A METHOD FOR MANAGING MULTIPLE FILE STATES FOR REPLICATED FILES

RELATED APPLICATIONS

[0001] This patent application relates to commonly assigned copending U.S. Patent Application No. AA/BBB,CCC, filed Month DD, YYYY under attorney docket number MSFT-1651 and entitled "Managing File Replication in Applications."

FIELD OF THE INVENTION

[0002] This invention relates to the field of document management, and in particular to a method and system for managing multiple file states during the use of a file copied from a server.

BACKGROUND OF THE INVENTION

[0003] It is common practice in a networked computer environment for a computer user to collaborate on a document with another user on the network. Such a network may be a local area network (LAN), wide area network (WAN), the Internet or the like. In such a collaboration, each user may create and edit an electronic file by using a software application such as general application software, software designed specifically for the type of file that is the object of the collaboration, or the like.

[0004] A collaboration may be made possible by maintaining a master version of the electronic file in a location that is accessible to all users who will be collaborating on the file. In a conventional networked computer environment, there may exist a server computer and one or

more user computers. The server is accessible to each user computer and therefore may store and control the master version of the file being collaborated on by the users.

[0005] When collaborating on an electronic file, a user accesses the file on the server. The user may then work on the file by altering the file in any way made available to the user by the application. For example, if the application is a word processor, spreadsheet, database, and/or the like, the application may provide means for performing tasks such as appending to, editing, copying, and/or deleting the file. At the completion of any alterations, the user may save changes to the version of the file stored on the server if the user wishes to retain such alterations to the file and has the requisite permissions.

[0006] While having a centrally-stored file is good for collaboration, requiring access to the server has a downside. A user must be connected to the server in order to access the server copy of the file. Replication engines are used to solve this problem. Such engines allow the user to have a local copy of the file on the server. This local copy allows the user to work with or view the file even when the server copy is not accessible. The replication engine manages the replication of the server state of the file to the user's local machine, giving the user offline access while attempting to maintain consistency between the server copy and the local copy.

[0007] This solution, however, has drawbacks, however, and such consistency is difficult or impossible to maintain using existing replication engines. The version of the file on the server computer may change while the user is not connected. For example, the version of the file on the server may be version A. This version A is copied to the user's computer by the replication engine. When the user is "offline" (disconnected from the server), the user edits the file, producing version B. Meanwhile, a second user edits the server copy of the file, creating version C. When the user again connects to the server, the user will have access to the user's current version (B) and the server version (C). Current replication engines provide only the ability to copy the user's version over the server version, in which case the second user's changes are lost, or to copy the server version over the user version. No information is available for the user regarding changes from version A. If the user synchronizes by saving version C and abandoning changes, version B will no longer be accessible to the user.

[0008] Because of these drawbacks, an improved way of managing replicated files from a server is desirable.

SUMMARY OF THE INVENTION

[0009] A server file replication technique is implemented which stores three copies of a replicated file. These three files allow improved conflict resolution for the user.

[0010] The shadow file is a copy of the last known good version of the file on the server. When the file is initially copied from the server, an extra copy is made to the shadow file. The shadow file is brought to the local machine to improve performance in synchronization and conflict management as well as afford an offline experience. The shadow file is updated from the server copy any time that a connection exists to the server and the server copy has changed from the version contained in the shadow copy. This is the only time this file is updated.

[0011] The local copy is the user's working copy. Whatever changes the user has made to the file are stored in the local copy. This local copy is used to update the server (in the case that the user's changes need to be pushed to the server) or to identify conflicts (when both the user's and the server versions have changed.)

