UPDATE DISTRIBUTION SYSTEM ARCHITECTURE AND METHOD FOR DISTRIBUTING SOFTWARE

FIELD OF THE INVENTION

The present invention relates to software and computer networks, and, in particular, the present invention relates to an update distribution system architecture and method for distributing software updates.

BACKGROUND OF THE INVENTION

Nearly all commercially available software products undergo a continual revision process to repair or update features of the software. Each revision of a software product frequently requires adding new files, replacing existing files with newer revisions, deleting obsolete files, or various combinations of these actions. This process of replacing older files, adding new files, deleting obsolete files of a software product, changing the software product's configuration and behavior by adding and/or removing functionality, will be referred to hereafter as "updating the product," and the data collection, including binary files, data files, update instructions, metadata, and the like, used in updating the product will be referred to hereafter more simply as an "update."

Once a software provider has created an update for a software product, either to fix a problem, enhance security, or add new features, the software provider will want to make that update widely available to its customer base. Quite often, such as when the update is directed at correcting a flaw in the product or addressing a critical security issue, the software provider will want that update installed on the customers' computers as soon as possible. Indeed, most software providers have a business incentive to distribute software updates to their customers as quickly and as trouble-free as possible.

The computer industry has experienced an explosive growth in the number of computers connected to networks, and in particular, to the Internet. Due to this explosive growth, and due to the communication abilities available through a connection to the Internet, the Internet has become an important and integral channel for software providers to distribute updates to their customers. In fact, the Internet has become the primary distribution channel for many software providers to provide software updates to their customers. It is often in the best interest of software providers to distribute software updates

over the Internet, as electronic update distribution over the Internet reduces their overall costs and enables customers to obtain the software updates as soon as they are available. More and more frequently, these software updates are conducted automatically over the Internet, without any user intervention.

While the Internet is now commonly used as a conduit for distributing software updates from software providers, several issues frequently arise. Two such issues include (1) efficiency relating to the update distribution infrastructure/resources, and (2) administrative control over the distribution and installation of software updates.

In regard to efficiency of the distribution resources, networks, including the Internet, possess only a finite amount of communication resources, often referred to as bandwidth. A finite amount of communication bandwidth frequently results in bottlenecks, especially in regard to software updates for popular software products, such as Microsoft Corporation's Windows® family of operating systems and related productivity products. Such bottlenecks exist even when software updates are made available on multiple download locations distributed throughout the Internet. One reason that such bottlenecks occur is the unstructured access model made available by the Internet. For example, if a first user at computer A requests the latest download of a software product, the download passes through the first user's independent service provider (ISP). Furthermore, the request is treated as a single, individualized access, meaning that the request is treated independent of, and unrelated to, any other network traffic and/or request. As such, if a second user at computer B, who also happens to have the same ISP, requests the same download as the first user, the request from the second user is also treated as a single, individualized access. In this example, the same download will be transmitted over the same infrastructure twice, because each request was treated in isolation. Clearly, if the number of users increases substantially, the finite communication bandwidth will become a bottleneck. In this example, which is quite common, it would have been much more efficient if the download could have been cached at a local location, and each user request satisfied from the local cache.

With regard to control of distribution, many organizations, especially large organizations, have legitimate reasons to control the distribution of updates to their computers. For example, unfortunately some updates have or introduce flaws, frequently

referred to as bugs, that "break" features of a software product. These broken features may be insignificant, but all too often they can disrupt a business's mission-critical features. Alternatively, mission-critical features may be "broken" if they are removed or their behaviors substantially altered by a particular software update. As a business cannot afford to lose its mission-critical features, a responsible business will first evaluate and test each software update within a controlled environment for some period of time prior to releasing the update to the remainder of their computers. This evaluation period permits the organization to validate whether an update will adversely affect a mission-critical feature. Only after it has been satisfactorily determined that an update will not bring down any mission critical feature is the update permitted to be distributed to the remainder of the organization's computers. Clearly, most organizations must exercise control over the installation of software updates on their computers.

Another reason that a business or an organization often needs to control distribution of software updates is to ensure consistency among the computers in the organization. It is very important for information service departments to have a standardized, target platform upon which all computers operate, whether it is for a word processor or an operating system. Without a standard, software and computer maintenance may be unnecessarily complex and difficult.

Still another reason that local control is important is for billing purposes. In large organizations, it is often inefficient to individually install software on a computer, or to individually maintain licenses for a particular software product for each computer in the organization. Instead, a single site license permits an organization to run a software product on numerous computers. Thus, an organization may be required to report the number of computers running a product under the site license, or may need to limit the number of computers running a product under a site license. All of these reasons often require local control over software update distribution.

In light of the various above-identified issues relating to software update distribution, what is needed is an extensible software update distribution architecture for providing control over the distribution of software updates, as well as increasing their distribution efficiency. The present invention addresses these and other issues found in the prior art.

SUMMARY OF THE INVENTION

In accordance with the present invention, a software update distribution system for distributing a software update over a communication network is presented. The software update distribution system includes a root update service node and at least one child update service node. According to aspects of the present invention, the root update service node and the at least one child update service node are organized in a hierarchical manner. The root update service node is the parent update service node of at least one child update service node. Each update service node, except the root update service node, has a parent update service node. Each child update service node may also be a parent update service node to at least one other update service node in the software update distribution system hierarchy. Additionally, the root update service node obtains a software update from a software provider. Each of the at least one child update service nodes obtains the software update from its parent update service node.

In accordance with additional aspects of the present invention, an update service node for distributing a software update to client computers and child update service nodes is presented. An update service node includes an update store for storing software updates. The update service node also includes a web service interface through which the update service node obtains software updates from its parent update service node, and through which the update service node distributes software updates to child update service nodes. The update service node further includes an administration application programming interface (API) through which an administrator, using an administration user interface, establishes rules for distributing software updates to child update service nodes. The update service node still further includes a child update module for determining which software updates are available to be distributed to the child update service nodes according to the established rules.

