A system and method of synchronizing electronic files over a network. A node on the network has a differencing module. The differencing module can generate a differencing file from a reference file and a modified file. When the differencing file is be sent to another device over the network, the other device can regenerate the modified file.
ABSTRACT

A system and method of synchronizing electronic files over a network. A node on the network has a differencing module. The differencing module can generate a differencing file from a reference file and a modified file. When the differencing file is sent to another device over the network, the other device can regenerate the modified file.
SYSTEM AND METHOD FOR DATA COMMUNICATION OVER A NETWORK

TECHNICAL FIELD

[0001] The following relates generally to data communication over a network.

BACKGROUND

[0002] In geographically remote locations, access to a reliable network may not be possible. For example, in developing countries or mining sites, the infrastructure required to provide a stable, continuous and high-bandwidth network connection may not be available. Access to a reliable network via portable infrastructure such as a satellite receiver, for example, may also be hindered by weather or other environmental conditions. Network access may also be intermittently available and could potentially be unavailable for unpredictable periods of time. Limitations and fluctuations in the bandwidth of the network may affect the performance of a network.

[0003] In certain applications, such as mining, research, or prospecting, for example, it may be required that information be shared between devices located at the site and devices located remotely, such as in an office or server site. Exchanging or synchronizing information between these devices may be difficult if the network connection has low bandwidth or is only intermittently available.

[0004] These issues can be exacerbated where the application requires transfer of relatively large files.

[0005] Certain disadvantages are apparent when using existing synchronization protocols. Some such protocols require a first device to communicate with a second device multiple times during synchronization. If the network is unreliable, the synchronization process stalls or fails. Additionally, some such protocols require time consuming file processing to determine optimal data exchange to accomplish synchronization. Where the network is unreliable, the aggregate time taken for communication and computation may be unreasonably long.

[0006] Many existing synchronization protocols also do not support compression and, therefore, depend even more heavily on maintaining a network connection between the devices. These may not be suitable for unreliable networks.
[0007] It is an object of the present invention to mitigate or obviate at least one of the above disadvantages.

SUMMARY

[0008] The present invention provides a system for enabling data communication over a network, the system comprising a network-connected device comprising a differencing module operable to generate a difference file from a reference file and a modified file, wherein the differencing file may be sent to another device over the network to enable the other device to generate the modified file.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] Embodiments will now be described by way of example only with reference to the appended drawings wherein:

[0010] FIG. 1 is a block diagram illustrating a system in accordance with the present invention;

[0011] FIG. 2 is a block diagram illustrating a node;

[0012] FIG. 3 is a flow diagram illustrating generating a difference file;

[0013] FIG. 4 is a flow diagram illustrating a version history of an example file; and

[0014] FIG. 5 is an example diagram illustrating various types of data and links therebetween relevant to a mining operation.

DETAILED DESCRIPTION OF THE DRAWINGS

[0015] The present invention provides a file management system. The file management system enables transfer and synchronization of files over a network enabling data communication between one or more devices. The network may be one that is unreliable or reliable and may exhibit characteristics comprising low bandwidth and/or low quality of service (QoS). Devices communicating with one another over the network may, for a particular implementation, benefit from a relatively higher rate of transfer and/or level of reliability (e.g. QoS) than is otherwise possible given the network characteristics.

[0016] Turning to Fig. 1, a plurality of nodes 10 is shown. A node 10 may be a computer device, such as a desktop computer, a laptop computer, a mobile device such as a
smartphone, a piece of industrial equipment (e.g. an automated drill), a piece of sensing equipment (e.g. an aerial gravitometer), a rack-mount server, a cloud-based server, or any other computer device. The node 10 may comprise or be linked to a processor 9 and a memory 8. The memory may have stored thereon computer instructions which, when executed by the processor, provide the functionality of the file management system as described herein.

[0017] The node 10 may further comprise or be linked to a transceiver 15. The transceiver 15 may be linked to a network 14, such as an intranet or the Internet. Further nodes 10 being substantially similar to the aforementioned node 10 may also be linked to the network. Each node 10 may be operable to communicate with one or more other nodes 10 via the network.

[0018] The node 10 may be user-controllable or automatically controlled.

[0019] Each node 10 may further comprise or be linked to a data store 16. The data store 16 is operable to have stored thereon at least one file. The at least one file may comprise at least one reference file, and at least one difference file associated with each reference file.

[0020] A reference file may be a file that can be considered a complete file, in that if a node 10 provides the reference file to another node 10, the other node 10 can receive and read the reference file for the purposes of operating upon it (i.e., opening file, modifying the file, etc.).