[0012] A rollback file is also stored. This rollback file is a copy made of the local copy if the local copy changes (for example, when conflicts are resolved or the server version is copied to the local copy.) This copy ensures the ability to return the user to the last known copy that the user has seen.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] The foregoing summary, as well as the following detailed description of preferred embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the invention, there is shown in the drawings exemplary embodiments of the invention; however, the invention is not limited to the specific methods and instrumentalities disclosed. In the drawings:

[0014] Figure 1 is a block diagram showing an exemplary computing environment in which aspects of the invention may be implemented;

[0015] Figure 2 is a functional diagram illustrating an example of a relationship between a server computer, a network, a user computer, and other computers;

[0016] Figure 3 is a state diagram showing state transitions in accordance with one embodiment of the present invention;

[0017] Figure 4 is a flowchart showing file replication in accordance with an embodiment of the invention; and

[0018] Figure 5 is a flowchart showing file replication in accordance with an embodiment of the invention.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

Overview

[0019] A server file replication technique is implemented which stores three copies for each file on a server for while replication is in use. A local file stores is the user's working copy of the file. The shadow file is a copy of the last known good version of the server file (the version of the file on the server.) A rollback file stores a copy of the last local copy before a conflict resolution or replication has updated it.

[0020] When the file is initially copied from the server as the local copy, an extra copy is made to the shadow file. This is done in order to improve synchronization performance and implement conflict management.

[0021] When the user's computer is connected to the server, replication occurs. The shadow file is compared to the server version. If they are the same, then no update has been made to the server version of the file since the last synchronization. If they are not the same, the server version is copied to the shadow version. The local version is saved to rollback, and the shadow version is saved to the local version.

[0022] Versioning states track the state of changes to the local and server files. Information regarding versioning states and options for conflict management are provided to the user.

Exemplary Computing Device

[0023] FIG. 1 and the following discussion are intended to provide a brief general description of a suitable computing environment in which the invention may be implemented. It should be understood, however, that handheld, portable and other computing devices and computing objects of all kinds are contemplated for use in connection with the present invention, as described above. Thus, while a general purpose computer is described below, this is but one example, and the present invention may be implemented with other computing devices, such as a thin client having network/bus interoperability and interaction. Thus, the present invention may be implemented in an environment of networked hosted services in which very little or minimal client resources are implicated, e.g., a networked environment in which the client device serves merely as an interface to the network/bus, such as an object placed in an appliance, or other computing devices and objects as well. In essence, anywhere that data may be stored or from which data may be retrieved is a desirable, or suitable, environment for operation according to the invention.

[0024] Although not required, the invention can be implemented via an operating system, for use by a developer of services for a device or object, and/or included within

application software that operates according to the invention. Software may be described in the general context of computer-executable instructions, such as program modules, being executed by one or more computers, such as client workstations, servers or other devices. Generally, program modules include routines, programs, objects, components, data structures and the like 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. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer configurations. Other well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers (PCs), automated teller machines, server computers, hand-held or laptop devices, multi-processor systems, microprocessor-based systems, programmable consumer electronics, network PCs, appliances, lights, environmental control elements, minicomputers, mainframe computers and the like. 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/bus or other data transmission medium. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices, and client nodes may in turn behave as server nodes.

[0025] Fig. 1 thus illustrates an example of a suitable computing system environment 100 in which the invention may be implemented, although as made clear above, the computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0026] With reference to Fig. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer system 110. Components of computer system 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. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture

(MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus (also known as Mezzanine bus).

[0027] Computer system 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer system 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, Random Access Memory (RAM), Read Only Memory (ROM), Electrically Erasable Programmable Read Only Memory (EEPROM), flash memory or other memory technology, Compact Disk Read Only Memory (CDROM), digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can accessed by computer system 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[0028] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer system 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Fig. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0029] The computer system 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Fig. 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a

magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156, such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0030] The drives and their associated computer storage media discussed above and illustrated in Fig. 1 provide storage of computer readable instructions, data structures, program modules and other data for the computer system 110. In Fig. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer system 110 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus 121, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A graphics interface 182, such as Northbridge, may also be connected to the system bus 121. Northbridge is a chipset that communicates with the CPU, or host processing unit 120, and assumes responsibility for accelerated graphics port (AGP) communications. One or more graphics processing units (GPUs) 184 may communicate with graphics interface 182. In this regard, GPUs 184 generally include on-chip memory storage, such as register storage and GPUs 184 communicate with a video memory 186. GPUs 184, however, are but one example of a coprocessor and thus a variety of coprocessing devices may be included in computer system 110. A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190, which may in turn communicate with video memory 186. In addition to monitor 191, computer systems may

also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195.