In accordance with further aspects of the present invention, a method for a child update service node in a software update distribution system to obtain a software update from its parent update service node is presented. The child update service node obtains a list of software products from its parent update service node. The list of software products identifies those software products for which the parent update service node distributes

software updates. The child update service node selects a software product for which software updates are sought. The child update service node then submits a request to the parent update service node identifying those software products for which updates are sought. In response, the child update service node receives an update list identifying software updates for the selected software products available to the child update service node. The child update service node then obtains each software update in the software update list from the parent update service node.

In accordance with yet further aspects of the present invention, a method for distributing software updates to a child update service node, executed on a parent update service node, is presented. A request is received from the child update service node requesting a software update catalog. A software update catalog, identifying software products for which the parent update service node distributes software updates, is returned to the child update service node. A request is then received for a list of software updates for a selected software product. In response, the parent update service node determines whether any software updates for the selected software product are available for the child update service node. An update list identifying those software updates for the selected software product determined to be available for the child update service node are returned to the child update service node. The update list identifies available software updates according to a unique update identifier. A update request is received requesting metadata corresponding to an available software update identified in the update list, the update request identifying the available software update by its unique identifier. In response, the update metadata is returned to the client update service node. The update metadata includes information corresponding to a software update including a reference for obtaining the corresponding update payload.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same become better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

FIGURE 1 is a pictorial diagram of an exemplary update distribution system formed in accordance with aspects of the present invention;

FIGURE 2 is a block diagram illustrating exemplary logical components of an update service node formed in accordance with aspects of the present invention;

FIGURE 3 is a block diagram illustrating exemplary logical components of a root update service node formed in accordance with aspects of the present invention;

FIGURE 4 is a block diagram illustrating an exemplary exchange between a parent update service node and a child update service node in providing a software update from the parent update service node to the child update service node in accordance with aspects of the present invention;

FIGURE 5 is a flow diagram illustrating an exemplary routine executed on a child update service node to periodically obtain updates from its parent update service node;

FIGURE 6 is a flow diagram of an exemplary subroutine suitable for use in the exemplary routine of FIGURE 5 for obtaining an update catalog from a parent update service node;

FIGURE 7 is a flow diagram of an exemplary subroutine suitable for use in the exemplary routine of FIGURE 5 for obtaining a software update from a parent update service node; and

FIGURE 8 is a flow diagram of an exemplary routine for processing an update request from a child update service node.

DETAILED DESCRIPTION

According to aspects of the present invention, an update distribution system, organized in a hierarchical fashion, for distributing software updates is presented. For purposes of the present invention, as already mentioned, a software update may involve replacing or removing older and/or obsolete files, adding additional files, altering a software product's configuration, and the like. Additionally, while not specifically directed to a software update of a software product in the technical and literal sense, aspects of the present invention may be advantageously utilized to distribute software products. Accordingly, while the following discussion is made in terms of distributing software updates in an update

distribution system, it is for illustration purposes only, and should not be construed as limiting upon the present invention.

FIGURE 1 is a pictorial diagram of an exemplary update distribution system 100 formed in accordance with aspects of the present invention. According to the present invention, at the "top" of an update distribution system, such as the illustrated update distribution system 100, is a root update service node 102. Software providers, such as software provider 110, distribute their software updates through the update distribution system 100 by submitting the updates to the root update service node 102. According to aspects of the present invention, software providers, such as software provider 110, may submit their software updates to the root update service node 102 through a network, such as the Internet 108.

A hierarchical update distribution system, such as the exemplary update distribution system 100, will likely include at least one other update service node in addition to the root update service node 102. As illustrated in FIGURE 1, the exemplary update distribution system 100 includes root update service node 102 and two additional update service nodes: update service node 104 and update service node 106. According to the present invention, each hierarchical update distribution system is organized in a tree-like structure underneath the root update service node 102. In other words, each update service node in an update distribution system has zero or more child update service nodes. Thus, while the exemplary update distribution system 100 shows that each parent update service node, i.e., the root update service node 102 and update service node 104, have only one child, it is for illustration purposes only, and should not be construed as limiting upon the present invention. Furthermore, with the exception of the root update service node 102, according to one embodiment of the present invention, each update service node in an update distribution system has one parent update service node. Accordingly, as shown in FIGURE 1, update service node 104 is a child node to the root update service node 102, and update service node 106 is a child node to update service node 104. In an alternative embodiment, a child update service node may receive updates from multiple parent update service nodes (not shown.) As can be seen, each update service node, with the exception of the root update service node 102, can be both a child update service node and a parent update service node.

As illustrated in the exemplary update distribution system 100, the root update service node 102 communicates with update service node 104 through the Internet 108. However, it should be understood that this is illustrative only, and should not be construed as limiting upon the present invention. Each update service node in an update distribution system need only be able to communicate with its parent and/or children through some communication network. Thus, while update service node 104 communicates with its parent, root update service node 102, through the Internet 108, it may alternatively communicate with its child update service nodes, such as update service node 106, via a local area network 124.

Also shown in FIGURE 1, update service node 106 resides within a sub-network 126 of the local area network 124. As an example, local area network 124 may correspond to an organization's general corporate network, and update service node 104 represents the corporation's link to the update distribution system 100, via its connection to its parent, root update service node 102. Further, sub-network 126 may correspond to an identifiable group of computers within the corporate network, such as a test/evaluation group, a remotely located office, or a mission critical group. As will be described in greater detail below, according to aspects of the present invention, an administrator on update service node 104 is able to control the distribution of updates to update service node 106, and ultimately to client computers.

It should be appreciated that each update service node, including both the root update service node 102 and update service nodes 104 and 106, is configured to distribute software updates to both child update service nodes as well as client computers. As shown in FIGURE 1, the exemplary update distribution system 100 includes client computers 112-122. Each update service node, including the root update service node 102, distributes updates to child update service nodes and client computers according to local configuration information. According to one embodiment, an administrator defines groups and associates update distribution rules with those groups. Each update service node has at least one distribution group.

