upload the file to the server. An update module 118 uploads the requested file to the synchronization server 108 (see synchronizing files 120).” [0033] “It should be understood that, in one implementation, the synchronization server 108 also includes a scanning module, which detects changes within the synchronization server's file system and issues the server's synchronization report to the appropriate synchronization client(s). Also, in the same implementation or in an alternative implementation, the synchronization server 108 includes an updating module, which responds to requests from synchronization clients for synchronizing files from the synchronization

server 108.” [0056] “In an exemplary implementation, scanning modules, update modules, and other modules may be incorporated as part of the operating system, application programs, or other program modules. File system hierarchies, scan queues, folder queues, event queues, schedule times, file objects, folder objects, and other data may be stored as program data.”

- Rao at, e.g., 3:5-13 “When the invention is embodied in the general context of computer-executable instructions, the embodiment may comprise program modules, executed by one or more systems, computers, or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Typically, the functionality of the program modules may be combined or distributed as desired in various embodiments.”
- Braginsky at, e.g., [0020] “FIG. 1 is a block diagram of one embodiment of a file management system 100 including a one or more client systems 102 (e.g., laptop computer, desktop computer, personal digital assistant (PDA), mobile phone, media player, etc.) coupled to one or more server systems 104 via a network 106. The network 106 can be one or more networks having one or more types of topologies, including but not limited to the Internet, intranets, local area networks (LANs), wireless networks, Ethernet, Storage Area Networks (SANs) and the like. The client system 102 generally includes a processor 118 coupled to a display device 114 for presenting a locally cached file system 116 to the user. In some embodiments, the file system 116 manages files stored in a cache data structure on one or more data storage devices 119 (e.g., random access memory (RAM), hard disk, optical disk, portable storage device, etc.). The term “cache” or “cache data structure” is not limited to cache memory but can include any data structure on the computer-readable medium 119 that is accessible to the client system 102.” [0021] “The server system 104 (e.g., a file server) generally includes a processor 108

coupled to one or more computer-readable mediums. The computer-readable medium 110 includes a file system 112, herein sometimes called a “remote file system” because it is remotely located relative to the client system 102. In some embodiments, the locally cached file system 116 has a same or similar file structure as the remote file system 112.” [0024] “In some embodiments, the file system 116 is implemented as a “local server” in the client system 102. In such an embodiment, the file system 116 can be part of an application that is installed on the client system 102. In some embodiments, the file system 116 is installed on the client system 102 as a virtual hard drive that is integrated into the native file system to provide all the functionality of a local physical hard drive.” [0084] “Each of the above identified modules and applications in client device 800 corresponds to a set of instructions for performing one or more functions described above. These modules (i.e., sets of instructions) need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise re-arranged in various embodiments. In some embodiments, computer-readable medium 808 may store a subset of the modules and data structures identified above. Furthermore, computer-readable medium 808 may store additional modules and data structures not described above.” [0086] “Each of the above identified modules and applications in server system 102 corresponds to a set of instructions for performing one or more functions described above. These modules (i.e., sets of instructions) need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise re-arranged in various embodiments. In some embodiments, memory 1008 may store a subset of the modules and data structures identified above. Furthermore, memory 1008 may store additional modules and data structures not described above.”

- Shappell at, e.g., 2:1-11 “A shared space is presented as an entry in the file system such that user selectable management tasks and items may be activated. Such action includes various operations that are performed with respect to shared files within the space. Upon creation of the shared space, other users can access and/or transmit files, and perform other file management operations such as drag and drop shared files, move, copy and other file system management tasks. In addition, shared files and directories may be linked into the shared space. This provides a clear sense of user operations that are available for shared files, to thereby create an ease of use of group spaces.” 2:17-32 “Once created, users can drag and drop files into the shared space itself as well as perform other file system operations with respect to the shared files and directories. In accordance with the invention, the shared files include associated metadata that is propagated to other members of the shared space. Members who connect to the group later are still able to access such metadata and thereby continue to operate within the shared space. Upon receipt of the metadata, the infrastructure based on user settings can now handle the shared files as desired, such as by copying the shared files locally or not. In the case where a file is to be copied locally, a connection is established with the source of the file metadata, and the shared file is transferred from the source to the local computer. The process is secure in that only a valid member of the group may copy the shared file to its space.” 3:42-49 “Although it is not required for practicing the invention, the invention is described as it is implemented by computer-executable instructions, such as program modules, that are executed by a computing device. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types.”
- Moromisato at, e.g., 7:25–28 “The file synchronizer 308 is a software program that insures that

any changes made to a file in the Windows shell file system 302 are also made to copies of that file that are stored in the collaborative system and vice versa.” 11:44–51 “Alternatively, if a folder that has synchronization is opened and the collaborative system is not running, then a task pane can be displayed that prompts the user to run the collaborative application. Similarly, if the collaborative system is running, but the account that enabled synchronization is not logged-in, then another task pane is displayed that prompts the user to log in (unless the account has no password and the log-in process can be automated.)”

- Brown ‘847 at, e.g., 2:56–67 “An embodiment of the invention is a client/server application that allows users to synchronize data on multiple machines. The server component is a server Synchronization File System (SFS) with a functional interface and metadata organization that enables efficient and reliable synchronization by the clients. (A glossary of acronyms can be found at the end of this document.) 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.” 4:19–28 “The term client refers to various kinds of machines that can be connected to a network. Client 130 represents an ordinary desktop computer, as might be present at a user's place of work or home. Portable computer 135 represents a notebook or laptop computer, as a user might take with him to a hotel room on business. To the extent that the client software can be installed on other types of devices, these other devices can be clients. For example, a user might use a personal digital assistant (PDA) to synchronize with server 105, if the PDA can install the client SA.”
- Brown ‘826 at, e.g., 2:60–3:3 “An embodiment of the invention is a client/server application that allows users to synchronize data on multiple machines. The server component is a server

Synchronization File System (SFS) with a functional interface and metadata organization that enables efficient and reliable synchronization by the clients. (A glossary of acronyms can be found at the end of this document.) 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.”

4:19–28 “The term client refers to various kinds of machines that can be connected to a network. Client 130 represents an ordinary desktop computer, as might be present at a user’s place of work or home. Portable computer 135 represents a notebook or laptop computer, as a user might take with him to a hotel room on business. To the extent that the client software can be installed on other types of devices, these other devices can be clients. For example, a user might use a personal digital assistant (PDA) to synchronize with server 105, if the PDA can install the client SA.”

- iFolder at, e.g., <https://www.novell.com/documentation/ifolder1/pdfdoc/ifolder/ifolder.pdf> at 10. “iFolder server software: ... Once you have installed the iFolder server software on your server, your users can install the iFolder client in order to access their iFolder files, and you can access the Server Management Console and the default iFolder Web site to manage your iFolder user accounts. ... iFolder client software: ... The iFolder client must be installed on every workstation that you will use to access your iFolder files.”
- BeInSync at, e.g., [https://web.archive.org/web/20060330064008/http://www.beinsync.com/bis\\_datasheet.pdf](https://web.archive.org/web/20060330064008/http://www.beinsync.com/bis_datasheet.pdf) at 2. “Simply install BeInSync on each of your PCs, choose what you’d like to synchronize (your files, desktop, emails, contacts or Internet favorites) and you’re ready to go. From now on, any change made on one PC will automatically be reflected on your other PCs. And what’s more, you can easily share data with others. ... Any file added to a share

will automatically be synchronized among all the share's participants.” <https://web.archive.org/web/20060112125007/http://www.pcmag.com/article2/0,1895,1645668,00.asp> “The service is surprisingly easy to set up and use. First, you install a small app on one system, check the items you'd like to synchronize, and sign up for a user name and password. Then, you simply set up the same app on one or two other machines and log on with the same user name and password. From then on, the service will automatically synchronize all your chosen items.”

- FolderShare at, e.g., <https://web.archive.org/web/20060313171107/https://www.foldershare.com/download/files/FolderShareGuide.pdf> at 2. “To get started, you will need to install the FolderShare Satellite on each computer you want to run FolderShare. During installation you will create an account. Once the Satellite is installed and running click the FolderShare icon in your system tray and select ‘My FolderShare’ to login to the FolderShare website to start using the service.” <https://web.archive.org/web/20040528145753/http://www.foldershare.com/library/info/aboutReplication.php> “Running the FolderShare application on multiple computers allows you to easily keep an identical set of files on all your computers. Upload a file on your home computer and it will instantly be placed on your office computer. Make a change to a file on your laptop and your office computer will automatically change its version of the file.”
- SharpCast discloses desktop and mobile device application software and web server software to implement synchronization.
- GoodSync discloses application software on each device, including devices implemented as servers.
- Subversion at, e.g., <https://web.archive.org/web/20071124042709/http://www.eclipse.org/subversive/documentati>

[on/gettingStarted/subversion/svn\\_intro.php](http://on/gettingStarted/subversion/svn_intro.php) “Subversion® is an open-source version control system. Subversion® allows developers to share there projects on the repositories, where they are stored afterwards. Repository is much similar to a file server, except the fact, that it not only stores the copy of the file system, but its previous states and changing history. Subversion® access its repositories using network, so it provides a probability for a person to work over some shared files and watch for every possible changes made by other developers.”

<https://web.archive.org/web/20071116172448/http://svnbook.red-bean.com/en/1.4/svn-book.pdf> at xviii-xix. “Subversion is a free/open-source version control system. That is, Subversion manages files and directories, and the changes made to them, over time. This allows you to recover older versions of your data, or examine the history of how your data changed. In this regard, many people think of a version control system as a sort of ‘time machine’. Subversion can operate across networks, which allows it to be used by people on different computers. At some level, the ability for various people to modify and manage the same set of data from their respective locations fosters collaboration. Progress can occur more quickly without a single conduit through which all modifications must occur. And because the work is versioned, you need not fear that quality is the trade-off for losing that conduit—if some incorrect change is made to the data, just undo that change. Some version control systems are also software configuration management (SCM) systems. These systems are specifically tailored to manage trees of source code, and have many features that are specific to software development—such as natively understanding programming languages, or supplying tools for building software. Subversion, however, is not one of these systems. It is a general system that can be used to manage any collection of files. For you, those files might be source code—for others, anything from grocery shopping lists to digital video mixdowns and beyond.”

- TortoiseCVS at, e.g., [https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 3. “It is assumed that the TortoiseCVS software has already been installed on the client machine and that the user has an account on CropForge.”
- Apple iDisk at, e.g., <https://www.peachpit.com/articles/article.aspx?p=420902> at 1. “iDisk is a .Mac utility that lets you store your files on servers maintained by Apple. These servers store files for thousands of .Mac subscribers. When one of those subscribers connects to his or her iDisk using a Mac, the iDisk appears on the Desktop as if it were a network volume on a file server sitting in a closet down the hall. Users can treat the iDisk almost as if it were a hard drive connected directly to their Macs.” <https://www.peachpit.com/articles/article.aspx?p=420902> at 2. “If you’re using Mac OS X, there are a few different ways of connecting to your iDisk, depending on what version of Mac OS X you’re running. If you’re using Mac OS 9, you’ll need additional software to connect to your iDisk ... Mac OS 9 doesn’t have WebDAV built in, so it can’t by itself connect to an iDisk. Fortunately, there’s a handy bit of Mac OS 9 software, called Goliath, that does the trick. ... Windows XP uses an Apple-created utility to connect to an iDisk; you can download it from your .Mac account.”
- Apple iSync at, e.g., <https://flylib.com/books/en/3.343.1/> at 82. “As Bluetooth technology spread from Europe and Asia to North America, more and more people ‘Mac users’ found themselves with Bluetooth-enabled cell phones and PDAs, but didn’t have a way to synchronize the devices with their Mac. Apple realized that something needed to be done to help Mac users share the information on their Mac with their devices, and the result was iSync (<http://www.apple.com/isync>). iSync is a small application that synchronizes contacts,

calendars, and bookmarks from your Mac with your .Mac account and other devices that are part of your digital life.”

#### **(4) “Me-To-Me” References**

The Asserted Claims recite features including a system implementation where files may be transfer between multiple devices that are each associated with a user. Nothing about these claim limitations add anything novel or non-obvious over the prior art. To the contrary, this feature was well-known, routine, conventional, and widely available long before the alleged invention. As the attached claim charts demonstrate, this subject matter is disclosed in numerous references. For purposes of these contentions, references whose disclosures are cited in these contentions with respect to this claimed subject matter as reflected in the claim charts (including without limitation the specific references cited below) are hereby labeled the “Me-To-Me” references.

The implementation of the claimed features would have been obvious to a person of ordinary skill in the art least because doing so would have involved the combination of prior art elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the application of a known technique to a known device ready for improvement to yield predictable results, and/or would have been obvious to try. Moreover, a person of ordinary skill in the art would have readily appreciated the well-known and widespread use of these features and the associated benefits and advantages of their use. Persons of ordinary skill in the art would have been motivated to combine the teachings of these references with any system and/or disclosure that relates to the synchronization of files or other data content, where it was well-known that synchronizing files between multiple devices of the same user (for example, a given user’s work computer and home computer, or desktop and laptop computers) was desirable and advantageous in order to permit a given user to work on a

given file while maintaining the current version of the file across multiple devices without needing to manually transfer copies of the file across devices. Synchronizing files between multiple devices of the same user can also lead to minimize the use of wide area network (WAN) bandwidth when those devices are located on the same local area network (LAN). For example, references disclose the following motivations to combine, such as the following as well as the other disclosures that are cited in the prior art claim charts submitted herewith:

- Sigurdsson at, e.g., [0028] “In one implementation, when a user at the Client A **102** performs a file save or an open operation, Client A Content **108** is extracted from the file and is sent to the Server **106**, as indicated by arrow **110**. For example, the user may save a text file on his home computer. The Client A Content **108**, which can include a copy of the user's file, is sent automatically to the Server **106**. Similarly, when the user employs the Client B **104** to save or open a file stored on the Client B **104**, Client B Content **112** is extracted from the file stored on the Client B **104** and is sent to the Server **106**, as indicated by arrow **114**. For example, the Client B **104** may be the user's work computer. Because content from the Client A **102** (e.g., the user's home computer) has been transferred to the user's work computer, the user has access to files not previously resident on his work computer. Similarly, the user's home computer now has files previously resident only on the user's work computer.” [0045] “The system **100** includes the Client A **102**, the Client B **104**, and the Server **106**. The Client A **102** and the Client B **104** each represent a computer used by a user including, for example, a personal computer at home, a computers at work, and portable computers, such as a laptop computer, a PDA, and a cell phone. The Server **106** can act as a temporary storage location that facilitates synchronization between the user's clients. The Client A **102** posts the Client A Content **108** to the Server **106** after content is changed on the Client A **102**. Posting of the Client A's

Content **108** occurs, for example, whenever the user saves a file or document, or when he views a webpage.”

- Venkataraman at, e.g., [0002] “Electronic, mobile personal digital assistants (PDAs) and the like (hereinafter referred to as remote devices) are well known in the art. In essence, such devices allow a user to use many of the functions traditionally found in more conventional computing systems, such as desktop personal computers, by allowing the user of the remote device to store, manipulate and access various types of data. For example, users of remote devices may work with personal or business contact information, calendar data, email and a variety of other data types through their remote devices.”
- Kaasten at, e.g., 2:14-26 “Selected data objects (files and folders) are copied onto selected computing devices. A data synchronization service running on each selected device monitors the selected data objects for changes, in some embodiments by intercepting calls to the device's file system. When a change is detected, the data synchronization service sends a notification of the change to the other devices so that they can update their copies of the data object. Thus, the copies of the data object are kept in synchrony on all of the selected devices. A user can access a data object from any of the selected devices, knowing that he will retrieve the latest version of the data object. If one device is temporarily not available, then the latest version can still be accessed from another device.”
- Braginsky at, e.g., [0003] “Today, most people's computer files (e.g., documents, photos, songs, movies, etc.) and other items (e.g., calendar events, emails, tasks, etc.) exist on one or more personal physical devices (e.g., laptops, desktops, PDAs, mobile phones, etc.). This hinders the management and production of information in a number of ways. For example, access to files is typically not ubiquitous across multiple physical devices. It is generally

difficult for a user to walk into an Internet café or grab a friend's computer and view or edit a draft document that was started on a different computer. File synchronization is also difficult if a user works on the same document on multiple devices. To ensure the most current version of a document is available, a user may have to repeatedly email modified versions of the document to himself, or remember to consistently store a copy of the current version on a portable drive (e.g., a USB drive), which are techniques that are prone to errors. This problem is compounded during collaboration where multiple document versions may have to be managed via email or merged manually. Finally, if hard disks fail or a laptop is stolen, valuable information may be lost if the user lacks the know-how or the discipline to back-up data.”

[0020] “FIG. 1 is a block diagram of one embodiment of a file management system 100 including a one or more client systems 102 (e.g., laptop computer, desktop computer, personal digital assistant (PDA), mobile phone, media player, etc.) coupled to one or more server systems 104 via a network 106.”

- Shappell at, e.g., 5:35-43 “FIG. 2 is a schematic diagram of a network environment within which an embodiment of the invention may be implemented. In particular, a plurality of user computing devices 201, 203, 205, and 207 are illustrated as being communicably linked via a network 209. Each user device will typically be used by or associated with a different user. The exact number of user devices so linked is not important for the present invention, although a network communication will typically involve at least two users.”
- Brown ‘847 at, e.g., 1:24–46 “But as computers have become more affordable, people find themselves working with several machines. It is increasingly common for people to find themselves working on one machine at the office, a second machine at home, and having use of a portable computer when they need to have computer access while traveling. ... But with

the increasing number of machines a person might find himself using comes an added complexity. Since a person typically accesses the same files from the various computers, the user needs to be certain that the files he is accessing are current. ... And if the user forgets to bring the files with him as he moves around, or forgets to move the latest versions of the files off the computer he most recently used, the user can find himself with several versions of the files, each of which contain desired portions.” 4:19–28 “The term client refers to various kinds of machines that can be connected to a network. Client 130 represents an ordinary desktop computer, as might be present at a user's place of work or home. Portable computer 135 represents a notebook or laptop computer, as a user might take with him to a hotel room on business. To the extent that the client software can be installed on other types of devices, these other devices can be clients. For example, a user might use a personal digital assistant (PDA) to synchronize with server 105, if the PDA can install the client SA.”

- Brown ‘826 at, e.g., 1:24–47 “But as computers have become more affordable, people find themselves working with several machines. It is increasingly common for people to find themselves working on one machine at the office, a second machine at home, and having use of a portable computer when they need to have computer access while traveling. ... But with the increasing number of machines a person might find himself using comes an added complexity. Since a person typically accesses the same files from the various computers, the user needs to be certain that the files he is accessing are current. ... And if the user forgets to bring the files with him as he moves around, or forgets to move the latest versions of the files off the computer he most recently used, the user can find himself with several versions of the files, each of which contain desired portions.” 4:19–28 “The term client refers to various kinds of machines that can be connected to a network. Client 130 represents an ordinary desktop

computer, as might be present at a user's place of work or home. Portable computer 135 represents a notebook or laptop computer, as a user might take with him to a hotel room on business. To the extent that the client software can be installed on other types of devices, these other devices can be clients. For example, a user might use a personal digital assistant (PDA) to synchronize with server 105, if the PDA can install the client SA.”

- Hesselink at, e.g., 48:08–21 “The present system may also be optionally employed to perform synchronization of files among a user's multiple computers. Referring back to the operations described above, the system is designed to automatically update at least the "master copy" of a file at a remote location (i.e., the file at the location of the device module from where the file was accessed by the user module) as the user makes changes to the file locally on the computer associated with the user module. However, the user may be provided with alternative synchronization options by the present system. For example, the user may choose to always synchronize all copies of a file, wherein, whenever the user makes a change to that file locally, at any location, then all of the devices which subscribe to that file are automatically updated immediately.” 39:45–51 “As noted above, when data is updated, such as when a file is edited, for example, the system may be configured to automatically update all or some designated subset of computers on the network as soon as the update has been performed at one location. Computers that are offline at such a time will be automatically updated as soon as those computers go online again.”
- iFolder at, e.g., <https://www.novell.com/documentation/ifolder21/pdfdoc/enduser/enduser.pdf> at 7. “As you work, the iFolder client intelligently tracks and logs updates to your files. It can automatically and transparently synchronize those changes through Internet or network connections with your files on the iFolder server and the various workstations that you use.”

<https://www.novell.com/documentation/ifolder21/pdfdoc/qsifolder/qsifolder.pdf> at 1. “Novell iFolder is a Net services software solution that allows your files to automatically follow you everywhere—online, offline, all the time—across multiple workstations and the Net. It no longer matters where you are; iFolder securely synchronizes any local files that you've placed in your iFolder directory across all your workstations and the iFolder server. All you need is an active Internet or network connection and the iFolder client or a Web browser. Novell iFolder provides you with the flexibility of working on your local files, online or offline, from your different computers at work, school, home—even from a computer halfway around the world.”

- BeInSync at, e.g., [https://web.archive.org/web/20060204061917/http://www.beinsync.com/what\\_is\\_beinsync.php](https://web.archive.org/web/20060204061917/http://www.beinsync.com/what_is_beinsync.php) “BeInSync enables you to seamlessly and securely access and use your latest data anytime, anywhere - at the office, from home or on the go. Using innovative Secure Peer-to-Peer Technology that automatically synchronizes all your computers (home PC, office PC, laptop), BeInSync ensures that wherever you go, your data will follow.” [https://web.archive.org/web/20060330064008/http://www.beinsync.com/bis\\_datasheet.pdf](https://web.archive.org/web/20060330064008/http://www.beinsync.com/bis_datasheet.pdf) at 2. “Simply install BeInSync on each of your PCs, choose what you'd like to synchronize (your files, desktop, emails, contacts or Internet favorites) and you're ready to go. From now on, any change made on one PC will automatically be reflected on your other PCs. And what's more, you can easily share data with others. ... Any file added to a share will automatically be synchronized among all the share's participants.”
- FolderShare at, e.g., <https://web.archive.org/web/20030801073433/http://www.foldershare.com/index.php> “Run FolderShare on multiple personal computers (like your home and office machines) and you will have a powerful personal file system at your disposal. FolderShare can

automatically create an identical set of files on all of your computers. Make a file at home and it will instantly be copied to your office computer. Edit a file on your laptop and your office computer will automatically update its version of the file. If you are on the road, use FolderShare to view and download files from your office computer via any web browser.”

<https://web.archive.org/web/20060313170513/https://www.foldershare.com/info/aboutFolderShare.php> “Synchronize all your devices - Retrieve work files at home or access photos at work. With your devices in sync, you no longer have to be frustrated that your information is on another computer.”

- SharpCast discloses synchronization between a user’s devices, such as PCs and mobile phones. “Sharpcast services automatically synchronize your PCs, mobile phone and the web, to keep all our media and information always up to date and instantly accessible on whatever device you’re using and whether you’re online or not.”

<https://web.archive.org/web/20060720195133/http://sharpcast.com/products/photos/experience/>

- GoodSync discloses synchronization between a user’s devices, such as home and office computers or laptop and desktop computers. “GoodSync can be used to synchronize data between your desktop PC and laptop, home and office computers, computer and removable devices (USB Key, Flash Drive, CDRW disc), over a local network or the Internet.”

<https://web.archive.org/web/20060417232608/http://www.goodsync.com/index.html>

- SmartSync Pro at <http://web.archive.org/web/20070204114427/http://www.smartsync.com:80/smartsyncpro/key-features.html>. “Common features:

- Attractive and easy-to-use interface.
- Synchronizes your data only when it needs synchronizing.