[0031] The computer system 110 may operate in a networked or distributed environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer system 110, although only a memory storage device 181 has been illustrated in Fig. 1. The logical connections depicted in Fig. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks/buses. Such networking environments are commonplace in homes, offices, enterprise-wide computer networks, intranets and the Internet.

[0032] When used in a LAN networking environment, the computer system 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer system 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer system 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Fig. 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0033] Various distributed computing frameworks have been and are being developed in light of the convergence of personal computing and the Internet. Individuals and business users alike are provided with a seamlessly interoperable and Web-enabled interface for applications and computing devices, making computing activities increasingly Web browser or network-oriented.

[0034] For example, MICROSOFT®'s .NET platform includes servers, building-block services, such as Web-based data storage and downloadable device software. While exemplary embodiments herein are described in connection with software residing on a computing device, one or more portions of the invention may also be implemented via an operating system, application programming interface (API) or a "middle man" object between any of a coprocessor, a display device and requesting object, such that operation according to the

invention may be performed by, supported in or accessed via all of .NET's languages and services, and in other distributed computing frameworks as well.

Managing Multiple File States for Replicated Files

Replication

[0035] With reference to Figure 2, a server computer 200 is connected via network 215 to user computer 220. When the request is received by the server, the server allows the user computer 220 to open the file. File replication allows the user computer 220 to work, along with users on other user computers 270, on a local version of a server file. The authoritative version of the file is stored as server copy 210 on server computer 200. File replication manager 230 creates three copies of the file in user computer 220. These are the local copy 232, the shadow copy 234, and the rollback copy 238.

[0036] The local copy 232 is opened on the user computer 220 by an application 240. In one embodiment, there is a copy of the local copy 232 in the user's namespace on user computer 220. In this embodiment, the local copy 232 tracks changes in the namespace copy and the namespace copy is used by application 240.

[0037] Where a namespace copy is used, the namespace copy isn't managed by the file replication manager 230. In this embodiment, the namespace file is the file that the user generally accesses. This file could be a file that they've created or a file that they've received via e-mail. Where the file is received via e-mail, the user receives an "always live attachment" (with server information) and save a copy to their local computer. Just the act of saving this file means that there are two copies, the one in the mail store and the one on the local machine. The local machine copy is the namespace copy. The server information in the attachment allows for file replication and the creation of the local copy 232, the shadow copy 234, and the rollback copy 238. Namespace copies are synchronized by the replication manager 220 with the local copy 232 when they are being used by a user.

[0038] The shadow copy 234 maintains the version of the server copy 210 last known to user computer 220. When the user computer 220 is connected to the server computer 200, the shadow copy 234 is the same as the server copy 210.

[0039] The rollback copy 238 stores the last known copy that the user has viewed or edited. When the local copy 232 has been overwritten in the replication process but the user wishes to return to the previous state, the rollback copy 238 is used to provide the last copy the user has read or edited.

[0040] The file replication manager 230 may be separate from the application 240, as shown, or it may be integrated into application 240. In either case, messages and information for the user from the file replication manager, and user interfaces collecting user input and displaying messages for the user may be integrated into the display presented to the user application 240. Information presented to the user regarding changes made between two version of a file (further described below) may also be presented using the display of application 240. Additionally, the production of such information for display to the user may utilize functionality of application 240.

Versioning States

[0041] The action of the file replication techniques according to one embodiment of the invention can be understood by reference to versioning states. For each replicated document, a versioning state describes what is occurring with the document at user computer 220. These versioning states are based on previous versioning states and the actions occurring in the local copy 232, the shadow copy 234, and the server copy 210 (the master copy of the document).

[0042] The first state is "current". The versioning state is current when the local copy 232 is the same as shadow copy 234. For example, when a file which is saved on user computer 220 is uploaded to server computer 200 for collaboration, the local copy 232 is the same as shadow copy 234. Similarly, when a file on server computer 200 is first replicated to user computer 220, the local copy is equal to shadow copy 234. As shown in the state diagram of Fig. 3, when a change is made to the local copy 232, so that it is no longer the same as shadow copy 234, the state transitions from "current" state 300 to "modified" state 310. When this modification is pushed up to the server or abandoned, so that the local copy 232 is again equal to the shadow copy 234, then the state returns to "current" state 300.