As an example to illustrate how the update distribution system operates, assume that local area network 124 corresponds to a business organization's corporate network.

According to one embodiment of the present invention, an administrator, on update service node 104, may define multiple distribution groups for the corporate network 124, including an evaluation group, corresponding to the sub-network 126 including update service node 106 and client computers 120 and 122, for evaluating the suitability of an update for the general corporate network 124, as well as a general corporate group including the update service node 104 and client computers 114-118.

With regard to the evaluation group, the administrator includes the update service node 106 as a member, and associates rules with that group such that updates are immediately distributed to the evaluation group's members as they become available. Alternatively, with regard to the general corporate group, the administrator adds client computers 114-118, and associates a rule such that updates are only distributed to the general corporate group members if specifically authorized by the administrator. Assume also that an administrator for child update service node 106 creates a default group consisting of the client computers 120 and 122 in the evaluation sub-network 126, to which any new software update may be immediately distributed.

Continuing the above example, a software provider 110 submits a software update to the root update service node 102. According to rules established at the root update service node 102, the update is eventually distributed to the corporate update service node 104. Upon receiving the update, per the rules established by the administrator, the corporate update service node 104 distributes the update to the members of the evaluation group (defined as only the child update service node 106), but withholds the update from the general corporate group pending specific authorization to distribute the update to that group.

Continuing the above example, upon receiving the update, the evaluation update service node 106 processes the update with respect to each defined group. In this example, the evaluation update service node 106 has only one group. However, as previously mentioned, in an actual implementation, there may be multiple groups defined, each with a unique set of associated distribution rules. For this example, the evaluation update service node 106 immediately makes the update available for distribution to client computers 120 and 122. Client computers 120 and 122 may now be updated and the evaluation period/process may begin.

Still continuing the above example, when the administrator on the corporate update service node 104 is sufficiently satisfied that the update is suitable for distribution over the entire corporate network 124, the administrator then explicitly authorizes the update to be distributed to the members of the general corporate group. The corporate update service node 104 correspondingly makes the update available to client computers 114-118. It should be understood that the evaluation update service node 106 may also be included in the general corporate group. However, because the evaluation update service node 106 has already been updated, no additional update-related action is needed for distributing the update to the evaluation sub-network 126.

As can be seen by the above example, the present invention offers significant benefits in terms of local distribution control and download efficiency. In addition to the above-described aspects of local distribution control, significant savings in communication bandwidth are also realized. For example, while the exemplary corporate network 124 illustrated in FIGURE 1 includes five client computers, the software provider's update was downloaded from the root update service node 102 to the corporate update service node 104 only one time. Clearly then, as the number of client computers serviced by an update service node increases, the communication bandwidth usage between a parent update service node and a client update service node remains constant, thereby substantially reducing the amount of communication bandwidth that would otherwise be used. Additionally, the update distribution system is both extensible and scalable. The update distribution system is extensible in at least two ways: any number of child update service nodes may be added to a parent update service node, and child update service nodes may also be a parent update service node. Each sub-tree of the update distribution system may therefore be tailored to meet individual needs.

Yet another benefit realized by the present invention is the ease in which an administrator establishes and enforces a consistent software platform over the client computers and network for which the administrator is responsible. For example, the current process that an administrator ensures a consistent software platform over client computers is generally the following: make a determination as to what the software platform is to be, i.e., which updates should be installed and which should not; determine the status of the software

platforms on each client computer; and push the appropriate software updates down to the client computers for installation. Even after the initial effort, an administrator must frequently repeat the process in order to ensure that the software platform remains consistent. In contrast to these prior art approaches, by utilizing the update distribution system described above, as well as the infrastructure support for grouping of clients machines into different target groups and the ability to target different updates at different target groups (described below), an administrator can specify the desired state for software update installation of the networks and or client computers for which the administrator is responsible. An administrator specifies the desired state through an Administration API (also described below) for any number of client computers and/or groups of client computers. Once the desire update state is established/specified, the administrator can then rely on the update distribution system to enforce the distribution of the updates throughout the update distribution system. The update distribution system works to make sure that the clients in the system conform to the desired state dictated by the administrator. Through periodic actions, the update distribution system constantly tries to bring those client computers that do not conform with the desired state into conformance. Additionally, by associating specific installation attributes with the software updates, such as a mandatory installation attribute or a install-by deadline installation attribute, the update distribution system relieves the administrator of a substantial portion of the implementation details associated with maintaining a consistent software platform. A more detailed description of associating installation attributes with a software update to control installation behaviors on a client computing device is set forth in copending and commonly assigned U.S. Patent Application entitled "Controlling Installation Update Behaviors On a Client Computer," filed March 12, 2004, attorney docket no. MSFT-1-22459, which is incorporated herein by reference.

FIGURE 2 is a block diagram illustrating exemplary logical components of an update service node 200, such as the corporate update service node 104 (FIGURE 1) or the evaluation update service node 106 (FIGURE 1), formed in accordance with aspects of the present invention. As shown in FIGURE 2, an update service node 200 includes an update web service 202, a client update module 204, a child update module 206, and a reporting module 208. The exemplary update service node 200 also includes an authentication/

authorization module 210, an administration application programming interface (API) 212, an update content store 214, an administration user interface 218, and an update information store 216.

The update web service 202 provides a common set of Web services through which client computers, child update service nodes, and a parent update service node can communicate with an update service node. For example, with reference to FIGURE 1, in order for the child/evaluation update service node 106 to obtain a software update from the parent/corporate update service node 104, the client communicates through the parent's update web service 202. Similarly, when a parent update service node, such as root update service node 102, has information, including updates, to communicate to its child update service node 104, the parent update service node communicates through the child's update web service 202.

The client update module 204 handles communications between a client computer and the update service node 200 in regard to updates and update information stored on the update service node. The update-related communications include, but are not limited to, distributing updates in response to client requests and providing a list of available software products and associated updates for the client computer. The client update module 204 is also responsible for determining whether a client computer is authorized to obtain a particular update according to associated distribution rules, and responds to a client computer with the update-related information that the client computer is authorized to access.