- Flexible filtering system to include only the files you want.
  - Tree-like structure of profiles and profile groups.
  - Drag and drop support for folders, profiles and profile groups.
  - Full support for unicode file names (Japanese, Korean, Chinese, etc.).
  - Activity log.
  - Multi-threaded background processing.
  - Easy-to-use New Profile Wizard for creating new profiles.
  - Color highlighting for new, modified and deleted files and file conflicts.
  - Detailed synchronization progress for all operations.
  - Resolves file conflicts when the same files have been modified on both sides.
  - Adjusts for different time zones, Daylight Saving Time.
  - Scheduling support to run profiles whenever you want.
  - Can dial-up before synchronization and hang up afterward.
  - Can run external programs before and after synchronization.
  - Profile access can be restricted to selected users only.
  - Detects locked files and asks to retry to overwrite them.
  - Command line support.
  - Can be run as service under NT4/2000/XP.”
- Nokia Intellisync at [https://www.novell.com/documentation/groupwise\\_mobile\\_3/pdfdoc/admin\\_gde/admin\\_gde.pdf](https://www.novell.com/documentation/groupwise_mobile_3/pdfdoc/admin_gde/admin_gde.pdf). “Nokia Intellisync Mobile Suite is a collection of products designed to help you support your mobile workforce through a single-source solution to meet your mobile computing needs. Nokia Intellisync Mobile Suite consists of both server and client components that facilitate

interaction with the server. The server is the data synchronization host to which mobile devices connect. The server Admin Console controls functions such as security, user authentication, communication, logging, and reporting, among others.”

- Unison File Synchronizer at <https://web.archive.org/web/20071026022954/https://www.cis.upenn.edu/~bcpierce/unison/index.html>

. Unison is a file-synchronization tool for Unix and Windows. It allows two replicas of a collection of files and directories to be stored on different hosts (or different disks on the same host), modified separately, and then brought up to date by propagating the changes in each replica to the other.

Unison shares a number of features with tools such as configuration management packages (CVS, PRCS, Subversion, BitKeeper, etc.), distributed filesystems (Coda, etc.), uni-directional mirroring utilities (rsync, etc.), and other synchronizers (Intellisync, Reconcile, etc). However, there are several points where it differs:

- Unison runs on both Windows and many flavors of Unix (Solaris, Linux, OS X, etc.) systems. Moreover, Unison works across platforms, allowing you to synchronize a Windows laptop with a Unix server, for example.
- Unlike simple mirroring or backup utilities, Unison can deal with updates to both replicas of a distributed directory structure. Updates that do not conflict are propagated automatically. Conflicting updates are detected and displayed.
- Unlike a distributed filesystem, Unison is a user-level program: there is no need to modify the kernel or to have superuser privileges on either host.
- Unison works between any pair of machines connected to the internet, communicating over either a direct socket link or tunneling over an encrypted ssh connection. It is careful with network bandwidth, and runs well over slow links such as PPP

connections. Transfers of small updates to large files are optimized using a compression protocol similar to rsync.

- Unison is resilient to failure. It is careful to leave the replicas and its own private structures in a sensible state at all times, even in case of abnormal termination or communication failures.
  - Unison has a clear and precise specification.
  - Unison is free; full source code is available under the GNU Public License.
- Apple iDisk at, e.g., <https://www.peachpit.com/articles/article.aspx?p=24288> at 1. “The applications for iDisk are tremendous. Since you can access iDisk from any computer with an Internet connection, you can use it to store files that you want to access while you are away from home or work.” [https://appleinsider.com/articles/03/06/23/idisk\\_sync\\_in\\_panther](https://appleinsider.com/articles/03/06/23/idisk_sync_in_panther). “The all-new iDisk architecture keeps a copy of your files on your local hard disk, then regularly and automatically synchronizes any changes you make to Apple's servers. Your latest changes are accessible from anywhere, so if you use multiple Macs you'll have the same up-to-date files available directly from the Finder on any of them.”
  - Apple iSync at, e.g., <https://flylib.com/books/en/3.338.1/> at 154. “Many of us work on two or more computers, perhaps one at the office and one at home, or you may have multiple computers in one office or home. To synchronize all the computers so you have the same information everywhere you go, perform a Mac-to-Mac synchronization. This procedure is exactly the same as a Mac-to-.Mac sync, except that you sign in to the same .Mac server with a different computer (a Mac of course).” <https://flylib.com/books/en/3.343.1/> at 82. “As Bluetooth technology spread from Europe and Asia to North America, more and more people ‘Mac users’ found themselves with Bluetooth-enabled cell phones and PDAs, but didn't have

a way to synchronize the devices with their Mac. Apple realized that something needed to be done to help Mac users share the information on their Mac with their devices, and the result was iSync (<http://www.apple.com/isync>). iSync is a small application that synchronizes contacts, calendars, and bookmarks from your Mac with your .Mac account and other devices that are part of your digital life.”

#### **(5) “Transfer When Modified” References**

The Asserted Claims recite features including the automatic transfer of a file when the file is modified and/or saved by a user. Nothing about these claim limitations add anything novel or non-obvious over the prior art. To the contrary, this feature was well-known, routine, conventional, and widely available long before the alleged invention. As the attached claim charts demonstrate, this subject matter is disclosed in numerous references. For purposes of these contentions, references whose disclosures are cited in these contentions with respect to this claimed subject matter as reflected in the claim charts (including without limitation the specific references cited below) are hereby labeled the “Transfer When Modified” references.

The implementation of the claimed features would have been obvious to a person of ordinary skill in the art least because doing so would have involved the combination of prior art elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the application of a known technique to a known device ready for improvement to yield predictable results, and/or would have been obvious to try. Moreover, a person of ordinary skill in the art would have readily appreciated the well-known and widespread use of these features and the associated benefits and advantages of their use. Persons of ordinary skill in the art would have been motivated to combine the teachings of these references with any system and/or disclosure that relates to the synchronization of files or

other data content. Synchronizing files when the files are modified was one of the well-known design options for a file synchronization system. Well-known benefits in the prior art of this design were to help ensure timely updates and currency of files across multiple devices, so that changes to a given file cause the file to be promptly synchronized across devices as opposed to delaying so that copies of early versions of the file may become substantially outdated over time. Synchronizing files when the files are modified also reduces the likelihood of conflicts arising from modifying the same file on different devices. For example, references disclose the following motivations to combine, such as the following as well as the other disclosures that are cited in the prior art claim charts submitted herewith:

- Sigurdsson at, e.g., [0026] “Exchange of content between the computer devices can occur automatically without a user specifying that the content should be sent to another computer device. For example, a user may edit a word document file at his home computer. When the user saves the document, the documents content is automatically extracted and transmitted to the user's computer at work.” [0028] “In one implementation, when a user at the Client A **102** performs a file save or an open operation, Client A Content **108** is extracted from the file and is sent to the Server **106**, as indicated by arrow **110**. For example, the user may save a text file on his home computer. The Client A Content **108**, which can include a copy of the user's file, is sent automatically to the Server **106**. Similarly, when the user employs the Client B **104** to save or open a file stored on the Client B **104**, Client B Content **112** is extracted from the file stored on the Client B **104** and is sent to the Server **106**, as indicated by arrow **114**. For example, the Client B **104** may be the user's work computer.”
- Salas at, e.g., 11:12-41 “Because multiple users may concurrently, and even simultaneously, perform work on a project, the page builder must ensure that the objects from the local database

and locally stored files are not stale before inserting them into the eRoom template (step 506). Put another way, the client workstation's local project database 20 must be synchronized with the server's project database 20 to ensure data coherency. Synchronization may be done in at least four different ways: (1) periodically in the foreground; (2) event triggered in the foreground; (3) periodically in the background; and (4) event-triggered in the background. If synchronization is done in the foreground, then the user is blocked from performing any work while the synchronization occurs. Background synchronization allows the user to continue working. For example, the object ID for an object is used to query the local database 20. The object record may include a modification tag value (as described above), or each data object may be provided with one or more state bits which can be set to indicate the file or data is stale. If the modification tag value or state bits indicate that the object needs to be synchronized, the updated object may be requested from the server in the foreground or in the background. Alternatively, a client workstation 12 may periodically search its entire local database 20 for objects which need to be updated. This may take the form of a database query for objects having a modification tag value less than the current value, or a database query for objects having a particular value for state bit fields. Objects returned by the query are requested from the server as discussed above."

- Venkataraman at, e.g., [0026] "At any time prior to or substantially simultaneous with steps 302 through 306, the central data may be updated by the synchronization application at step 320. For example, while a user is away from the office, his or her assistant may enter local data on a local device that requires synchronization with the central data. The changes to the local data may cause synchronization to be initiated automatically. Conversely, the synchronization application may periodically query the local device to determine if any updates to the central

data are needed. Operation of a synchronization application in this manner is well known in the art. Regardless, the central data serves as the primary storage point for up to date data.”

- Matsubara at, e.g., [0025] “For the sake of discussion, suppose peer client **22a** has subscribed to File-X **32**, which is owned by peer client **22b**. In the context of the present invention, a “subscription” to an information source, the act of “subscribing” to an information source, and other variants, means that the subscriber will be notified of any changes that occur to the subscribed information source. Thus, if peer client **22b**, who owns File-X, modifies the file, then peer client **22a**, who subscribed to File-X, will be made aware of the modification. The notification server **14** manages a plurality of subscribed information sources as peer clients send subscription requests to it.” [0034] “After the information is accessed, it can be displayed, or played back, or otherwise processed by the peer client. It is noted that in another implementation, the peer client software can be configured to automatically start accessing the information without direction from the user to access the information.”
- Kaasten at, e.g., 2:18-22 “When a change is detected, the data synchronization service sends a notification of the change to the other devices so that they can update their copies of the data object. Thus, the copies of the data object are kept in synchrony on all of the selected devices. A user can access a data object from any of the selected devices, knowing that he will retrieve the latest version of the data object. If one device is temporarily not available, then the latest version can still be accessed from another device.”
- Savage at, e.g., [0031] “In one implementation, the scanning module **112** detects that a file on the synchronization client has changed, generates and schedules a synchronization report for that file, and transfers the synchronization report to the synchronization server **108** -at the appropriate time. In one implementation, the transfer may be accomplished by a transmission

from the client to the server over a communications network. However, other implementations may transfer the synchronization report via an intermediary, such as a shared storage location, or by having the report read from the client by the server.” [0032] “Upon receipt of a synchronization report for a file, the synchronization server 108 checks to determine the state of the file within its own files system. If the synchronization server 108 needs to receive the file from the client in order to synchronize with the client, the synchronization server 108 request that the client upload the file to the server. An update module 118 uploads the requested file to the synchronization server 108 (see synchronizing files 120).”

- Rao at, e.g., 4:56-60 “The client system 104 is synchronized with the server system 102. When synchronized, the server's item repository 110 is synchronized with the client item repository 130. On the initial synchronization, the server system 102 may merely copy the entire item repository 110 to the client system 104.” 4:61-5:5 “Subsequent synchronization events may involve detecting which items have been updated or changed since the last synchronization event. If an item has been updated, but the corresponding item on the opposite system has not been changed since the last synchronization, the updated item may be copied to the opposite system. If both the item and its corresponding item on the opposite system have been updated, a decision must be made to select one or the other item. In some embodiments, rules may be used for determining which item to keep, while in other embodiments, a user may be queried. Synchronization of files may follow the same methodology as synchronization of items.” 5:6-15 “In some embodiments, an indicator may be assigned to each metadata item or raw data file at synchronization. The indicator may be used to determine whether or not an item or raw data has been updated. In some embodiments, an incremental counter may be used as a version identifier. Upon each update of metadata or raw data, the counter may be incremented, so that

when a synchronization event occurs, the version identifiers of the server and client metadata or raw data may be compared to determine if an update has occurred since synchronization.”

5:16-23 “In other embodiments, a timestamp may be used as a version indicator. The timestamp may include date and time, or may be another indicator of time such as a coded time signal. In still other embodiments, a simple flag may indicate whether or not an item or set of raw data has been updated.”

- Braginsky at, e.g., [0040] “A request received from the OS/FS interface 204 that modifies a meta-entry 308 will set its IS\_DIRTY flag. In some embodiments, setting the IS\_DIRTY flag causes the synchronization module 208 to automatically commit the file to the server system 104 for file synchronization. An open file request may sometimes fail, because the requested file is not locally available. If the data is unavailable (e.g., has\_data=0) and the server system 104 is reachable, the synchronization module 208 automatically schedules a download of the data while blocking other requests to modify the same data until the data is available. In some embodiments, when a request to access a file fails, a request to download that file is automatically given high priority within a predefined prioritization scheme for ordering file downloads to the client system.” [0044] “The synchronization module 208 commits “dirty” meta-entries to the server system 104, gets updates from the server system 104, and integrates the updates in the meta-directory 206 at the appropriate meta-entries 308. It also schedules files for upload/download via the worker module 214 and maintains a synchronized clock with the server system 104. If a user is logged in to the file system 116 and the client system 102 is connected to the network 106, then the synchronization module 208 uses available bandwidth to maintain synchronization with the remote file system 112 in server system 104. In some embodiments, the synchronization module 208 automatically polls the server system 104 for

the latest meta-data and file information and requests and handles conflict resolution instructions from the user (obtained via the Client UI 212 and the Event Master 210).” [0048]

“Tasks 410 are received from the event master 210 and stored in the task queue 404. For example, when files need to be uploaded to the server system 104 or downloaded from the server system 104 a corresponding task is added to the task queue 404. The tasks 410 in the task queue 404 are ordered according to the priority policy 408. In some embodiments, the priority policy 408 can be modified or specified by the user via the client UI 212. In some embodiments, each task 410 includes a progress state and a run method that gets called repeatedly until the task 410 completes. The task manager 406 repeatedly gets an unfinished task 410 from the task queue 404 and calls its run method. It then sleeps for a specified delay, allowing for bandwidth throttling. In other embodiments, other mechanisms may be used for bandwidth throttling. If the task 410 fails before it completes, it may be marked as failed and kept on the task queue 404 so that it can be run again at a future time in accordance with the priority policy 408 then in effect. Some examples of a file synchronization process flow 700 and a background process flow 800 performed by the synchronization module 208 are described with respect to FIGS. 7 and 8, respectively.” [0065]

“To write a file a user selects a save file option in the application (e.g., a word processing application) being used by the user to create, modify or copy the file. The O/S 202 receives the request and forwards it to the OS/FS interface 204. The OS/FS interface 204 looks-up the file in the meta-directory 206, to determine if the file already exists. The OS/FS interface 204 then writes the file to the local cache 224, and creates or updates a corresponding entry in the meta directory 206. If the file previously existed, and already has a file ID assigned to it, the meta-entry 308 for the file will retain the previously assigned file ID, which is stored in the file ID field of the meta-entry 308

for the file. The OS/FS interface 204 sets the IS\_DIRTY flag in the meta-entry 308 to indicate that the file contains new or updated content. During the next synchronization cycle, the synchronization module 208 will process the new or updated meta-entry 308 for the file, and will determine that a corresponding upload task is required for sending the new file content to the file server. The worker module 214 stores the upload task in the task queue 404 in accordance with the priority policy 408. The synchronization module 208 can then upload the file to the remote file system 112 when the task reaches the top of the task queue 404.” [0067]

“In some embodiments, the synchronization module 208 periodically or episodically commits changes made by the client system to the server system 104, and also receives meta-data and file content changes from the server system 102. During this synchronization process, the server system 104 accepts file meta-data and content changes from the client that do not conflict with changes to the same files made by other clients, and the client accepts file meta-data and content changes from the server when those changes do not conflict with changes to the same files by the client. However, when the server system has newer changes that have been made to the same file ID by another client system 102, or has a file with the same filename as a client system file, but having a different file ID, the existence of a conflict is detected and the user of the client system 104 is requested to resolve that conflict by selecting the client or server version of the document, or by removing the conflict by renaming or moving the client file to a different filename or file path.” [0069] “FIG. 5 is a flow diagram of an embodiment of a metadata synchronization process 500. The metadata synchronization process may be performed separately for each share, or it may be performed for all shares to which a user has access rights. However, since different shares may be stored on different file servers, the following explanation is directed to synchronizing the metadata for a single share. The

metadata synchronization process 500 is repeatedly performed periodically (e.g., once every N seconds, where N is in the range of 1 to 10) or episodically (e.g., in response to the occurrence of any one of a predefined set of trigger conditions). As explained in more detail below, some synchronization operations require more than one iteration of the metadata synchronization process 500 to be completed.”

- Shappell at, e.g., 13:20-36 “When a modification of a file contained in a shared space occurs, a notification to other members in the group occurs (see steps 1902, 1904 and 1906 in FIG. 19). In a preferred embodiment, upon receipt of such notification, member machines that have previously obtained a copy of the shared file will remove the local copy of the out-of-date file. This will change the visual representation of the file to a “missing” file within the shared space folder when the application setting does not automatically update files, as is shown in step 2010 in FIG. 20. On the other hand, if automatic replication of the shared file is enabled, the local machine obtains an updated version of the file, as is shown in step 2006 in FIG. 20. Finally, an appropriate visual indicator is presented to the user as shown in step 2008. In order for a group space member to access an updated that is not locally stored, the file must first be transmitted to the local machine.”
- Sagar at, e.g., 4:1-11 “A synchronized folder 210 may represent a set of folders that are maintained to be consistent across a plurality of nodes or devices 201-204. In other words, a local copy of a synchronized folder at each node may be consistent with every other local copy of the synchronized folder at every other node. Maintaining consistency of the synchronized folder among a plurality of nodes may be extremely difficult where each node may modify their local synchronized folder (e.g., a local copy of the synchronized folder) and when multiple nodes may each be required to provide a local view of the synchronized folder

consistent with local node semantics.” 4:23-32 “Generally, updates or changes to content may be propagated using a broadcast model in which a change in the synchronized folder at one node initiates a broadcast to all nodes of the mesh of the synchronized folder to indicate the change. Synchronization conflicts may represent conflicts between different update data (e.g., originated by updates at different nodes) for the same item or related item of the synchronized folder (where an item may be a folder or a file). Other conflicts may also arise due to the capabilities of the underlying file system.” 6:4-16 “FIG. 4 illustrates a synchronization process embodiment. At block 401, an item in a feed is updated (or created) on a device or node. MOE may sense that this item is updated (or created) 402. The sensing may be performed, for example, using a watcher component. A watcher component may be a monitoring component that is configured to receive update indications from a node of the mesh and facilitate alerting other nodes of the mesh about the update. The watcher component may further be configured to facilitate the update process by executing certain functions. The watcher may provide indications of updates along with the content of an update using feeds, such as an atom feed.”

- Moromisato at, e.g., 20:8–16: "Once the file synchronizer 308 determines that the file system has been changed it passes the changes to the document share engine in the shared workspace which thereupon disseminates the changes. If a file has been edited, then the document share engine may either disseminate the entire updated file contents or determine the changes made to the file by comparing the current file contents against a copy of the contents that it maintains and disseminate only the differences." 7:25–28. ““The file synchronizer 308 is a software program that insures that any changes made to a file in the Windows shell file system 302 are also made to copies of that file that are stored in the collaborative system and vice versa.”
- Aboulhosn at, e.g., 6:60–7:15: "A member updates a folder by adding, deleting, or modifying

a file of the shared folder. The member that updates the folder sends an out-of-synchronization message 621 to the group owner, unless it is the group owner itself that updates the folder. In response, the group owner sends a synchronization request message 622 to the updating member. The updating member responds with a synchronization response message 623 that contains the metadata describing the update. The group owner upon receiving the metadata updates to the metadata associated with the shared folder. The group owner then notifies each other group member of the update. The group owner notifies a member by first sending an out-of-synchronization message 624 to the member. The member then responds by sending to the group owner a synchronization request message 625. In response, the group owner sends the metadata to the members via a synchronization response message 626. Upon receiving the metadata, the member updates its folder. In one embodiment, certain messages relating to updating a folder are not queued if a member is offline. When that member comes online, the online synchronization process will result in synchronization of the folders"

- Brown '847 at, e.g., Abstract "Distributed clients access the server, to learn about changes made to the files on the server, and to push local changes of the files onto the server. A synchronization application is used to synchronize the clients and server, synchronizing metadata and selected files." 3:47–50 "The client SA monitors local file system activity within the directory (sometimes called a folder) on the client so that it can efficiently locate changes to send to the server SFS during synchronization." 7:13–20 "During synchronization, when the client SA pulls down changes from the server and makes changes to the user's directory, it updates the client metadata to reflect those changes. Also, when the client SA pushes changes to the server during the second part of the synchronization process, it records new SID and FSI values returned by the server SFS into the client metadata file and directory items." 8:15–20

“Specifically, filter driver 350 watches for other applications accessing the files in folder 302, so as to determine which files on the client have been changed. When the client later synchronizes with the server, the client can use the information provided by the filter driver to identify which files to push to the server.”

- Brown ‘826 at, e.g., Abstract “Distributed clients access the server, to learn about changes made to the files on the server, and to push local changes of the files onto the server. A synchronization application is used to synchronize the clients and server, synchronizing metadata and selected files.” 3:49–52 “The client SA monitors local file system activity within the directory (sometimes called a folder) on the client so that it can efficiently locate changes to send to the server SFS during synchronization.” 7:14–21 “During synchronization, when the client SA pulls down changes from the server and makes changes to the user's directory, it updates the client metadata to reflect those changes. Also, when the client SA pushes changes to the server during the second part of the synchronization process, it records new SID and FSI values returned by the server SFS into the client metadata file and directory items.” 8:13–18 “Specifically, filter driver 350 watches for other applications accessing the files in folder 302, so as to determine which files on the client have been changed. When the client later synchronizes with the server, the client can use the information provided by the filter driver to identify which files to push to the server.”
- Hesselink at, e.g., 41:25–39 “As the user works on the file, such as by adding to the file, modifying or editing, each time that the user saves his/her work, the system compares the differences between the saved version of the file and the previous version of the file, and, as long as the user module remains connected to the network, the changes/differences are transmitted over the network to the remote device module, which are saved there, thereby

maintaining the home file (i.e., official reference file, source file, master file or DataClient file) up to date with the latest revisions, as described above. This functionality saves the user a great deal of time, since there is no need to synchronize files when the user returns to the site of the device module. Further, there is no concern over having two different versions of a file, or concern that a user is not using the latest, most up to date version of a file.” 48:08–21 “The present system may also be optionally employed to perform synchronization of files among a user's multiple computers. Referring back to the operations described above, the system is designed to automatically update at least the "master copy" of a file at a remote location (i.e., the file at the location of the device module from where the file was accessed by the user module) as the user makes changes to the file locally on the computer associated with the user module. However, the user may be provided with alternative synchronization options by the present system. For example, the user may choose to always synchronize all copies of a file, wherein, whenever the user makes a change to that file locally, at any location, then all of the devices which subscribe to that file are automatically updated immediately.”

- iFolder at, e.g., [http://www.pt-solutions.com/pdf/Novell\\_iFolder.pdf](http://www.pt-solutions.com/pdf/Novell_iFolder.pdf) at 6. “The Novell iFolder client software runs on Windows 95, 98, ME, NT, 2000, and XP workstations. It stores user data in a local Novell iFolder directory and watches for any changes to user files. ... [T]he Novell iFolder client runs as a filter driver and plugs right into the file system to facilitate its ability to watch for file system events that indicate that a modification has been made to file. ... When any change or discrepancy is detected in a user’s personal Novell iFolder directory, the Novell iFolder client communicates the changed data in an encrypted format using Blowfish encryption to the Novell iFolder server for distribution to the user’s other subscribing clients.” [http://www.pt-solutions.com/pdf/Novell\\_iFolder.pdf](http://www.pt-solutions.com/pdf/Novell_iFolder.pdf) at 12. “Anytime a user saves a

file, the data is immediately stored on the local computer and then automatically backed up and stored on the Novell iFolder server, as well as on any other workstations that the user has connected to the Novell iFolder server.”