[0043] When the user computer 220 is connected to server computer 200 and notes that the server copy 210 has been updated, a copy of the new server copy 210 is made to shadow copy 234. The local copy 232 is saved as rollback copy 238, and the shadow copy 234 is then copied to local copy 232. In one embodiment, the last copy viewed by the user is tracked. Thus, if the user has viewed version A, and version B is downloaded from the server, the rollback will be a copy of version A. But if version C now is downloaded from the server, the rollback will not be changed to version B unless version B has been accessed by the user. Thus the rollback holds the last version looked at by the user.

[0044] As seen in Fig. 3, if a server file change is downloaded to the shadow file and a rollback copy is stored, and the state is "current" state 300, the new state will be "rollback

available" 320. If, however, this occurs in "modified" state 310, then there have been both local changes to the a previous version of the server copy, which are saved in the local copy 232, and some other changes (from other users) resulting in a new version of the server copy 210. This results in the "conflicted" state 330.

[0045] When a "conflicted" state 330 occurs, a resolution of the two possibly divergent sets of changes in local copy 232 and shadow copy 234 must occur. For example, the user could choose to abandon changes which had been made in local copy 232 or to overwrite the changes. The file resulting from this resolution is stored as local copy 232, and the state returns to "modified." As noted previously, when the changes in local copy 232 are pushed up to the server or abandoned in favor of the current server copy, the state is changed to "current."

Uploading Changes to the Server

[0046] In each of the versioning states, synchronization options are available to a user. These options may be separately available from the file replication manager 230 or may be integrated into application 240. If changes have been made, uploading (or "pushing") the changes to the server copy 210 is one option. This synchronization option allows the user to connect to the server computer 200 (if a connection is available) and synchronize the local copy 232 with the server copy 210.

[0047] If the versioning state is current 300 or rollback available 320, then no modifications have been made, and so no file is pushed to the server. If the versioning state is the modified state 310, the differences between the local copy 232 and the shadow copy 234 are determined. This process is known as "generating a diff" (for difference) of the two files. If the versioning state was modified state 310, the shadow copy 234 is the same as the server copy 210 (otherwise the versioning state would be conflicted state 330). Therefore the entire file does not need to be uploaded to the server – a file which includes all information regarding differences between the local copy 232 and shadow copy 234 contains all necessary information. This diff file is used by server computer 200 to update the server copy 210. Once the server copy 210 is updated, it should be copied to the shadow copy 234 and the versioning state changed appropriately.

[0048] The generation of the diff file may be done off-line. The user may decide to request synchronization at a time when the server computer 200 is not reachable from the user computer 220. If this happens, then the diff may be created and the action queued so that next time the server computer 200 is accessible, the uploading process may continue.

[0049] If the versioning state is conflicted state 330, then a conflict user interface must be begun. This conflict user interface should allow the user to resolve the differences between the local copy 232 and the shadow copy 234, for example, by saving the local copy 232 elsewhere on user computer 220 and copying the shadow copy 234 as the new local copy 232. The user interface may also be a more complex interface, as described in the related application listed above.

[0050] In one embodiment, checks are made to ensure that no local change has been made when the file is in the current state 300 or the rollback available state 320, and to ensure that the upload of the diff file was successful. If these checks fail, a modified state 310 or conflicted state 330 should result.

Rollback Version

[0051] Another option available to the user is when the versioning state is rollback available 320 or modified 310. A user interface will then be available to the user which will allow them to roll back to the version of the document previously viewed or edited by the user, which is stored in rollback copy 238.

[0052] In one embodiment, the user interface is dnisplayed to the user, indicating that the document has been updated by another user. The user interface allows the user to select the previous version, stored in the rollback copy 238, or to view the differences between the previous version, stored in the rollback copy 238, and the current version, stored in local copy 232.

[0053] The rollback copy is only present in some embodiments of the invention. As shown in figure 4, file replication of a server file occurs on a user computer according when a local copy is stored (step 400) and a shadow copy is stored (step 410). Access is permitted to the local copy for modification (step 420). In some embodiments, this is done through a namespace copy of the local copy. When the user computer is connected to the server computer and the master file changes, then the shadow copy is updated (step 430). In one embodiment, versioning states are tracked for the replication.

