(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 December 2005 (29.12.2005)

PCT

(10) International Publication Number
WO 2005/124554 A2

(51) International Patent Classification[7]: G06F 11/30

(21) International Application Number:
PCT/US2005/020988

(22) International Filing Date: 9 June 2005 (09.06.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/866,647 10 June 2004 (10.06.2004) US

(71) Applicant *(for all designated States except US)*: **CISCO TECHNOLOGY, INC.** [US/US]; 170 W. Tasman Drive, San Jose, California 95134-1706 (US).

(72) Inventors; and

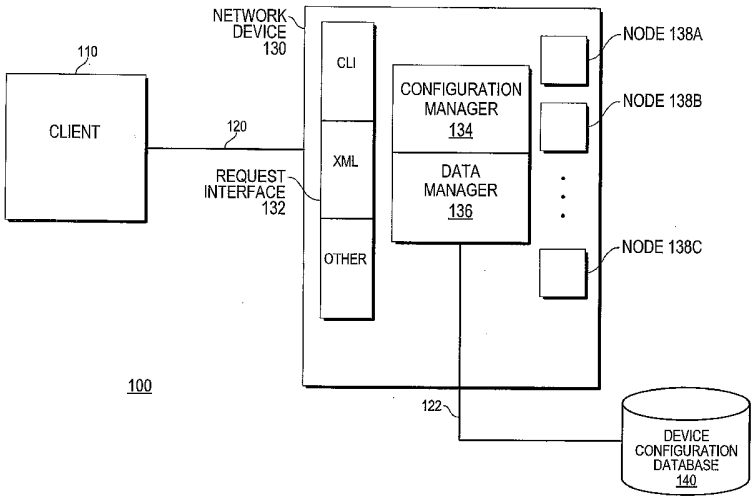
(75) Inventors/Applicants *(for US only)*: **FRESKOS, Mark A.** [US/US]; PO Box 756, Morrisville, North Carolina 27560 (US). **HAWKE, Michelle D.** [GB/GB]; 25 Bellview, St Albans Hertfordshire (GB). **JAIN, Dhanendra** [IN/US]; 1775 Milmont Drive, Apt. N307, Milpitas, California 95035 (US). **JAIN, Kapil** [IN/US]; 80 Descanso Drive, #1429, San Jose, California 95134 (US). **LETCHWORTH, Charles M.** [US/US]; 102 Bayreuth Pl, Cary, North Carolina 27513 (US). **MCDONNELL, Neal** [GB/GB]; 12 Preston Court, Eastfield Close, Fernhill Heath Worcester WR3 7TU (GB). **MOVASSAGHI, Yassin** [US/US]; 2741 Farnborough Road, Raleigh, North Carolina 27613 (US). **PUVVALA, Sukumar** [IN/IN]; Flat No. C-203, Adarsh Garden, 47th Cross, Jayanagar 8th Block, Bangalore (IN). **XIE, Xiaobing** [CN/US]; 2961 Cortina Drive, San Jose, California 95132 (US). **WARD, David D.** [US/US]; 301 221st Avenue, Somerset, Wisconsin 54025 (US).

(74) Agents: **PALERMO, Christopher, J.** et al.; HICKMAN PALERMO TRUONG & BECKER LLP, 2055 Gateway Place, Suite 550, San Jose, California 95110-1089 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE,

[Continued on next page]

(54) Title: TWO-STAGE NETWORK DEVICE CONFIGURATION PROCESS



(57) Abstract: A method and apparatus for modifying the configuration of a network device, such as a router, using a two-stage configuration model is provided. A first request for a change in configuration of a network device is received. Configuration data that describes the change in configuration of the network device is stored in a buffer. A second request to modify the current operational state of the network device to reflect the configuration data stored in the buffer is received. An exclusive lock on the network device is obtained. The current operational state of the network device is modified to reflect the configuration data stored in the buffer. Multiple users may modify the network device without interfering with one another because conflicts are avoided through use of an exclusive lock. Requests of different management operations may be contained within XML documents that are transmitted from the client to the network device.

WO 2005/124554 A2

KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

TWO-STAGE NETWORK DEVICE CONFIGURATION PROCESS

FIELD OF THE INVENTION

[0001] The present invention relates to configuring a network device using a two-stage model.

BACKGROUND

[0002] It is often necessary to modify the configuration of a network device, such as a router. Typically, to effect a configuration change in a router a user would issue one or more command line interface (CLI) commands to the router. As each CLI command is submitted to the router, the router interprets the commands and executes the command immediately. As each CLI command is executed, the command effects a change to the current operational configuration of the router.

[0003] This approach has several disadvantages. First, more than one user may wish to modify the configuration of the network device at the same time. If two or more users are submitting CLI commands to the router simultaneously, then the commands from each user will be executed as they are entered. As a result, the configuration changes entered by one party will interfere with the configuration changes entered by any other party.

[0004] Second, as each command is interpreted and executed as it is submitted to the router, it is possible that one command submitted by a user may be executed, while another command submitted by the same user may not be executed, e.g., the command is not executed because the command contains a syntax error or the command may not be a valid command. Consequently, only a partial set of the desired configuration changes may be performed on the network device. However, this may result in an undesirable configuration for the network device, as the resulting configuration was not intended.

[0005] Consequently, there is a need in the art to effect a modification of a network device without incurring the disadvantages associated with the above described approaches. The approaches described in this section are approaches that could be pursued, but not necessarily approaches that have been previously conceived or pursued. Therefore, unless otherwise indicated, it should not be assumed that any of the approaches described in this section qualify as prior art merely by virtue of their inclusion in this section.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Embodiments of the invention are illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0011] FIG. 1 is a block diagram of a system according to an embodiment;

[0012] FIG. 2 is a flow chart illustrating the high level functional steps according to an embodiment; and

[0013] FIG. 3 is a block diagram that illustrates a computer system upon which an embodiment may be implemented.

DETAILED DESCRIPTION

[0014] In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present disclosure. It will be apparent, however, that embodiments may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the description of the embodiments herein.

FUNCTIONAL OVERVIEW

[0015] Embodiments of the invention provide for modifying the configuration of a network device, such as a router, using a two-stage configuration model. In the first stage, a first request is received for a change in configuration of a network device is received. Configuration data identified by the first request is stored, but the first request is not performed. In the second stage, after obtaining an exclusive lock on the network device in response to receiving a second request to modify the current operational state of the network device to reflect the stored configuration data, the current operational state of the network device is modified to reflect the configuration data. Thus, the configuration of a network device is modified only after obtaining the exclusive lock on the network device in the second stage of the two-stage model. The first and second request may be comprised in an XML document.

[0016] More specifically, in an embodiment, the network device receives an XML document containing a first request for a change in configuration of the network device to a potential operational state from a current operational state of the network device. The configuration data describes the change in configuration of the network device.

[0017] Next, the configuration data is stored in a buffer. At this point, processing is still "in the first stage;" consequently, the configuration data stored in the buffer may be parsed to check for syntax errors, but the configuration of the network device is not changed.

[0018] Thereafter, an XML document containing a second request is received, at the network device, which requests the current operational state of the network device to be modified to reflect the configuration data stored in the buffer. Before the current operational state of the network device can be modified, it is necessary for the user associated with the first and second request to obtain an exclusive lock on the network device. Having possession of the exclusive lock prevents another user from changing the current operational state of the network device. In effect, once the exclusive lock is obtained on the network device, processing enters the "second stage." After the user has obtained the exclusive lock, the current operational state of the network device is modified to reflect the configuration data.