[0021] A difference file may be a file providing information sufficient to generate a file that can be operated upon when used in conjunction with a reference file. For example, a difference file may map the differences between two files. For example, a node 10 having received a difference file and a reference file can recover information relating to modifications that have been made relative to the reference file. The node 10, having the reference file and the difference file, can generate a modified file corresponding to the modifications made relative to the reference file and can operate upon the modified file.

[0022] As will be appreciated, if the amount of modification made to a file is relatively small, a difference file may be substantially smaller than the size of a reference file.

Transferring only a difference file, rather than the entire reference file between nodes 10
over the network 14 may substantially reduce the amount of data that must be transferred over the network 14, which may enable faster and more efficient synchronization of files.

[0023] In one aspect, one of the nodes 10 may be referred to as a "server" for the purposes of storing one or more particular reference files and associated difference files.

[0024] In another aspect, which may be considered a distributed embodiment of the above, a plurality of nodes 10 may each be referred to as a server, wherein each node 10 may be a server of particular reference files and associated difference files. For example, a first reference file may be provided on a first node 10 which is the server of the first reference file, while a second reference file may be provided on a second node 10 which is the server of the second reference file, and so on for any number of reference files. As will be appreciated, any particular node 10 could also be the server of a plurality of such reference files.

[0025] The server may track and store data associated with files, for example, the version number of the file. Newer versions of a file may comprise incremental changes from an older version of the file. Version information controlled by the server may be accessed by the nodes 10. By centralizing storage the version number for a particular file, the server is operable to determine whether each node 10 is operating upon the most recent version of that file.

[0026] In certain embodiments, at least one of the nodes 10 may further comprise or be linked to a database 17. The database may be used to store information relating to files stored on the data store 16. For example, a server may store on its database 17 a version number for each reference file (which may correspond to the number of associated difference files for that reference file) stored on its data store, as well as one or more annotations for each such reference file and/or difference file. These annotations may be implemented by metatags associated with such file. Nodes other than the server may also comprise or be linked to a database for storing information relating to files stored on their respective data stores, for example version numbers of such files.

[0027] Referring now to Fig. 2, each node 10 may comprise a differencing module 20 and a synchronization module 22. The node 10 may further comprise a modification detection module 28, a compression module 24, a search module 26 and a file monitor 30.
[0028] The differencing module 20 is operable to generate a difference file from a reference file and modifications made relative to the reference file (which may be embodied in a permanently or temporarily stored modified file). The modified file need not be stored on the date store at any time, though it may be stored in memory 8 while the node 10 is operating upon it. When provided with a difference file (or a differencing file, which is described below), the differencing module 20 is also operable to apply the difference file to the reference file (or a differencing file) to generate a modified file to be operated upon.

[0029] The synchronization module 22 may be operable to facilitate the synchronization of files between a node 10 and a server. The synchronization module 22 may cooperate with the node's transceiver 15 to enable a node 10 to communicate (receive and transmit) both reference files and difference files and differencing files with another node 10.

[0030] The node's modification detection module 24 is operable to detect modification to a file on a node 10 regardless of whether the node 10 is connected to a server over a network 14.

[0031] The node 10 and/or the server may comprise a compression module 24. The compression module 24 may be operable to compress and decompress reference files and difference files. Compression of files may be provided for reducing bandwidth requirements of communicating files between nodes 10 in order to reduce the amount of data that must be sent over the network 14, thereby reducing the time required to send the file.

[0032] The node's search module 26 may be operable to perform searches for files regardless of the connectivity state of the node 10 to the network 14. For example, the node's search module 26 may enable searching for files on other nodes despite an unreliable connection. Typically, performing a search from a node 10 for information on a server requires a connection between the server and the node 10. Over an unreliable network, the connection may not always be available, which may cause long search times and timed-out searches.

[0033] In operation, in one example, a node 10 (or a user of the node 10) may request to access a particular file from a server. The node's synchronization module 22 determines whether any version of the requested file is stored on the node's data store 16.
[0034] If there is no such file stored on the node’s data store 16, the node’s synchronization module 22 may transmit a request to the server’s synchronization module 22 to provide the node 10 with the current version of the particular file.

[0035] The server’s synchronization module 22 may access the database 17 to determine the version number of the requested file, and correspondingly access its data store to retrieve the reference file and any associated difference files (the number of which may be derived from the version number). The server’s synchronization module 22 may request the server’s differencing module 20 to generate a modified file from the reference file and associated difference files. For example, a modified file generated from a reference file associated with four difference files may have a version number of five. The server’s synchronization module 22 may transmit the modified file and the version number to the node’s synchronization module 22.