[0054] Where the rollback copy is kept, as shown in Figure 5, again, file replication of a server file occurs on a user computer according when a local copy is stored (step 500) and a shadow copy is stored (step 510). Access is permitted to the local copy for modification (step 520). When the user computer is connected to the server computer and the master file changes, then the shadow copy is updated (step 530). In this step, when the shadow copy is updated, the rollback copy is also updated, if necessary. First, the local copy is stored as the rollback copy,

and the local copy is then updated with the shadow copy. In one embodiment, this is done only if the local copy has been accessed since the previous rollback.

[0055] The file replication manager 220 may be a separate application or may be wholly or partially integrated into the application used with the file being replicated.

Conclusion

[0056] As mentioned above, while exemplary embodiments of the present invention have been described in connection with various computing devices and network architectures, the underlying concepts may be applied to any computing device or system in which it is desirable to implement collaboration via server. Thus, the methods and systems of the present invention may be applied to a variety of applications and devices. While exemplary programming languages, names and examples are chosen herein as representative of various choices, these languages, names and examples are not intended to be limiting. One of ordinary skill in the art will appreciate that there are numerous ways of providing object code that achieves the same, similar or equivalent systems and methods achieved by the invention.

[0057] The various techniques described herein may be implemented in connection with hardware or software or, where appropriate, with a combination of both. Thus, the methods and apparatus of the present invention, or certain aspects or portions thereof, may take the form of program code (i.e., instructions) embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other machine-readable storage medium, wherein, when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for practicing the invention. In the case of program code execution on programmable computers, the computing device will generally include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. One or more programs that may utilize the signal processing services of the present invention, e.g., through the use of a data processing API or the like, are preferably implemented in a high level procedural or object oriented programming language to communicate with a computer. However, the program(s) can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language, and combined with hardware implementations.

[0058] The methods and apparatus of the present invention may also be practiced via communications embodied in the form of program code that is transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via any other form of transmission, wherein, when the program code is received and loaded into and

executed by a machine, such as an EPROM, a gate array, a programmable logic device (PLD), a client computer, a video recorder or the like, or a receiving machine having the signal processing capabilities as described in exemplary embodiments above becomes an apparatus for practicing the invention. When implemented on a general-purpose processor, the program code combines with the processor to provide a unique apparatus that operates to invoke the functionality of the present invention. Additionally, any storage techniques used in connection with the present invention may invariably be a combination of hardware and software.

[0059] While the present invention has been described in connection with the preferred embodiments of the various figures, it is to be understood that other similar embodiments may be used or modifications and additions may be made to the described embodiment for performing the same function of the present invention without deviating therefrom. Furthermore, it should be emphasized that a variety of computer platforms, including handheld device operating systems and other application specific operating systems are contemplated, especially as the number of wireless networked devices continues to proliferate. Therefore, the present invention should not be limited to any single embodiment, but rather should be construed in breadth and scope in accordance with the appended claims.

What is claimed:

1. A method for file replication on a first computer of a master file on a second computer, comprising:
storing a local copy on said first computer of said master file;
storing a shadow copy on said first computer of said master file;
permitting access to said local copy for modification; and
updating said shadow copy when said master file changes, where said step of updating said shadow copy when said master file changes comprises:
storing said local copy as a rollback copy; and
updating said local copy with said shadow copy.

2. The method of claim 1, where said method further comprises:
storing a versioning state associated with said replication,
and where said step of permitting access to said local copy for modification comprises:
changing said versioning state when said local copy is modified.

3. The method of claim 2, where said step of changing said versioning state when said local copy is modified comprises:
setting said versioning state to a modified state indicating that said local copy has been modified.

4. The method of claim 1, where said method further comprises:
storing a versioning state associated with said replication;
and where said step of updating said shadow copy when said master file changes comprises:
changing said versioning state when said updating occurs.

5. The method of claim 1, where said step of updating said shadow copy when said master file changes comprises:
detecting that said local copy has been modified; and
resolving conflicts between said local copy and said updated shadow copy.