[0019] Employment of the two-stage configuration model allows multiple users to modify a network device without interfering with one another because conflicts are avoided through use of the exclusive lock. An exclusive lock may be obtained explicitly, through a user command to obtain the lock, or implicitly by requesting the network device to commit a set of configuration , Thus, a user who has not explicitly requested and received an exclusive lock may still issue requests to the network device to commit configuration data, and the network device will interpret the request to commit any configuration changes to the network device as an implicit request for an exclusive lock. Also, as configuration data is saved at the network device, a user can modify the configuration of the network device to reflect a prior configuration or may view an earlier configuration of the network device. Moreover, the network device provides a mechanism for a user to modify, save, and error check submitted requests in real time.

[0020] Other embodiments are described in further detail herein.

ARCHITECTURE OVERVIEW

[0021] FIG. 1 is a block diagram of a system 100 according to an embodiment. The system 100 of FIG. 1 may be used to modify the configuration of a network device using a two-stage configuration model. System 100 includes a client 110, communications links 120 and 122, a network device 130, and a device configuration database 140.

[0022] A client, such as client 110, may be implemented by any medium or mechanism that provides for the transmission of a command or request to a network device. Client 110 may be implemented in software or in hardware. Examples of client 110 include, without limitation, a web browser, a software application executing on a machine, a wireless device, and a

management console. While only client 110 is shown in FIG. 1, embodiments may include any number of clients in system 100.

[0023] Communications link 120 may be implemented by any medium or mechanism that provides for the exchange of data between client 110 and network device 130. Communications link 122 may be implemented by any medium or mechanism that provides for the exchange of data between network device 130 and device configuration database 140. Examples of communications links 120 and 122 include, without limitation, a network such as a Local Area Network (LAN), Wide Area Network (WAN), Ethernet or the Internet, or one or more terrestrial, satellite or wireless links.

[0024] A network device, such as network device 130, may be implemented by device that is accessible to a network and is capable of being configured. Examples of network device 130 include, without limitation, a router, a server, a PC, a wireless device, a firewall, and a cell phone. While only network device 130 is shown in FIG. 1, embodiments may include any number of network devices in system 100.

[0025] Network device 130 includes a request interface 132, a configuration manager 134, a data manager, and one or more nodes 138. A request interface, such as request interface 132, may be implemented by any software component executing on network device 130 that is capable of exchanging communications with client 110. Request interface 132 may exchange communications using a variety of transport protocols. Request interface 132 may also process communications encoded using a variety of protocols, including, but not limited to, CLI and XML.

[0026] A configuration manager, such as configuration manager 134, may be implemented by any software component executing on network device 130 that is capable of managing the configuration of the network device. For example, configuration manager 134 may process any request received by request interface 132 that concerns the configuration of the network device.

[0027] A data manager, such as a data manager 136, may be implemented by any software component executing on network device 130 that is capable of managing the persistent storage of data to a device configuration database.

[0028] A node, such as node 138A, 138B, and 138C, may be implemented by any hardware or software component of network device 130 that may be separately configurable. Examples of node 138A, 138B, and 138C include, without limitation, a line card and a software module that is configurable.

[0029] A device configuration database, such as device configuration database 140, as broadly used herein, refers to any medium or mechanism that provides for the persistent storage

of data. Examples of device configuration database 140 include, without limitation, a relational database, an object-oriented database, a multidimensional database, a hierarchical database, a file server, and an EPROM chip.

OPERATION OF TWO-STAGE CONFIGURATION MODEL

[0030] FIG. 2 is a flow chart illustrating the functional steps according to an embodiment. Through the performance of the functional steps of FIG. 2, network device 130 may be configured using a two-stage configuration model. The functional steps of FIG. 2 shall be described below with reference to the illustrative system 100 depicted in FIG. 1.

[0031] In step 210, a first request from client 110 to network device 130 over communications link 120 is received. In an embodiment, request interface 132 receives the first request of step 210.

[0032] Request interface 132 may receive requests containing one or more commands from client 110 using a variety of transport protocols. Request interface 132 may process requests encoded using a variety of protocols. Request interface 132 may comprises one or more components that parse communications encoded using different protocols, such as CLI and XML. For example, request interface 132 may comprise a component that parses CLI commands, as shown in FIG. 1. Thus, when client 110 transmits a CLI command to request interface 132, request interface 132 is able to process the CLI command.

[0033] In another example, request interface 132 may comprise a component that parses XML communications. Request interface 132 may contain a component, as shown in FIG. 1, that can read XML documents and process XML tags and associated information that are contained therein. The processing of requests sent from client 110 to network device 130 that are contained within an XML document shall be explained in greater detail below. Request interface 132 may also expose an API that allows client 110 to issue requests to network device 130.

[0034] In an embodiment, the first request of step 210 may contain one or more commands, e.g., the communication may be an XML document that contains one or more commands. One or more the functions listed in Table 1 may be performed by the request received in step 210. Note that Table 1 is merely illustrative, as the request received in step 210 may perform other functions than those listed in Table 1.

TABLE 1

Modify the current operational configuration according to a set of configuration data
Lock the current operational configuration
Unlock the current operational configuration
Retrieve the change made to the configuration data stored in the buffer
Retrieve the current operational configuration
Retrieve a merged configuration reflecting both the configuration data in the buffer and the current operational configuration
Retrieve the configuration changes resulting from a commit operation
Retrieve the configuration changes resulting from a rollback operation
Load the buffer with configuration data stored in a device configuration database
Load the buffer with the failed configuration from the most recent commit operation
Save the contents of a buffer containing configuration data to a device configuration database
Commit the contents of the buffer to cause the current operational state to reflect the configuration data stored in the buffer
Clear the contents of the buffer
Rollback a set of configuration changes
Retrieve the configuration history regarding a set of commits
Retrieve the configuration history regarding all users that are currently configuring the network device

[0035] Note that the request received in step 210 may be expressed in a variety of forms, including: CLI commands, commands contained within an XML document, one or more calls

through an exposed API of request interface 132, or another protocol which request interface 132 is configured to process.

[0036] To illustrate the functional steps of the two-stage configuration model, an example shall be described wherein a request is received in step 210 to change the configuration of a network device to a potential state from a current operational state of the network device. The request may accompany or reference configuration data. Configuration data is data that describes a change in the configuration of a network device. Configuration data may describe one or more specific configuration changes made to an operational state of the network device. The current operational state of the network device is the configuration of the network device as it is current operating.

[0037] Note that in this example, processing is currently in the first stage of the two-stage model, because prior to entering the second stage of the two-stage model, one needs to obtain the exclusive lock on the network device. After the performance of step 210, processing proceeds to step 220.

[0038] In step 220, the configuration data that describes the change in configuration of the network device as requested in step 210 is stored in a buffer. A buffer is any portion of volatile or non-volatile memory on network device 130 that may store configuration data. Upon receipt of the first request of step 210, request interface 132 may forward to configuration manager 134 any request that concerns the configuration of the network device. In an embodiment, configuration manager 134 stores the configuration data in the buffer in step 220.

[0039] After configuration data is stored in the buffer after the performance of step 220, the configuration data may be viewed by a user. A user may transmit a request from client 110 to network device 130 to view the configuration data stored in the buffer. In response to receiving such a request, the configuration manager 134 may create and provide a view to the user of the configuration data stored in the buffer. For example, the configuration manager 134 may retrieve the configuration data stored in the buffer, and transmit the retrieved configuration data to the user.

[0040] In an embodiment, the network device 130 may comprise a set of one of more buffers. In such an embodiment, each buffer of the set of one or more buffers may be associated with a single user. Each buffer of the set of one or more buffers may only store configuration data associated with the user to which the buffer is associated. For example, network device 130 may comprise 100 buffers, and each of the 100 buffers only stores configuration data for a single user at a time. When a request is received at the network device, the user associated with the request is assigned to a buffer. Thereafter, configuration data associated with that user is

stored in the buffer to which the user is assigned. After a period of time elapses, the user may no longer no assigned to a particular buffer; consequently, the next time the user submits a request to network device 130, that user may be assigned to a different buffer.