- BeInSync at, e.g., [https://web.archive.org/web/20060330064008/http://www.beinsync.com/bis\\_datasheet.pdf](https://web.archive.org/web/20060330064008/http://www.beinsync.com/bis_datasheet.pdf) at 2. “Simply install BeInSync on each of your PCs, choose what you'd like to synchronize (your files, desktop, emails, contacts or Internet favorites) and you're ready to go. From now on, any change made on one PC will automatically be reflected on your other PCs. And what's more, you can easily share data with others. ... Any file added to a share will automatically be synchronized among all the share's participants.” <https://web.archive.org/web/20060112125007/http://www.pcmag.com/article2/0,1895,1645668,00.asp>

“Synchronization occurs on the fly and, as long as your systems remain online, around the clock. If you change a file, the change will propagate across all your online systems in a matter of minutes—sometimes seconds.”

- FolderShare at, e.g., <https://web.archive.org/web/20060313171107/https://www.foldershare.com/download/files/FolderShareGuide.pdf> “FolderShare runs silently in the background constantly monitoring the files in the folders you specified to keep in sync for any changes. Once changes are detected, they are replicated to all other computers. You can continue to access and work with the files in this synced folder as you normally do.” <https://web.archive.org/web/20060313171026/https://www.foldershare.com/info/faq/AboutFolderShare.php> “FolderShare works in the background to automatically recognize modified files and auto-upload them to other computers connected to your library.” <https://www.pcmag.com/archive/foldershare-147101> “By default, the service automatically synchronizes your folders around the clock. As long as your machines remain online, when

you change a file in one of the folders, the change propagates to each system in a matter of minutes—if not seconds.”

- SharpCast at, e.g.,

<https://web.archive.org/web/20060829025503/http://www.sharpcast.com/products/photos/>

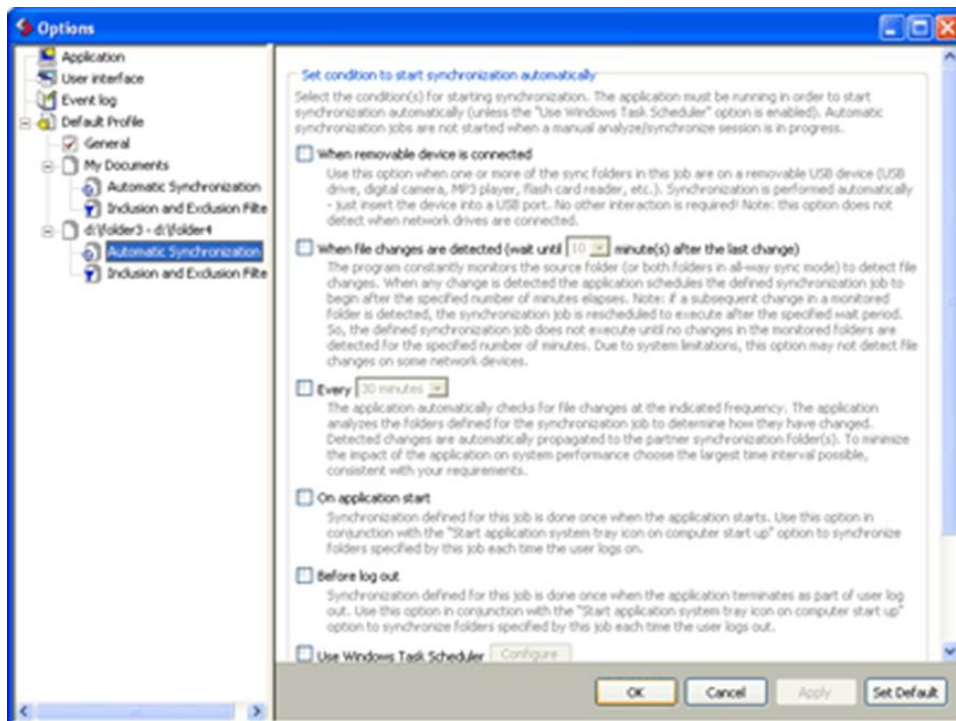
“Changes you make anywhere, even offline, are automatically made everywhere, so you can pick up right where you left off.” “Changes you make on the phone are instantaneously synchronized with the web and your PC, and vice versa.”

<https://web.archive.org/web/20060829025503/http://www.sharpcast.com/products/photos/>

- GoodSync at, e.g.,

<https://web.archive.org/web/20060417232608/http://www.goodsync.com/index.html> “A

variety of automatic synchronization options include; when file changes are detected, a removable device is connected, a time period has elapsed, on user logon/logout, or on a set time schedule.”



<https://web.archive.org/web/20060427042153/http://www.goodsync.com/screenshot.html>

- Subversion at, e.g., <https://web.archive.org/web/20071116172448/http://svnbook.red-bean.com/en/1.4/svn-book.pdf> at 335. “Autoversioning is an optional feature defined in the DeltaV standard. A typical DeltaV server will reject an ignorant WebDAV client attempting to do a PUT to a file that's under version control. To change a version-controlled file, the server expects a series of proper versioning requests: something like MKACTIVITY, CHECKOUT, PUT, CHECKIN. But if the DeltaV server supports autoversioning, then write-requests from basic WebDAV clients are accepted. The server behaves as if the client had issued the proper series of versioning requests, performing a commit under the hood. In other words, it allows a DeltaV server to interoperate with ordinary WebDAV clients that don't understand versioning. Because so many operating systems already have integrated WebDAV clients, the use case for this feature can be incredibly appealing to administrators working with non-technical users: imagine an office of ordinary users running Microsoft Windows or Mac OS. Each user “mounts” the Subversion repository, which appears to be an ordinary network folder. They use the shared folder as they always do: open files, edit them, save them. Meanwhile, the server is automatically versioning everything. Any administrator (or knowledgeable user) can still use a Subversion client to search history and retrieve older versions of data. This scenario isn't fiction: it's real and it works, as of Subversion 1.2 and later.” <https://web.archive.org/web/20071116172448/http://svnbook.red-bean.com/en/1.4/svn-book.pdf> at 335. “When SVNAutoversioning is active, write requests from WebDAV clients result in automatic commits.”
- SmartSync Pro discloses bi-directional synchronization, which includes automatically transferring a modified first file to a first device. The On Folder Changes option triggers sync

operations when any difference in the source file is detected, such as by doing a save operation on a modified file. SSP Ex. 7 at 4, 31.

- TortoiseCVS at, e.g., [https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 13. “A commit merges the changes that were made in your local copy of the source code into the remote CVS repository. The commit process first compares the local revision number of each affected file with the latest revision number of the corresponding file in the remote CVS repository. If the revision numbers are the same, the commit process proceeds.” TortoiseCVS at, e.g., [https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 9. “An update merges the differences that have occurred in the remote CVS repository since your last update into your current local copy of the source code. The local TortoiseCVS client transmits the current local revision numbers to the remote CVS server, which then generates the differences between those revisions and the latest revisions available on the remote server. These differences will be applied to your current local source code files. Such differences are usually the edits that other developers have made to the source code, and committed to the remote CVS repository.”
- Apple iDisk at, e.g., <https://flylib.com/books/en/3.324.1/> at 74. “The default behavior is for your iDisk to be synchronized automatically, whenever Mac OS X detects that your local or remote iDisk has been changed.” <https://flylib.com/books/en/3.343.1/> at 35 “You’ll notice that there are two options for synchronization: Automatically (the default) and Manually. If you have iDisk synchronization set to Automatically, your local iDisk will sync with your online iDisk any time you make a change to the contents of the local copy.”

<https://www.informit.com/articles/article.aspx?p=401141&seqNum=5> “From the perspective of the end user, iDisk synchronization is entirely transparent—your iDisk appears like any other disk whether working with the contents online or using the local copy. As you make changes to the files, they are noted and automatically uploaded to the .Mac server in the background.”

- Apple iSync at, e.g., <https://flylib.com/books/en/3.338.1/> at 153. “Click the "Synchronize with .Mac" checkbox, then from the pop-up menu set how often you want your computer to sync the selected files. If you choose ‘Automatically,’ iSync will sync whenever a change is made to the selected file type.” <https://flylib.com/books/en/3.324.1/> at 127. “When you configure Address Book to synchronize itself with an Exchange server, Address Book is actually passing off the synchronization duties to iSync. While you update contacts in Address Book, iSync operates in the background, finding new and changed entries in the Exchange server and propagating them into your Address Book.”

#### **(6) “Replace Older Version” References**

Certain Asserted Claims recite features including replacing an older version of a file with a copy of a modified version of a file that is transferred across devices. Nothing about these claim limitations add anything novel or non-obvious over the prior art. To the contrary, this feature was well-known, routine, conventional, and widely available long before the alleged invention. As the attached claim charts demonstrate, this subject matter is disclosed in numerous references. For purposes of these contentions, references whose disclosures are cited in these contentions with respect to this claimed subject matter as reflected in the claim charts (including without limitation the specific references cited below) are hereby labeled the “Replace Older Version” references.

The implementation of the claimed features would have been obvious to a person of

ordinary skill in the art least because doing so would have involved the combination of prior art elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the application of a known technique to a known device ready for improvement to yield predictable results, and/or would have been obvious to try. Moreover, a person of ordinary skill in the art would have readily appreciated the well-known and widespread use of these features and the associated benefits and advantages of their use. Persons of ordinary skill in the art would have been motivated to combine the teachings of these references with any system and/or disclosure that relates to the synchronization of files or other data content. It was well-known to implement file synchronization techniques where an updated or modified version or copy of a file would be received by a device and an outdated earlier version or copy of the file would be deleted and replaced by the updated or modified version, as a basic mechanism of maintaining updated current copies of files while maintaining disk storage space by replacing older versions that are no longer needed. Replacing the older version of a file also reduces the likelihood of conflicts arising from modifying the same file on different devices. For example, references disclose the following motivations to combine, such as the following as well as the other disclosures that are cited in the prior art claim charts submitted herewith:

- Venkataraman at, e.g., [0028] “In a preferred embodiment, three different synchronization modes are possible. In the first, remote data having temporal priority (i.e., most recently created/modified) is preferred and overwrites data found in the central data. In the second, the opposite is true, and central data having temporal priority is preferred and overwrites data found the remote data. In the third mode, data having temporal priority is preferred regardless of its source. These three options are presented to the user at step **310**, along with an option to cancel the synchronization procedure. Upon receipt, the user’s selection in response to the

prompt is provided to the synchronization application. With the synchronization mode information, the synchronization application synchronizes the remote and central data accordingly at step 326. In this manner the central data is now synchronized to reflect the changes necessitated, if at all, by the remote data. The synchronized data (i.e., the now-updated central data) is transmitted by the synchronization application at step 328 and received by the synchronization device at step 312. The form of the synchronized data is a matter of design choice. For example, the synchronized data may comprise a copy of all of the relevant central data, or information indicative of the modifications that must be made to the remote data in order to synchronize it to the central data. Regardless, the synchronization application, at step 330, subsequently updates (either periodically or on-demand) the local data on any local devices that subscribe to synchronization services for the central data.” [0017] “Note that the remote data 120 and central data 122 are of the same type as the local data, i.e., including, but not limited to, contact data (e.g., phone numbers, addresses, etc.), calendar data, email, shared files and the like.”

- Kaasten at, e.g., 7:40-44 “Some embodiments monitor the time of the most recent modification of a synchronized data object. If that time is significantly different from the most recent modification time of a counterpart data object, then the more recent version should replace the out-of-date version.”
- Savage at, e.g., [0032] “Upon receipt of a synchronization report for a file, the synchronization server 108 checks to determine the state of the file within its own files system. If the synchronization server 108 needs to receive the file from the client in order to synchronize with the client, the synchronization server 108 request that the client upload the file to the server. An update module 118 uploads the requested file to the synchronization server 108 (see

synchronizing files 120). Alternatively, if the synchronization server 108 determines a conflict (e.g., the client file is edited and the server file is also edited), the synchronization server 108 executes a conflict resolution (e.g., asking the user to chose which version of the file to record at both the client and the server). Upon synchronizing the file with the synchronization client 106, the synchronization server 108 transmits a synchronization confirmation to inform the synchronization client 106 that the synchronization is complete. Note: If there is an error in the synchronization, the synchronization confirmation can also be used to indicate this to the synchronization client 106.” [0032]

- Rao et al., e.g., 6:55-58 “FIG. 3 is a pictorial representation of an embodiment 300 of a sequence for synchronizing. The process shows an initial connection 301, a period where the systems are disconnected 325, and the actions at reconnection 326.” 6:59-67 “In the initial connection 301, a server 302 and client 304 are connected. The server 302 has a storage system 306 that contains items 308 in an item repository 310. Some of the items 308 have related files 312 and 314. During the initial connection 301, the client storage 316 is populated with a synchronized item repository 318 that contains items 320, which are copies of items 308 from the server 302. The two server referenced files 312 and 314 are ghosted on the client 304 as files 322 and 324.” 7:1-3 “During the initial connection 301, the item repositories 310 and 318 are synchronized and the raw data files 312 and 314 are ghosted on the server as ghosted files 322 and 324.” 7:4-9 “Block 325 shows the state of the systems when connection is severed. Server 302 maintains storage 306. Client 304 has storage 316 that contains an item repository 318 that enables various metadata and other functions to be performed. While disconnected, a ghosted file 324 may be selected for future synchronization.” 7:10-15 “Block 326 shows the systems upon reconnect. Server 302 and client 304 are reconnected. The item repositories 310 and 318

are synchronized between storage systems 306 and 316. As part of the synchronization routine, the file 314 replaces ghosted file 324 so that the raw data of file 324 may be available when disconnected.”

- Braginsky at, e.g., [0027] “If a user is connected to the server system 104 via the network 106, then the file system 116 returns the most recent version of the requested file, whether cached locally in file system 116 or stored remotely in file system 112. In some embodiments, the selection of the latest version of the requested file can be based on a comparison of file timestamps located in a meta-directory stored on the client system 102, as described below with respect to FIG. 5. In some circumstances, the entire file may have to be downloaded from the server system 104 to the client system 102 to service a request. This may happen, for instance, when the server has a newer version of the file than the client, and the process of downloading the newest version to the client did not begin prior to the request.” [0028] “If the client system 102 is not connected to the server system 104, then the file system 116 returns the locally cached version of the requested file. If there is no cached version of the file, then the file system 116 returns an error and notifies the user (e.g., through a callout bubble) that the requested file is not available offline.” [0065] “To write a file a user selects a save file option in the application (e.g., a word processing application) being used by the user to create, modify or copy the file. The O/S 202 receives the request and forwards it to the OS/FS interface 204. The OS/FS interface 204 looks-up the file in the meta-directory 206, to determine if the file already exists. The OS/FS interface 204 then writes the file to the local cache 224, and creates or updates a corresponding entry in the meta directory 206. If the file previously existed, and already has a file ID assigned to it, the meta-entry 308 for the file will retain the previously assigned file ID, which is stored in the file ID field of the meta-entry 308 for the

file. The OS/FS interface 204 sets the IS\_DIRTY flag in the meta-entry 308 to indicate that the file contains new or updated content. During the next synchronization cycle, the synchronization module 208 will process the new or updated meta-entry 308 for the file, and will determine that a corresponding upload task is required for sending the new file content to the file server. The worker module 214 stores the upload task in the task queue 404 in accordance with the priority policy 408. The synchronization module 208 can then upload the file to the remote file system 112 when the task reaches the top of the task queue 404.” [0067]

“In some embodiments, the synchronization module 208 periodically or episodically commits changes made by the client system to the server system 104, and also receives meta-data and file content changes from the server system 102. During this synchronization process, the server system 104 accepts file meta-data and content changes from the client that do not conflict with changes to the same files made by other clients, and the client accepts file meta-data and content changes from the server when those changes do not conflict with changes to the same files by the client. However, when the server system has newer changes that have been made to the same file ID by another client system 102, or has a file with the same filename as a client system file, but having a different file ID, the existence of a conflict is detected and the user of the client system 104 is requested to resolve that conflict by selecting the client or server version of the document, or by removing the conflict by renaming or moving the client file to a different filename or file path.” [0068] “For example, Client A modifies an existing version 0.0 of File X and commits the modified version to the server system 104 for synchronization. The server system 104 performs the synchronization and uploads File X from Client A, and labels it as version 1.0 of File X. Client B then attempts to commit a modified version 0.0 of File X to the server system 104. However, the server system 104 will not accept

the modified version 0.0 of File X from Client B, because it conflicts with the version of File X from Client A. Instead, Client B receives from the server system 104 information that identifies the conflict. Client B then provides the user with options for solving the conflict. For example, the user of Client B could be presented with a dialog box, using client UI 212, which asks the user if she wants to overwrite File X, version 1.0. If the user selects the overwrite option, then Client B's modified version 0.0 will be committed to the server system 104 as version 1.0, and will replace or overwrite Client A's version 1.0, which was previously committed to the server system 104. If the user doesn't select the overwrite option, and instead selects a "discard" option, Client B's local version of File X will be overwritten with version 1.0 of File X."

- Shappell at, e.g., 13:20-36 "When a modification of a file contained in a shared space occurs, a notification to other members in the group occurs (see steps 1902, 1904 and 1906 in FIG. 19). In a preferred embodiment, upon receipt of such notification, member machines that have previously obtained a copy of the shared file will remove the local copy of the out-of-date file. This will change the visual representation of the file to a "missing" file within the shared space folder when the application setting does not automatically update files, as is shown in step 2010 in FIG. 20. On the other hand, if automatic replication of the shared file is enabled, the local machine obtains an updated version of the file, as is shown in step 2006 in FIG. 20. Finally, an appropriate visual indicator is presented to the user as shown in step 2008. In order for a group space member to access an updated that is not locally stored, the file must first be transmitted to the local machine."
- Sagar at, e.g., 7:5-16 "Processing of a feed received from the Storage Service may be performed in two blocks. The update items in the feed from the Storage Service may be merged

with a local copy of the feed at a node. In one embodiment, merging the feed data may include associating or matching each item of the feed with corresponding items of the local feed copy based on an identifier of each item. In one embodiment, merging updates with the local feeds includes modifying the local feed to incorporate the updates. This may include overriding local feed information with the updates. In the described system, where the updates are incremental updates, the local feed may simply be updated to incorporate only the most recent changes.”

- Moromisato at, e.g., 16:49–51 “When the file arrives, the stub file representing the actual file is replaced by the actual file and the file system snapshot is updated to indicate that the file contents have changed.” 21:7–9 “In particular, file synchronizer 308 associates a unique version identifier and the last modified time with each change made to a synchronized file.” 19:20–23 “Binary difference support can be disabled through a local-only setting on folders and files. When binary difference support is disabled, document share engine 312 does not store old file versions in the collaborative system storage.”
- Brown ‘847 at, e.g., 13:11–23 “The client also computes a MDA from the file. The client SA then requests the MDA from the file to be downloaded and compares the two MDAs. If the two arrays are sufficiently similar, the client SA performs a special download where it requests the specific 4K blocks that have differing message digest values. It creates the download file by modifying the copy of the file with the requested downloaded 4K blocks. On the other hand, if less than a threshold number of message digests in the MDAs match, then the entire file is downloaded from the server. Once the download file is completely constructed, the client inserts the download file into its final location, replacing an older version of the file if it exists.” 12:61–13:3 “The client SA compares the two MDAs and if a sufficiently high number of message digests match, it performs a special upload where only the differing message digests

and their corresponding 4K data blocks are uploaded. The server constructs the new version of the file by starting with a copy of the previous version and modifying it with the uploaded data. Once the file has been completely uploaded, the server then stores the file in the specified directory and updates the file metadata.”

- Brown ‘826 at, e.g., 13:2–14 “The client also computes a MDA from the file. The client SA then requests the MDA from the file to be downloaded and compares the two MDAs. If the two arrays are sufficiently similar, the client SA performs a special download where it requests the specific 4K blocks that have differing message digest values. It creates the download file by modifying the copy of the file with the requested downloaded 4K blocks. On the other hand, if less than a threshold number of message digests in the MDAs match, then the entire file is downloaded from the server. Once the download file is completely constructed, the client inserts the download file into its final location, replacing an older version of the file if it exists.” 12:53–61 “The client SA compares the two MDAs and if a sufficiently high number of message digests match, it performs a special upload where only the differing message digests and their corresponding 4K data blocks are uploaded. The server constructs the new version of the file by starting with a copy of the previous version and modifying it with the uploaded data. Once the file has been completely uploaded, the server then stores the file in the specified directory and updates the file metadata.”
- Hesselink at, e.g., 41:25–39 “As the user works on the file, such as by adding to the file, modifying or editing, each time that the user saves his/her work, the system compares the differences between the saved version of the file and the previous version of the file, and, as long as the user module remains connected to the network, the changes/differences are transmitted over the network to the remote device module, which are saved there, thereby

maintaining the home file (i.e., official reference file, source file, master file or DataClient file) up to date with the latest revisions, as described above. This functionality saves the user a great deal of time, since there is no need to synchronize files when the user returns to the site of the device module. Further, there is no concern over having two different versions of a file, or concern that a user is not using the latest, most up to date version of a file.”

- iFolder at, e.g., <https://www.novell.com/documentation/ifolder21/pdfdoc/admin/admin.pdf> at 16–17. “iFolder does away with version inconsistency, making it simple for users to access the most up-to-date version of their documents from any connected desktop, laptop, Web browser, or handheld device. In preparation to travel or work from home, users no longer need to copy essential data to their laptop from various desktop and network locations. The iFolder client can automatically update users' laptop and desktops with the most current versions of their files ... With iFolder, you always have convenient and secure access to the most recent version of your documents. All you need is an active network or Internet connection and the Novell iFolder client software or a Web browser.”
- BeInSync at, e.g., <https://web.archive.org/web/20060202155233/http://www.smallbusinesscomputing.com/biztools/article.php/3410531> “From then on, BeInSync is responsible for making sure the latest version of each file and folder is always present on every computer.”
- FolderShare at, e.g., <https://web.archive.org/web/20040528145753/http://www.foldershare.com/library/info/aboutReplication.php> “Replication: Running the FolderShare application on multiple computers allows you to easily keep an identical set of files on all your computers. Upload a file on your home computer and it will instantly be placed on your office computer. Make a change to a file on your laptop and your office computer will automatically change its version of the file.” <https://www.heraldnet.com/business/web-services-make-it-easy-to->

[synchronize-files](#) “I set up FolderShare to keep five different folders, containing more 36,000 files, synchronized, and it did so without breaking a sweat. For instance, I linked folders called “columns” on every machine, and linked the “My Music” folders on my three Windows machines to the “Music” folders on my three Macs. Now, all the latest versions of my columns and notes, including those in Microsoft Word, Excel and PowerPoint files, and Adobe PDF files, are on all my Windows PCs and Macs.”

- SmartSync Pro discloses an option to delete obsolete files, such as old versions of modified files. SSP Exs. 1-7.
- Nokia Intellisync discloses an option to delete obsolete files, such as old versions of modified files in the number of file differences to keep per file option. NIS Ex. 2 at 44.
- Unison File Synchronizer discloses a “backupversion” option which controls the number of previous versions kept, which results in the replacement of older files. UFS Ex. 3 at 2, 8, 11-12, 13-14, 49.

### **(7) “Transfer Metadata” References**

Certain Asserted Claims recite features including transferring metadata for files, where metadata for a given file is transferred across devices. Nothing about these claim limitations add anything novel or non-obvious over the prior art. To the contrary, this feature was well-known, routine, conventional, and widely available long before the alleged invention. As the attached claim charts demonstrate, this subject matter is disclosed in numerous references. For purposes of these contentions, references whose disclosures are cited in these contentions with respect to this claimed subject matter as reflected in the claim charts (including without limitation the specific references cited below) are hereby labeled the “Transfer Metadata” references.