[0036] The node’s synchronization module 22 may store the modified file on its data store along with its corresponding version number on its database. The node 10 may then operate upon the modified file.

[0037] Alternatively, if the node’s data store 16 already has any version of the requested file, the node’s synchronization module 22 may determine its corresponding version number, for example, by accessing the node’s database which may store the version number for each file stored on the node’s data store. The node’s synchronization module 22 may transmit the version number to the server’s synchronization module 22. The server’s synchronization module 22 may access its database 17 to determine the version number corresponding to the requested file on the server’s data store. If the version numbers are the same, the server need not transmit any file to the node 10. If the version numbers are different, the server’s synchronization module 22 may direct its differencing module to generate a differencing file corresponding to the set of difference files for the intervening version numbers. The server’s synchronization module 22 may transmit the differencing file and version number to the node’s synchronization module 22. The node’s differencing module 20 may then generate a modified file by applying the differencing file to its version of the file. The node 10 may store the modified file in its data store 16 and may store the version number in its database 17. The node 10 may overwrite its previous version of the requested file with the modified file.
In either of the above examples, the server may store in its database an indicator that the node has accessed the file.

The node 10 may then operate upon the file. Once the node 10 has finished modifying the file (resulting in a further modified file), the differencing module 20 may construct a difference file based on the further modified file and the (received) modified file. The node may additionally store the further modified file on its data store and an updated version number on its database. The node may overwrite the (received) modified file with the further modified file on its data store. The node’s synchronization module 22 may transmit the difference file to the server’s synchronization module 22. The server’s differencing module 20 may save the difference file to its data store 17 and update the version number corresponding to that file on its database 16.

The file management system may further enable a first node 10 and a second node 10 to synchronize a file based on one of said nodes 10 having made modifications to the file. Once modifications are made on a first node 10, the updated file may be provided to the second node 10. This may be done, for example, by the node 10 providing the difference file to a server in accordance with the foregoing, the server determining from its database other nodes that have accessed the file (in this case, the second node), the server updating the version number of the file in accordance with the foregoing, the server requesting that the second node 10 provide it with its version number for the file, and the server 16 correspondingly generating a differencing file to enable the second node 10 to generate a modified file in accordance with the foregoing.

Such a request from the server 16 to any one or more nodes may be provided as follows. The server’s synchronization module 22 may transmit a notification to other nodes 10 that have previously accessed the reference file associated with a received difference file (from a node 10). Any such other node’s synchronization module 22 may correspondingly determine the version number of its corresponding copy of the file and send the version number to the server. The server’s synchronization module 22 may transmit to each node 10 a differencing file to update the node’s file to the current version (note that any two or more of such nodes 10 may receive a distinct differencing file as they may not have been previously updated to the same version number, for example if any such nodes 10 were offline at the previous update). Such nodes’ differencing module 20 may
update such file on its data store, along with a corresponding version number on its
database when it receives the differencing file.

[0042] The file management system also enables a node 10, or a user operating a node
10, to view the difference the actual contents of the file, as well as the metatags associated
with the state of the locally modified file and the state of the file on the server. With an
unreliable network connection, the server may be out of contact for a period of time while
the file is being modified at the node 10.

[0043] The reference file preferably corresponds to the file on the server at the time that
the file was most recently synchronized. When a network connection between a first node
and the server is unavailable, a difference file may be generated based on the reference file
and the modified version of the file. The modification detection module 24 may be operable
to determine whether the reference file differs from a more recent version of the file on the
node. The modification detection module 24 may provide updated information regarding the
state of the local file with respect to the state of a reference file in the temporary absence of
a network connection. The modification detection module 24 may also provide an indication
to a user that the reference file differs from a modified file, as is described below. Although
the file on the server may have been modified since the last synchronization by a second
node, the first node or a user at the first node may be able to determine, from the indication,
which of the files have been modified at the node since the time of the last synchronization.