[0041] Two or more users may transmits request for a change in configuration of the network device contemporaneously because the configuration data associated with each user will be stored in a separate buffer. A user can save configuration data to the network device independent of the activity of any other user. A user can modify the configuration of the network device when other users are transmitting requests for a change in configuration of the same network device, except as discussed below, e.g., a user may be prevented from modifying the configuration of the network device if that user cannot obtain an exclusive lock. After the processing of step 220, processing proceeds to step 230.

[0042] In step 230, a second request to modify the current operational state of the network device to reflect the configuration data stored in the buffer is received. Request interface 132 may receive the second request from client 110. The second request of step 230 is transmitted from the same user or party as the first request of step 210. As explained above, when request interface 132 determines that the second request of step 230 concerns the configuration of the network device, request interface 132 communicates with configuration manager 134 to inform configuration manager 134 of the second request of step 230. If the network device comprises more than one buffer, then the second request of step 230 refers to the buffer that is associated with the user or party that transmitted the second request of step 230. After the performance of step 230, processing proceeds to step 240.

[0043] In step 240, an exclusive lock on the network device is obtained, through either an explicit or implicit request. In an embodiment, configuration manager 134 may obtain the exclusive lock for a user associated with the second request. Having possession of the exclusive lock prevents another user from changing the current operational state of the network device. In effect, once the exclusive lock is obtained on the network device, the "second stage" is entered.

[0044] In one embodiment, a user may obtain an exclusive lock by submitting an explicit request for the exclusive lock using a specified command. In that embodiment, after the network device processing a request from a user for the exclusive lock, that user has the exclusive lock until the exclusive lock is released. In another embodiment, whenever a user submits a request to modify the current operational state of the network device to reflect the configuration data stored in a buffer, the network device interprets the request as an implicit request for a lock, and that user may automatically obtain the exclusive lock unless another user already holds the exclusive lock. In an embodiment, if a user is unable to obtain the exclusive

lock, that user may be notified that the request was not performed because the user could not obtain the exclusive lock. The user must wait until the lock is released and attempt the commit operation again. Requesting the lock explicitly provides a way to ensure that the lock is obtained before the commit operation is requested. However, obtaining an exclusive lock does not guarantee that a commit of a configuration will succeed; for example, if a back-end system failure occurs, then the configuration for which a commit is requested may not become part of the operational state of the network device. After the performance of step 240, processing proceeds to step 250.

[0045] In step 250, the current operational state of the network device is modified to reflect the configuration data. Note that the current operational state of the network device is modified to reflect the configuration data is step 250 only upon obtaining the exclusive lock either explicitly or implicitly. Step 240 may be performed by configuration manager 134. As a result of configuration manager 134 performing step 250, the current operational state of the network device reflects the configuration data that was stored in the buffer in step 220. In an embodiment, after the performance of step 240, the configuration data stored in the buffer is removed.

[0046] As all configuration changes identified in the configuration data are made to network device 130 contemporaneously in step 250, significant performance benefits are achieved. Effecting multiple configuration changes contemporaneously is more efficient than applying each configuration change to network device 130 individually.

APPLICATIONS OF STORING CONFIGURATION DATA IN DEVICE CONFIGURATION DATABASE

[0047] Embodiments store configuration data to enable a user to modify the configuration of the network device to reflect the configuration of the network device at an earlier point in time. A historical record of the configuration data that has been used to modify the current operational state of the network device may also be viewed by a user associated with client 110. Configuration data may describe any changes made to the operational state of network device 130 or any node 138 on network device 130.

[0048] Whenever a request to modify the current operational state of the network device to reflect a set of configuration data is performed, such as when step 250 of FIG. 2 is performed, the configuration data is persistently stored. In an embodiment, in performing step 250, configuration manager 134 instructs data manager 136 to store the configuration data, or a reference to where the configuration data is stored, in device configuration database 140. In an

alternate embodiment, data manager 136 may persistently store the configuration data, or a reference to where the configuration data is stored, at network device 130.

[0049] In an embodiment, data manager 136 stores the configuration data in a binary file in device configuration database 140, where device configuration database 140 is a hierarchical database. The binary file references information that describes, for each of the one or more configuration changes described in the configuration data, details about the configuration change. For example, the binary file could reference information that describes, for each configuration change, when the configuration change was made (for example, a timestamp), what user initiated the configuration change, which client application transmitted the request to make the configuration change, and a location from which the configuration change was initiated (for example, which client or port on the client initiated the request).

[0050] In an embodiment, device configuration database 140 stores configuration history data. Configuration history data is data that describes all changes in the operational state of the network device that occur over a period of time. Configuration history data may be generated by aggregating the configuration data stored in device configuration database 140.

[0051] Configuration manager 134 can process a request from client 110 to view the configuration history data. Configuration manager 134 may retrieve configuration history data associated with a particular point in time or a particular state of network device 130 and transmit the configuration history data to client 110. In this manner, client 110 may view configuration history data of network device 130 associated with any point in time or any state of network device 130. In an embodiment wherein the configuration data describes a set of changes made between operational states of network device 130, rather than fully describing the complete configuration of network device 130, configuration manager 134 may dynamically determine information that fully describes the configuration of network device 130 at the desired particular point in time or state by applying the set of changes described in the configuration data to a base configuration, as described in further detail below.

[0052] Client 110 may view the set of configuration changes made from a first operational state to a second operational state of network device 130. If client 110 transmits a request to network device 130 to view configuration data with reference to a first point in time and a second point in time, configuration manager 134 can use the configuration history data to determine a set of configuration changes between the operational state of the network device associated with the first point in time and the operational state of the network device associated with the second point in time.

[0053] The set of configuration changes generated by configuration manager 134 between two operational states of the network device may be generated either from a forward-looking perspective or from a backward-looking perspective. In other words, for a given starting point in time, the configuration manager 134 can use the configuration history data to generate a set of configuration changes associated with an operational state that is earlier than the starting point or later than the starting point. The requested information about the configuration changes may then be transmitted from network device 130 to client 110.

[0054] In an embodiment, for a particular configuration change made to network device 130, device configuration database 140 may only store data that describes only a set of configuration options that changed from a first operational state of network device 130 to a second operational state of network device 130, rather than storing data that fully describes the second operation state of network device 130. For example, if only 10% of the configuration changed from a first operational state of the network device to a second operational state of the network device, then only the configuration data that reflects the 10% of the configuration of the network device that changed is stored in device configuration database 140. As the configuration history data allows configuration manager 134 to identify the operational state of the network device at an earlier point in time, only the difference between operational states of the network device needs to be stored in order for configuration manager 134 to determine the complete state of the network device at any point in time since configuration history data was stored.

[0055] Since data that describes all changes in the operational state of the network device that occur over a period of time is stored in the device configuration database as configuration history data, the current operational state of the network device may be "rolled back" or returned to an operational state associated with an earlier point in time. A request from a user maybe processed wherein the current state of the network device is changed to reflect the configuration data associated with an earlier point in time. Since a user associated with client 110 may view the configuration data associated with any operational state of network device 130 that is reflected in the configuration history data, the user may view prior configuration data applied to the operational state of the network device 130 and roll back the current operational state of the network device 130 to reflect that configuration data. Consequently, any user of client 110 can alter the configuration of network device 130 to correspond to any prior configuration state, and that user can view information that describes the configuration of any prior state of network device 130, which enable the user to understand exactly what the configuration of network device 130 will be before the rollback operation is made.