6. The method of claim 5, where said method further comprises:
storing a versioning state associated with said replication;
and where said step of resolving conflicts between said local copy and said updated shadow copy comprises:
changing said versioning state to a conflicted state indicating that a conflict exits.

7. The method of claim 5, where said step of resolving conflicts between said local copy and said updated shadow copy comprises:
presenting a conflict interface;
accepting a conflict resolution command; and
implementing said conflict resolution command.

8. The method of claim 7, where said implementation of conflict resolution commands comprises:
determining if a change should be made to said master file; and communicating said change to be made to said second computer.

9. The method of claim 8, where said step of communicating said change to be made to said second computer comprises:
generating difference information regarding differences between said local copy and said master file; and
communicating said difference information to said master file.

10. The method of claim 1, where said step of storing said local copy as a rollback copy comprises:
presenting a rollback interface;
accepting a rollback command; and
displaying said rollback copy in response to said rollback command.

11. The method of claim 1, where said method further comprises:
storing a versioning state associated with said replication;
and where said step of storing said local copy as a rollback copy comprises: changing said versioning state to a rollback available state.

12. The method of claim 1, further comprising:
tracking whether the local copy has been accessed;
and where said step of storing said local copy as a rollback copy comprises: storing said local copy as a rollback copy only if said local copy has been accessed.

13 . At least one of an operating system, a computer readable medium having stored thereon a plurality of computer-executable instructions, a co-processing device, a computing

device, and a modulated data signal carrying computer executable instructions for performing the method of claim 1.

14. A computer system for replication of a master file, said master file located on a second computer, said computer system at least intermittently operably connected to said second computer comprising:
local copy storage for storing a local copy on said first computer of said master file;
shadow copy storage storing a shadow copy on said first computer of said master file;
local copy modification module for permitting access to said local copy for modification; and
shadow copy updating module updating said shadow copy storage when said master file changes, where said shadow copy updating module comprises:
rollback storage for storing said local copy as a rollback copy; and
local copy update module for updating said local copy with said shadow copy.

15. The computer system of claim 14, further comprising:
versioning state storage for storing a versioning state associated with said replication,
and where said local copy modification module changes said versioning state when said local copy is modified.

16. The computer system of claim 15, where said where said local copy modification module changes sets said versioning state to a modified state indicating that said local copy has been modified.

17. The computer system of claim 14, where said system further comprises: versioning state storage for storing a versioning state associated with said replication;
and where said shadow copy updating module changing said versioning state when said updating occurs.

18. The computer system of claim 14 , where shadow copy updating module comprises:
local copy modification detection module for detecting that said local copy has been modified; and conflict resolution module for resolving conflicts between said local copy and said updated shadow copy.

19 The computer system of claim 18, where said method further comprises:
versioning state storage for storing a versioning state associated with said replication;

and said conflict resolution module changes said versioning state to a conflicted state indicating that a conflict exits.

20. The computer system of claim 18, where said conflict resolution module comprises:
conflict interface module for presenting a conflict interface;
conflict resolution input module for accepting a conflict resolution command; and
conflict resolution implementation for implementing said conflict resolution command.

21. The computer system of claim 20, where said conflict resolution implementation comprises:
change determination module for determining if a change should be made to said master file; and
second computer change communication module for communicating said change to be made to said second computer.

22. The computer system of claim 14, where said second computer change communication module comprises:
difference generation module for generating difference information regarding differences between said local copy and said master file; and
second computer communication module for communicating said difference information to said master file.

23. The computer system of claim 14, where said local copy update module comprises:
a rollback interface presentation module presenting a rollback interface; rollback command acceptance module accepting a rollback command; and rollback copy display for displaying said rollback copy in response to said rollback command.

24. The computer system of claim 14, where said system further comprises: versioning state storage for storing a versioning state associated with said replication,
and where said rollback storage changes said versioning state to a rollback available state.

25. The computer system of claim 14, said system further comprising:
local copy access tracking module for tracking whether the local copy has been accessed;
and where said rollback storage for storing said local copy as a rollback copy comprises:
local copy access logic for storing said local copy as a rollback copy only if said local copy has been accessed.