[0044] The node may be provided with a display, the display being operable to provide a
displayed list of files on a node. The list of files may be provided with a visual indication that
provides the user with an indication that the version of the file on the node differs from the
reference file, based on information received from the modification detection module 24.
The indication that the file on the node differs from the reference file may comprise, for
example, a pair of arrows that point in opposite directions when the files differ. The
indication may also comprise further details, for example, the date and time that the file was
last modified as well as the date and time that the last synchronization occurred, or a
percentage outlining what percent of the blocks in the file are identical. Other indications
that provide the user with a sensory experience based on differences in the files may also
be possible.
The file management system may further provide compression of files. In certain implementations, depending on the size of a file and the bandwidth and reliability of the network 14, the amount of time required to transfer the file between nodes 10 over the network 14 may be inconveniently or prohibitively long. To reduce the amount of data that must be transferred over the network 14, a node's compression module 24 may compress the file (that is being transmitted) by applying a compression algorithm. The compression algorithm may comprise, for example, VCDIFF, another format for encoding compressed and/or differencing data, or any appropriate compression algorithms for the types of files being transmitted. By compressing the file, the size of the file may be reduced and, consequently, the time required to transfer the file over the network 14 may be reduced.

Referring now to Fig. 3, a method of generating a difference file is now provided. In step 36, the differencing module 20 may segment a reference file into a plurality of blocks. The size of the blocks may be determined based on preconfigured segmenting parameters which the differencing module 20 may adaptively adjust. Examples of block sizes may be 4092 bytes, 8192 bytes, etc. The differencing module 20 may compute a hash of each block, and assign an identifier (e.g., a number) to each of the blocks in step 38. The differencing module 20 may segment the corresponding modified file in step 40, assign an identifier (e.g., a number) and compute the hash of each of its blocks in step 42. The hash of the blocks of the modified file may be compared with the hash of the blocks of the reference file in step 44. The differencing module 20 may determine which blocks have been modified between the modified file and the reference file in step 46. The differencing module 20 may generate a difference file comprising the modified blocks and modified block identifiers in step 48. It will be appreciated that the differencing module 20 may alternatively compare the blocks without hashing. It will also be appreciated that certain of the foregoing steps can be performed in different sequences without an affect on functionality.

The differencing module 20 may segment blocks by setting markers based on the contents of the file, enabling a modified file to be compared to a reference file even if it comprises significant rearrangement. For example, each block may be hashed. The hashes of one file may then be compared with the hashes of another file to determine which blocks are identical and which blocks must be transmitted in a difference file over the network 14, to synchronize the files.
Preferably, the hashes are compared by the node's differencing module 20 rather than at the server, however the hashes may be transmitted to the server for comparison by its differencing module 20. Since only those blocks that have been changed may need to be transferred over the network 14, the generated difference file may comprise the data located in these blocks.

To further overcome the drawbacks of existing difference file-based synchronization methods, and to increase the speed and reliability of the file management system operating over the unreliable network 14, the differencing module 20 may calculate and cache the difference file locally before the synchronization process is initiated. For example, if the differencing module 20 calculates the difference file on a node 10, the difference file may be cached on the node's memory 8 prior to the synchronization process being initiated.

In other words, instead of computing the difference files at the time that a network connection is re-established with the server, the difference files may be computed by the node prior to re-establishing the network connection. For example, the node may compute the difference files in the background while performing other tasks. Once the network connection is re-established, the difference files are computed and ready to be transmitted over the network. By distributing the difference file calculation over a longer period of time, including while a network connection is unavailable, the computation steps required once the network connection is re-established may be dramatically lower than if the difference files were calculated at the time that the network is re-established. By having the difference files ready to synchronize at the time that the network connection is re-established, the synchronization process may be completed more quickly, with lower risk of again losing the connection during synchronization, and with less interruption to other activities that require use of the node's processor.

The reference file may be used to calculate the difference file. The differencing module 20 may calculate at least a portion of the difference file prior to initiating the synchronization process, reducing the length of time required to perform the synchronization process. The differencing module 20 may also compute the difference file while the modifications to the file are taking place. If so desired, the difference file may also be calculated during the synchronization process.
[0052] If the network 14 is unreliable, for example the network connection between the node 10 and the server is lost during synchronization, the synchronization module 22 may apply a break and resume transfer algorithm to continue synchronization when the network connection is re-established. The break and resume transfer algorithm may be any algorithm enabling a file to be transferred where it has previously not been transferred or only partially transferred.

[0053] Additionally, some files may comprise references to other files, for example to provide input data or other information. For example, a first file may comprise a spreadsheet that retrieves data from a comma separated value (.csv) file to perform a pre-determined calculation. If the node's synchronization module 22 is set to synchronize the spreadsheet, all other related data, for example, the .csv file, may also be synchronized. As the spreadsheet file may not be used without the .csv file, associating the .csv file to the spreadsheet for the purposes of the synchronization process prevents erroneous or non functional files. If one of the files cannot be synchronized, a warning message may appear to alert a user at a node 10 that a related file is missing. Alternatively, the file management system may block the file from being shown or delete the file, as this file may contain or initiate an error at another node 10. The newly synchronized file may remain hidden until all associated files may be synchronized.