The child update module 206 handles update-related communications between a parent update service node and its child update service nodes. The update-related communications include, but are not limited to, identifying lists of software products and associated updates available to a child update service node, as well as responding to update requests from a child update service node. The downstream update module 206 is responsible for determining whether a child update service node is authorized to obtain a particular update according to associated distribution rules, and responds to a child update service node with the update-related information that the child update service node is authorized to access.

The reporting module 208 generates update-related reports, such as which groups have or have not received a particular update, which client computers have or have not downloaded/installed an update, what updates are available on the update service node, and the like. These reports may be used internally, such as by an administrator, and also submitted to the parent update service node, via the parent's update service interface 202. As described above, it is often necessary for corporations to determine which client computers have a particular update installed, such as for billing purposes or for maintenance purposes. Information/reports generated by the reporting module 208 may be the basis of these reports.

The authentication/authorization module 210 is responsible for authenticating, i.e., determining the identity of, a particular client computer or child update service node, and determining whether a client computer or child update service node is authorized to access available updates at the update service node 200. To those client computers and child update service nodes that are authenticated and authorized to access updates on an update service node, the authentication/authorization module 210 issues an authorization token that must be used in conjunction with obtaining updates. The issuance and use of an authorization token is described in greater detail below in regard to FIGURES 4A and 4B.

The administration API 212 represents the application interface through which control of the update service node 200 is exercised, and through which updates ultimately are stored and distributed. When the update web service 202 receives various update-related requests from client computers and child update service nodes, these requests are ultimately broken into calls into the administration API 212, either directly or indirectly through the client update module 204 and the child update module 206. In conjunction with the administration user interface 218 or some other program installed on the update service node 200 suitably configured to use the administration API 212, an administrator ultimately controls all aspects of the update process for that update service node, as well as any child update service nodes and client computers.

Through the administration user interface 218, administrators may configure and maintain an update service node 200, via the administration API 212. Thus, through the administration user interface 218, an administrator creates, modifies, and deletes groups, as well as associating rules for each group. Furthermore, using the administration user

interface 218, an administrator establishes to which group a client computer or child update service node belongs. Through the administration user interface 218, an administrator may also explicitly authorize the distribution of updates to client computers or child update service nodes, configure the update service node 200 to periodically query its parent update service node for new updates, configure reporting parameters and view internal reports, and the like. As mentioned above, while the administration user interface 218 permits an administrator to exercise control over aspects of the update service node 200, another application residing on the update service node 200, suitably adapted to operate with the administration API 212, may be used instead of the administration user interface 218.

As mentioned above, according to one embodiment of the present invention, an update service node 200 includes both an update content store 214 and an update information store 216. The update content store 214 stores the actual files representing the software updates, such as binaries and patch files. In contrast, the update information store 216 stores information and metadata corresponding to the updates available on the update service node 200, including the update files stored in the update content store 214. According to one embodiment, the update content store 214 and the update information store 216 are both relational databases. While the exemplary update service node 200 is shown as having two data stores, the present invention should not be so limited. In an alternative embodiment, both the update content store 214 and the update information store 216 may be combined in a single information store.

In accordance with aspects of the present invention, a software update may be presented as being "available" on an update service node 200 to client computers and child update service nodes even though the update is not stored physically in the update content store 214. More particularly, rather than immediately downloading and storing the actual update files on an update service node 200, a link referencing the update files on the parent update service node or elsewhere, may instead be stored on the update service node. Thus, if a client computer requests the update, or a child update service node requests the actual update, the update is then brought down from the parent update service node and stored in the update content store 214, in preparation for delivering it to the client computer or child update service node. Those skilled in the art will recognize this type of update access is

referred to as just-in-time downloading. In this manner, an "available" update, need not be distributed over the various network channels until it is actually requested. According to aspects of the present invention, an administrator of an update service node 200 may selectively determine whether to obtain software updates in a just-in-time manner.

While the above description of FIGURE 2 illustrates various components of an exemplary update service module 200, it should be appreciated that other components of an update service module may also exist. Furthermore, the above described components should be understood to be logical components, not necessarily actual components. In an actual implementation, the above identified components may be combined together and/or with other components according to implementation determinations. Additionally, it should be appreciated that while an update service node 200 may be viewed as a server computer on a network, in an actual implementation, an update service node may be implemented on any number of types of computing devices. For example, each update service node 200 may be implemented and/or installed on a single stand-alone computer system or, alternatively, on a distributed computing system comprising multiple computing devices.

FIGURE 3 is a block diagram illustrating exemplary logical components of a root update service node 300, such as the root update service node 102 illustrated in FIGURE 1, formed in accordance with aspects of the present invention. Similar to the logical components of an update service node 200 (FIGURE 2), a root update service node 300 includes an update web service 202, a child update module 206, and an authentication/authorization module 210. Additionally, an exemplary root update service node 300 also includes an administration API 212, an update content store 214, and an update information store 216. Optionally, the root update service node 300 may also include a client update module 204, a reporting module 208, and an administration user interface 218.

The client update module 204 is an optional component for a root update service node 300 depending on whether the root update service node provides software updates directly to client computers. For example, with reference to FIGURE 1, root update service node 102 would include the optional client update module 204 as the root update service

node that directly services client computer 112. However, if a root update service node 300 were not to directly service client computers, the client update module 204 could be omitted.

The reporting module 208 is optional for a root update service node 300 because a root update service node has no parent update service node to whom update reports are provided. However, to the extent that update reports are desirable to the root update service node's administrator, the reporting module 208 may be optionally included.

In addition to comprising the logical components included in an update service node 200 (FIGURE 2), the root update service node 300 also includes a software provider interface 302. The software provider interface 302 provides the communication interface by which a software provider 110 (FIGURE 1) submits software updates directly to the root update service node 300, and indirectly to the exemplary update distribution system 100.