[0056] In an embodiment, the configuration of the current operational state of the network device may only be returned to an operational state associated with an earlier point in time if a user has a sufficient privilege level. For example, the user may need to be a "root" user to perform a rollback operation. To illustrate, assume a request to change the configuration of the network device from the current operational state of the network device to a prior operational state of the network device is received by request interface 132. Thereafter, request interface 132 forwards the request to configuration manager 134. Configuration manager 134 determines if the user associated with the request has a sufficient privilege level for the request to be performed. Configuration manager 134 only changes the configuration of the network device from the current operational state of the network device to a prior operational state of the network device specified in the request upon determining that the user has the sufficient privilege level for the request to be performed

[0057] Data manager 136 may periodically perform a rebase operation. A rebase operation creates a new base configuration from the set of configuration history data stored in device configuration database 140. Network device 130 loads (or "boots") a base configuration whenever network device 130 is initially turned on. If one or more configuration changes have been made to the base configuration, then the network device 130 applies those configuration changes to the base configuration to ensure the configuration of network device 130 is current. Performing a periodic rebase operation advantageously reduces the number of configuration changes that need to be applied to the base configuration. Data manager 136 may perform a rebase operation in response to a variety of events, e.g., (a) the number of commits performed on network device 130 exceeds a configurable threshold, or (b) the size of the configuration data stored in device configuration database 140, or a portion therefore, exceeds a configurable threshold.

[0058] Data manager 136 may periodically perform a trim operation. A trim operation is an operation to reduce the amount of configuration changes that are stored in the device configuration database 140 by deleting the oldest configuration changes in the configuration history data. A trim operation reduces the amount of storage space required to store configuration history data. A trim operation may advantageously remove configuration changes made to network device 130 that are no longer needed, e.g., a rebase operation may make storing a particular configuration change made to network device 130 unnecessary if the base configuration already reflects that configuration change. Data manager 136 may perform a trim operation in response to a variety of events, e.g., (a) the number of commits performed on network device 130 exceeds a configurable threshold, (b) the size of the configuration data

stored in device configuration database 140, or a portion therefore, exceeds a configurable threshold, (c) the passage of a configurable amount of time, or (d) in response to a request issued by client 110.

ERROR CHECKING

[0059] In an embodiment, configuration manager 134 may comprise a parser. A parser is any component that is capable of determining whether a request contains an error or is otherwise unable to be performed. The parser may be used by configuration manager 134 to determine whether a request contains one or more syntax errors. In an embodiment, only a received request for a change in the configuration of the network device associated with a user that has not yet obtained the exclusive lock on the network device is processed to determine whether the request contains one or more syntax errors.

[0060] In response to a determination that a request contains one or more syntax errors, a communication may be transmitted from the network device to the user that transmitted the request containing the one or more syntax errors. The communication may comprise information about the determination that a request contains one or more syntax errors, e.g., a description of the one or more syntax errors that are contained with the request. Alternatively, if the communication sent to the user does not describe the one or more syntax errors that are contained with the command, a second communication that does describe the one or more syntax errors that are contained with the command may be sent to the user in response to receiving a request for that information from the user.

[0061] In an embodiment, configuration manager 134 may determine whether a request contains one or more semantic errors or one or more verification errors. In an embodiment, configuration manager 134 only determines whether a request contains one or more semantic errors or one or more verification errors if a user associated with the request has obtained an exclusive lock on the network device, and the user has transmitted a request to network device 130 to modify the current operational state of network device 130 to reflect configuration data stored in the buffer. Semantic errors and verification errors generally arise from back end processing entities that cannot process the request. For example, semantic errors and verification errors include a duplicate IP address contained within the request and inclusion of a user name or user group that does not exist.

[0062] In response to determining that a request contains one or more semantic errors or one or more verification errors, configuration manager 134 may transmit a communication to the user issuing the request that indicates information about the determination that the request

contains one or more semantic errors or one or more verification errors, e.g., the communication may describe the one or more semantic errors or one or more verification errors found within the request.

EXECUTING ATOMIC AND BEST EFFORT CONFIGURATION CHANGES

[0063] Embodiments provide for processing a request based on whether the particular request is an "atomic" request or a "best effort" request. An "atomic" request is a request that is performed only if it is determined that each of the one or more configuration changes described by the configuration data associated with the request is capable of being performed. Thus, if a request is an atomic request, if any of the one or more configuration change described by the configuration data associated with the request cannot be performed, then none of the configuration changes described by the configuration data associated with the request are performed.

[0064] In an embodiment that processes atomic requests, configuration manager 134 determines if a request requires that each of the one or more configuration changes described by the configuration data associated with the request be performed. In response to a determination that the request requires that each of the one or more configuration changes described by the configuration data associated with the request be performed, the configuration manager 134 determines if each of the one or more configuration changes is capable of being performed. Thereafter, if each of the one or more configuration changes is capable of being performed, then the configuration manager 134 modifies the current operational state of the network device to reflect the configuration data.

[0065] A "best effort" request, on the other hand, is a request that is executed regardless of whether a particular configuration change described by the configuration data associated with the request is not capable of being performed. Thus, if a request is a best effort request, even if one or more of the configuration changes described by the configuration data associated with the request cannot be performed, then the one or more configuration changes described by the configuration data that are capable of being performed are still performed.

[0066] In an embodiment that processes best effort requests, configuration manager 134 determining if the request requires that each of the one or more configuration changes described by the configuration data associated with the request be performed. In response to a determination that the request does not require that each of the one or more configuration changes described by the configuration data associated with the request be performed, then the configuration manager 134 modifies the current operational state of the network device to

reflect any of the one or more configuration changes described by the configuration data that can be performed, even if one or more configuration changes described by the configuration data associated with the request are not capable of being performed.

XML INTERFACE FOR TWO-STAGE CONFIGURATION OPERATIONS

[0067] Client 110 may transmit a XML document over communications link 120 to request interface 132. The XML document may contain one or more requests that are formatted according to any of a variety of request syntax conventions, such as the Cisco CLI syntax. The XML document may be sent to the network device using any of several transport mechanisms including CORBA, Telnet, SSH, etc. Any request may be contained in the XML document, e.g., any request in Table 1 may be contained within an XML document. Also, a request contained within an XML document may be associated with different types of management operations. For example, the request may be to (a) manipulate the native management data on the network device, e.g., by processing operations to get, set (i.e., modify), create, or delete instances of management data, (b) process an operation regarding more advanced configuration services on the network device, e.g., a lock operation, an unlock operation, a commit operation, or a rollback operation, or (c) perform a command line interface (CLI) operation.

[0068] Note that subsequent requests from the client to the network device and/or responses from the network device to the client which are contained within XML documents may be associated with different types of management operations than of the prior requests. Each request from a client to a network device may view the effects of other requests, even if those requests are of different types of management operations. For example, a first request contained within an XML document may cause configuration data to be stored in a buffer, while a second request contained within an XML document may be associated may view the configuration data stored in the buffer, even if the second request is associated with a different type of management operation than the first request.

[0069] Request interface 132 may process the received XML document to extract any requests in the XML document, and thereafter forward those requests to configuration manager 134 so that the requests may be processed. An illustrative example of an XML document containing multiple requests (or operations) sent from client 110 to network device 130 is described below in the pseudocode of Example 1.