[0054] Turning to Fig. 4, a graphical representation of a plurality of versions of a file is shown. Preferably, only one node 10 may modify a particular file at any given time, to reduce the likelihood that two nodes 10 will operate upon a particular version of the file and attempt to synchronize the file. Thus, a node 10 may be restricted to updating only the most recent version of the file on the server. If a node 10 accesses a particular file, the server’s synchronization module 22 may indicate in its database that the file is “checked out” by the node 10. The node 10 that has checked out the file may be given the authority to designate a file as a master copy when the synchronization module 22 synchronizes the file with the server. The master copy designation may be saved on the database 17. The node 10 may then check in the file to allow other nodes 10 to designate the file as a master copy. The node 10 may save the file as a new version and store the version information in the database 17.
The version information, which may comprise information indicating whether the file is a master copy, may be stored on the database 17, whereas the reference file itself, and any associated difference files, may be stored on the data store 16. If another node 10 attempts to access the file, the other node 10 may be provided the file, however, because the first node 10 had already checked out the file, any modifications made by the second node 10 will not be applied as differencing files for that file. The second node 10 may save its modifications to a new file, however.

In one embodiment, when the synchronization module 22 synchronizes a file on a node 10 with the corresponding file in the data store 16, the file may remain on the server. Similarly, when the synchronization module 22 updates a file on a node 10 with a newer version that is available on the data store 16 and delivered through the server, the older version of the file may not be deleted. The older version of the file may be retained and the update may be stored by way of storing one or more difference files that are associated with the file.

For example, if the reference file on node 10 is version one of a file and has since been updated twice to yield version three, the synchronization module 22 on node 10 provides the node 10 with both updates in the form of two difference files or a differencing file. A first difference file may provide the differencing module 20 with the information necessary to construct version two. The second difference file may provide the differencing module 20 with the information necessary to construct version three of the file. In this embodiment, both the first version of the file, as well as the difference files that enable the differencing module 20 to construct the second and third versions of the file, may be necessary if all versions of the file are to be accessed. Alternatively, the node may apply a differencing file to obtain the corresponding version of the file.

As can be seen in Fig. 4, the synchronization module 22 on a node 10 may cause the server to update its data store 16. For example, if modifications to the file were made on a node 10 that had checked out the file, the synchronization module 22 may provide the server with the difference file that had been calculated by the node’s differencing module 20. The server may store this difference file on its data store. The difference file may then be used by the differencing module of the server to construct the second version of the file 52. Alternatively, the server may store the original file on the data
store 16 as well as each of the difference file updates. It may be noted that in this example, the original file, as well as the difference files provided to the data store 16, are saved on the data store of the server.

[0059] For example, the original file 50 may be uploaded to the data store 16 on the server. The synchronization module 22 of a node 10 may then check out the original file. While the file is checked out, a second node 10 may access the original file 50 on the data store 16 of the server by downloading the original file. Once the second node 10 has finished modifying the file, the second node’s synchronization module 22 may provide the modifications to the data store 16 on the server. To provide the modifications to the data store 16 on the server, the second node’s differencing module 20 may calculate a difference file locally on the second node based on the original file 50 and the modified file. The second node’s synchronization module 22 may then provide the difference file 54 to the server to store on its data store 16. Since the original file 50 was checked out by the first node 10, the file produced by the second node 10 may not be designated as a master copy. Hence, the difference file may be saved separately. The information relating to the file’s version may be stored on the database 17 of the server.

[0060] If the second module 10 later performs further modifications to the file, a second difference file 56 may be saved on the data store 16 of the server. Once the first node 10 has finished modifying the file, the first node’s differencing module 20 may compute the difference file, which may then be provided to the data store 16 of the server by the synchronization module 22. As the file was checked out by the first node 10, the difference file uploaded to the data store 16 of the server corresponds to the second master version. Similarly, if the first node 10 then made a further modification to the file, the synchronization module 22 may provide a further difference file 58 to the data store 16 of the server as a master version of the file and save the corresponding version information on the database 17 of the server. The first node 10 may then check in the file once the first node 10 has completed the edits.