Similar to the update service node 200 of FIGURE 2, the above description of FIGURE 3 illustrates various components of an exemplary root update service module 300. However, it should be appreciated that other components of a root update service module may also exist. Furthermore, the above described components should be understood to be logical components, not necessarily actual components. In an actual implementation, the above identified components may be combined together and/or with other components according to implementation determinations. Additionally, it should be appreciated that while a root update service node 200 may be viewed as a server computer on a network, in an actual implementation, an update service node may be implemented on any number of computing devices. For example, the root update service node 300 may be implemented and/or installed on a single stand-alone computer system or, alternatively, on a distributed computing system comprising multiple computing devices.

In order to better understand how an update is distributed from the root update service node throughout an update distribution system 100, an illustration of an exemplary exchange between a parent update service node and a child update service node is warranted. FIGURE 4 is a block diagram illustrating an exemplary exchange 400 between a parent update service node 402 and a child update service node 404 in propagating a software update from the parent update service node to the child update service node, in accordance with aspects of the present invention. As can be seen, the exemplary diagram 400 is divided

in half, the left half of which corresponds to actions and events of the parent update service node 402, and the right half corresponding to actions and events of the child update service node 404.

For purposes of discussion with regard to FIGURE 4, it should be further understood that the parent update service node 402 may or may not be the root update service node in the update distribution system 100. Additionally, for purposes of this discussion, it is assumed that the parent update service node 402 has been configured by an administrator such that the child update service node 404 may not receive software updates unless explicitly authorized to do so by the administrator.

As shown in the exemplary exchange 400, beginning at event 406, the parent update service node 402 receives a software update from a software provider 110, either directly, if the parent update service node is the root update service node 102, or indirectly through the update distribution system 100. At some point after the parent update service node 402 receives the software update from the software provider 110, the child update service node 404 begins a process for obtaining software updates from the parent update service node.

According to one embodiment, a child update service node 404 can be configured to automatically obtain the software updates available from a parent update service node 202 on a periodic basis. More particularly, an administrator, via the administration user interface 218, may selectively configure the child update service node 404 to automatically obtain the latest software updates available on the parent update service node 402 on a periodic basis. As one example, an administrator may configure the child update service node 404 to obtain the latest software updates from its parent update service node 402 on a daily and/or hourly basis, as well as specify the time-of-day that the automatic update process is to commence. Other periodic schedules and criteria may also be utilized. Similarly, an administrator may manually initiate the update process through the administration user interface 218.

To begin the updating process, at event 408 the child update service node 404 authenticates and authorizes itself with the parent update service node 402. Authenticating and authorizing with the parent update service node 402 provides an element of control over

the distribution of software updates, limiting update distribution to authorized update service nodes. Authenticating and authorizing techniques are well known in the art, any number of which may be employed to authenticate and authorize a child update service node 404 with the parent update service node 402. The present invention is not restricted to any one technique.

After properly authenticating and authorizing with the parent update service node 402, at event 410 the parent update service node 402 returns an authorization token to the child update service node 404. According to one embodiment, an authorization token is a time sensitive token providing the child update service node 404 authorization to conduct further update activities with the parent update service node for a limited amount of time. Thus, if the child update service node 404 is not properly authenticated and authorized with the parent update service node, no authorization token is returned and the child update service node is unable to perform any other update-related activities except authentication and authorization. Similarly, after the update token has expired, the child update service node 404 is unable to perform any further update-related activities with the parent update service node 402 except reauthentication and reauthorization.

After receiving the authorization token, at event 412 the child update service node 404 submits a request to the parent update service node for a product update catalog along with the authorization token. A product update catalog represents a listing, or table of contents, of software products for which the parent update service node 402 distributes software updates.

According to aspects of the present invention, a child update service node 404 is not required to propagate all software updates available on its parent update service node 402. For example, with reference to the exemplary update distribution system of FIGURE 1, the corporate update service node 104 may have site licenses to only a fraction of software products available on the root update service node 102. Accordingly, it would be unnecessary for the corporate update service node 104 to obtain all software updates available at the root update service node 102, as most would never be used. Accordingly, an administrator on an update service node may selectively establish which software product updates will be available on the update service node. In addition to selecting a software

product, an administrator may also selectively identify additional, more specific, aspects relating to the software product in order to minimize the amount of network bandwidth consumed in the update process. For example, in addition to selecting a software product, an administrator may selectively specify that only English versions of software updates for the selected software product are requested. Similarly, an administrator may selectively specify that only software updates having a specific attribute are download. As an example of these specific attributes, an administrator may request, for the selected software product, only security-related software updates, or those that the software provider 110 considers to be critical. In sum, according to aspects of the present invention, an administrator may request a list of software updates for a selected software product according to any identifiable attribute or aspect of a software update.

According to one aspect of the present invention, the update product catalog, obtained from a parent update service node 402, identifies all software products for which updates are available, whether or not the child update service node 404 is configured to distribute updates for each product. However, according to an alternative aspect of the present invention, the update product catalog, obtained from a parent update service node 402, identifies only those software products for which the requesting child update service node is configured to distribute updates. For example, limiting which software products are listed in the product update catalog may be determined according to the group or groups to which the child update service node 404 belongs.

At event 414, the parent update service node 402 returns a product update catalog to the child update service node 404. At event 416, the child update service node 404 selects those products from the product update catalog for which the latest updates are currently desired. It should be noted that even though the product update catalog may list only those software products that the child update service node 404 distributes, the child update service node may be configured to obtain updates for different software products at different times or on different periodic schedules.

At event 418, the child update service node 404 submits an update synchronization request, along with the authorization token, identifying the selected products for whose updates the child update service node is currently seeking. Included in the synchronization

request is information identifying the latest update available for a product on the child update service node 404. Information identifying the latest update for a product is hereafter referred to as an "update anchor." Update anchors for each software product are typically stored in the update information store 216 (FIGURE 2). In one embodiment, an update anchor includes a revision number and a date associated with the revision number.