Example 1:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Request MajorVersion = "1" MinorVersion = "0">
    <Operation 1>
    ...
    Operation 1 data is contained here.
    ...
    </Operation 1>
    <Operation 2>
    ...
    Operation 2 data is contained here.
    ...
    </Operation 2>
</Request>
```

[0070] In response to receiving the XML document illustrated in Example 1, configuration manager 134 processes the requests contained therein. Request interface 134 may forward any embedded commands within the XML document that relate to the configuration of network device 130 to configuration manager 134. After configuration manager 134 has processed the request, request interface 132 may transmit response data that describes a result of processing the request on network device 130 to client 110. An illustrative example of an XML document containing response data sent from network device 130 to client 110 is described below in the pseudocode of Example 2.

Example 2:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Response MajorVersion = "1" MinorVersion = "0">
    <Operation 1>
    ...
    Operation 1 response data is contained here.
    ...
    </Operation 1>
    <Operation 2>
    ...
    Operation 2 response data is contained here.
```

```
...
</Operation 2>
</Response>
```

[0071] Note that the XML document sent from client 110 of Example 1 circumscribes requests with a "Request" tag, while the XML document sent from network device 130 of Example 2 circumscribes response data with a "Response" tag.

[0072] Client 110 may transmit an XML document to network device 130 to request the current running BGP configuration of network device 130, as shown below in Example 3.

Example 3:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Request MajorVersion = "1" MinorVersion = "0">
    <Get>
        <Configuration Source ="CurrentConfig">
            <BGP MajorVersion = "1" MinorVersion ="0"/>
        </Configuration>
    </Get>
</Request>
```

[0073] In response to receiving the XML document of Example 3, network device 130 may transmit an XML document containing the current running BGP configuration of network device 130, as shown below in Example 4.

Example 4:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Response MajorVersion = "1" MinorVersion = "0">
    <Get>
        <Configuration>
            <BGP MajorVersion = "1" MinorVersion ="0">
                <AS>
                    <Naming>
                        <AS>3</AS>
                    </Naming>
                    <Global>
                        <DefaultMetric>5</DefaultMetric>
                        <GlobalTimers>
```

```
                                        <Keepalive>30</Keepalive>
                                        <Holdtime>90</Holdtime>
                                    </GlobalTimers>
                                ...
                                More BGP config data returned here.
                                ...
                            </BGP>
                        </Configuration>
                    </Get>
                </Response>
```

[0074] Client 110 may transmit to network device 130 an XML document containing configuration data which will be stored in the buffer of network device 130, as shown below in Example 5.

Example 5:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Request MajorVersion = "1" MinorVersion = "0">
    <Set>
        <Configuration>
            <BGP MajorVersion = "1" MinorVersion = "0">
                <AS>
                    <Naming>
                        <AS>3</AS>
                    </Naming>
                    <Global>
                        <DefaultMetric>10</DefaultMetric>
                        <GlobalTimers>
                            <Keepalive>60</Keepalive>
                            <Holdtime>180</Holdtime>
                        </GlobalTimers>
                    </Global>
                </AS>
            </BGP>
        </Configuration>
    </Set>
```

```
</Request>
```

[0075] In response to receiving the XML document of Example 5, network device 130 may transmit an XML document containing an acknowledgement that the configuration data has been received and stored in the buffer, as shown below in Example 6.

Example 6:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Response MajorVersion = "1" MinorVersion = "0">
    <Set>
        <Configuration/>
    </Set>
</Response>
```

[0076] Client 110 may transmit to network device 130 an XML document containing a request to commit configuration data stored in the buffer of network device 130, as shown below in the Example 7.

Example 7:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Request MajorVersion = "1" MinorVersion = "0">
    <Commit Mode ="Atomic" Label ="BGPUpdate"
     Comment = "Sample BGP config update"/>
</Request>
```

[0077] In response to receiving the commit request contained within the XML document of Example 7, network device 130 may transmit an XML document containing an acknowledgement that the commit request has been performed, as shown below in Example 8.

Example 8:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Response MajorVersion = "1" MinorVersion = "0">
    <Commit Mode ="Atomic" Label ="BGPUpdate"
     Comment ="Sample BGP config update"/>
</Response>
```

[0078] Note that a request from client 110 to network device 130, which is contained within an XML document, may correspond to a first type of management operation, and the response

from the network device 130 to the client 110, which is contained within another XML document, may correspond to a second type of management operation. This may be advantageous when a user associated with client 110 is more accustomed to issuing requests to network device 130 that correspond to a first type of management operation, but prefers to view information about the results of processing the request on network device 130 in accordance with a second type of management operation. Example 9, shown below, illustrates a request from client 110 to network device 130, contained within an XML document, that corresponds to a manipulation of the native management data on network device 130. Example 10, shown below, illustrates a response from network device 130 to client 110, contained within an XML document, that corresponds to a CLI management operation that is made in response to the request of Example 9.

Example 9:

```
<?xml version="1.0" encoding="UTF-8"?>
<Request>
  <Set>
    <Configuration>
      <BGP>
        <AS>
          <Naming>
            <AS>65001</AS>
          </Naming>
          <Global>
            <GlobalAFTable>
              <GlobalAF>
                <Naming>
                  <AF>IPv4Unicast</AF>
                </Naming>
                <SourcedNetworkTable>
                  <SourcedNetwork>
                    <Naming>
                      <Network>
                        <IPV4Address>202.202.11.11</IPV4Address>
                        <IPV4PrefixLength>32</IPV4PrefixLength>
                      </Network>
```

```
                    </Naming>
                  </SourcedNetwork>
                </SourcedNetworkTable>
              </GlobalAF>
            </GlobalAFTable>
          </Global>
        </AS>
      </BGP>
    </Configuration>
  </Set>
</Request>

<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
    <CLI>
      <Configuration>
          show config
      </Configuration>
  </CLI>
</Request>
```

Example 10:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <CLI>
    <Configuration>
        Building configuration...
        router bgp 65001
        address-family ipv4 unicast
        network 202.202.11.11/32
        !
        !
        end
    </Configuration>
```

</CLI>
</Response>

[0079] Note that any request of any type of management operation that client 110 may issue to network device 130 may be contained within an XML document and processed according to the above explanation. Each command of any management operation may be identified by a particular tag and associated values in an XML document. Accordingly, not every request has been shown in an illustrative example; however, those skilled in the art will appreciate that any type of management operation that client 110 may issue to network device 130 may be contained within an XML document and processed according to embodiments of the invention.

CONFIGURATION SESSION MANAGEMENT AND LOCKING

[0080] As explained above, users of clients, e.g., client 110, may obtain an exclusive lock on network device 130 which prevents another user to effect configuration changes on network device 130 while that user has the lock. If a first user attempts to modify the current operational state of network device 130 while a second user has an exclusive lock, then the request of the first user will be aborted by the configuration manager 134.

[0081] There are two types of exclusive locks: implicit and explicit. An implicit lock is obtained whenever a user initiates a request to modify the current operational state of network device 130 to reflect the configuration data stored in a buffer, e.g., when step 250 of FIG. 2 is performed. This implicit lock prevents a second user from making any configuration changes to network device 130 while a first user is in the process of making configuration changes to network device 130.

[0082] On the other hand, an explicit lock is obtained whenever a user specifically requests a lock unaccompanied by a request to modify the operational state of network device 130. An explicit lock is advantageous where a user does not wish another user to be able to modify the operational state of network device 130.

[0083] A client may transmit to network device 130 a request to determine which users are operational connected to network device 130 and which user has the exclusive lock. Configuration manger 134 maintains information about which users are connected to network device 130 and which user has the exclusive lock, including information about the session id of each user connected to network device 130, a timestamp associated with each user connected to network device 130, a username associated with each user connected to network device 130, a location identifier associated with each user connected to network device 130, and whether each user connected to network device 130 has an exclusive lock. Configuration manager 134 may

process a request for this information received by network device 130 and thereafter cause a response to be transmitted to the requesting client that contains any information maintained by configuration manager 134 about the users connected to network device 130, such as information about which user has the exclusive lock.

[0084] Client 110 may transmit an XML document over communications link 120 to network device 130 that contains a request to obtain an exclusive lock. If the request is successful, network device 130 transmits a communication informing client 110 that client 110 has the exclusive lock. On the other hand, if the request is not successful, network device 130 transmits a communication informing client 110 why the exclusive lock was not obtained, e.g., an error code or error message may be provided to client 110. Example 11 illustrates a portion of a XML document that client 110 may transmit to network device 130 to request an exclusive lock.

Example 11:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Request MajorVersion = "1" MinorVersion = "0">
        <Lock/>
</Request>
```

[0085] In response to receiving the request for an exclusive lock contained within the XML document of Example 11, network device 130 may transmit the XML document of Example 12 to client 110 to indicate that client 110 has the exclusive lock.