[0061] If, for example, a third node 10 then checks out and modifies the file, the synchronization module 22 may provide the resulting difference file, as outlined in the process explained above, to the data store 16 of the server. This difference file 62 may be stored on the data store 16 of the server with the master copy designation. The information
relating to the master copy designation may be saved in the database 17 of the server. However, if while the third node 10 had checked out the file, the first made further modifications to the file corresponding to difference file 58, the modifications may be uploaded to the data store 16 of the server in the form of a difference file 60 by the synchronization module 22 but may not be saved as a master copy. If the first node 10 then modified the file further, the synchronization module 22 may provide a copy of the most recent difference file 64 computed by the differencing module 20 to the data store 16 of the server. Hence, at each new update of the original file, a new difference file is provided to the data store and no files are deleted.

The node 10 may further log operations performed on files stored on its data store, for example to determine the history of file updates, particularly if the node 10 is a server for such files. The operations performed on the data store 16 may be identified with, for example, a timestamp, identification of the node 10 (and/or its user), location of the node 10 (and/or its user), MAC address/computer ID, etc.

It may also be possible for a node 10 to request a version of the file from the server that is not the most recent version of the file stored on the server. The node 10 may also request a copy of the file that is more than one version behind the most recent version. To enable a node 10 to access older versions of the file as well as update the file from an older version to the most recent version, the server's differencing module may be operable to generate a checkpoint file. A checkpoint file is a complete file that can be accessed by a node 10 without requiring the server to apply a difference file to a reference file. A checkpoint file may be saved at predetermined intervals to reduce the number of computations that must be performed by the server's differencing module 20 if there are many versions of file. The file management system may be operable to track the number of times a new version of a particular file has been saved. The file management system may be also operable to save a checkpoint file based on other parameters, for example, the number of version changes, the date, the time since the last checkpoint was saved, the amount of content that has changed between version updates, etc.

The file management system may also be operable to save checkpoint files at intervals that are based upon how often nodes 10 update files and how often nodes 10 request older versions of the file. In the case that the node 10 is requesting a version of the
file that is identical to the checkpoint file, the server can transmit the checkpoint file to the node 10. If the node 10 already has another version of the file, it may also be possible to transmit a difference file that directly maps the difference between the file currently at the node 10 and the file that the node 10 is requesting. If, however, the node 10 is requesting a 
version of the file that is not identical to the checkpoint file, the server's differencing module 20 may be operable to compute the requested version of the file by applying a difference file to the most appropriate checkpoint file. For example, if a node 10 requests from the server the tenth version of a file that is currently at its twentieth version, and checkpoint files are saved for every fifth version, the tenth version of a checkpoint file may be provided to the node 10 by the server. If the node 10 requests the eleventh version of the same file, the server may calculate the eleventh version by applying a difference file to the tenth version. The server may also, depending on the difference files that are stored, apply one or more difference file to the fifteenth version to obtain the tenth version. The node's synchronization module 22 may be operable to provide the requested version of the file to the node 10 through the network 14. Alternatively, if the node 10 currently has access to the twentieth version but would like to access the ninth version, the server may compute a difference file mapping the differences between the twentieth version and the ninth version and transmit this difference file to the node 10.

[0065] By transmitting the difference file rather than the entire version nine file, the transmission may occur much more quickly. Once the node 10 receives the difference file, the node's differencing module 20 may be operable to compute version nine of the file. By saving only a certain number of version files but saving enough difference files to enable the intermediate version files to be calculated by the server's differencing module, the required amount of memory on the data store 16 of the server may be reduced.

[0066] To expedite the process of providing files that are more than one version behind the most recent version, the differencing module may be operable to calculate relevant combinations of difference files in coordination with the checkpoint files. Considering an example where the server saves a checkpoint file for every ten versions of updates, and there are a total of fifty five versions of a particular file, then forty five difference files may be required to provide access to each of the versions in between the checkpoints. To conserve space in the data store, it may be assumed that most nodes 10 will have a version
that is no more than ten versions old and will want to update the version of the file to the most recent version. This reduces the number of required difference files to ten. In situations where the difference files are very large or space on the data store 16 may be limited, the difference files enabling a particular number of past versions to be updated to the most recent version can be stored in order to reply to any node's request for an update more rapidly.

[0067] In a further aspect, a node 10 may be operable to generate a synchronization list to request a plurality of files from the server. All files on the synchronization list may be updated when there is access to the network 14. In one example, where the node 10 is used by a user, the node 10 may update the files on the synchronization list when the user is away from the node 10, resulting in more bandwidth being available for synchronization processes. In order to synchronize each of the files in the synchronization list without transferring the entire file, some version of each file may need to be stored on the node 10. Thus, the node's synchronization module 22 requests that the server 16 provide it with corresponding difference files for each such file.