In response to the update synchronization request, at event 420 the parent update service node 402 determines which, if any, new updates are available for the child update service node 404. As mentioned above, this determination is based on the specific rules associated with particular software updates and the group or groups of which a child update service node 404 is a member, as well as the update anchor. For this example, as previously mentioned, the previously received software update was explicitly not authorized for the child update service node 404. Therefore, the software update received at event 406 is not determined to be "available" to the child update service node 404. Accordingly, at event 422 an update list is returned to the child update service node 404 without identifying the software update received at event 406. According to aspects of the present invention, the update list identifies all of the updates "available" on the parent update service node 402 according to the synchronization request. In one embodiment, the update list identifies each "available" update information by a unique identifier associated with an update.

At event 424, because the update list is empty, i.e., no updates are currently "available" on the parent update service node 402, the update process of the child update service node 404 simply delays, or sleeps, for a predetermined amount of time. According to the current example, during this delay period, at event 426, an administrator at the parent update service node 402 authorizes the software update, received at event 406, to be distributed to the child update service node 404.

At event 428 (FIGURE 4B), the child update service node 404 again begins the automatic update process by authenticating and authorizing itself with the parent update service node 402. In response, at event 430, the parent update service node 402 returns an authorization token to the child update service node 404.

At event 432, the child update service node 404 submits a request, along with the authorization token, to the parent update service node 402 for a product update catalog. At

event 434, the parent update service node 402 returns the product update catalog to the child update service node 404. At event 436, the child update service node 404 selects the products for the update catalog for which updates are desired. At event 438, the child update service node 404 submits the update synchronization request identifying those selected products with the authorization token.

Because the child update service node 404 has been authorized to obtain the software update previously received at event 406, at event 440 the parent update service node 402 determines that the software update is "available" for the child update service node and includes corresponding update information in the update list. Thereafter, at event 442, the parent update service node 402 returns the update list, now identifying the software update received at event 406, to the child update service node 404.

With an update list identifying an "available" update on the parent update service node 402, the child update service node 404 now has the information necessary to obtain the software update. According to one embodiment of the present invention, a child update service node 404 obtains the software update from the parent update service node 402 in two parts: obtaining update metadata, and obtaining the update content or file, hereafter referred to as the update payload. According to additional aspects of the present invention, the update metadata describes pertinent aspects of the software update, including, but not limited to: an update identifier that uniquely identifies the update, revision number information associated with the software update, whether the software update should be considered a priority, language specific information, relationships to other software updates, location of the update payload for downloading purposes, installation handler routines, and the like.

Some of the reasons that it is often beneficial to download the entire software update in two parts, i.e., the update metadata and the update payload, is that the update payload is often substantially larger than the update metadata, and the update payload is not always immediately needed, i.e., needed for installation on a client computer, if it is ever needed. Thus, according to one embodiment of the present invention, the update payload is downloaded separately from the update metadata, and only when needed. Those skilled in the art will recognize this downloading technique as lazy downloading, or alternatively as just-in-time downloading. According to aspects of the present invention, an administrator

may configure an update service node to obtain the update payload in a just-in-time fashion, or immediately upon obtaining the update metadata. Furthermore, in an alternative embodiment, both update metadata and the update payload may be downloaded jointly.

As shown in FIGURE 4B, with an update identified in the update list, at event 444, the child update service node 404 requests the update metadata for the "available" software update according to its unique identifier in the update list. As with most other communication exchanges with the parent update service node 402, the update request is submitted with the authorization token. It should be noted that while in the illustrated example, all update metadata is downloaded in one access, according to alternative aspects of the present invention (not shown), the update metadata may be downloaded in more than one access. For example, in a first access, only elements of the update metadata to are necessary to determine whether a software update is applicable and/or desirable is first downloaded, such as applicability rules and dependencies upon other software updates. Then, after it is determined that an update is applicable and/or desirable, the remainder of the update metadata may be obtained. In response, at event 446 the parent update service node 402 returns the update metadata for the software update child update service node 404, which in turn stores the update metadata into the update information store 216.

Optionally, at event 448, the child update service node 404 submits a request to download the update payload from the parent update service node 402. In response, at event 450, the parent update service node 402 returns the update payload to the child update service node 404, which in turn stores it in the update content store 214. In an alternative embodiment, the child update service node 404 downloads the update payload from a storage location, which may not be the parent update service node 402, from a location specified in the update metadata.

Because update activity has now occurred on the child update service node 404, at event 452, the child update service node generates and submits an update report to the parent update service node 402 outlining the update activities that have just recently occurred. Thereafter, the child update service node 404 again delays until the next time that the update process is scheduled to run (not shown).

Those skilled in the art will appreciate that the above described events are for illustration purposes, and reflect one particular exemplary set of events and circumstances. Clearly, other events may also occur according to specific details and circumstances which will cause some variation to the above described events. Additionally, it should be understood that while the child update service node 404 is obtaining the latest "available" software updates from the parent update service node 402, the child update service node may simultaneously be processing update requests from its child update service nodes. Accordingly, the above sequence of events should be viewed as illustrative only, and not limiting upon the present invention.

FIGURE 5 is a flow diagram illustrating an exemplary routine 500 executed on a child update service node, such as the corporate update service node 104 of FIGURE 1, for periodically obtaining updates from its parent update service node. Beginning at block 502, the child update service node obtains a synchronized update list of "available" updates from the parent update service node. Obtaining a synchronized update list of "available" updates from the parent update service node is described below with regard to FIGURE 6.

FIGURE 6 is a flow diagram of an exemplary subroutine 600, suitable for use in the exemplary routine 500 of FIGURE 5, for obtaining a synchronized update list of "available" updates from a parent update service node. Beginning at block 602, as previously discussed with regard to FIGURES 4A and 4B, the child update service node authenticates and authorizes itself with the parent update service node and, in response to proper authentication and authorization, receives an authorization token. At block 604, in conjunction with the authorization token, the child update service node establishes communication parameters with the parent update service node. Establishing communication parameters permits the parent and child update service nodes to properly establish a common basis that both the parent and child understand. The communication parameters include, but are not limited to: communication update protocols or versions; product groupings; and the like.