Example 12:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Response MajorVersion = "1" MinorVersion = "0">
        <Lock/>
</Response>
```

[0086] Example 13 illustrates a portion of a XML document that client 110 may transmit to network device 130 to release the exclusive lock.

Example 13:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Request MajorVersion = "1" MinorVersion = "0">
        <Unlock/>
</Request>
```

[0087] In response to receiving the request to release the exclusive lock contained within the XML document of Example 13, network device 130 may transmit the XML document of Example 14 to client 110 to indicate that client 110 has released the exclusive lock.

Example 14:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<Response MajorVersion = "1" MinorVersion = "0">
    <Unlock/>
</Response>
```

[0088] Multiple users may be operational connected to network device 130. In an embodiment, each user of a plurality of users may view the configuration data that another user of the plurality of users has saved in a buffer of network device 130. In this way, one user can view the configuration changes that another user is making to network device 130.

[0089] As mentioned above, configuration manger 134 maintains information about which users are connected to network device 130 and which user has the exclusive lock Network device 130 maintains information. Consequently, a user of client 110 may transmit a request to network device 130, which when processed by configuration manager 134, causes information to be sent to client 110 that describes which users are actively editing network device 130 and whether an exclusive lock has been assigned to any other user.

IMPLEMENTING MECHANISMS

[0090] In accordance with an embodiment, client 110 or network device 130 may be implemented on a computer system. FIG. 3 is a block diagram that illustrates a computer system 300 upon which an embodiment may be implemented. Computer system 300 includes a bus 302 or other communication mechanism for communicating information, and a processor 304 coupled with bus 302 for processing information. Computer system 300 also includes a main memory 306, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 302 for storing information and instructions to be executed by processor 304. Main memory 306 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 304. Computer system 300 further includes a read only memory (ROM) 308 or other static storage device coupled to bus 302 for storing static information and instructions for processor 304. A storage device 310, such as a magnetic disk or optical disk, is provided and coupled to bus 302 for storing information and instructions.

[0091] Computer system 300 may be coupled via bus 302 to a display 312, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 314, including alphanumeric and other keys, is coupled to bus 302 for communicating information and command selections to processor 304. Another type of user input device is cursor control 316, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 304 and for controlling cursor movement on display 312. This input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

[0092] The invention is related to the use of computer system 300 for implementing the techniques described herein. According to one embodiment of the invention, those techniques are performed by computer system 300 in response to processor 304 executing one or more sequences of one or more instructions contained in main memory 306. Such instructions may be read into main memory 306 from another machine-readable medium, such as storage device 310. Execution of the sequences of instructions contained in main memory 306 causes processor 304 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

[0093] The term "machine-readable medium" as used herein refers to any medium that participates in providing data that causes a machine to operation in a specific fashion. In an embodiment implemented using computer system 300, various machine-readable media are involved, for example, in providing instructions to processor 304 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 310. Volatile media includes dynamic memory, such as main memory 306. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 302. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infrared data communications.

[0094] Common forms of machine-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

[0095] Various forms of machine-readable media may be involved in carrying one or more sequences of one or more instructions to processor 304 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 300 can receive the data on the telephone line and use an infrared transmitter to convert the data to an infrared signal. An infrared detector can receive the data carried in the infrared signal and appropriate circuitry can place the data on bus 302. Bus 302 carries the data to main memory 306, from which processor 304 retrieves and executes the instructions. The instructions received by main memory 306 may optionally be stored on storage device 310 either before or after execution by processor 304.

[0096] Computer system 300 also includes a communication interface 318 coupled to bus 302. Communication interface 318 provides a two-way data communication coupling to a network link 320 that is connected to a local network 322. For example, communication interface 318 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 318 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 318 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

[0097] Network link 320 typically provides data communication through one or more networks to other data devices. For example, network link 320 may provide a connection through local network 322 to a host computer 324 or to data equipment operated by an Internet Service Provider (ISP) 326. ISP 326 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 328. Local network 322 and Internet 328 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 320 and through communication interface 318, which carry the digital data to and from computer system 300, are exemplary forms of carrier waves transporting the information.

[0098] Computer system 300 can send messages and receive data, including program code, through the network(s), network link 320 and communication interface 318. In the Internet example, a server 330 might transmit a requested code for an application program through Internet 328, ISP 326, local network 322 and communication interface 318.

[0099] The received code may be executed by processor 304 as it is received, and/or stored in storage device 310, or other non-volatile storage for later execution. In this manner, computer system 300 may obtain application code in the form of a carrier wave.