[0068] The node 10 may request an extensive list of files that are to be synchronized. To conserve memory on the node 10, the files on the synchronization list may be compressed by the node's compression module 24 and stored in a compressed format prior to transmission over the network 14. Similarly, the files transmitted by the server 16 to the node 10 may be compressed prior to transmission.

[0069] By performing differencing calculations prior to any synchronization process, the time required for synchronization can be reduced. This can be particularly advantageous if network access is only available for a limited number of hours in a day. By performing the differencing calculations before and after the data transfer over the network 14, the use of the network 14 may be maximized while it is available. Furthermore, as explained above, since only one file is being transferred, the transfer may be interrupted and resumed in an intermittently available network without significant loss. Since only some versions must be saved as checkpoint versions and the other versions can be calculated based on difference files by the differencing module 20, the amount of space required from the data store 16 may be significantly reduced.
[0070] By saving difference calculations, the difference file can be used for updating future files without requiring an extra computation step. This increases the efficiency of the synchronization system and reduces the load on both the server and the nodes 10. Moreover, since the server may distribute updates to each of the nodes 10 in the form of difference files as a more recent version of the file is created, the number of difference files that must be calculated may be reduced. Moreover, since difference files are typically smaller than reference files, there may be a lower probability of corruption during transmission.

[0071] Another advantage of the synchronization process of the current invention is that compression may be applied to a particular difference file, further reducing the quantity of data that must be transmitted over the network 14.

[0072] In another aspect, the file management system enables metadata tagging of files in the data store 16 of the server or locally on the node. Metadata tags may be stored in a database 17 of the node, as well as in database 17 of the server. By storing metadata tags on the database 17 of the node, the metadata tags may be used to perform searches during a temporary interruption of the network connection. The database 17 of the node may provide metadata to the search module 26 of the node. The search module 26 of the node provides search functionality to the nodes 10. In a relational database, each file on a server may be tagged by the node 10, the server, or a search module 26, based on the class of the file. Similarly, metadata searches may be performed by the search module on the server using metadata in the database 17 of the server.

[0073] Classes may be user-created or may be automatically created by the search module 26. Classes may exist for particular work sites, particular types of projects, particular employee types and/or particular file types, for example. The file may also be tagged based, for example, on the creator or editor of the file, the date that the file was created, the program used to create the file, the content in the file, the number of times that the file has been accessed, and particular information in the file. For example, in a mining operation, a certain file may be tagged as belonging to the class containing drill-hole data. Each of the files in this class may have a unique set of properties and the node's search module 26 may be operable to search for files based on their class.
[0074] Metadata tagging in accordance with the foregoing may optimally be applied in connection with data transmission over unreliable networks. For example, when a server’s synchronization module 22 provides an updated file to a node 10, the class and tagging information may also be provided to the node 10. This may ensure that class information as well as other metadata tags associated with the file are available to a node’s search module 26. The node 10 may save the class and tagging data. The server may also provide the tagging data of other files that have been tagged as being similar to the synchronized files. The server may also provide a larger subset of the tagging metadata available on the data store 16 or may provide all metadata associated with the tagging to the node 10.

[0075] Depending on the amount of metadata downloaded from the server, a user at a node 10 may apply metadata tags to search the entire body of files on the data store 16 or a subset of the files on the data store 16. For example, if the user is working at a node 10 that has downloaded the metadata tags for all the files on the data store 16, the user may search for all files of a specific class or all files tagged with particular information. For example, a user may wish to search information from all drill holes bored using tool steel bits in a particular area. A corresponding search may bring up all files in the class of drill holes bored using tool steel bits in the particular area. The user may then select to have particularly relevant files incorporated into the synchronization list to enable the user to view the file and maintain the file in its most recent version.

[0076] Further, files created by the node 10 can be added to the list of files that must be synchronized. Since no copy of the file may exist on the data store 16 of the server, the server’s synchronization module 22 may upload the file to the data store 16 during the synchronization process. If modifications are made to the file either at the user’s node 10 or at the node 10 of another user, the differencing module 20 may incorporate updates into the data store 16 if the node’s synchronization module 22 provides the differing file to the differencing module 20.

[0077] To assist with tagging of documents, a specific template with relevant metadata may be recommended for each type of file. The user may combine templates as well as add or remove new tags and classes to optimize the metadata such that the file can easily be found in future searches. Metadata may also comprise folder information that may be relevant to the contents of a particular file. For example, if an existing folder structure is
uploaded into the data store 16, the server may create metadata from the folder names or other information associated with the folders being uploaded.