The implementation of the claimed features would have been obvious to a person of

ordinary skill in the art least because doing so would have involved the combination of prior art elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the application of a known technique to a known device ready for improvement to yield predictable results, and/or would have been obvious to try. Moreover, a person of ordinary skill in the art would have readily appreciated the well-known and widespread use of these features and the associated benefits and advantages of their use. Persons of ordinary skill in the art would have been motivated to combine the teachings of these references with any system and/or disclosure that relates to the synchronization of files or other data content. Metadata was ubiquitous in the prior art along with file contents, and it was well-known to use metadata for many different advantageous features in file management and file synchronization systems. Among other features, it was well-known to separately transfer and synchronize metadata for files so that an indication and record of an updated file could be efficiently recorded and updated across devices without needing to update and transfer the complete file contents. For example, references disclose the following motivations to combine, such as the following as well as the other disclosures that are cited in the prior art claim charts submitted herewith:

- Salas at, e.g., 11:42-12:6 “Synchronization is enabled by storing all records in the server database with an associated modification tag. The tag is a positive integer which is taken from an ever-increasing counter. The counter increments each time it is read, i.e., each time a new modification tag is assigned to a data object stored on the server **14**. When a client workstation **12** synchronizes its local databases and files, it also receives the current modification tag, i.e., it also receives the current value of the counter. Alternatively, the current modification tag value can be included as extra information in each “wrapper page.” The client workstation **12**

includes the last modification tag value it received when it makes a subsequent synchronization request. The server **14** transmits to the client workstation **12** any data objects to which the user has appropriate access rights that also have a modification tag value greater than the modification tag value sent with the synchronization request. The client workstation **12** stores the received data objects in its local database and stores the new received modification tag value. Client workstations **12** and servers **14** may be separated by relatively slow, lossy channels such as telephone networks. Accordingly, synchronization requests can sometimes fail for various reasons, e.g., a connection node is out of service or a necessary transmission line suffers too many errors to allow a reliable connection to be made. In this event, the synchronization request fails and should be retried later. Once synchronization has been accomplished and local database metadata has been updated, the appropriate data objects and values are inserted into the eRoom where indicated by Replace Properties, and the eRoom is displayed to the user (step **508**) by the browser application in a traditional manner (refer to FIG. 4).” 13:52-57 “File metadata may include: the name of the file; the size of the file; the date the file was created; the date the file was last modified; access information such as which users may open, view, and edit the file, and information regarding the edit Status of the file, such as whether an edit lock has been set by a user.” 13:46-51 “If a file modification is occurring, the server **14** updates the metadata contained by the data object associated with the file and the file is transmitted to the server **14** using HTTP (step **712**). The server **14** associates the uploaded file with the newly-created data object.”

- Kaasten at, e.g., 2:37-45 “Some files may be very large, such as audio or video clips. Instead of incurring the costs of storing such a file on every computing device, a user can choose to ‘ghost’ the file on some devices. A ghosting device stores only metadata about the file (such

as its name and version date) rather than the entire file. The user can still access the file through the ghost: the access requests are sent to a device that holds the actual contents, and those contents are then presented to the user as if they were stored locally.” 4:63-5:7 “Minimally, the ghost file **142** only contains a reference to the very big file **140** on its host computing device **A 102**. However, the ghost file **142** is synchronized with the very big file **140** so that a user on the laptop **106** can access the very big file **140** through the ghost file **142**. If the user wishes to change the contents of the very big file **140**, he can make that change on the laptop **106** just as if he were working directly with the very big file **140**. The change is then sent, by means of the ghost file **142**, to the computing device **102** and is applied to the very big file **140** itself. Thus, the user of the laptop **106** can read and modify the very big file **140** as if a synchronized copy of it existed on the laptop **106**.”

- Savage at, e.g., [0014] “In one implementation, the synchronization client 106 maintains a synchronization state datastore 114 as an array of folder objects. A folder object is associated with a folder in the file system of the synchronization client 106 (and therefore the corresponding folder in the file system of the synchronization server 108) and maintains a database of the file objects associated with files in each folder. A file object maintains the synchronization state of the associated file. An example synchronization state for a file may be represented by the following fields of a file or folder object, although other state representations may be employed.” [0015] “Unreported state—set to add, delete, or edit when a file change event is detected; reset to none when synchronization client reports the state of the file to the synchronization server.” [0019] “File—the name of the file associated with the watch object.” [0020] “FullPath—the path name in the file system of the file associated with the watch object.” [0021] “LastUploadTime—the time of the last upload of the changed file

to the synchronization server.” [0022] “LastUploadSize—the amount of data in the last upload of the changed file to the synchronization server.” [0023] “LastWriteTime—the last time the file was edited on the client (also referred to as the last file change time).” [0031] “In one implementation, the scanning module 112 detects that a file on the synchronization client has changed, generates and schedules a synchronization report for that file, and transfers the synchronization report to the synchronization server 108 -at the appropriate time. In one implementation, the transfer may be accomplished by a transmission from the client to the server over a communications network. However, other implementations may transfer the synchronization report via an intermediary, such as a shared storage location, or by having the report read from the client by the server.”

- Rao at, e.g., 4:56-60 “The client system 104 is synchronized with the server system 102. When synchronized, the server's item repository 110 is synchronized with the client item repository 130. On the initial synchronization, the server system 102 may merely copy the entire item repository 110 to the client system 104.” 4:61-5:5 “Subsequent synchronization events may involve detecting which items have been updated or changed since the last synchronization event. If an item has been updated, but the corresponding item on the opposite system has not been changed since the last synchronization, the updated item may be copied to the opposite system. If both the item and its corresponding item on the opposite system have been updated, a decision must be made to select one or the other item. In some embodiments, rules may be used for determining which item to keep, while in other embodiments, a user may be queried. Synchronization of files may follow the same methodology as synchronization of items.” 5:6-15 “In some embodiments, an indicator may be assigned to each metadata item or raw data file at synchronization. The indicator may be used to determine whether or not an item or raw data

has been updated. In some embodiments, an incremental counter may be used as a version identifier. Upon each update of metadata or raw data, the counter may be incremented, so that when a synchronization event occurs, the version identifiers of the server and client metadata or raw data may be compared to determine if an update has occurred since synchronization.”

5:16-23 “In other embodiments, a timestamp may be used as a version indicator. The timestamp may include date and time, or may be another indicator of time such as a coded time signal. In still other embodiments, a simple flag may indicate whether or not an item or set of raw data has been updated.”

6:59-67 “In the initial connection 301, a server 302 and client 304 are connected. The server 302 has a storage system 306 that contains items 308 in an item repository 310. Some of the items 308 have related files 312 and 314. During the initial connection 301, the client storage 316 is populated with a synchronized item repository 318 that contains items 320, which are copies of items 308 from the server 302. The two server referenced files 312 and 314 are ghosted on the client 304 as files 322 and 324.”

7:1-3 “During the initial connection 301, the item repositories 310 and 318 are synchronized and the raw data files 312 and 314 are ghosted on the server as ghosted files 322 and 324.”

7:43-47 “When the client and server are reconnected in block 402, any changes to the items are synchronized as well as any changes to previously synchronized raw data files in block 404. Any files that were ‘unghosted’ or tagged to be made available offline may also be synchronized in block 404.”

7:48-60 “A database containing raw data and metadata is synchronized in two phases. In the first phase, metadata is synchronized between a server and client while the raw data remains on the server system. The client system, when connected or disconnected, may use and manipulate the metadata for any purpose whatsoever, including selecting one or more raw data files that are to be available on the client system. The second phase of synchronization is to

synchronize the selected raw data to the client system. The synchronized raw data may be manipulated, changed, and used on the client system when disconnected. Upon reconnect, any changes made to the metadata or the raw data are synchronized between the systems.”

- Braginsky at, e.g., [0038] “The meta-entries 308 (hereinafter also referred to as “meta-data” or “entry”) include information that is associated with files. For example, a meta-entry 308 could include one or more of the following fields listed in Table I.” See Table I. [0040] “A request received from the OS/FS interface 204 that modifies a meta-entry 308 will set its IS\_DIRTY flag. In some embodiments, setting the IS\_DIRTY flag causes the synchronization module 208 to automatically commit the file to the server system 104 for file synchronization. An open file request may sometimes fail, because the requested file is not locally available. If the data is unavailable (e.g., has\_data=0) and the server system 104 is reachable, the synchronization module 208 automatically schedules a download of the data while blocking other requests to modify the same data until the data is available. In some embodiments, when a request to access a file fails, a request to download that file is automatically given high priority within a predefined prioritization scheme for ordering file downloads to the client system.” [0044] “The synchronization module 208 commits “dirty” meta-entries to the server system 104, gets updates from the server system 104, and integrates the updates in the meta-directory 206 at the appropriate meta-entries 308. It also schedules files for upload/download via the worker module 214 and maintains a synchronized clock with the server system 104. If a user is logged in to the file system 116 and the client system 102 is connected to the network 106, then the synchronization module 208 uses available bandwidth to maintain synchronization with the remote file system 112 in server system 104. In some embodiments, the synchronization module 208 automatically polls the server system 104 for the latest meta-data and file

information and requests and handles conflict resolution instructions from the user (obtained via the Client UI 212 and the Event Master 210).” [0067] “In some embodiments, the synchronization module 208 periodically or episodically commits changes made by the client system to the server system 104, and also receives meta-data and file content changes from the server system 102. During this synchronization process, the server system 104 accepts file meta-data and content changes from the client that do not conflict with changes to the same files made by other clients, and the client accepts file meta-data and content changes from the server when those changes do not conflict with changes to the same files by the client. However, when the server system has newer changes that have been made to the same file ID by another client system 102, or has a file with the same filename as a client system file, but having a different file ID, the existence of a conflict is detected and the user of the client system 104 is requested to resolve that conflict by selecting the client or server version of the document, or by removing the conflict by renaming or moving the client file to a different filename or file path.” [0069] “FIG. 5 is a flow diagram of an embodiment of a metadata synchronization process 500. The metadata synchronization process may be performed separately for each share, or it may be performed for all shares to which a user has access rights. However, since different shares may be stored on different file servers, the following explanation is directed to synchronizing the metadata for a single share. The metadata synchronization process 500 is repeatedly performed periodically (e.g., once every N seconds, where N is in the range of 1 to 10) or episodically (e.g., in response to the occurrence of any one of a predefined set of trigger conditions). As explained in more detail below, some synchronization operations require more than one iteration of the metadata synchronization process 500 to be completed.”

- Shappell at, e.g., Abstract “A computer implemented method and system enable users to share files in a server-less shared space. By providing access to such 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.” 12:22-29 “For sharing files among group members, metadata is transmitted to all members when files become available, such as is shown in the steps 1904 and 1906 in the flow chart in FIG. 19. Such metadata are sufficient to render an icon of the shared file and may include date, time, thumb-nail information, name size, and optionally the source of the information. This may include an identifier for the source computing machine and the creator identity.” 12:40-65 “The following table illustrates one such implementation of required and optional metadata concerning a shared file.

---

|             |                                                            |
|-------------|------------------------------------------------------------|
| // required |                                                            |
| WCHAR       | *m_pwzFilename; //Name of file                             |
| WCHAR       | *m_pwzRealPath; //Path to file                             |
| WCHAR       | *m_pwzParentId; //What the identifier for the container of |
| the file is |                                                            |
| WCHAR       | *m_pwzCreatorMachinelId; //on which machine the file is    |
|             | located                                                    |
| BOOL        | m_fIsFolder; //whether the file is actually a folder       |
| LONG        | m_cbSize; // what the size of the file is                  |
| FILETIME    | m_ftModification; //modification time                      |
| // optional |                                                            |
| WCHAR       | *m_pwzThumbnail; // thumbnail for the file                 |
| WCHAR       | *m_pwzUrlIconFile; // URL of Icon file                     |

---

As shown, the name and path to the file are included in the metadata of a transmitted shared file. In addition, an identifier for the container of the shared file is provided as well as the location of the machine on which the file resides. An indication of the type of file, the file size and modification date are also included. Optionally, the transmitted file may include a thumbnail and Uniform Resource Locator for the icon associated with the shared file.”

- Nurminen at, e.g., [0056] “Although the mobile terminal **120** generally does not send the lower

priority synchronization items using the first less desirable data connection, in one embodiment of the present invention, the mobile terminal **120** is configured to send metadata related to the lower priority synchronization items to the other electronic device using the first less desirable data connection, as also illustrated by block **350**. For example, the metadata may simply include an indication that a database entry has been inserted, updated, or deleted. In another example, the metadata may include a description of the inserted data, the update, or the deletion. In yet another example, the metadata may include a compressed or an otherwise abbreviated version of the synchronization item. The idea behind sending metadata is to create a smaller object from the large synchronization item that can be transferred using the first less desirable connection without adding considerable extra cost. Then, at a later time when the second more desirable connection becomes available, the original synchronization item can be transferred to the other electronic device. In this regard, where metadata related to a lower priority synchronization item is communicated using the first less desirable data connection, the lower priority synchronization item is still generally queued for transmission via the second more desirable data connection when such a data connection becomes available. As is described below in relation to FIG. 6, in one embodiment, once metadata is sent, the mobile terminal may receive an externally originated request for the item that may cause the mobile terminal to transfer the item even over the slower and/or more costly network. In one embodiment, the metadata provides a user of an external device the option of requesting the item over the slower and/or more costly network.” [0057] “In one embodiment, generic or application specific compressors/data extractors may be utilized for generating abbreviated versions of the synchronization item. For example, if a large image was added to the first mobile terminal's database **125**, a generic image compressor component may be used to create

a scaled-down version of the image. This scaled-down version may be sent to the other electronic device. Likewise, if the image includes annotation information or other metadata, such information may also be inserted in the scaled-down version. Where the synchronization item contains, for example, a large document where the format is dictated by a particular application, then the application may be used to provide an extractor that extracts the portions of the document so that the document portions may be transferred via a first less desirable connection. In this way, although the abbreviated representation of the document does not contain all the details of the large original field, it can still be interpreted by the application without any modifications to its logic. Application specific compressors and extractors may be embodied as software, such as software written in a plug-in like manner, provided in stored procedures, or provided in higher level languages than the synchronization script. Such compressors and extractors may be automatically called by the synchronization script. The creation of a smaller representation of the large synchronization data is sometimes referred to herein as the “deferral” of the data.”

- Sagar at, e.g., 4:64-5:7 “In one embodiment, the described system and process may use an atom feed with FeedSync for the synchronization process. Generally, an atom feed may be a set of items, where each item includes an attribute for referencing an enclosure (e.g., a link) and includes attributes that describe a relationship of the item with another item. The enclosure may be a link to data that represents, for example, a file. An example of a system using atom feeds is Atom Publishing Protocol or APP. This protocol may be widely used for atom feeds and some embodiments of the described method and system may implement the atom feed as APP.” 5:8-15 “The atom feed may be configured to include a link attribute that is used to specify a location of a enclosure. Since enclosures (files) can be of arbitrary size, they may not

be present inline in the FeedSync feeds but instead may be referenced out of band (e.g., using an attribute such as <link rel="enclosure">). The implication is that enclosures may be downloaded using a separate mechanism after a feed has been synchronized.” 5:16-31 “Feeds allow a hierarchical structure to be captured among feed items. For example, items in the feed that contain an enclosure (e.g., a file) are allowed to have a parent item which represents a folder. Such folder items in turn can have a parent folder. Multiple top level file or folder items are allowed. To preserve hierarchical structures, the feed may include a parent ID attribute that may be used to specify a hierarchy of files and folders to store a feed's enclosures on a device. This attribute, if present, may be required to have the same value of an FeedSync ID attribute of some other feed entry. The other entry may represent a parent folder and may also be required to have a link sub-element with an attribute type set to “folder.” If the sub-element is omitted, the entry may manifest as a top level file or folder on a device. Multiple top level files or folders may be allowed.”

- Moromisato at, e.g., 5:65–67 "The term “file” as used herein means a set of persistent data with associated metadata such as name, last modification date, or custom attributes specified by a user. " 4:12–17 "Still another embodiment may have a notification mechanism that alerts members when monitored objects change. Such a notification mechanism can alert the user by means of an alert mechanism such as an operating system task bar or side bar or another mechanism such as an audio alert, email alert or instant message alert." 7:67–8:8 "Each document share engine stores a “file descriptor” for each file in a synchronized folder. The file descriptor consists of the file metadata (such as filename, size, and modification time) and is used to keep track of the file in the collaborative system. During collaborative system operation, each engine also sends data change requests containing the file descriptors to each

collaborator in its respective shared space in order to keep all collaborators synchronized."

15:14–22 "If the download settings are such that file downloading is deferred to a later time, that file is represented in the OS file system as a "stub" file that is very similar to a Windows shortcut (.lnk file) in that it represents a target file with a small file that contains information necessary to find the target file. The stub file is represented in the user interface by the same icon as the target file, but it may have an overlay of some sort that indicates that it is a stub file in a manner similar to the hand overlay shown in FIG. 1." 16:49–51 "When the file arrives, the stub file representing the actual file is replaced by the actual file and the file system snapshot is updated to indicate that the file contents have changed."

- Aboulhosen et al., e.g., 2:17–23: "Whenever a shared file is modified, the file owner sends updated metadata for that file to the other members of the group. In this way, the members use their local file systems to represent each shared file of the group and defer downloading a shared file until it is accessed by a member who is not the file owner." 2:27–30: "For example, the metadata may identify the file name, the file owner, the create date of the file, the last modified date of the file, the size of the file, and so on." 2:37–42: "Whenever the actual file is changed at the file owner, the file owner sends the updated metadata for the file to the other members, who update the metadata associated with their corresponding virtual file as appropriate. The members, who are not the file owner, use the metadata when accessing the shared file." 6:60–7:4: "The member that updates the folder sends an out-of-synchronization message 621 to the group owner, unless it is the group owner itself that updates the folder. In response, the group owner sends a synchronization request message 622 to the updating member. The updating member responds with a synchronization response message 623 that contains the metadata describing the update. The group owner upon receiving the metadata updates to the metadata

associated with the shared folder. The group owner then notifies each other group member of the update. "

- Brown '847 at, e.g., Abstract "Distributed clients access the server, to learn about changes made to the files on the server, and to push local changes of the files onto the server. A synchronization application is used to synchronize the clients and server, synchronizing metadata and selected files." 7:44–50 "The client SA synchronizes with the server by synchronizing the client metadata with the server metadata. This is an ID-based process because SIDs are carried in both the client and server metadata. The client metadata has both a client and SID because a new file or directory is not assigned a SID until the file is uploaded or the directory is created on the server." 7:13–20 "During synchronization, when the client SA pulls down changes from the server and makes changes to the user's directory, it updates the client metadata to reflect those changes. Also, when the client SA pushes changes to the server during the second part of the synchronization process, it records new SID and FSI values returned by the server SFS into the client metadata file and directory items."
- Brown '826 at, e.g., Abstract "Distributed clients access the server, to learn about changes made to the files on the server, and to push local changes of the files onto the server. A synchronization application is used to synchronize the clients and server, synchronizing metadata and selected files." 7:45–50 "The client SA synchronizes with the server by synchronizing the client metadata with the server metadata. This is an ID-based process because SIDs are carried in both the client and server metadata. The client metadata has both a client and SID because a new file or directory is not assigned a SID until the file is uploaded or the directory is created on the server." 7:14–21 "During synchronization, when the client SA pulls down changes from the server and makes changes to the user's directory, it updates

the client metadata to reflect those changes. Also, when the client SA pushes changes to the server during the second part of the synchronization process, it records new SID and FSI values returned by the server SFS into the client metadata file and directory items.”

- Hesselink at, e.g., 25:44–49 “File overhead information pertains to file attributes or properties that may include but are not limited to file name, file size, folder status, last modified date, created date, etc. Every file or folder has file overhead information that can be used by local applications to get properties information about a file or a folder.” 27:29–46 “The user, through the use of local applications, such as application 72 a, for example, may make changes to the downloaded file overhead information as well as create or delete files and folders. When this occurs, user module 72 su sends any such modifications, creations and deletions to device module 74 sd. After receiving the modifications, creations and/or deletions, device module 74 sd consults the file overhead subscription information to identify any other connected user modules that are actively subscribed to the modified file overhead information. Subsequently, modifications of file overhead information are transmitted by device module 74 sd to all of the identified user modules. Further, device module 74 sd may also listen to file access events from its local operating system to detect any changes to files and folders that are actively subscribed by connected user modules. Device module 74 sd transmits any locally detected file overhead modifications (detected through the described listening) to all connected and actively subscribed user modules.”
- iFolder at, e.g., <https://www.novell.com/documentation/ifolder21/pdfdoc/admin/admin.pdf> at 194. “Every time the iFolder client logs in to the iFolder server, it compares filemaps (metadata that describes information about the actual file in your local iFolder) and dirmaps (metadata information on your local iFolder directory) between itself and the iFolder server. Filemaps

and dirmaps are located on the local workstation at c:\program files\novell\iFolder\username\home. If discrepancies are found between the filemaps and dirmaps, the iFolder client first downloads the new files from the server and then uploads any new local files.” <https://www.novell.com/documentation/ifolder21/pdfdoc/enduser/enduser.pdf> at 10–11. “4. When it is time to synchronize, the iFolder client reconciles changes in the iFolder directory with those on the server. It compares the metadata for the files and directories to determine if there have been changes since the last synchronization. 5. The iFolder server downloads any new files from the iFolder server to the local iFolder directory. ... It might transfer the entire file, depending on how the application in use saved file changes. ... 6. The iFolder client uploads any new files or changes to files from the local iFolder directory to the iFolder server. ... It might transfer the entire file, depending on how the application in use saved file changes. ... 7. The iFolder server receives the new files and increases its synchronization index. ... 9. Another of your iFolder client workstations connects to the iFolder server, ... . The file changes from the first workstation are downloaded to second. The file changes from the second are uploaded to the iFolder server. 10. When the iFolder server next synchronizes with the first workstation, it downloads the new data it received from the second workstation to the first workstation. In this way, iFolder captures information about the changes you make locally so that it can make those changes to files on the centralized iFolder server and, subsequently, to all your workstations.”

- BeInSync at, e.g., [https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 40.

When a file is received from one of your remote computers, BeInSync will create a stub (placeholder) for the file. The file will appear on your remote computers as follows:



You may always double-click on the file and select 'sync now' to move it up in the priority queue of your unsynced items.

Once the content of the file arrives it will automatically replace the placeholder file.

[https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 35.

|                               |                                                                                                                                          |                                                  |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| <b>When a file is updated</b> | Whenever a file is updated in an existing private or group share, you can turn this notification on so that you will be alerted of this. | By default this notification is turned <b>on</b> |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|

[https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 30.

Sync States

Sync and unsynced indications

Every item in the view area has an icon next to it indicating whether it's **synced**  or **unsynced** 

When a file is in its unsynced state you can either wait for it to synchronize or you can click on the **Sync Now** button.

- FolderShare at, e.g., <https://www.networkworld.com/article/2312749/syncing-with-foldershare.html> “The synchronization process can be continuous or on-demand. Once a library is created and the participating computers defined, the directory tree details for either end are exchanged. Files not yet transferred are allocated placeholders - empty files using the file's original full name (say, gearhead.doc) with the file type .p2p appended (thus our example becomes gearhead.doc.p2p). In continuous mode the files are eventually downloaded, while in on-demand or continuous modes opening a .p2p file forces its download. When the file is completely downloaded the original file name will be restored. The on-demand mode is very important for users on low-speed dial-up connections.”

<https://web.archive.org/web/20040511110758/http://www.foldershare.com/library/info/about>

[Placeholders.php](#) “When you connect to a FolderShare Library, the application creates a set of Placeholders on your computer that represent the files in the Library. ... If you connect to this library via Automatic Replication, the Placeholders will immediately begin to change into the actual files.”