After having established communication parameters with the parent update service node, at block 606, the child update service node obtains a product update catalog describing software products for which the parent update service node provides/distributes updates. At block 608, the child update service node selects those software product updates for which

updates are currently sought. At block 610, the child update service node submits an update synchronization request to the parent update service node, including both the authorization token and an "anchor" associated with the selected software products identifying the current revision and updates already on the child update service node.

In response to the update synchronization request, at block 612, the child update service node obtains an update list from the parent update service node, synchronized according to the software updates "available" on the parent update service node according to what is currently stored on the child update service node. As mentioned above, the update list identifies, by a unique identifier, those software updates on the parent update service node that are "available" to the child update service node. Thereafter, the exemplary subroutine 600 terminates.

With reference again to FIGURE 5, after having obtained a synchronized update list from the parent update service node, at decision block 504, a determination is made as to whether any software updates are currently "available" for downloading from the parent update service node. This determination is made according to whether there are any update identifiers listed in the synchronized update list. If no software updates are currently "available" for downloading, the exemplary routine 500 proceeds to delay block 510, where the exemplary routine delays/sleeps until the next update period occurs. Alternatively, if there are updates "available" for downloading from the parent update service node, at block 506, the child update service node obtains the updates from the parent update service node. Obtaining "available" updates from the parent update service node is described below with regard to FIGURE 7.

FIGURE 7 is a flow diagram of an exemplary subroutine 700, suitable for use in the exemplary routine 500 of FIGURE 5, for obtaining "available" software updates from a parent update service node. Beginning at block 702, a first update identifier in the update list is selected. At block 704, the child update service node obtains the update metadata corresponding to the selected update identifier from the parent update service node and stores it in the update information store 216.

According to one embodiment, at block 706, the child update service node obtains the update payload corresponding to the selected update identifier from the parent update service

node, and stores the update payload in the update content store 212. Optionally, the update content need not be immediately downloaded to the child update service node. As previously mentioned, a child update service node may be selectively configured to download updates from a parent update service node in a just-in-time fashion. According to this optional treatment, as illustrated in FIGURE 7, rather than proceeding from block 704 to block 706, the exemplary subroutine 700 optionally proceeds from block 704 to decision block 708.

At decision block 708, after having obtained the update metadata for the selected update identifier, and optionally the update payload, a determination is made as to whether there are any additional update identifiers in the update list. If there are additional update identifiers, at block 710, the next update identifier in the update list is selected, and the subroutine 700 returns to block 704 for additional processing. The routine 700 continues until, at decision block 708, it is determined that there are no more update identifiers in the update list, whereupon the exemplary subroutine 700 terminates.

Returning again to FIGURE 5, after having obtained the "available" updates from the parent update service node, at block 508, the child update service node reports the update activities to the parent update service node. Thereafter, at delay block 510, the exemplary routine 500 delays/sleeps for a predetermined amount of time until the next update period, and then proceeds to block 502 to repeat the above-identified update procedures.

As illustrated in FIGURE 5, at decision block 504, even when no updates are "available" on a parent update service node, a child update service node may be optionally configured to report its update activities to the parent update service node. According to this alternative configuration, when there are no updates available, the exemplary routine 500 may proceed to block 508 to report the update activities.

FIGURE 8 is a flow diagram of an exemplary routine 800, implemented on a parent update service node, for generating a synchronized update list identifying "available" updates in response to an update synchronization request from a child update service node. Beginning at block 802, the parent update service node receives an update synchronization request from a child update service node for an update list identifying "available" updates.

At block 804, the first software product identified in the update synchronization request is selected.

At decision block 806, a determination is made as to whether there are any available updates for the identified software product. This determination is made according to metadata for the software product stored in the update information store 216, according to the update anchor provided by the child update service node, and according to distribution rules associated with the group to which the child update service node belongs. According to this determination, if there are updates "available," at block 808, unique update identifiers associated with the "available" updates are written into an update list. After having written unique update identifiers for "available" updates into the update list, at decision block 810, a determination is made as to whether there are any more additional software products identified in the update synchronization request. If there are additional update software products in the update synchronization request, at block 814, the parent update service node selects the next software product identified in the update synchronization request, and returns to decision block 806 for determining whether there are "available" updates for the selected software product. Alternatively, if there are not more software products identified in the update synchronization request, at block 814, the update list is returned to the child update service node. Thereafter, the exemplary subroutine 800 terminates.

While various embodiments of the present the invention have been illustrated and described, including the preferred embodiment, it will be appreciated that various changes can be made therein without departing from the spirit and scope of the invention.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. A software update distribution system for distributing a software update over a communication network for distribution to client computers, comprising:

a root update service node; and

a plurality of child update service nodes;

wherein the root update service node and the at least one child update service node are organized in a hierarchical manner such that the root update service node is a parent update service node to at least one child update service node, wherein each update service node, except the root update service node, has a parent update service node, wherein each of the plurality of child update service nodes is configured to operate as a parent update service node to another child update service node, and wherein at least one child update service node of the plurality of child update service nodes is a parent update service node to another child update service node of the plurality of child update service nodes; and

wherein the root update service node obtains a software update from a software provider, and wherein each of the at least one child update service nodes obtains the software update for distribution to client computers by obtaining the software update from its parent software update node.

2. The software update distribution system of Claim 1, wherein the root update service node comprises:

an update store for storing software updates;

an update web service through which the root update service node distributes software updates to its child update service nodes over the communication network; and

a software provider interface through which a software provider submits its software update over the communication network to the root update distribution node.