[0078] In a further aspect, the file management system enables off-line synchronization. A node 10 may determine the synchronization status of each file on the node 10 or particular files on the node 10. In order to determine how much a file has been modified, the system may implement a file monitor 30. The file monitor is operable to determine the difference between modified files and the most recent version of the file downloaded to the node 10 from the server. Since, as explained above, the node 10 stores the most recent copy of the file downloaded from the server as a reference file, the node’s modification detection module 24 may compare the modified version of the file to the reference file. If the file has not been synchronized with the server for a pre-determined period of time or if the differences between the reference file and the modified file are greater than a certain threshold, the node’s modification detection module 24 may provide a warning to the user that the file should be synchronized when access to a network 14 becomes available.

[0079] The file management system may further prioritize the synchronization of files that are most different from the version that had been downloaded from the server. For example, if the node’s synchronization module 22 is set to synchronize two files, the file that has been most modified compared with the file last accessed from the server will be synchronized first. The user may also provide a manual priority ranking of which of the files on the node 10 should be synchronized first. The priority ranking may also be determined based on metadata tags or classes applied to the file. By synchronizing higher priority files first, it ensures that the most high priority files have been synchronized if there is an interruption in network access 14.

[0080] Since the files may be modified without the node’s knowledge, for example, if the file comprises information gleaned during a drilling process monitored by a sensor, the file may be updated in the background on the node 10. To notify the node 10 that the file must be synchronized to provide the information relating to the drill process to the server, the node’s modification detection module 24 may monitor for differences between the locally stored version of the most recent file during the last synchronization and the most recently updated file on the node 10. The node’s modification detection module 24 may be registered with the node’s operating system in order to capture file changes from a plurality
of programs and processes, similarly to a virus scanning program. All files that should be
synchronized with the server may then be synchronized once the network 14 becomes
available.

[0081] By coupling the caching of server files with the ability to search all files on the
server based on locally stored metadata, as well as the ability to monitor for file changes,
the file management system may be better suited for use with unreliable networks than past
systems. This allows minimal data transfer and ensures that the files that should be
synchronized are synchronized as soon as possible. Furthermore, by enabling a user to
search on a node 10 even when the server may be unavailable, the file management
system may retrieve files for the user without the user monitoring the system.

[0082] Referring to FIG. 5, an example of operation of the file manager is shown in one
example implementation, in which various types of data relating to the preparation,
construction and operation of a mine are shown. It shall be understood that this is but one
example, and that numerous other example implementations, and processes related to this
implementation, may be provided.

[0083] In this example, two types of data may be stored by the server. The first type of
data may be updated frequently, for example, blast-hole data 32. The second type of data
may be updated infrequently, for example, the data for the design of a mine. The design of
a mine 34 can potentially change on a weekly or monthly basis and must be well controlled,
as unintended changes to a design document may have significant consequences.

[0084] For example, within a working group there may be many versions of a given file
that are saved. Even for an individual user, there may be multiple iterations of a file in which
each version is stored. At some point the work is “complete enough” to share with a broader
audience, at which point the current version is “published”. This is a very important concept
in the mining industry in particular, as a lot is at stake when data is published, and it can
often only be done by people with specific certifications.

[0085] Older versions of a file may not be replaced; however, they may be accessed,
edited as saved as a new file or a new version. This ensures that the historical order of the
files can always be retrieved from the data store 16. When a user wishes to edit a
document, the user must check out the document to make edits in the master copy. Other
users may access the same document; however, changes made by these other users may
not be saved as a newer version of the document. These changes may be saved as a side branch of the document, as is shown in FIG. 4. Only the user who has checked out the document may save the master version of the document.

[0086] Although the above has been described with reference to certain specific example embodiments, various modifications thereof will be apparent to those skilled in the art without departing from the scope of the claims appended hereto.
Claims

1. A system for enabling data communication over a network, the system comprising a network-connected device comprising a differencing module operable to generate a difference file from a reference file and a modified file, wherein the differencing file may be sent to another device over the network to enable the other device to generate the modified file.
Application number / Numéro de demande: 2469773

Figures: 5

Pages:

Unscannable items received with this application
(Request original documents in File Prep. Section on the 10th floor)

Documents reçu avec cette demande ne pouvant être balayés
(Commander les documents originaux dans la section de la préparation
des dossiers au 10ième étage)
FIG. 3

Segment unmodified file

Number blocks

Compute hash

Segment modified file

Number blocks

Compute hash

Compare hashes

Determine modified blocks

Generate difference file

36

37

38

40

41

42

44

46

48