- SharpCast at, e.g., <https://web.archive.org/web/20060829025503/http://www.sharpcast.com/products/photos/>  
“Your images and all metadata (captions, album names, etc.) are automatically secured online - you don’t have to do a thing!”
- GoodSync at, e.g., <https://web.archive.org/web/20060417215451/http://goodsync.com/synchronization.html>  
“File and folder metadata for each synchronization session is collected and stored in a database. . . . Recent changes are detected from file attributes, size and timestamp (not file modification time). . . . Can be used to synchronize data other than files: registry keys, database records, messages, contacts, playlists, etc.”
- Subversion at, e.g., <https://web.archive.org/web/20071116172448/http://svnbook.red-bean.com/en/1.4/svn-book.pdf> at 31. “Your Subversion repository is like a time machine. It keeps a record of every change ever committed, and allows you to explore this history by examining previous versions of files and directories as well as the metadata that accompanies them.” <https://web.archive.org/web/20071116172448/http://svnbook.red-bean.com/en/1.4/svn-book.pdf> at 331. “A new feature of Subversion is that you can attach arbitrary metadata (or “properties”) to files and directories. Properties are arbitrary name/value pairs associated with files and directories in your working copy. To set or get a property name, use the svn propset and svn propget subcommands. To list all properties on an object, use svn

proplist. For more information, see the section called ‘Properties’.”

- TortoiseCVS at, e.g., [https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 9. “An update merges the differences that have occurred in the remote CVS repository since your last update into your current local copy of the source code. The local TortoiseCVS client transmits the current local revision numbers to the remote CVS server, which then generates the differences between those revisions and the latest revisions available on the remote server. These differences will be applied to your current local source code files. Such differences are usually the edits that other developers have made to the source code, and committed to the remote CVS repository.” TortoiseCVS at, e.g., [https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 7 “Windows Explorer can be configured to display the CVS file attributes, in addition to the normal attributes shown.” TortoiseCVS at, e.g., [https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 8. “These additional CVS file attributes are very helpful in maintaining your local source code version.”
- Apple iDisk at, e.g., <https://flylib.com/books/en/3.343.1/> at 35. “When the synchronization process begins, you’ll see an iDisk icon appear on your Desktop, with just the plain name of ‘iDisk’ ... Your local iDisk, simply named iDisk, is created on your Desktop, while the progress meter keeps track of what’s being synced. The progress meter lets you know how far along you are in the synchronization process, while messages appear below the progress meter to let you know: How many files there are to synchronize; How many files have been synchronized; The name of the file currently being transferred”

- Apple iSync at, e.g., <https://flylib.com/books/en/3.343.1/> at 83. “And the synchronization process is a two-way street, meaning that the data on your Mac is compared with any existing data on the device. If there is a slight change or something new, iSync will alert you to those changes and allow them to be made to the data on your Mac.” <https://flylib.com/books/en/3.343.1/> at 86. “iSync's Safeguard window alerts you when it finds differences in data on the destination device when compared with the source device. ... [T]he Safeguard window is broken up into a table view with four columns :

- Device: This column lists the devices with which you're synchronizing data.
- Add: This column lists the number of new records that will be added to the device.  
...
- Delete: This column lists the number of records that will be deleted from the device. iSync determines whether a record will be deleted based on a previous sync. ...
- Modify: This column lists the number of records that will be modified on your Mac. When performing the sync, iSync looks at the records on your Mac and the device and compares the two. If it finds any changes, the number of records that will be modified is noted in the Modify column. ...

After reviewing the Safeguard panel, if you're okay with the changes that will be made, click on the Proceed button to allow the changes to be made to both devices.”

#### **(8) “Transfer Metadata Before File” References**

Certain Asserted Claims recite features including transferring metadata for files, where metadata for a given file is updated and transferred across devices, and specifically where the metadata is transferred before the file contents, which may involve assigning priorities to metadata

and to files so that a first priority for metadata is used to transfer metadata prior to transferring file content. Nothing about these claim limitations add anything novel or non-obvious over the prior art. To the contrary, this feature was well-known, routine, conventional, and widely available long before the alleged invention. As the attached claim charts demonstrate, this subject matter is disclosed in numerous references. For purposes of these contentions, references whose disclosures are cited in these contentions with respect to this claimed subject matter as reflected in the claim charts (including without limitation the specific references cited below) are hereby labeled the “Transfer Metadata Before File” references.

The implementation of the claimed features would have been obvious to a person of ordinary skill in the art least because doing so would have involved the combination of prior art elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the application of a known technique to a known device ready for improvement to yield predictable results, and/or would have been obvious to try. Moreover, a person of ordinary skill in the art would have readily appreciated the well-known and widespread use of these features and the associated benefits and advantages of their use. Persons of ordinary skill in the art would have been motivated to combine the teachings of these references with any system and/or disclosure that relates to the synchronization of files or other data content. Metadata was ubiquitous in the prior art along with file contents, and it was well-known to use metadata for many different advantageous features in file management and file synchronization systems. Among other features, it was well-known to separately and first transfer and synchronize metadata for files so that an indication and record of an updated file could be efficiently recorded and updated across devices first, quicker and more efficiently, before providing to transfer the complete file contents. It was also well known to transfer metadata before

the file to determine which portion of the file had been modified. For example, references disclose the following motivations to combine, such as the following as well as the other disclosures that are cited in the prior art claim charts submitted herewith:

- Kaasten at, e.g., 2:37-45 “Some files may be very large, such as audio or video clips. Instead of incurring the costs of storing such a file on every computing device, a user can choose to ‘ghost’ the file on some devices. A ghosting device stores only metadata about the file (such as its name and version date) rather than the entire file. The user can still access the file through the ghost: the access requests are sent to a device that holds the actual contents, and those contents are then presented to the user as if they were stored locally.” 4:54-5:7 “The situation just described does not apply to the second flavor of data synchronization, termed “ghosted synchronization.” Consider the very big file **140** in the folder A2 **116** on computing device A **102**. This very big file **140** may be a video clip or large database. A user wishes to access the very big file **140** when logged into the laptop **106**, but does not wish to incur the enormous costs of storing a copy of this file locally. Instead, a “ghost” file **142** is created on the laptop **106**. This ghost **142** does not contain the full contents of the very big file **140**. Minimally, the ghost file **142** only contains a reference to the very big file **140** on its host computing device A **102**. However, the ghost file **142** is synchronized with the very big file **140** so that a user on the laptop **106** can access the very big file **140** through the ghost file **142**. If the user wishes to change the contents of the very big file **140**, he can make that change on the laptop **106** just as if he were working directly with the very big file **140**. The change is then sent, by means of the ghost file **142**, to the computing device **102** and is applied to the very big file **140** itself. Thus, the user of the laptop **106** can read and modify the very big file **140** as if a synchronized copy of it existed on the laptop **106**.” 5:16-26 “These two flavors of data synchronization, full-

copy and ghosting, can be intermingled in a number of ways. A folder is set up for full-copy synchronization while a file within that folder is excluded from full-copy and is instead ghosted. A folder can be full-copy synchronized between two computing devices and ghosted to a third. A ghost can locally contain a copy of some attributes or metadata of the remote file. Such as its name, a thumbnail display of the file, and the like. Changes to these attributes are then synchronized in the same way that changes to a full-copy file are synchronized, while changes to the bulk of the file are synchronized by ghosting.”

- Savage at, e.g., [0031] “In one implementation, the scanning module 112 detects that a file on the synchronization client has changed, generates and schedules a synchronization report for that file, and transfers the synchronization report to the synchronization server 108 -at the appropriate time. In one implementation, the transfer may be accomplished by a transmission from the client to the server over a communications network. However, other implementations may transfer the synchronization report via an intermediary, such as a shared storage location, or by having the report read from the client by the server.” [0032] “Upon receipt of a synchronization report for a file, the synchronization server 108 checks to determine the state of the file within its own files system. If the synchronization server 108 needs to receive the file from the client in order to synchronize with the client, the synchronization server 108 request that the client upload the file to the server. An update module 118 uploads the requested file to the synchronization server 108 (see synchronizing files 120). Alternatively, if the synchronization server 108 determines a conflict (e.g., the client file is edited and the server file is also edited), the synchronization server 108 executes a conflict resolution (e.g., asking the user to chose which version of the file to record at both the client and the server). Upon synchronizing the file with the synchronization client 106, the synchronization server 108

transmits a synchronization confirmation to inform the synchronization client 106 that the synchronization is complete. Note: If there is an error in the synchronization, the synchronization confirmation can also be used to indicate this to the synchronization client 106.”

- Rao at, e.g., 7:48-60 “A database containing raw data and metadata is synchronized in two phases. In the first phase, metadata is synchronized between a server and client while the raw data remains on the server system. The client system, when connected or disconnected, may use and manipulate the metadata for any purpose whatsoever, including selecting one or more raw data files that are to be available on the client system. The second phase of synchronization is to synchronize the selected raw data to the client system. The synchronized raw data may be manipulated, changed, and used on the client system when disconnected. Upon reconnect, any changes made to the metadata or the raw data are synchronized between the systems.”
- Braginsky at, e.g., [0069] “FIG. 5 is a flow diagram of an embodiment of a metadata synchronization process 500. The metadata synchronization process may be performed separately for each share, or it may be performed for all shares to which a user has access rights. However, since different shares may be stored on different file servers, the following explanation is directed to synchronizing the metadata for a single share. The metadata synchronization process 500 is repeatedly performed periodically (e.g., once every N seconds, where N is in the range of 1 to 10) or episodically (e.g., in response to the occurrence of any one of a predefined set of trigger conditions). As explained in more detail below, some synchronization operations require more than one iteration of the metadata synchronization process 500 to be completed.” [0070] “In a first phase (operations 502-506), sometimes called the commit phase, the client system sends to the server all client metadata meta-directory

entries that have been modified by the client (502). In some embodiments, client meta-directory entries that have been modified are marked with an IS\_DIRTY flag. In some embodiments, the entire content of each changed meta-directory entry is sent to the server, while in other embodiments only changed fields of the entry are sent. The server receives the meta-directory entries from the client, identifies any received entries that conflict with entries in the server's corresponding meta-directory, and rejects the conflicting entries (i.e., the received entries that conflict with corresponding entries in the server's meta-directory) (504). In some embodiments, rejected entries are not processed and thus their changed fields are not applied to the corresponding entries in the server's meta-directory. The remaining client meta-directory entries, which do not conflict with entries in the server's corresponding meta-directory, are used to update the server's meta-directory (504). For example, the updating may be performed in some embodiments by updating changed fields in existing meta-directory entries, assigning file IDs to new meta entries and inserting the new meta-directory entries into the server's metadata table.” [0071] “The server may assign a version number to each new server meta-directory entry and each updated server meta-directory entry. Alternately, or in addition, it may store a timestamp in the server meta-directory entry to indicate the date and time of the last update made to the server meta-directory entry. The server meta-directory entries modified in response to the client meta-directory entries sent to the server are sent to the client (504). Optionally, the server may also send to the client information indicating which client meta-directory entries were rejected because they conflict with entries in the server's meta-directory.” [0072] “The client processes the received server meta-directory entries (506, 508). One aspect of this processing is identifying meta-directory entries that indicate revised file content located at the client, and scheduling content uploads of those files to the server

(506). In some embodiments, the meta-directory entries include a content checksum field, and an update flag for that field that indicates whether the content checksum field contains an updated value. When the upload for a file is completed, the server changes its meta entry to clear the update flag, and that update is copied to the corresponding client meta-directory entry during a next iteration of the metadata synchronization process. Another aspect of processing the received server meta-directory entries is updating or overwriting the corresponding client meta-directory entries with the information in the received server meta entries (508). For example, the received server meta entries may have new version numbers that need to be copied into the corresponding client meta entries. In summary, operations 502 through 508 synchronize updated client meta-directory entries with the corresponding server meta-directory entries, excluding client meta-directory entries that have updates that conflict with server meta-directory entries.” [0073] “Next, in a second phase of the metadata synchronization process, sometimes called the get updates phase, the client requests from the server copies of all server meta-directory entries revised since the last metadata synchronization (510). As noted above, each meta-directory entry includes a timestamp (synch\_ts) indicating the last time the entry was changed on the server. The server identifies and sends to the client the requested server meta-directory entries (512). For ease of explanation, separate discussions are provided for the handling of server meta-directory entries that do not conflict with client meta-directory entries and for the handling of those server meta-directory entries that do conflict with client meta-directory entries. It may be noted that the server meta-directory entries sent to the client at 512 include server meta-directory entries corresponding to any client meta-directory entries rejected by the server at 504 due to conflicting updates.” [0074] “When a received server meta-directory entry does not conflict

with any corresponding client meta-directory entries (i.e., entries having the same file ID and/or the same filename), the metadata changes in the server meta-directory entry are applied to the corresponding client meta-directory entry (514). Alternately, the received server meta-directory entry is used to overwrite the corresponding client meta-directory entry, if any. When there is no corresponding client meta-directory entry, a new client meta-directory entry is generated. In addition, if the server meta-directory entry indicates revised file content located at the server, the client schedules a content download of the files from the server (514).” [0075]

“When a received server meta-directory entry conflicts with one or more corresponding client meta-directory entries (i.e., entries having the same file ID and/or the same filename), the process requires a user to resolve the conflict (516). As noted above, the user may resolve the conflict by selecting a client or server version of a file (and its metadata) as the “winner,” in which case the losing file and/or its metadata will be overwritten by the winning file and/or its metadata, or the user may rename or move the conflicting client file so as to eliminate the conflict. Changes are applied to the client meta-directory in accordance with the user specified resolution of the conflict (516). This may include deleting or revising one or more client meta-directory entries. In addition, the client schedules any file content uploads or downloads needed to implement the user specified resolution of the conflict (516). For instance, when the server meta-directory entry is selected by the user as the winning entry, and that entry includes an update flag or other data that indicates that the content of the corresponding server file is new or updated, a file download is scheduled. More generally, if the server meta-directory entry survives the conflict resolution process and includes an update flag or other data that indicates that the content of the corresponding server file is new or updated, a file download is scheduled. On the other hand, if the conflict resolution process results in a client meta-directory entry that

includes an update flag or other data that indicates that the content of the corresponding client file is new or updated, a file upload will be scheduled during the next metadata synchronization cycle (i.e., when operations 502-506 are next performed).”

- Shappell at, e.g., Abstract “A computer implemented method and system enable users to share files in a server-less shared space. By providing access to such 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.” 12:22-29 “For sharing files among group members, metadata is transmitted to all members when files become available, such as is shown in the steps 1904 and 1906 in the flow chart in FIG. 19. Such metadata are sufficient to render an icon of the shared file and may include date, time, thumb-nail information, name size, and optionally the source of the information. This may include an identifier for the source computing machine and the creator identity.” 12:40-65 “The following table illustrates one such implementation of required and optional metadata concerning a shared file.

---

|             |                                                            |
|-------------|------------------------------------------------------------|
| // required |                                                            |
| WCHAR       | *m_pwzFilename; //Name of file                             |
| WCHAR       | *m_pwzRealPath; //Path to file                             |
| WCHAR       | *m_pwzParentId; //What the identifier for the container of |
| the file is |                                                            |
| WCHAR       | *m_pwzCreatorMachineId; //on which machine the file is     |
|             | located                                                    |
| BOOL        | m_flFolder; //whether the file is actually a folder        |
| LONG        | m_cbSize; // what the size of the file is                  |
| FILETIME    | m_ftModification; //modification time                      |
| // optional |                                                            |
| WCHAR       | *m_pwzThumbnail; // thumbnail for the file                 |
| WCHAR       | *m_pwzUrlIconFile; // URL of Icon file                     |

---

As shown, the name and path to the file are included in the metadata of a transmitted shared file. In addition, an identifier for the container of the shared file is provided as well as the location of the machine on which the file resides. An indication of the type of file, the file size

and modification date are also included. Optionally, the transmitted file may include a thumbnail and Uniform Resource Locator for the icon associated with the shared file.” 12:66-13:13 “To inform a user of a group shared space file that is not present locally, display icons are presented to the user. FIG. 15 illustrates a graphical interface 1500 for a Shared Space folder that contains files that are not local. As shown, a left hand pane 1502 presents a display icon 1504 to denote a “missing” file that is available on another system. This icon is ghosted or semi-transparent. As is shown in the flow chart in FIG. 20, the system renders such a display upon receipt that metadata concerning a shared file is available in the peer network, as is shown in method step 2002. Other files that are “missing” may be unavailable to the user, such as when a system that is hosting the file of interest is offline. In this instance, a display icon 1506 is presented as a ghosted icon that includes a small image of a red “X” or other suitable visual indicia that the file is unavailable to the user.” 13:14-19 “Additionally, because content in a group folder is likely to change over time and the organization of content does not enable the user to discern what content is new or has been recently altered, a “new” glyph may be presented on folders and files that are new or have been recently changed. This glyph provides a visual indicator of new content.” 13:20-36 “When a modification of a file contained in a shared space occurs, a notification to other members in the group occurs (see steps 1902, 1904 and 1906 in FIG. 19). In a preferred embodiment, upon receipt of such notification, member machines that have previously obtained a copy of the shared file will remove the local copy of the out-of-date file. This will change the visual representation of the file to a “missing” file within the shared space folder when the application setting does not automatically update files, as is shown in step 2010 in FIG. 20. On the other hand, if automatic replication of the shared file is enabled, the local machine obtains an updated version of the file, as is shown in step

2006 in FIG. 20. Finally, an appropriate visual indicator is presented to the user as shown in step 2008. In order for a group space member to access an updated that is not locally stored, the file must first be transmitted to the local machine.”

- Nurminen at, e.g., [0056] “The idea behind sending metadata is to create a smaller object from the large synchronization item that can be transferred using the first less desirable connection without adding considerable extra cost. Then, at a later time when the second more desirable connection becomes available, the original synchronization item can be transferred to the other electronic device. In this regard, where metadata related to a lower priority synchronization item is communicated using the first less desirable data connection, the lower priority synchronization item is still generally queued for transmission via the second more desirable data connection when such a data connection becomes available. As is described below in relation to FIG. 6, in one embodiment, once metadata is sent, the mobile terminal may receive an externally originated request for the item that may cause the mobile terminal to transfer the item even over the slower and/or more costly network. In one embodiment, the metadata provides a user of an external device the option of requesting the item over the slower and/or more costly network.” [0057] “In one embodiment, generic or application specific compressors/data extractors may be utilized for generating abbreviated versions of the synchronization item. For example, if a large image was added to the first mobile terminal's database 125, a generic image compressor component may be used to create a scaled-down version of the image. This scaled-down version may be sent to the other electronic device. Likewise, if the image includes annotation information or other metadata, such information may also be inserted in the scaled-down version. Where the synchronization item contains, for example, a large document where the format is dictated by a particular application, then the

application may be used to provide an extractor that extracts the portions of the document so that the document portions may be transferred via a first less desirable connection.” [0060] “The second mobile terminal **130** receives the higher priority synchronization items as well as the metadata for the lower priority synchronization items and updates the replicated database **135** stored in the memory of the second mobile terminal **130**, as illustrated by block **420**. If the user of second mobile terminal **130**, after viewing the metadata related to the lower priority synchronization items, decides that he or she would like the lower priority synchronization items communicated to the second mobile terminal **130** without waiting until the first mobile terminal **120** finds a more desirable data connection, then the user may communicate a request **435** to the first mobile terminal **120** for immediate transmission of a lower priority synchronization item, as represented by block **430**. The first mobile terminal **120** may receive this request **435**, as represented by block **440**, and, in response to the request, send the requested lower priority synchronization item **455** to the recipient device, as represented by block **450**. The second mobile terminal **130** receives the requested lower priority synchronization item and updates its database **135** accordingly.”

- Sagar at, e.g., 6:4-16 “FIG. 4 illustrates a synchronization process embodiment. At block 401, an item in a feed is updated (or created) on a device or node. MOE may sense that this item is updated (or created) 402. The sensing may be performed, for example, using a watcher component. A watcher component may be a monitoring component that is configured to receive update indications from a node of the mesh and facilitate alerting other nodes of the mesh about the update. The watcher component may further be configured to facilitate the update process by executing certain functions. The watcher may provide indications of updates along with the content of an update using feeds, such as an atom feed.” 6:17-23 “In some

embodiments, the node may be configured to transmit a change or update indication to the storage service component when a change or modification has been made to the synchronized folder (e.g., a folder or item has been updated, moved, or deleted) of the local node. The watcher may be implemented as part of each node of the mesh or part of a service or both.”

6:24-40 “Node updates may be received at the storage service at block 403. The storage service may use FeedSync attributes to determine what changes should be brought up to the service, and logically pull those device-side changes MOE into the storage service. Note that although this process may logically appear to be a service-side pull, the retrieval may in fact be implemented via a device-side push because from a security perspective the service may not have permission to actually pull changes from the client. When the Storage Service “pulls”, the Storage Service may receive a batch of updates with only portions of the updated feed that have been changed since the last time the service-based copy of the feed was updated by this device. This state, or knowledge (semantics defined by FeedSync rather than a node level application), may then be stored on the service (e.g., one entry for each mesh device). Each entry may contain state for a direction of data: a client pull direction and a service pull direction.”

6:41-44 “The Storage Service may merge the item changes into its feed at block 404. For example, the update data is merged with a storage service copy of the feed to result in an updated feed.”

6:45-52 “The update of the local feed may initiate generation of a notification that may be enqueued to the notification service at block 405 for processing. At block 406, the Notification Service may inform other devices that updates are available and may prompt nodes to pull changes down. Because the system implements a feed service, devices that come online (e.g., connect with the mesh) may receive updates once they subscribe for the notifications.”

6:61-7:2 “Devices may use the same FeedSync algorithm to pull down

changes (e.g., just the new updates) from the Storage Service and merge the changes or updates into their local copies of the feed at block 407. Nodes may realize the new feed or updates at block 408. Devices may also locate and replicate enclosures over if required at block 409 (whether from peers or from the Storage Service, as illustrated in FIG. 3, depending upon communications topology and where it's dynamically determined to be best to retrieve the enclosures).” 7:5-16 “Processing of a feed received from the Storage Service may be performed in two blocks. The update items in the feed from the Storage Service may be merged with a local copy of the feed at a node. In one embodiment, merging the feed data may include associating or matching each item of the feed with corresponding items of the local feed copy based on an identifier of each item. In one embodiment, merging updates with the local feeds includes modifying the local feed to incorporate the updates. This may include overriding local feed information with the updates. In the described system, where the updates are incremental updates, the local feed may simply be updated to incorporate only the most recent changes.”

- Moromisato at, e.g., 4:12–17 “Still another embodiment may have a notification mechanism that alerts members when monitored objects change. Such a notification mechanism can alert the user by means of an alert mechanism such as an operating system task bar or side bar or another mechanism such as an audio alert, email alert or instant message alert.” 7:67–8:8 “Each document share engine stores a “file descriptor” for each file in a synchronized folder. The file descriptor consists of the file metadata (such as filename, size, and modification time) and is used to keep track of the file in the collaborative system. During collaborative system operation, each engine also sends data change requests containing the file descriptors to each collaborator in its respective shared space in order to keep all collaborators synchronized.” 15:14–22 “If the download settings are such that file downloading is deferred to a later time,

that file is represented in the OS file system as a “stub” file that is very similar to a Windows shortcut (.lnk file) in that it represents a target file with a small file that contains information necessary to find the target file. The stub file is represented in the user interface by the same icon as the target file, but it may have an overlay of some sort that indicates that it is a stub file in a manner similar to the hand overlay shown in FIG. 1.” 16:49–51 “When the file arrives, the stub file representing the actual file is replaced by the actual file and the file system snapshot is updated to indicate that the file contents have changed.”