[0100] In the foregoing specification, embodiments of the invention have been described with reference to numerous specific details that may vary from implementation to implementation. Thus, the sole and exclusive indicator of what is the invention, and is intended by the applicants to be the invention, is the set of claims that issue from this application, in the specific form in which such claims issue, including any subsequent correction. Any definitions expressly set forth herein for terms contained in such claims shall govern the meaning of such terms as used in the claims. Hence, no limitation, element, property, feature, advantage or attribute that is not expressly recited in a claim should limit the scope of such claim in any way. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. A computer apparatus, comprising:

means for receiving a first request for a change in a configuration of a network device to a potential operational state from a current operational state of the network device, wherein electronically stored configuration data describes the change in the configuration of the network device;

means for storing the configuration data in a buffer memory;

means for receiving a second request to modify the current operational state of the network device to reflect the configuration data stored in the buffer memory;

means for obtaining an exclusive lock on the network device; and

means for modifying the current operational state of the network device to reflect the configuration data only upon obtaining the exclusive lock, wherein operation of the network device is improved by preventing unauthorized modifications to the current operational state.

2. The apparatus of Claim 1, wherein the network device is a router for a packet-switched network.

3. The apparatus of Claim 1, wherein the means for modifying the current operational state has means for storing the configuration data from the buffer memory to a device configuration database.

4. The apparatus of Claim 1, further comprising means for creating and storing information identifying, for each of the one or more configuration changes, when the configuration change was made, what user initiated the configuration change, and a location from which the configuration change was initiated.

5. The apparatus of Claim 3, wherein the configuration data stored in the device configuration database specifies only a set of configuration options that changed from a first operational state of the network device to a second operational state of the network device.

6. The apparatus of Claim 3, wherein the device configuration database stores configuration history data that describes all changes in an operational state of the

network device that occur over a period of time; and wherein the apparatus includes means for creating and providing a view of the configuration history data stored in the device configuration database associated with a particular point in time.

7. The apparatus of Claim 6, wherein the apparatus includes means for generating, based on the configuration history data, a set of configuration changes that describe the change in configuration between two different operational states of the network device.

8. The apparatus of Claim 1, wherein the buffer memory is a particular buffer memory in a set of buffer memories, and wherein the apparatus includes means for determining that the particular buffer memory in the set of buffer memories is associated with a particular user associated with the first request.

9. The apparatus of Claim 1, wherein the apparatus includes means for determining whether the second request contains one or more semantic errors or one or more verification errors; and means for informing a user associated with the second request that the second request contains one or more semantic errors or one or more verification errors in response to a determination that the first second contains one or more semantic errors or one or more verification errors.

10. The apparatus of Claim 1, wherein the change in configuration described by the configuration data includes one or more configuration changes, and wherein the apparatus includes means for determining if the second request requires that each of the one or more configuration changes described by the configuration data be performed; means for determining if each of the one or more configuration changes is capable of being performed in response to a determination that the second request requires that each of the one or more configuration changes described by the configuration data be performed; and means for modifying the current operational state of the network device to reflect the configuration data only if each of the one or more configuration changes is capable of being performed in response to the determination that the second request requires that each of the one or more configuration changes described by the configuration data be performed.

11. The apparatus of Claim 1, wherein the change in configuration described by the configuration data includes one or more configuration changes, and wherein the apparatus includes means for determining if the second request requires that each of the

one or more configuration changes described by the configuration data be performed; and means for modifying the current operational state of the network device to reflect any of the one or more configuration changes described by the configuration data that can be performed, even if one or more configuration changes described by the configuration data are not capable of being performed in response to a determination that the second request does not require that each of the one or more configuration changes described by the configuration data be performed.

12. The apparatus of Claim 1, wherein the first request is associated with a first user that holds an exclusive lock, wherein the exclusive lock indicates that only the holder of the exclusive lock may effect change to the current operational state of the network device, and wherein the apparatus includes means for determining that a third request to change the configuration of the network device is associated with a second user who does not have the exclusive lock; and means for notifying the second user that the third request is not performed.

13. The apparatus of Claim 1, wherein the apparatus includes means for receiving a third request to change the configuration of the network device from the current operational state of the network device to a prior operational state of the network device; means for determining if a user associated with the third request has a sufficient privilege level for the third request to be performed; and means for changing the configuration of the network device from the current operational state of the network device to the prior operational state of the network device upon determining that the user has the sufficient privilege level for the third request to be performed.

14. An apparatus comprising one or more electronic digital data processors, an interface to a network for receiving and transmitting packet data from and to the network, and a memory storing a plurality of digital program instructions which, when executed by one or more processors, cause the one or more processors to perform the steps of:

receiving a first request for a change in a configuration of a network device to a potential operational state from a current operational state of the network device, wherein electronically stored configuration data describes the change in configuration of the network device;

storing the configuration data in a buffer memory;