3. The software update distribution system of Claim 2, wherein each of the plurality of child update service nodes comprises:

an update store for storing software updates;

an update web service through which the child update service node obtains software updates from its parent update service node over the communication network, and through which the child update service node distributes software updates to its child update service nodes over the communication network;

an administration application programming interface (API) through which an administrator, using an administration user interface, establishes rules for distributing software updates to its child update service nodes; and

a child update module for determining which software updates are available to be distributed to its child update service nodes according to the established rules.

4. The software update distribution system of Claim 3, wherein each of the at least one child update service nodes further comprises:

a reporting module for generating and sending update activity reports to the parent update service node;

an authentication and authorization module for determining whether an update service node is authorized to obtain software updates from the child update service node; and

a client update module for distributing software updates to client computers.

5. The software update distribution system of Claim 4, wherein the update store comprises an update content store in which the update payload for the software update is stored, and an update information store in which update metadata for the software update is stored.

6. The software update distribution system of Claim 5, wherein the child update service node obtains the software update from the parent update service by obtaining update metadata for the software update from the parent update service node, and separately obtaining the update payload for the software update from the parent update service node.

7. The software update distribution system of Claim 6, wherein the child update service node obtains the update payload for the software update from the parent update service node in a just-in-time fashion.

8. The software update distribution system of Claim 4, wherein the client update module distributes software updates to client computers according to rules established by an administrator via the administration API using the administration user interface.

9. The software update distribution system of Claim 8, wherein the root update service node further comprises a client update module for distributing software updates to client computers.

10. The software update distribution system of Claim 3, wherein the child update service node may be selectively configured to periodically obtain available software updates from the parent update service node.

11. An update service node for distributing software updates to client computers and to child update service nodes, wherein the update service node is organized in a hierarchy of a plurality of similarly configured update service nodes, the update service node comprising:

an update store for storing software updates;

an update web service through which the update service node obtains software updates from a parent update service node over a communication network, and through which the update service node distributes software updates to child update service nodes in the hierarchy over the communication network;

an administration application programming interface (API) through which an administrator, using an administration user interface, establishes rules for distributing software updates to the child update service nodes; and

a child update module for determining which software updates are available to be distributed to the child update service nodes according to the established rules.

12. The update service node of Claim 11, further comprising:

a reporting module for generating update activity reports to the parent update service node;

an authentication and authorization module for determining whether a child update service node is authorized to obtain software updates from the update service node; and

a client update module for distributing software updates to client computers.

13. The update service node of Claim 12, wherein the update store comprises an update content store in which the update payload for the software update is stored, and an update information store in which update metadata for the software update is stored.

14. The update service node of Claim 13, wherein the update service node obtains the software update from the parent update service by obtaining update metadata for the software update from the parent update service node, and separately obtaining the update payload for the software update from the parent update service node.

15. The update service node of Claim 14, wherein the update service node obtains the update payload for the software update from the parent update service node in a just-in-time fashion.

16. The update service node of Claim 12, wherein the update module distributes software updates to client computers according to rules established by an administrator via the administration API using the administration user interface.

17. The update service node of Claim 11, wherein the child update service node may be selectively configured to periodically obtain available software updates from the parent update service node.

18. A method for distributing software updates to a first child update service node from a second child update service node in a software update distribution system over a communication network, wherein the first and second child update services nodes are organized in a hierarchy of update service nodes originating with a root update service node, and wherein the second child update service node is a parent update service node to the first child update service node, the method comprising as executed on the second child update service node:

(a) receiving a request for a software update catalog from the first child update service node, the software update catalog identifying software products for which the second child update service node provides software updates;

(b) returning a software update catalog to the child update service node;

(c) receiving a request for a list of software updates for a selected software product available for the child update service node;

(d) determining whether any software updates for the selected software product are available for the child update service node;

(e) returning a software update list identifying those software updates for the selected software product determined to be available for the child update service node to the first child update service node, the software update list identifies available software updates according to a unique update identifier;

(f) receiving an update request for metadata corresponding to an available software update from the first child update service node, the update request identifying the available software update by its unique update identifier; and

(g) returning update metadata corresponding to the software update identified in the update request, the update metadata including information corresponding to the software update including a reference for obtaining the corresponding update payload.

19. The method of Claim 18 further comprising:

prior to steps (a)-(g):

receiving an authentication and authorization request from the first child update service node;

determining whether the first child update service node is authorized to obtain software updates from the second child update service node; and

if the first child update service node is authorized to obtain software updates from the second child update service node, returning an authorization token to the first child update service node; and

wherein each subsequent communication from the first child update service node to the second child update service node is made with the authorization token.

20. The method of Claim 19 further comprising:

receiving an update request for the update payload corresponding to the software update identified in the update request; and

returning the update payload corresponding to the software update identified in the update request.

21. The method of Claim 18, wherein the request for a list of software updates for a selected software product includes an update anchor identifying the latest update information on the first child update service node corresponding to the selected software product; and

wherein determining whether any software updates for the selected software product are available for the first child update service node comprises determining whether there are any updates available to the first child update service node according to the update anchor and according to rules associated with the distribution of software updates to the first child update service node.

22. A tangible computer-readable medium bearing computer-executable instructions which, when executed on a first child update service node wherein the first and second child update services nodes are organized in a hierarchy of update service nodes originating from a root update service node, and wherein the first child update service node is a parent update service node to the second child update service node, carry out the method comprising the steps of:

(a) receiving a request for a software update catalog from the second child update service node, the software update catalog identifying software products for which the first update service node provides software updates;

(b) returning a software update catalog to the child update service node;

(c) receiving a request for a list of software updates for a selected software product available for the second child update service node;

(d) determining whether any software updates for the selected software product are available for the second child update service node;

(e) returning a software update list identifying those software updates for the selected software product determined to be available for the second child update service node to the second child update service node, the software update list identifies available software updates according to a unique update identifier;

(f) receiving an update request for metadata corresponding to an available software update from the second child update service node, the update request identifying the available software update by its unique update identifier; and

(g) returning update metadata corresponding to the software update identified in the update request, the update metadata including information corresponding to the software update including a reference for obtaining the corresponding update payload.