- Hesselink at, e.g., 25:44–49 “File overhead information pertains to file attributes or properties that may include but are not limited to file name, file size, folder status, last modified date, created date, etc. Every file or folder has file overhead information that can be used by local applications to get properties information about a file or a folder.” 27:47–62 “When the user in the example above browses through remote directory structures as described above, user module 72su may only subscribe to file overhead information of the associated files and folders that are browsed. At this point, the local application 72a being used by the user may not yet have requested user module 72su to perform any data access functionalities such as a read operation of the actual binary contents of the files, for example. At this time, the user may only be browsing or navigating through files and directories before actually accessing any file binary contents. Consequently, no file binary contents are downloaded at this time during the process. This avoids unnecessary data transmissions and associated network bandwidth requirements. Once a request is made by an application to perform an operation on the actual file data, however, a file data Subscription and management process is run by the system.”
- iFolder at, e.g., <https://www.novell.com/documentation/ifolder21/pdfdoc/admin/admin.pdf> at 194. “Every time the iFolder client logs in to the iFolder server, it compares filemaps (metadata

that describes information about the actual file in your local iFolder) and dirmaps (metadata information on your local iFolder directory) between itself and the iFolder server. Filemaps and dirmaps are located on the local workstation at c:\program files\novell\iFolder\username\home. If discrepancies are found between the filemaps and dirmaps, the iFolder client first downloads the new files from the server and then uploads any new local files.” <https://www.novell.com/documentation/ifolder21/pdfdoc/enduser/enduser.pdf> at 10–11. “4. When it is time to synchronize, the iFolder client reconciles changes in the iFolder directory with those on the server. It compares the metadata for the files and directories to determine if there have been changes since the last synchronization. 5. The iFolder server downloads any new files from the iFolder server to the local iFolder directory. ... It might transfer the entire file, depending on how the application in use saved file changes. ... 6. The iFolder client uploads any new files or changes to files from the local iFolder directory to the iFolder server. ... It might transfer the entire file, depending on how the application in use saved file changes. ... 7. The iFolder server receives the new files and increases its synchronization index. ... 9. Another of your iFolder client workstations connects to the iFolder server, ... . The file changes from the first workstation are downloaded to second. The file changes from the second are uploaded to the iFolder server. 10. When the iFolder server next synchronizes with the first workstation, it downloads the new data it received from the second workstation to the first workstation. In this way, iFolder captures information about the changes you make locally so that it can make those changes to files on the centralized iFolder server and, subsequently, to all your workstations.”

- BeInSync at, e.g., [https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 40.

When a file is received from one of your remote computers, BeInSync will create a stub (placeholder) for the file. The file will appear on your remote computers as follows:

 Beinsync\_file.bis

You may always double-click on the file and select 'sync now' to move it up in the priority queue of your unsynced items.

Once the content of the file arrives it will automatically replace the placeholder file.

[https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 35.

|                               |                                                                                                                                          |                                                  |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|
| <b>When a file is updated</b> | Whenever a file is updated in an existing private or group share, you can turn this notification on so that you will be alerted of this. | By default this notification is turned <b>on</b> |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|

[https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 30.

### Sync States

#### Sync and unsynced indications

Every item in the view area has an icon next to it indicating whether it's **synced**   or **unsynced** 

When a file is in its unsynced state you can either wait for it to synchronize or you can click on the **Sync Now** button.

- FolderShare at, e.g., <https://www.networkworld.com/article/2312749/syncing-with-folder-share.html> “The synchronization process can be continuous or on-demand. Once a library is created and the participating computers defined, the directory tree details for either end are exchanged. Files not yet transferred are allocated placeholders - empty files using the file's original full name (say, gearhead.doc) with the file type .p2p appended (thus our example becomes gearhead.doc.p2p). In continuous mode the files are eventually downloaded, while in on-demand or continuous modes opening a .p2p file forces its download. When the file is completely downloaded the original file name will be restored. The on-demand mode is very important for users on low-speed dial-up connections.”

<https://web.archive.org/web/20040511110758/http://www.foldershare.com/library/info/about>

[Placeholders.php](#) “When you connect to a FolderShare Library, the application creates a set of Placeholders on your computer that represent the files in the Library. ... If you connect to this library via Automatic Replication, the Placeholders will immediately begin to change into the actual files.”

- Apple iDisk at, e.g., <https://flylib.com/books/en/3.343.1/> at 35. “When the synchronization process begins, you'll see an iDisk icon appear on your Desktop, with just the plain name of ‘iDisk’ ... Your local iDisk, simply named iDisk, is created on your Desktop, while the progress meter keeps track of what's being synced. The progress meter lets you know how far along you are in the synchronization process, while messages appear below the progress meter to let you know: How many files there are to synchronize; How many files have been synchronized; The name of the file currently being transferred”
- Apple iSync at, e.g., <https://flylib.com/books/en/3.343.1/> at 83. “And the synchronization process is a two-way street, meaning that the data on your Mac is compared with any existing data on the device. If there is a slight change or something new, iSync will alert you to those changes and allow them to be made to the data on your Mac.”  
<https://flylib.com/books/en/3.343.1/> at 86. “iSync's Safeguard window alerts you when it finds differences in data on the destination device when compared with the source device. ... [T]he Safeguard window is broken up into a table view with four columns :

- Device: This column lists the devices with which you're synchronizing data.
- Add: This column lists the number of new records that will be added to the device.

...

- Delete: This column lists the number of records that will be deleted from the device. iSync determines whether a record will be deleted based on a previous sync. ...
- Modify: This column lists the number of records that will be modified on your Mac. When performing the sync, iSync looks at the records on your Mac and the device and compares the two. If it finds any changes, the number of records that will be modified is noted in the Modify column. ...

After reviewing the Safeguard panel, if you're okay with the changes that will be made, click on the Proceed button to allow the changes to be made to both devices.”

#### **(9) “Graphical Availability Indication” References**

Certain Asserted Claims recite features including presenting a graphical availability indication that an updated version of a file is available to be transferred (which may be presented proximate a file icon representing the file), which may be caused by transfer of metadata from one device to another device that causes the display of the graphical indication. Nothing about these claim limitations add anything novel or non-obvious over the prior art. To the contrary, this feature was well-known, routine, conventional, and widely available long before the alleged invention. As the attached claim charts demonstrate, this subject matter is disclosed in numerous references. For purposes of these contentions, references whose disclosures are cited in these contentions with respect to this claimed subject matter as reflected in the claim charts (including without limitation the specific references cited below) are hereby labeled the “Graphical Availability Indication” references.

The implementation of the claimed features would have been obvious to a person of ordinary skill in the art least because doing so would have involved the combination of prior art

elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the application of a known technique to a known device ready for improvement to yield predictable results, and/or would have been obvious to try. Moreover, a person of ordinary skill in the art would have readily appreciated the well-known and widespread use of these features and the associated benefits and advantages of their use. Persons of ordinary skill in the art would have been motivated to combine the teachings of these references with any system and/or disclosure that relates to the synchronization of files or other data content. Among other features, it was well-known to provide a graphical availability indication that an updated version of a file is available to be transferred to communicate that information to a user in an efficient manner. For example, references disclose the following motivations to combine, such as the following as well as the other disclosures that are cited in the prior art claim charts submitted herewith:

- Salas at, e.g., 3:62-4:3 “Objects may also be represented by a table which includes as fields identification codes for each data object, one or more flags which are used to distinguish various objects, one or more flags which are used to determine the behavior of objects (editability, searchability, and others), a field indicating the date the object was created, a field indicating who created the object, a field identifying the parent of the object, and a field identifying the date the object was last modified, among others.” 4:37-54 “However, the database **20** stored on the client workstation **12** may contain tables which are not stored by the server database **20**. For example, a client workstation **12** may store in its database an “unread” table which indicates which objects have been modified since the user of the client workstation **12** has last accessed those objects. An unread table may include a member identification field and a modification tag indicating the last modification date and time of an object. All records

may be read from this table to identify to the client workstation **12** every item in a particular eRoom page which has not been read by the user, or a selective database query may be done to return only those objects belonging to a particular set of eRoom pages that have not been read by the user. If it is desired to provide this functionality, an additional entry in the unread table must be made to allow data objects to be distinguished based on some indication of affiliation.” 6:4-31 “In the embodiment shown in FIG. **4**, three icons are provided: an “icon display” icon **494** (currently selected) which causes items to be displayed as large icons with identifying text underneath; a “list display” icon **496** which causes items to be displayed as small icons with identifying text to one side of the icon; and a “report display” icon which causes items to be displayed as a list. The displayed list may be alphabetized, ordered by size of item, ordered by creation date, ordered by modification data, or ordered by some other data field associated with each item. Items in the item box may include a graphical indication that it, or items contained within it, are unread. This may imply that the item has been newly created, or the item may have been modified since the viewing user last read it. In either event, the graphical indication signals the user that the item should be read. In FIG. **4**, the “Brainstorms” folder **482** has an indication **484** that it is unread. eRoom pages also may include a shortcut bar **410**. The shortcut bar is a list of shortcuts which provide the viewer with a convenient way to access other eRoom pages. For example, in the embodiment shown in FIG. **4**, a shortcut to the directory of eRooms is provided, as well as shortcuts to the page currently viewed **414** and a shortcut **416** the folder **482** displayed in the item box **408**. The folder shortcut **416** includes a graphical indication that there are unread items in the folder **417**. The shortcut to the front page of the eRoom currently being viewed **414** also includes a graphical indication that unread items exists in the page **415**.”

- Matsubara at, e.g., [0027] “In a step **105**, the notification server **14**, in response to receiving the message from the management server **12**, communicates an alert message to each peer client that has subscribed to File-X. In the example shown in FIG. **1**, peer client A is notified by the notification server that File-X has been modified. In a particular implementation, the technology for communicating the alert message is known as “push” technology, also known as “webcasting” and “Point Casting” by which information is sent directly to the computer system rather than being retrieved by the computer. The alert message can include information indicating the time the operation was executed, the name of the file/folder, the type of operation (creating folder, deleting folder, registering file, deleting file, updating file, etc.), a user ID of the user who executed the operation, an ID of the virtual folder where the folder/file is located.” [0029] “In a step **106**, the peer client software of peer client **22a** can be configured to display an appropriate alert indication in response to receiving the alert message from the notification server **14**, to inform the peer client as to the fact of the modified File-X. As noted above, the alert message can include location information as to the location of the modified file.” [0048] “FIG. **4** depicts an example of a peer client user interface for viewing alert messages. A window **302** displays a “buddy list” which shows presence information for one or more peer members in the P2P network. The notion of presence information is discussed in greater detail in U.S. application Ser. No. 10/411,941. A window **304** provides a list of all the alerts that were sent from the notification server **14**. This list can be ordered by time of receipt of the; alert. A window **303** can be provided which presents the received alerts based on category. For example, the alerts can be arranged based on the type of information that is being alerted, such as conference meeting alerts, file modification alerts, directory modification alerts (e.g., some changes the access permission of a directory). Alerts can be sorted according to the peer

member, to show the activity occurring on a particular peer member's machine.” [0052] “The user can click a button **503** to jump to the information source. When this button is clicked, a GUI window for viewing the information source can be opened to that particular information source. The user can presented with a virtual space/folder where the originating information source of the alert is located. An icon or text of the actual object (conference sessions, BBS messages, files, etc.) can be highlighted to identify the object of the alert as it appears in the virtual space. It can be appreciated that a similar “jump” action can occur when the user clicks on the button **504.**” [0040] “FIG. 3 is a schematic representation of an illustrative example for a peer client user interface **201** to view information sources for conferences. A graphic representing a window **202** contains a directory tree that shows the structure of a folder directory. Conferences are typically organized into “conference rooms. Each conference room holds one or more “conference sessions.” Conferences can be represented in much the same way that files are represented, i.e., hierarchically. Conference rooms can be represented by folders (conference folders) in a graphical representation of a directory. Each conference folder may contain zero or more objects which represent conference sessions. A conference session can be represented by different icons to represent different states of the conference session; e.g., active, inactive, voting, and so on. Thus, in a graphical representation of a directory structure, some of the folders displayed may be conference folders, within which are documents (objects) which represent conference sessions.”

- Kaasten at, e.g., 5:21-26 “A ghost can locally contain a copy of some attributes or metadata of the remote file. Such as its name, a thumbnail display of the file, and the like. Changes to these attributes are then synchronized in the same way that changes to a full-copy file are synchronized, while changes to the bulk of the file are synchronized by ghosting.”

- Rao at, e.g., 5:24-30 “File 140 is not synchronized with file 120, but is shown as ‘ghosted.’ File 140 may have some attributes of a file, such as a placeholder in a directory system, but the raw data underlying file 140 may not be available on the client system 104. When the client system 104 is disconnected from server system 102, the item 138 that relates to file 140 may be available, but the raw data in the file 140 may not be available.” 8:9-13 “While the systems are disconnected in state 325, placeholders may be used for the files 322 and 324. The files are ‘ghosted’ and may or may not be seen in the directory structure. For example, ghosted files may be displayed as grayed out or semi-transparent.”
- Braginsky at, e.g., [0041] “The client UI 212 displays file status information to the user, prompts the user for information and allows the user to issue commands, such as commands for monitoring shares (groups of files to which the user has access rights), changing users or exiting the file system 116. In embodiments using WINDOWS, most interaction with the user can be done using a taskbar icon, sometimes called a quick launch icon or system tray icon. The taskbar icon can use different images to indicate different connectivity and synchronization states. For instance, the taskbar icon can indicate whether there are unresolved conflicts. When the user right-clicks on the icon, the user is presented with a menu from which different commands can be selected. In addition to a taskbar icon, the user can be presented with notification panels to provide messages such as upload/download status and the like.” [0042] “In some embodiments, the client UI 212 communicates with the file system 116 over a socket and displays status information to the user. For example, in some embodiments, when a file is being downloaded, the client UI 212 presents a progress indicator (e.g., a progress bar) and an estimated time of completion for the file. Whenever the file system 116 needs to get input from the user, it requests the information from the client UI 212, which then prompts the

user for the requested information. The client UI 212 can also send user initiated requests to the file system 116, such as mounting a new share or shutting down the file system 116. In some embodiments, the client UI 212 can be used to schedule and manage automatic updating from the server system 104, as described with respect to FIG. 5.” [0075] “When a received server meta-directory entry conflicts with one or more corresponding client meta-directory entries (i.e., entries having the same file ID and/or the same filename), the process requires a user to resolve the conflict (516). As noted above, the user may resolve the conflict by selecting a client or server version of a file (and its metadata) as the “winner,” in which case the losing file and/or its metadata will be overwritten by the winning file and/or its metadata, or the user may rename or move the conflicting client file so as to eliminate the conflict. Changes are applied to the client meta-directory in accordance with the user specified resolution of the conflict (516). This may include deleting or revising one or more client meta-directory entries. In addition, the client schedules any file content uploads or downloads needed to implement the user specified resolution of the conflict (516). For instance, when the server meta-directory entry is selected by the user as the winning entry, and that entry includes an update flag or other data that indicates that the content of the corresponding server file is new or updated, a file download is scheduled. More generally, if the server meta-directory entry survives the conflict resolution process and includes an update flag or other data that indicates that the content of the corresponding server file is new or updated, a file download is scheduled. On the other hand, if the conflict resolution process results in a client meta-directory entry that includes an update flag or other data that indicates that the content of the corresponding client file is new or updated, a file upload will be scheduled during the next metadata synchronization cycle (i.e., when operations 502-506 are next performed).”

- Shappell at, e.g., 5:65-6:14 “In accordance with the invention, a shared space is presented through a graphical user interface with the look and feel of existing file system features. For example, the invention may use Windows Explorer-style dialog boxes such as the Open dialog box or the like. Such file system operations are accessible through the created shared space. In one embodiment, the invention is implemented as a namespace extension to Windows Explorer that presents user interface elements in a similar or same manner as a file system would create those elements, such as presenting display windows, icons and other graphic presentations to render the appearance of a file system. For example, the invention may implement namespace extensions and basic folder object interfaces that enable Microsoft Windows Explorer file system capabilities. The application similarly enables various operations that are similar to accepted file system operations to be performed with respect to data residing in the shared space.” 9:46-54 “Unlike a traditional file system, a shared space allows two or more group members to contribute to a file with the same name. To permit the user to differentiate between these files, the user may select an option provided in the Settings dialog 900, shown in FIG. 9 as a File/Group Names option 902. By selecting a “Display contributor name with files” setting, the user may view the contributor name as a part of the filename. In a preferred embodiment, the default setting is a “Display only the name for files” setting.” 12:66-13:13 “To inform a user of a group shared space file that is not present locally, display icons are presented to the user. FIG. 15 illustrates a graphical interface 1500 for a Shared Space folder that contains files that are not local. As shown, a left hand pane 1502 presents a display icon 1504 to denote a “missing” file that is available on another system. This icon is ghosted or semi-transparent. As is shown in the flow chart in FIG. 20, the system renders such a display upon receipt that metadata concerning a shared file is available in the peer network, as is shown

in method step 2002. Other files that are “missing” may be unavailable to the user, such as when a system that is hosting the file of interest is offline. In this instance, a display icon 1506 is presented as a ghosted icon that includes a small image of a red “X” or other suitable visual indicia that the file is unavailable to the user.” 13:14-19 “Additionally, because content in a group folder is likely to change over time and the organization of content does not enable the user to discern what content is new or has been recently altered, a “new” glyph may be presented on folders and files that are new or have been recently changed. This glyph provides a visual indicator of new content.” 13:20-36 “When a modification of a file contained in a shared space occurs, a notification to other members in the group occurs (see steps 1902, 1904 and 1906 in FIG. 19). In a preferred embodiment, upon receipt of such notification, member machines that have previously obtained a copy of the shared file will remove the local copy of the out-of-date file. This will change the visual representation of the file to a “missing” file within the shared space folder when the application setting does not automatically update files, as is shown in step 2010 in FIG. 20. On the other hand, if automatic replication of the shared file is enabled, the local machine obtains an updated version of the file, as is shown in step 2006 in FIG. 20. Finally, an appropriate visual indicator is presented to the user as shown in step 2008. In order for a group space member to access an updated that is not locally stored, the file must first be transmitted to the local machine.” 13:46-57 “For adding to or removing files or directories from a shared space, the user may right-click an icon associated with the content of interest to present a graphical interface, such as a Context Menu 1700 shown in FIG. 17. The Context Menu 1700 is presented with the look and feel of a file system menu having various user selectable menu options such as Explore, Open, Search and the like. In addition, the Menu includes a “Share with” menu item 1702. By user selection of this item, a drop-down

list of available groups is presented. Each shared space that contains the selected file or directory will include a graphical representation such as a check mark 1704 as shown in FIG. 17.”

- Nurminen at, e.g., [0057] “In one embodiment, generic or application specific compressors/data extractors may be utilized for generating abbreviated versions of the synchronization item. For example, if a large image was added to the first mobile terminal's database 125, a generic image compressor component may be used to create a scaled-down version of the image. This scaled-down version may be sent to the other electronic device. Likewise, if the image includes annotation information or other metadata, such information may also be inserted in the scaled-down version. Where the synchronization item contains, for example, a large document where the format is dictated by a particular application, then the application may be used to provide an extractor that extracts the portions of the document so that the document portions may be transferred via a first less desirable connection.”
- Sagar at, e.g., 11:22-38 “The described system may use the concept of ghosting or ghost files to help reduce the possibility of file synchronization conflicts. Generally, a ghost file may represent a placeholder for items (folders and files) when an item has been updated and the system is in a process of downloading or realizing that update or when an old file is available and an update is unavailable. In some systems, a zero byte placeholder file is used to represent a ghosted item while a “.ghost” file may be created to indicate a ghost display (e.g., a ghost icon). The ghosted file may alert a user that a file that the user is about to modify is already in a transition state. While there may exist narrow windows of time where ghosting may not catch a concurrency issue, ghosting may help to reduce the possibility of this conflict. For example, when a user manages to begin editing an item in a local synchronized folder before an update

notification is broadcast to or received at the local node, a ghosting indication for the item may be initiated.”

- Moromisato at, e.g., 17:3–9 "Once a folder has been synchronized, the folder icon displayed in the OS file system user interface display that represents that file is provided with an icon overlay that indicates it is synchronized through the collaborative system. The icon overlay can be similar to the “hand” overlay 114 (FIG. 1) displayed by Windows XP Pro file system when a folder is shared over the network. The synchronized folder icon overlay is provided by an icon overlay shell extension, which is only available on Windows 2000 or later." 3:62–64 "In still another embodiment, the display of the operating system folder may be augmented with an area that displays meta-information about the shared folder or workspace." 1:39–46 "In one version of this system data that is being collaboratively modified may be stored centrally on the server. Then, each collaborator that wants to modify the data sends information to the server to effect a change in the server data. The server modifies its copy of the data and then sends information, synchronously or asynchronously, representing a “view” of the modified data to all collaborators, so that each collaborator can display the data locally."
- BeInSync at, e.g., [https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 30.

#### Sync States

##### Sync and unsynced indications

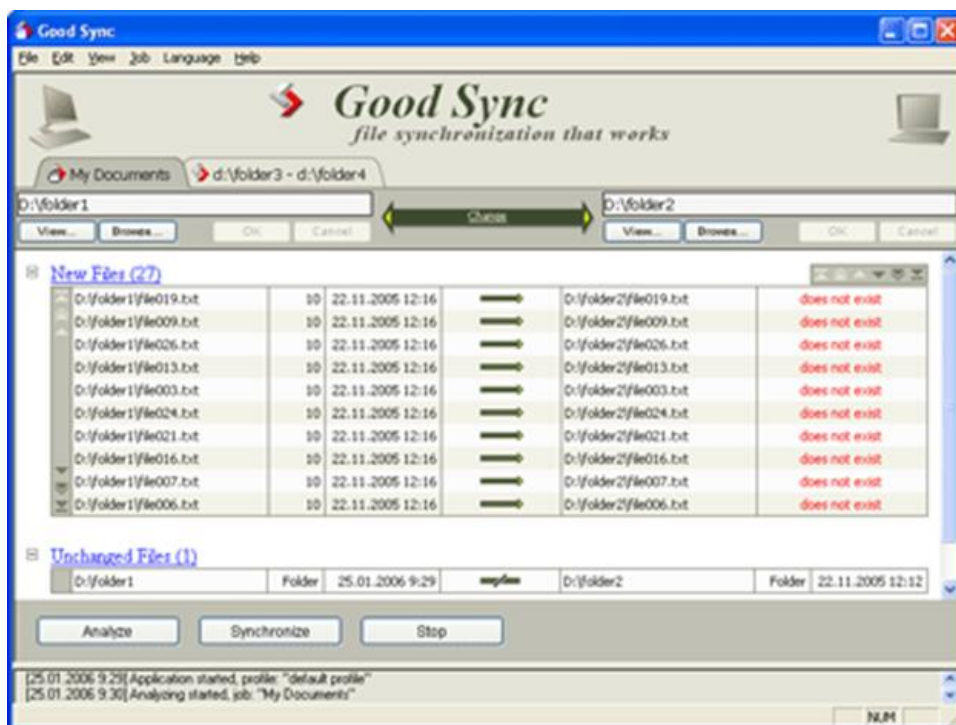
Every item in the view area has an icon next to it indicating whether it's **synced**   or **unsynced** 

When a file is in its unsynced state you can either wait for it to synchronize or you can click on the **Sync Now** button.

<https://web.archive.org/web/20060112125007/http://www.pcmag.com/article2/0,1895,1645668,00.asp> “As you parse through the items in this subdirectory or any other, the app tells you

when each item was last modified, how often you use it, and whether it's been synchronized yet.” [https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 38. “File Management: BeInSync enables you to manage all of your unsynced and conflicted files from one centralized location, on the right-side of the status screen. Current Activity: Click this tab to view the files which are currently being synchronized, files waiting to be downloaded, files being uploaded etc. You may also view the details of each file, immediately sync files by clicking the sync now button and view the file’s sync status and progress.”

- GoodSync discloses a user interface that presents synchronization and file status on multiple devices, which may be implemented using transferred metadata.



- Subversion at, e.g., <https://web.archive.org/web/20071116172448/http://svnbook.red-bean.com/en/1.4/svn-book.pdf> at 313-14. “For each file in a working directory, Subversion records two essential pieces of information in the .svn/ administrative area: what revision your

working file is based on (this is called the file's working revision), and a timestamp recording when the local copy was last updated by the repository.”

[https://web.archive.org/web/20071109123641/https://www.eclipse.org/subversive/documentation/teamSupport/workspace\\_synch.php](https://web.archive.org/web/20071109123641/https://www.eclipse.org/subversive/documentation/teamSupport/workspace_synch.php) “The synchronization state is shown with the help of the following icon decorations:

| Icon | Description                                                                                                                                                                                                                     |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|      | An incoming addition icon means, that the marked resource was added to a repository location, so it's new comparatively to the local copy. It will be added to the local copy after update action.                              |
|      | An incoming change icon means, that some changes to the marked file were made comparatively to your local copy. The changes will be added to the local copy after update action.                                                |
|      | An incoming deletion icon means, that this resource was deleted from a repository location, but it's still contained in the local copy. It will be deleted from the local copy after update.                                    |
|      | An outgoing addition icon means, that the marked resource was added to the local copy, so it's new comparatively to the repository location. It will be added to the repository after commit action.                            |
|      | An outgoing change icon means, that some changes to the marked file were made comparatively to the repository location. The changes will be added to the repository after commit action.                                        |
|      | An outgoing deletion icon means, that the marked resource was deleted from the local copy, but it's still contained in the repository location. It will be deleted the repository after commit action.                          |
|      | A conflict addition icon means, that to different resources with the marked resource name were added both to the repository location and to the local copy.                                                                     |
|      | A conflict change icon means, that the changes, that can't be merged automatically were made both with local copy and repository one. A manual merge is required. The parents of the decorated resource will be also decorated. |
|      | A conflict deletion icon means, that both local and repository's copy of the resource were deleted.                                                                                                                             |

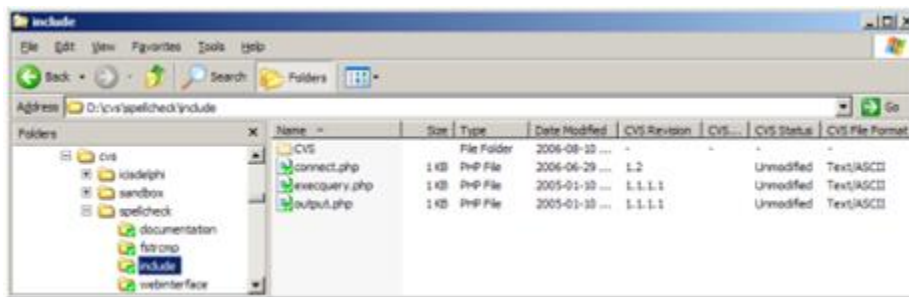
- Smartsync Pro discloses color highlighting for modified files. SSP Ex. 7 at 4, 34-39
- Nokia Intellisync discloses that, in addition to normal display of metadata which would indicate a newer version, color highlighting for modified files. NIS Ex. 2 at 19-25, 44, 82.
- Unison File Synchronizer at <https://web.archive.org/web/20071009214127/http://www.cis.upenn.edu:80/~bcpierce/unison/download/releases/beta/unison-manual.html>. “The main window shows all the file that have been modified.” “If a file has been modified . . . it will be displayed with an arrow indicating [the modification].”
- TortoiseCVS at, e.g.,

[https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 5. “TortoiseCVS integrates itself into Windows Explorer and

maintains additional information for directories and files that are under version control.

The most noticeable difference is the changed appearance of icons for directories and files that are under version control. Each directory under version control contains a “hidden” subdirectory named CVS in which TortoiseCVS maintains its administrative information.” TortoiseCVS at, e.g.,

[https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 5.



TortoiseCVS at, e.g.,

[https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 5. “TortoiseCVS uses the following icon overlays to display the

CVS status of the local files or folders within Explorer.” TortoiseCVS at, e.g.,

[https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS\\_Aug312007.pdf](https://assets.publishing.service.gov.uk/media/57a08bf7e5274a31e0000ee6/DailyTortoiseCVS_Aug312007.pdf) at 5.

|                                                                                   |                            |                                                                                                                                                                                                            |
|-----------------------------------------------------------------------------------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <b><i>Not Modified</i></b> | The local version of the file or folder is up-to-date with the version in the remote CVS repository.                                                                                                       |
|  | <b><i>Modified</i></b>     | The local version of file or folder has been modified and differs from the version in the remote CVS repository.                                                                                           |
|  | <b><i>Added</i></b>        | The file or folder has been added to the local copy of the source code under CVS, but has not yet been committed to the remote CVS repository.                                                             |
|  | <b><i>Conflict</i></b>     | The local file or folder has a conflict with the current file or folder in the remote CVS repository. A conflict occurs when two or more developers have changed the same few lines of a file in parallel. |
|  | <b><i>Not in CVS</i></b>   | The file or folder has not yet been added to the local copy of the source code under CVS.                                                                                                                  |
|  | <b><i>Ignore</i></b>       | The local file or folder is being ignored by CVS, thus it will not be affected by CVS operations.                                                                                                          |

#### (10) “Transfer Upon Communication” References

Certain Asserted Claims recite features including transferring a file responsive to determining that devices are in communication and/or transfer of files upon resuming communication between devices that were not in communication. Nothing about these claim limitations add anything novel or non-obvious over the prior art. To the contrary, this feature was well-known, routine, conventional, and widely available long before the alleged invention. As the attached claim charts demonstrate, this subject matter is disclosed in numerous references. For purposes of these contentions, references whose disclosures are cited in these contentions with respect to this claimed subject matter as reflected in the claim charts (including without limitation the specific references cited below) are hereby labeled the “Transfer Upon Communication” references.

The implementation of the claimed features would have been obvious to a person of ordinary skill in the art least because doing so would have involved the combination of prior art

elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the application of a known technique to a known device ready for improvement to yield predictable results, and/or would have been obvious to try. Moreover, a person of ordinary skill in the art would have readily appreciated the well-known and widespread use of these features and the associated benefits and advantages of their use. Persons of ordinary skill in the art would have been motivated to combine the teachings of these references with any system and/or disclosure that relates to the synchronization of files or other data content. Among other features, it was well-known that a user of a device might modify files while the device is not in communication (e.g., using a laptop while on a plane), so it was necessary to transfer file modifications when the device regained communication. For example, references disclose the following motivations to combine, such as the following as well as the other disclosures that are cited in the prior art claim charts submitted herewith:

- Sigurdsson at, e.g., [0061] “In this case, the system **400** automatically detects that a particular client is offline. In one implementation, determination of whether a client is online or offline can be made by monitoring polling by the client. If no request is received from a client, then the client is assumed to be offline. Similarly, the Server **106** can maintain a table of client IDs and whether each client is online or offline. When a client comes online, it can notify the Server **106** that it is online, and the Server **106** can update its table of clients with the online status.” [0069] “Step **612** can occur after the third client comes online, having been offline during the time that the content originated on the first client and was propagated to the second client and to the server. In step **612**, the third client requests a list of missing content from the server. For example, the Client C **406** requests a list of missing content from the Server **106** when it connects to network that links the server and other clients, such as the Internet.” [102] “In step

**818**, the server requests the second client to provide the missing content it needs to satisfy the original query received from the first client. In one implementation, the server can wait until it detects that the second client is online, then issue the request.”

- Venkataraman et al., e.g., [0025] “At step **304**, a user of a remote device couples the remote device to the synchronization device. In a preferred embodiment, the user mounts the remote device in a cradle specifically designed to accept the user's remote device. Regardless of the manner in which the coupling occurs, the synchronization device is subsequently able to communicate with the remote device and, at step **306**, to receive remote data from the remote device. By coupling the remote device to the synchronization device, the user is manifesting his or her intent to synchronize remote data using a synchronization application. In the case where synchronization of remote data is to be performed relative to central data located at a central device, it is anticipated that the user will have a pre-established subscription with the service provider maintaining the central device. In this case, the step of coupling the remote device to the synchronization device will either automatically, or in response to a user-entered command (for example, through the user interface **208** or through the remote device via the remote device interface **206**), cause the synchronization device, at step **308**, to initiate synchronization via the synchronization application.” [0008] “The synchronization device initiates synchronization of remote data found in the remote device with central data found on at least one central device by communicating with a synchronization application accessible through a communication network. In a preferred embodiment, the communication network is a publicly available network such as the Internet or World Wide Web and the at least one central device is one or more servers coupled thereto.” [0013] “The present invention may be more readily described with reference to FIGS 1-3. FIG. 1 is a block diagram of a communication system **100** in

accordance with the present invention. In particular, the communication system **100** comprises a communication network **102** coupled to a synchronization device **104**, a local device **108** and a central device **110**. In operation, the synchronization device **104** is coupled to a remote device **106**. The communication network **102** preferably comprises a publicly available network such as the Internet or World Wide Web. However, the communication network **102** may comprise a private or proprietary network exclusively, or in combination with a publicly available network. Furthermore, the communication paths within the communication network **102**, or between the communication network **102** and the various devices **104**, **108**, **110** may comprise wired or wireless links, as known in the art. The nature and use of such networks are well known in the art and therefore need not be described in greater detail herein.”

- Kaasten at, e.g., 2:7-13 “In view of the foregoing, the present invention provides the benefits of centralized data storage without incurring the costs, financial and administrative, of a central server. In a peer-to-peer computing environment, computing devices communicate among themselves to provide access to data and to synchronize changes to the data so that the latest versions are presented to the users.” 3:48-57 “The three peer computing devices **102**, **104**, and **106** of FIG.1a communicate with one another via a LAN **108**. Standard communications protocols exist for transporting data synchronization information among the computing devices in the shared environment **100**. Computing device B **104** also provides a link to the Internet **110** that is shared by all three computing devices **102**, **104**, and **106**. The Internet **110** is shown in FIG. 1a to indicate that the data synchronization services of the present invention can be performed even among remotely connected devices.” 7:1-26 “Finally, step **308** brings to-be-synchronized data objects into the same state before beginning the ongoing synchronization service. This step is important especially when one computing device has been inaccessible to

the other members of the synchronization environment **100**. For example, a user takes the laptop **106** along on a business trip and updates her work files that are stored on it. During the trip, the laptop **106** is not in communication with the other computing devices A **102** and B **104**. While she is away, that user's husband updates the couple's social calendar on computing device A **102**. Upon return, the user reconnects her laptop **106** to the data synchronization environment **100**. Because the user's work files on the laptop **106** are more up-to-date than their counterpart files on the other computing devices, in step **308**, those counterpart files are updated from the laptop **106**'s files. Similarly, the copy of the social calendar on the laptop **106** is updated from the computing device A **102**. Step **308** can be invoked even when less than a full computing device becomes accessible. For example, rather than taking the laptop **106**, the user decides to simply take a removable disk out of one of the computing devices. When the disk is returned and is again part of the data synchronization environment **100**, step **308** synchronizes the data objects on the disk with the remainder of the environment **100**. When step **308** is complete, all of the synchronized data objects on all of the accessible devices in the environment **100** are up-to-date.”

- <https://web.archive.org/web/20061016125144/https://www.foldershare.com/info/faq/Files.php>  
<https://web.archive.org/web/20061016125133/https://www.foldershare.com/info/faq/Syncin>  
[g.phphttps://www.pcmag.com/archive/foldershare-147101](https://www.pcmag.com/archive/foldershare-147101)
- Savage at, e.g., [0032] “Upon synchronizing the file with the synchronization client 106, the synchronization server 108 transmits a synchronization confirmation to inform the synchronization client 106 that the synchronization is complete. Note: If there is an error in the synchronization, the synchronization confirmation can also be used to indicate this to the synchronization client 106.” [0051] “If the decision operation 506 determined that the

Unreported state was not a modified state, another decision operation 518 determines whether the Reported state was a modified state, potentially indicated a previous failure of communication between the server and the client (e.g., a previously sent synchronization report was never received by the server). As such, a next report time is re-computed in a computing operation 520 to allow the synchronization report to be resent at an effective interval. Processing then proceeds to the setting operation 526.”

- Rao at, e.g., 6:55-58 “FIG. 3 is a pictorial representation of an embodiment 300 of a sequence for synchronizing. The process shows an initial connection 301, a period where the systems are disconnected 325, and the actions at reconnection 326.” 6:59-67 “In the initial connection 301, a server 302 and client 304 are connected. The server 302 has a storage system 306 that contains items 308 in an item repository 310. Some of the items 308 have related files 312 and 314. During the initial connection 301, the client storage 316 is populated with a synchronized item repository 318 that contains items 320, which are copies of items 308 from the server 302. The two server referenced files 312 and 314 are ghosted on the client 304 as files 322 and 324.” 7:1-3 “During the initial connection 301, the item repositories 310 and 318 are synchronized and the raw data files 312 and 314 are ghosted on the server as ghosted files 322 and 324.” 7:4-9 “Block 325 shows the state of the systems when connection is severed. Server 302 maintains storage 306. Client 304 has storage 316 that contains an item repository 318 that enables various metadata and other functions to be performed. While disconnected, a ghosted file 324 may be selected for future synchronization.” 7:10-15 “Block 326 shows the systems upon reconnect. Server 302 and client 304 are reconnected. The item repositories 310 and 318 are synchronized between storage systems 306 and 316. As part of the synchronization routine, the file 314 replaces ghosted file 324 so that the raw data of file 324 may be available when

disconnected.”

- Braginsky at, e.g., [0027] “If a user is connected to the server system 104 via the network 106, then the file system 116 returns the most recent version of the requested file, whether cached locally in file system 116 or stored remotely in file system 112. In some embodiments, the selection of the latest version of the requested file can be based on a comparison of file timestamps located in a meta-directory stored on the client system 102, as described below with respect to FIG. 5. In some circumstances, the entire file may have to be downloaded from the server system 104 to the client system 102 to service a request. This may happen, for instance, when the server has a newer version of the file than the client, and the process of downloading the newest version to the client did not begin prior to the request.” [0028] “If the client system 102 is not connected to the server system 104, then the file system 116 returns the locally cached version of the requested file. If there is no cached version of the file, then the file system 116 returns an error and notifies the user (e.g., through a callout bubble) that the requested file is not available offline.” [0078] “FIG. 7 is a conceptual flow diagram of one embodiment of a file synchronization process 700. In actual implementation, task execution and queue reordering may be executed as continuous background processes. The file synchronization process 700 determines if the client system 102 is connected to the server system 104 (702), and continues only when a connection is present. If an upload/download task is not in progress (704-No), then the tasks waiting in the task queue 404 of worker module 214 are executed in accordance with the priority policy 408 (712). If a task is in progress (704-Yes), but the task at the top of the task queue 404 has higher priority (706-Yes), then the task in progress is paused (708). Optionally, the task queue 404 is re-ordered (710) in accordance with the priorities assigned to the tasks in the task queue and/or in accordance with a priority

policy. The task at the top of the task queue is executed (712). If there are more tasks to execute (718-Yes), then those tasks are executed in accordance with the priority policy or order in the task queue 404.”

- Shappell at, e.g., 10:54-63 “When a group member is connected to the peer-to-peer network, the member entry in the Members list 1102 is selectable by the user. Otherwise, the entry appears grayed out in the list 1102. The members list 1102 may include fields containing the online status of the member in the shared space, the member's role and the last date in which the member connected with the group. In addition to these standard fields, the members list 1102 may further include a comment field for additional information concerning the member.” 14:13-17 “In a preferred embodiment, the application maintains metric for all peers connected to the source machine. Such metric indicate response time of the peer machine, the availability of the peer machines and like data for use in the decision of how the shared data will be obtained.” 14:17-22 “Thus, a download session can be suspended and resumed such that, if a portion of a shared file is downloaded and then suspended, the next time the group member is connected to the peer network or when they resume the download, the download resumes at the next place after the one that was last copied.”
- Sagar at, e.g., 6:45-52 “The update of the local feed may initiate generation of a notification that may be enqueued to the notification service at block 405 for processing. At block 406, the Notification Service may inform other devices that updates are available and may prompt nodes to pull changes down. Because the system implements a feed service, devices that come online (e.g., connect with the mesh) may receive updates once they subscribe for the notifications.”
- Aboulhosn at, e.g., 2:64–3:8 “When a member goes online, it contacts the group owner

(assuming that the group owner is currently online) to synchronize its copy of the group folder with the group owner. For example, files of the group folder may have been added, deleted, or updated while the member was offline. Also, the member while offline may have added, deleted, or updated files of the group folder." 6:59–7:14 "A member comes online by providing authentication information to the authentication server. Once authenticated, the member is online and sends its queued messages into the recipient computer systems that are currently online. The member then sends a synchronization request message 642 to the group owner. The group owner collects the metadata for the shared folder and sends a synchronization response message 643 to the new online member. When the new online member receives the metadata, it updates its folder accordingly. The new online member may also identify whether it made updates to the folder while it was offline. If so, the member notifies the group owner that it is out of synchronization, and the updating of the metadata is performed in a manner similar to that illustrated in FIG. 6B. "

- Brown ‘847 at, e.g., 17:62–18:1 “[A] person skilled in the art will recognize that authenticating a user is not needed while a user is making changes locally on a client. The filter drivers can track changes made locally, even while disconnected from the server. The user can later log in to the server and be authenticated, at which point changes can be migrated to the server.”
- Brown ‘826 at, e.g., 17:42–47 “[A] person skilled in the art will recognize that authenticating a user is not needed while a user is making changes locally on a client. The filter drivers can track changes made locally, even while disconnected from the server. The user can later log in to the server and be authenticated, at which point changes can be migrated to the server.”
- Hesselink at, e.g., 7:27–38 “By locally storing the files at the location of the user, the user can work on the files even when the user's is no longer actively connected to the remote computer

from which the files were originally accessed. Upon reconnecting to the remote computer, the files may be automatically updated at the remote location according to any changes made by the user of the user's computer (local location). Computer readable media containing instructions for operating one or more computers to carry out the methods described herein are also covered by the present invention.” 39:45–51 ““As noted above, when data is updated, such as when a file is edited, for example, the system may be configured to automatically update all or some designated subset of computers on the network as soon as the update has been performed at one location. Computers that are offline at such a time will be automatically updated as soon as those computers go online again.” 41:39–47 “Optionally, all computers that are connected to the network, that share a file that the user is working on and that have a copy of that file in cache memory (or on a remote device ) may be automatically updated concurrently with the updates of the device module file, each time an update is performed, when such computers are actively subscribed, as noted above. Computers that are not actively subscribed or are offline can be automatically updated when reconnecting with the network or initiating an active subscription.”

- iFolder at, e.g., <https://www.novell.com/documentation/ifolder21/pdfdoc/enduser/enduser.pdf> at 27–28 “[S]uppose that you have the iFolder client installed on two computers: computer A and computer B. At some point in the day, you disconnect both of these computers from the network and continue to work from both computers offline. ... When you reconnect computer B to the network, its change is uploaded to the iFolder server.” <https://www.novell.com/documentation/ifolder21/pdfdoc/admin/admin.pdf> at 17–18 “Novell iFolder ... [t]racks and logs changes made while you work offline and synchronizes those changes when you go online.”

- BeInSync at, e.g., [https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync\\_User\\_guide.pdf](https://web.archive.org/web/20060315123939/http://www.beinsync.com/help/Beinsync_User_guide.pdf) at 5. “BeInSync 1.6 also offers increased control over the synchronization process. Users can now pause and resume synchronization activity by choosing either the offline or online option from BeInSync's taskbar icon” <https://web.archive.org/web/20050329031828/http://www.beinsync.com:80/faq.php> “There are a few different types of connection modes in BeInSync: Connected - BeInSync is logged in and connected; synchronization and remote Web access are available. Disconnected - This means that BeInSync is not logged in, and won't synchronize with your other computers and shares in your BeInSync network.”
- Foldershare at, e.g., <https://web.archive.org/web/20061016125144/https://www.foldershare.com/info/faq/Files.php> “As a member of a library, what happens if file changes take place while my computer is offline? When your offline computer comes back online, it will be automatically updated with all changes.” <https://web.archive.org/web/20061016125133/https://www.foldershare.com/info/faq/Syncing.php> “What happens if I lose the connection between my computers while transferring a file? The transfer will begin once both computers are back online again at the same time.” <https://www.pcmag.com/archive/foldershare-147101> “By default, the service automatically synchronizes your folders around the clock. As long as your machines remain online, when you change a file in one of the folders, the change propagates to each system in a matter of minutes—if not seconds.”
- SharpCast at, e.g., <https://web.archive.org/web/20060829025503/http://www.sharpcast.com/products/photos/> (“Work offline - no need to leave your home PC on to view or manage your photos on another

PC or your phone! . . . Changes you make anywhere, even offline, are automatically made everywhere, so you can pick up right where you left off.”).

- Smartsync Pro at <http://web.archive.org/web/20070202015005/http://www.smartsync.com/>.

“You can easily synchronize computers (e.g. home and office) even not connected over LA

- Nokia Intellisync at <https://web.archive.org/web/20071028120238/http://usa.nokia.com/A4179068>.

“Automatically delivers content and updates to point-of-business, and pulls content from remote devices to store centrally. Connects clients automatically for updates, publishes information to PC and/or device.

- Apple iDisk at, e.g., [https://appleinsider.com/articles/03/06/23/idisk\\_sync\\_in\\_panther](https://appleinsider.com/articles/03/06/23/idisk_sync_in_panther)

“Offline Access to iDisk: Use iDisk files even when you're not connected to the Internet. When you reconnect, all your changes are synchronized to your iDisk online.”

<https://flylib.com/books/en/3.343.1/> at 35. “If you tend to work offline, you can copy files to your local iDisk, and then synchronize your local copy with your real iDisk the next time you go online.” <https://www.informit.com/articles/article.aspx?p=401141&seqNum=5> “iDisk does have one feature that sets it apart from a normal network share—the capability to keep a local copy of its contents that are synchronized automatically when you connect to the Internet. This means that you always have an up-to-date offline copy of the files in case you don’t have Internet access.”

#### **e. Obviousness Combinations**

The charts below disclose combinations of references demonstrating that the Asserted Claims would have been obvious in light of the prior art cited. This is done by reference to the particular categories identified above. For each instance in which such a category is referenced,

one or more of the references cited in that category (as identified above) may supply the claim element to which the category applies. The discussion above accompanying each category sets forth the detailed rationale and motivation for combining each category's respective references with respect to the claimed features they disclose.

Defendant identifies references in groups below, based on common subject matter relevant to the Asserted Claims, for the purpose of disclosing obviousness combinations and further disclosing motivations to combine. The absence of a particular reference from a group is not an admission or suggestion that the reference does not disclose any particular claim limitations.

| Asserted Claims                                                                                                                          | Application of Prior Art                                                                                                                                                                                                                                                                                                            |
|------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| '561 patent claims 1, 3, 4, 8, 10, 11<br>'942 patent claims 1, 3, 4, 10, 12, 13<br>'607 patent claims 1, 5, 7, 12, 17, 19                | Obvious over the combination of each reference from the following groups with each other reference in the following groups: Synchronize File; Server-Based; Device Application; Me-To-Me; Transfer When Modified; and/or Replace Older Version                                                                                      |
| '607 patent claims 3, 14, 20<br>'787 patent claims 1, 4, 8, 11, 13, 15, 16<br>'622 patent claims 1, 3, 5, 7, 11, 12, 14, 16, 17          | Obvious over the combination of each reference from the following groups with each other reference in the following groups: Synchronize File; Server-Based; Device Application; Me-To-Me; Transfer When Modified; Replace Older Version; Transfer Metadata; and/or Transfer Metadata Before File                                    |
| '607 patent claim 4<br>'787 patent claims 2, 9, 14<br>'823 patent claims 1, 2, 3, 4, 8, 9, 10, 11, 13, 14, 15, 16<br>'622 patent claim 4 | Obvious over the combination of each reference from the following groups with each other reference in the following groups: Synchronize File; Server-Based; Device Application; Me-To-Me; Transfer When Modified; Replace Older Version; Transfer Metadata; Transfer Metadata Before File; and/or Graphical Availability Indication |
| '561 patent claims 3, 4, 10, 11<br>'942 patent claims 3, 4, 12, 13<br>'607 patent claims 1, 5, 7, 12, 17, 19<br>'787 patent claims 3, 10 | Obvious over the combination of each reference from the following groups with each other reference in the following groups: Synchronize File; Server-Based; Device Application; Me-To-Me; Transfer When Modified; Replace Older Version; and/or Transfer Upon Communication                                                         |

| Asserted Claims              | Application of Prior Art                                                                                                                                                                                                                                                                                                                                         |
|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| '823 patent claims 3, 10, 15 | Obvious over the combination of each reference from the following groups with each other reference in the following groups: Synchronize File; Server-Based; Device Application; Me-To-Me; Transfer When Modified; Replace Older Version; Transfer Metadata; Transfer Metadata Before File; Graphical Availability Indication; and/or Transfer Upon Communication |

**f. Secondary Considerations**

Plaintiff bears the burden of establishing the existence of secondary considerations and the requisite nexus to the claimed inventions. To date, Plaintiff has not contended that any secondary considerations support any alleged non-obviousness of the Asserted Claims. To the extent that Plaintiff subsequently submits evidence and/or contentions whereby it contends that any secondary considerations support a conclusion that any of the Asserted Claims are not obvious in response to Defendant's interrogatory or otherwise, Defendant reserves the right to respond, including to amend and/or supplement these Invalidity Contentions. Moreover, as these Invalidity Contentions demonstrate, none of the Asserted Claims address any need that was either long felt or unmet, nor were their results unexpected. To the contrary, the extensive invalidating prior art detailed above and in Defendant's charts show that before the patents-in-suit, the prior art techniques, designs, and systems were widely accepted, understood, and available. Finally, although secondary considerations may be relevant to the obviousness analysis, secondary considerations of nonobviousness cannot overcome the strong prima facie case of obviousness set forth in these Invalidity Contentions.

C. **An Identification of Any Limitations the Defendant Contends are Indefinite or Lack Written Description Under Section 112**

1. **Section 112 – Preliminary Statements**

To be valid, a patent claim must “particularly point [] out and distinctly claim[] the subject matter which the inventor . . . regards as the invention.” 35 U.S.C. § 112(2). The Supreme Court has held that the proper standard for indefiniteness is as follows: “[A] patent is invalid for indefiniteness if its claims, read in light of the specification delineating the patent, and the prosecution history, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention.” *Nautilus, Inc. v. Biosig Instruments, Inc.*, 134 S. Ct. 2120, 2124, 189 L. Ed. 2d 37 (2014). The Supreme Court further explained that “a patent must be precise enough to afford clear notice of what is claimed, thereby apprising the public of what is still open to them.” *Id.* at 2129.

Indeed, to uphold claims—such as Plaintiff’s—that do not clearly define the scope of the invention would thwart the ultimate purpose of the patent laws: to “promote the Progress of Science and useful Arts.” *Festo Corp. v. Shoketsu Kinzoku Kogyo Kabushiki Co.*, 535 U.S. 722, 730 (2002) (quoting U.S. Const. art. I, § 8, cl. 8). As the Supreme Court explained in *Festo*, a patent claim “is a property right; and like any property right, its boundaries should be clear. This clarity is essential to promote progress, because it enables efficient investment in innovation. A patent holder should know what he owns, and the public should know what he does not.” *Id.* The identified claim terms of Plaintiff’s Asserted Claims “fall[] within the innovation-discouraging ‘zone of uncertainty’ against which the Supreme Court has warned.” *Interval Licensing LLC v. AOL, Inc.*, 766 F. 3d 1364, 1374 (Fed. Cir. 2014), quoting *Nautilus*, 134 S. Ct. at 2130 (internal quotations and brackets omitted).

The Asserted Claims suffer from similar defects that render them indefinite, at least under

Plaintiff's apparent view of the claim scope. Plaintiff's Infringement Contentions indicate that Plaintiff may interpret the claims in a manner that further renders them indefinite, particularly in view of the absence of sufficient supporting disclosure in the intrinsic evidence.

Defendant also notes that the attached claim charts identify disclosures in the prior art that correspond to limitations of the Asserted Claims; in some cases, Defendant has identified prior art disclosures corresponding to limitations separately identified below as indefinite. The identification of corresponding disclosures in the prior art for such limitations should not be construed as a concession that Defendant believes those limitations fulfill the definiteness requirement, or inform persons of ordinary skill in the art of the metes-and-bounds of the full scope of the alleged invention with reasonable certainty. Rather, Defendant's prior art disclosures are based on its best understanding (without the benefit of claim construction) informed by the Infringement Contentions served by Plaintiff, and in some cases, were chosen because they most closely mirror exemplary embodiments in the specifications of the Asserted Patents (and thus are likely to meet the disputed claim limitations in the event they are not adjudged to be indefinite).

Under the enablement requirement, the claims must be supported by a disclosure that, when filed, contained sufficient information regarding the subject matter of the claims as to enable one skilled in the pertinent art to make and use the full scope of the claimed invention without undue or unreasonable experimentation. *In re Wands*, 858 F.2d 731, 737 (Fed. Cir. 1988).

To satisfy the written description requirement, the specification must describe the claimed invention in sufficient detail that one skilled in the art can reasonably conclude that the inventor had possession of the claimed invention. *Vas-Cath, Inc. v. Mahurkar*, 935 F.2d 1555, 1562-63 (Fed. Cir. 1991). Specifically, the specification must describe the claimed invention in a manner understandable to a person of ordinary skill in the art and show that the inventor actually invented

the claimed invention. *Id.* Generic claim language in the original disclosure does not satisfy the written description requirement if it fails to support the scope of the genus claimed. *See LizardTech, Inc. v. Earth Res. Mapping, Inc.*, 424 F.3d 1336, 1343-46 (Fed. Cir. 2005).

Written description and enablement impose related requirements. *LizardTech*, 424 F.3d at 1344-45. For written description, “the test ... is whether the disclosure of the application relied upon reasonably conveys to those skilled in the art that the inventor had possession of the claimed subject matter as of the filing date.” *Ariad Pharms. v. Eli Lilly & Co.*, 598 F. 3d 1336, 1351 (Fed. Cir. 2010) (en banc). The “enablement requirement is satisfied when one skilled in the art, after reading the specification, could practice the claimed invention without undue experimentation.” *Auto. Techs. Int’l v. BMW of N. Am.*, 501 F.3d 1274, 1282 (Fed. Cir. 2007) (internal quotation omitted). The written description and enablement requirements are fundamental to the quid pro quo of the patent system, ensuring that the scope of the claims does not “overreach[] the scope of the inventor’s contribution.” *Ariad*, 598 F.3d at 1353-54.

The Asserted Claims are each invalid for lack of written description and non-enablement, particularly under Plaintiff’s apparent interpretation and application. The Asserted Patents speak in broad terms of wishes and goals for a system to achieve the desired functions. But merely reciting goals and wishes does not meet the requirements of 35 U.S.C. § 112. “Patent protection is granted in return for an enabling disclosure of an invention, not for vague intimations of general ideas that may or may not be workable. ... Tossing out the mere germ of an idea does not constitute enabling disclosure.” *Genentech v. Novo Nordisk A/S*, 108 F.3d 1361, 1366 (Fed. Cir. 1997); *Ariad*, 598 F.3d at 1356 (“[W]ritten description requires more than a mere wish or plan.”) (internal quotation omitted).

Moreover, despite the patents’ narrower disclosures, Plaintiff is attempting to read the

claims more broadly onto functionality that the patents nowhere describe or enable. To prevent such overreaching claims, the law requires the patentee to describe and enable the “full scope” of the claims. *LizardTech*, 424 F.3d at 1344-46; *Sitrick v. Dreamworks*, 516 F.3d 993, 999-1000 (“The full scope of the claimed invention must be enabled.... Because the asserted claims are broad enough to cover both movies and video games, the patents must enable both embodiments.”).

For example, a patentee may not describe and enable one embodiment and then capture other, “distinctly different” embodiments within the claim scope. *Auto. Tech.*, 501 F.3d at 1282-85 (affirming summary judgment of non-enablement because the patentee described and enabled only mechanical sensors but the generic “sensor” claims also covered electronic sensors); *see also Rivera v. Int’l Trade Commission*, 857 F.3d 1315 (Fed. Cir. 2017) (holding that disclosure of a single-brew coffee machine using pods did not also disclose a pod-less machine); *Synthes USA v. Spinal Kinetics*, 734 F.3d 1332, 1341 (Fed. Cir. 2013) (affirming invalidity, because “the as-filed disclosure did not demonstrate possession of an intervertebral implant that employed any sort of openings anywhere on the cover plates” as it “never discloses anything broader than using grooves to anchor the fiber system to the cover plates”); *Ariad*, 598 F.3d at 1376 (finding invalidity as a matter of law – “Here, the specification at best describes decoy molecule structures and hypothesizes with no accompanying description that they could be used to reduce NF-κB activity. Yet the asserted claims are far broader.”); *ICU Med. v. Alaris Med. Sys.*, 558 F.3d 1368, 1377-78 (Fed. Cir. 2009) (affirming summary judgment of invalidity for claim term that generically covered both valves with and without spikes where specification described and enabled only spiked valves); *Liebel-Flarsheim v. Medrad, Inc.*, 481 F.3d 1371, 1379 (Fed. Cir. 2007) (same regarding claim covering both jacketed and jacketless injectors where specification described and enabled only jacketed injector).

The Asserted Claims fail to enable the full scope of the invention, and fail to describe the invention sufficiently to convey to a person of skill in the art that the patentee had possession of claimed invention at time of application, *i.e.*, that patentee invented what is claimed, under 35 U.S.C. § 112 ¶ 1. Specifically, at least the limitations of the Asserted Claims that are identified in the Section 112 contentions chart provided below lack sufficient enablement and written description encompassing the full scope of the claimed invention, including as apparently interpreted and asserted by Plaintiff.

In view of the foregoing, Defendant identifies below the grounds of invalidity under Section 112. Defendant provides these contentions without the benefit of upcoming claim construction discovery and other ongoing discovery. Defendant may provide proposed claim constructions to be adopted in the alternative to the extent certain claim terms and elements are not determined to be indefinite. Where a certain claim term or limitation in one claim is identified as indefinite, Defendant contends that the same claim term or limitation is indefinite in every instance the term or limitation is recited throughout the Asserted Claims. Where Defendant has identified a phrase as indefinite, Defendant also contends that the subparts of the phrase are indefinite within the context recited in the claim.

## **2. Section 112 – Contentions**

As explained in the table below, the Asserted Claims are invalid for indefiniteness (in view of Plaintiff’s Infringement Contentions), lack of enablement, and/or lack of written description, at least for the reasons below. Also, any ground of invalidity under § 112 for any claim applies to any asserted claim that depends from it.

| <b>Claim(s)</b>       | <b>Limitation / Term</b>                                                  | <b>Grounds of Invalidity Challenge</b>            |
|-----------------------|---------------------------------------------------------------------------|---------------------------------------------------|
| ‘607 claims 1, 12, 17 | “the first metadata being assigned a first priority greater than a second | Lack of written description<br>Lack of enablement |

|                                                                                                    |                                                                                                                                                    |                                                   |
|----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|
| '787 claims 1, 8, 13<br>'823 claims 1, 8, 13<br>'622 claims 1, 11, 16                              | priority assigned to the copy of the first file"                                                                                                   |                                                   |
| '607 claims 3, 14, 17, 20<br>'787 claims 1, 8, 13<br>'823 claims 1, 8, 13<br>'622 claims 3, 14, 17 | "priority assignment configuration" and associated limitations                                                                                     | Lack of written description<br>Lack of enablement |
| '823 claim 1                                                                                       | "graphical availability indication is presented proximate a file icon representing the first file on a user interface of the second client device" | Lack of written description<br>Lack of enablement |

In addition to the foregoing, all of the Asserted Claims are invalid under Section 112 for lack of enablement and lack of written description of the full scope of the claimed inventions. The Asserted Claims refer to systems and functions for the transfer of files to and among various electronic devices. As recited in the Asserted Claims, each device is claimed as an "electronic device" rather than being limited to a particular type of electronic device. The scope of the term "electronic device" may include computers, phones, tablets, automobiles, refrigerators, thermostats, televisions, and so forth, just to provide a few examples. The Asserted Claims do not require that the various electronic devices be the same. In addition, the Asserted Claims do not require that the electronic devices execute the same operating software or the same application, let alone the same release and version of the software operating on the electronic devices. The Asserted Claims also recite high-level desired results for achieving the automatic transfer of files and metadata under certain conditions without limiting the claims to specific protocols, technologies, or hardware or software implementations. The written description of the asserted patents does not provide sufficient disclosure to describe and enable a person of ordinary skill in

the art to practice the full scope of the claims without undue experimentation in view of the many permutations and combinations of potential electronic devices, applications, files, and associated technologies that the Asserted Claims purport to encompass. Accordingly, the Asserted Claims are invalid for lack of enablement and lack of written description.

**D. An Identification of Any Claims the Defendant Contends are Directed to Ineligible Subject Matter Under Section 101**

Defendant has filed a Motion to Dismiss on Section 101 and incorporates that briefing by reference. As a brief summary, Section 101 provides that “[w]hoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.” 35 U.S.C. § 101. The Supreme Court has “long held that this provision contains an important implicit exception: Laws of nature, natural phenomena, and abstract ideas are not patentable.” *Alice Corp. v. CLS Bank Int’l*, 134 S. Ct. 2347, 2354 (2014) (quoting *Ass’n for Molecular Pathology v. Myriad Genetics, Inc.*, 133 S. Ct. 2107, 2116 (2013)). “[S]uch discoveries are manifestations of...nature, free to all men and reserved exclusively to none.” *Mayo Collaborative Servs. v. Prometheus Labs., Inc.*, 132 S. Ct. 1289, 1293 (2012) (quotations and citations omitted). The concern that underlies “this exclusionary principle” is “one of preemption.” *Alice*, 134 S. Ct. at 2354. Thus, patents that seek to preempt others from using “the basic tools of scientific and technological work” are invalid. *Id.*

The Supreme Court in *Alice* set forth a two-step test “for distinguishing patents that claim laws of nature, natural phenomena, and abstract ideas from those that claim patent-eligible applications of those concepts.” *Alice*, 134 S. Ct. at 2355.

1. First, the Court must “determine whether the claims at issue are directed to one of those patent-ineligible concepts,” such as an abstract idea. *Id.* If the Court determines that the

claims are directed to a patent-ineligible concept such as an abstract idea, then:

2. The Court must “consider the elements of each claim both individually and as an ordered combination to determine whether the additional elements transform the nature of the claim into a patent-eligible application.” *Id.* (citation and quotation marks omitted). The Supreme Court describes this second step “as a search for an inventive concept—*i.e.*, an element or combination of elements that is sufficient to ensure that the patent in practice amounts to significantly more than a patent upon the ineligible concept itself.” *Id.* (quotation marks and brackets omitted). For example, “[s]tating an abstract idea while adding the words ‘apply it’ is not enough for patent eligibility. Nor is limiting the use of an abstract idea ‘to a particular technological environment.’” *Id.* at 2358 (internal citations and quotations omitted).

Each and every Asserted Claim is invalid under 35 U.S.C. § 101 and applicable case law authority. *See, e.g., Alice*, 134 S. Ct. at 2354; *In re TLI Commc’ns LLC Patent Litig.*, 823 F.3d 607, 613 (Fed. Cir. 2016); *Intellectual Ventures I LLC v. Capital One Bank (USA)*, 792 F.3d 1363, 1369 (Fed. Cir. 2015); *Digitech Image Techs., LLC v. Elecs. for Imaging, Inc.*, 758 F.3d 1344, 1351 (Fed. Cir. 2014); *Content Extraction & Transmission LLC v. Wells Fargo Bank, Nat’l Ass’n*, 776 F.3d 1343, 1348 (Fed. Cir. 2014); *Ultramercial, Inc. v. Hulu, LLC*, 772 F.3d 709, 719 (Fed. Cir. 2014); *BuySAFE, Inc. v. Google, Inc.*, 765 F.3d 1350, 1355 (Fed. Cir. 2014); *Internet Patents Corp. v. Active Network, Inc.*, 790 F.3d 1343, 1347-48 (Fed. Cir. 2015); *Intellectual Ventures I LLC v. Capital One Fin. Corp.*, 850 F.3d 1332, 1341 (Fed. Cir. 2017); *FairWarning IP, LLC v. Iatric Sys., Inc.*, 839 F.3d 1089, 1093 (Fed. Cir. 2016); *Elec. Power Grp., LLC v. Alstom S.A.*, 830 F.3d 1350, 1354 (Fed. Cir. 2016); *Two-Way Media Ltd. v. Comcast Cable Commc’ns, LLC*, 874 F.3d 1329, 1337 (Fed. Cir. 2017); *Affinity Labs of Tex., LLC v. Amazon.com Inc.*, 838 F.3d 1266, 1267-68 (Fed. Cir. 2016); *BSG Tech v. Buyseasons*, 899 F.3d 1281, 1287 (Fed. Cir. Aug. 15, 2018);

*SAP America v. InvestPic*, 898 F.3d 1161, 1168 (Fed. Cir. 2018); *Burnett v. Panasonic*, No. 2018-1234, 2018 WL 3434533, at \*5 (Fed. Cir. July 16, 2018); *In re Villena*, No. 2017-2069, 2018 WL 4145863, at \*2 (Fed. Cir. Aug. 29, 2018); *Interval Licensing v. AOL*, 896 F.3d 1335, 1338, 1345-48 (Fed. Cir. 2018); *Intellectual Ventures I v. Symantec*, No. 2017-1814, 2018 WL 1324863, at \*1, n.1 (Fed. Cir. Mar. 15, 2018); *Zuili v. Google*, No. 2017-2161, 2018 WL 798666, at \*2 (Fed. Cir. Feb. 9, 2018); *WhitServe v. Donuts*, 809 Fed. Appx. 929, 932 (Fed. Cir. 2020); *Elec. Commc'n Techs. v. ShoppersChoice.com*, 958 F.3d 1178, 2020 WL 2479692, at \*1-2 (Fed. Cir. 2020); *Dropbox v. Synchronoss Techs.*, 815 F. App'x 529, 533 (Fed. Cir. 2020); *Am. Axle & Mfg. v. Neapco Holdings*, 967 F.3d 1285, 1295 (Fed. Cir. 2020); *Chamberlain Grp. v. Techtronic Indus.*, 935 F.3d 1341, 1348 (Fed. Cir. 2019), *cert. denied*, 2020 WL 5882260 (2020); *Innovation Scis. v. Amazon.com*, 778 F. App'x 859, 863 (Fed. Cir. 2019); *Ameranth v. Domino's Pizza*, 792 Fed. App'x 780, 788 (Fed. Cir. 2019); *ChargePoint v. SemaConnect*, 920 F.3d 759, 766 (Fed. Cir. 2019), *cert. denied*, 140 S. Ct. 983 (2020); *SAP Am. v. Investpic*, 898 F.3d 1161, 1166 (Fed. Cir. 2018); *Voter Verified v. Election Sys. & Software*, 887 F.3d 1376, 1384 (Fed. Cir. 2018); *BSG Tech v. Buyseasons*, 899 F.3d 1281, 1290 (Fed. Cir. 2018); *Glasswall Sols. v. Clearswift*, No. 2018-1407, 2018 WL 6720014, at \*2 (Fed. Cir. Dec. 20, 2018); *Interval Licensing v. AOL*, 896 F.3d 1335, 1346 (Fed. Cir. 2018); *Smartflash v. Apple*, 680 Fed. Appx. 977, 983 (Fed. Cir. 2018); *Location Based Services v. Niantic*, 295 F. Supp. 3d 1031, 1040, 1045-49 (N.D. Cal. 2017), *aff'd* 742 Fed. Appx. 506 (Fed. Cir. 2018).

The Asserted Claims are directed to an abstract idea. Specifically, the claims are directed to the abstract idea of synchronizing multiple versions of a file across networked computers, and particularly the idea of automatically transferring modified files and information across computer systems. Notably, nothing in the claims describes specifically *how* to achieve the desired

functional results in a non-abstract way. *Two-Way Media Ltd. v. Comcast Cable Comm'ns, LLC*, 874 F.3d 1329, 1338-39 (Fed. Cir. 2017) (holding claims directed to routing information to be abstract because they “[did] not sufficiently describe how to achieve these results in a non-abstract way”). The Asserted Claims, like those in *Two-Way Media*, are devoid of details on “how to engineer or program” the underlying components and recite only “result-based” functional aspirations. *Interval Licensing LLC v. AOL, Inc.*, 896 F.3d 1335, 1343-45 (Fed. Cir. 2018).

The specification further confirms that the claimed invention is directed to an abstract idea. See *ChargePoint, Inc. v. SemaConnect, Inc.*, 920 F.3d 759, 767 (Fed. Cir. 2019) (“[w]hile ‘[t]he § 101 inquiry must focus on the language of the Asserted Claims themselves,’” the specification may be useful “to understand ‘the problem facing the inventor’ and, ultimately, what the patent describes as the invention.”). The Asserted Patents’ shared specification admits that the purported invention is to be implemented with generic computing systems, rather than any specific inventive new hardware and/or software. *E.g.*, ’561, col. 4:57-59, 5:1-5, 5:14-16.

The Asserted Claims do not fare better at *Alice* Step Two, as they do not recite an inventive concept. When the abstraction is removed from the asserted claims, nothing is left other than generic computer functionality. *BSG Tech LLC v. Buyseasons, Inc.*, 899 F.3d 1281, 1291 (Fed. Cir. 2018) (“At *Alice* step two, it is irrelevant whether [the abstract idea] may have been non-routine or unconventional as a factual matter” when the claims fail to provide “significantly more” than the abstract idea itself.”). Specifically, each and every element of the Asserted Claims was well-understood, routine, and conventional prior to the purported invention of the subject matter recited in the asserted claims, including as reflected in the prior art references and claim charts submitted herewith. Nothing in any claim element provided any inventive concept sufficient to confer patent-eligibility.

**E. Other Prior Art**

In addition to the prior art references that Defendant has specifically discussed in these Invalidity Contentions, Defendant identifies each reference produced and cited in the accompanying document production of the same date. All of these references are relevant prior art, and Defendant reserves the right to rely on them based on its continuing research and analysis of the Asserted Claims, accused instrumentalities, and the prior art.

Dated: September 2, 2022

FISHER BROYLES, LLP

/s/ Carl D. Neff

Carl D. Neff (No. 4895)

CSC Station

112 S. French Street

Wilmington, Delaware 19801

Telephone: (302) 482-4244

carl.neff@fisherbroyles.com

*Of Counsel:*

Ryan T. Beard (*pro hac vice*)

ryan.beard@fisherbroyles.com

FISHER BROYLES, LLP

501 Congress Avenue

Suite 150

Austin, TX 78701

Phone: (512) 577-2673

*Attorneys for Defendant Egnyte, Inc.*

Christopher R. Kinkade (*pro hac vice*)

christopher.kinkade@fisherbroyles.com

FISHER BROYLES, LLP

100 Overlook Center, 2nd Floor

Princeton, New Jersey 08540

Phone: (609) 415-0101

Adam K. Yowell (*pro hac vice pending*)

FisherBroyles, LLP

5470 Kietzke Ln., Suite 300

Reno, NV 89511

Phone: (775) 230- 364

Adam.Yowell@fisherbroyles.com