receiving a second request to modify the current operational state of the network device to reflect the configuration data stored in the buffer memory;

obtaining an exclusive lock on the network device; and

modifying the current operational state of the network device to reflect the configuration data only upon obtaining the exclusive lock, wherein operation of the network device is improved by preventing unauthorized modifications to the current operational state.

15. The apparatus of Claim 14, wherein the network device is a router for a packet-switched network.

16. The apparatus of Claim 14, wherein the step of modifying the current operational state further comprises storing the configuration data from the buffer memory to a device configuration database.

17. The apparatus of Claim 14, wherein the memory further comprises one or more additional instructions which, when executed by the one or more processors, cause the one or more processors to perform creating and storing information identifying, for each of the one or more configuration changes, when the configuration change was made, what user initiated the configuration change, and a location from which the configuration change was initiated.

18. The apparatus of Claim 16, wherein the configuration data stored in the device configuration database reflects only a set of configuration options that changed from a first operational state of the network device to a second operational state of the network device.

19. The apparatus of Claim 16, wherein the device configuration database stores configuration history data that describes all changes in an operational state of the network device that occur over a period of time; and wherein the memory further comprises one or more additional instructions which, when executed by the one or more processors, cause the one or more processors to perform creating and providing a view of the configuration history data stored in the device configuration database associated with a particular point in time.

20. The apparatus of Claim 19, wherein the memory further comprises one or more additional instructions which, when executed by the one or more processors, cause the one or more processors to perform generating, based on the configuration history data, a set of configuration changes that describe the change in configuration between two different operational states of the network device.

21. The apparatus of Claim 14, wherein the buffer memory is a particular buffer memory in a set of buffer memories, and wherein the step of storing the configuration data further comprises determining that the particular buffer memory in the set of buffer memories is associated with a particular user associated with the first request.

22. The apparatus of Claim 14, wherein the memory further comprises one or more additional instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of determining whether the second request contains one or more semantic errors or one or more verification errors; and in response to a determination that the first second contains one or more semantic errors or one or more verification errors, informing a user associated with the second request that the second request contains one or more semantic errors or one or more verification errors.

23. The apparatus of Claim 14, wherein the change in configuration described by the configuration data includes one or more configuration changes, and wherein the memory further comprises one or more additional instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of determining if the second request requires that each of the one or more configuration changes described by the configuration data be performed; and in response to a determination that the second request requires that each of the one or more configuration changes described by the configuration data be performed: determining if each of the one or more configuration changes is capable of being performed, and modifying the current operational state of the network device to reflect the configuration data only if each of the one or more configuration changes is capable of being performed.

24. The apparatus of Claim 14, wherein the change in configuration described by the configuration data includes one or more configuration changes, and wherein the memory further comprises one or more additional instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of

determining if the second request requires that each of the one or more configuration changes described by the configuration data be performed; and in response to a determination that the second request does not require that each of the one or more configuration changes described by the configuration data be performed: modifying the current operational state of the network device to reflect any of the one or more configuration changes described by the configuration data that can be performed, even if one or more configuration changes described by the configuration data are not capable of being performed.

25. The apparatus of Claim 14, wherein the first request is associated with a first user that holds an exclusive lock, wherein the exclusive lock indicates that only the holder of the exclusive lock may effect change to the current operational state of the network device, and wherein the memory further comprises one or more additional instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of: determining that a third request to change the configuration of the network device is associated with a second user who does not have the exclusive lock; and notifying the second user that the third request is not performed.

26. The apparatus of Claim 14, wherein the memory further comprises one or more additional instructions which, when executed by the one or more processors, cause the one or more processors to perform the steps of receiving a third request to change the configuration of the network device from the current operational state of the network device to a prior operational state of the network device; determining if a user associated with the third request has a sufficient privilege level for the third request to be performed; and changing the configuration of the network device from the current operational state of the network device to the prior operational state of the network device upon determining that the user has the sufficient privilege level for the third request to be performed.

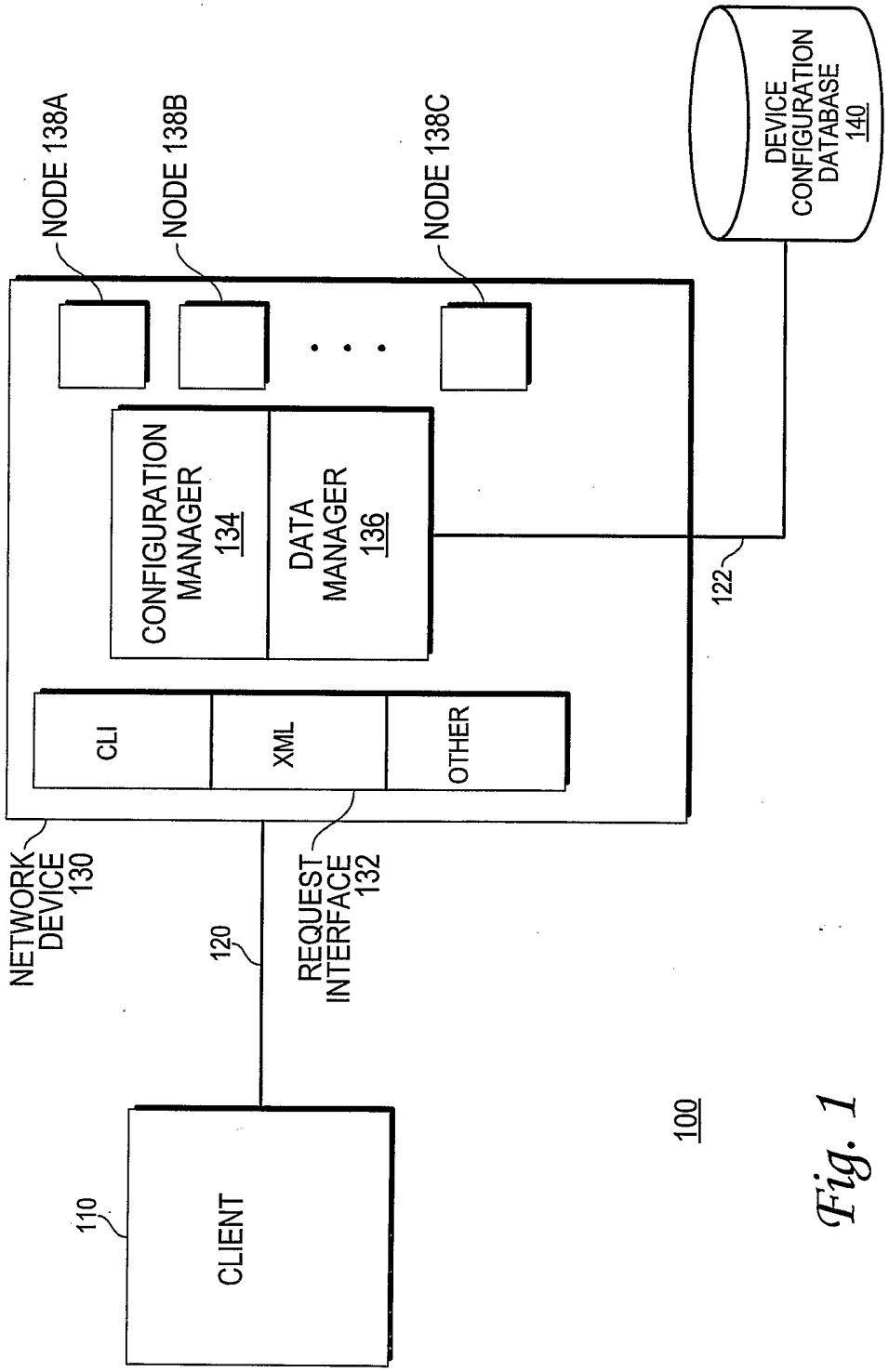

NODE 138A
NODE 138B
NODE 138C
DEVICE CONFIGURATION DATABASE 140
CONFIGURATION MANAGER 134
DATA MANAGER 136
122
CLI
XML
OTHER
NETWORK DEVICE 130
120
REQUEST INTERFACE 132
110
CLIENT
100

Fig. 1

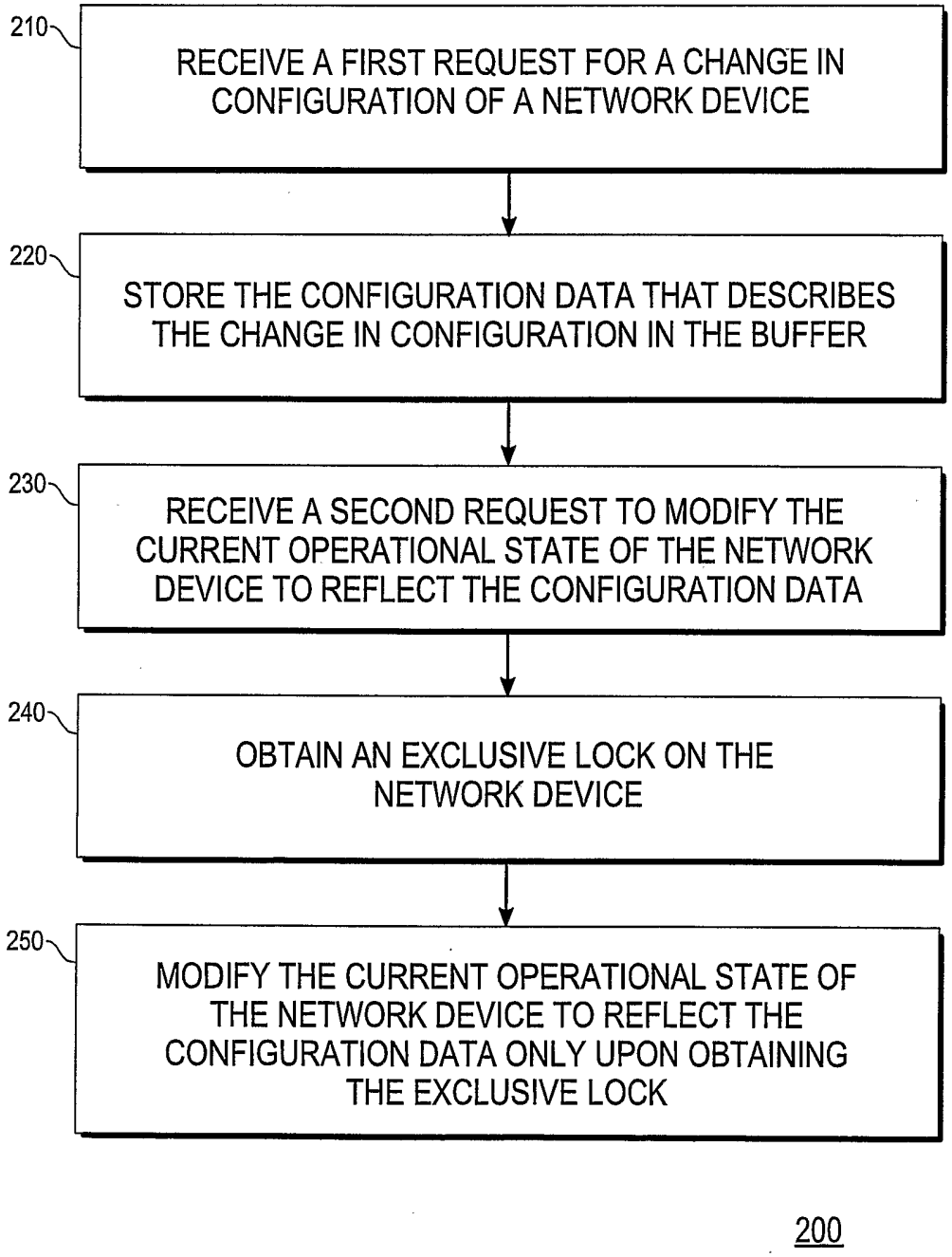

210
RECEIVE A FIRST REQUEST FOR A CHANGE IN CONFIGURATION OF A NETWORK DEVICE
220
STORE THE CONFIGURATION DATA THAT DESCRIBES THE CHANGE IN CONFIGURATION IN THE BUFFER
230
RECEIVE A SECOND REQUEST TO MODIFY THE CURRENT OPERATIONAL STATE OF THE NETWORK DEVICE TO REFLECT THE CONFIGURATION DATA
240
OBTAIN AN EXCLUSIVE LOCK ON THE NETWORK DEVICE
250
MODIFY THE CURRENT OPERATIONAL STATE OF THE NETWORK DEVICE TO REFLECT THE CONFIGURATION DATA ONLY UPON OBTAINING THE EXCLUSIVE LOCK
200

Fig. 2

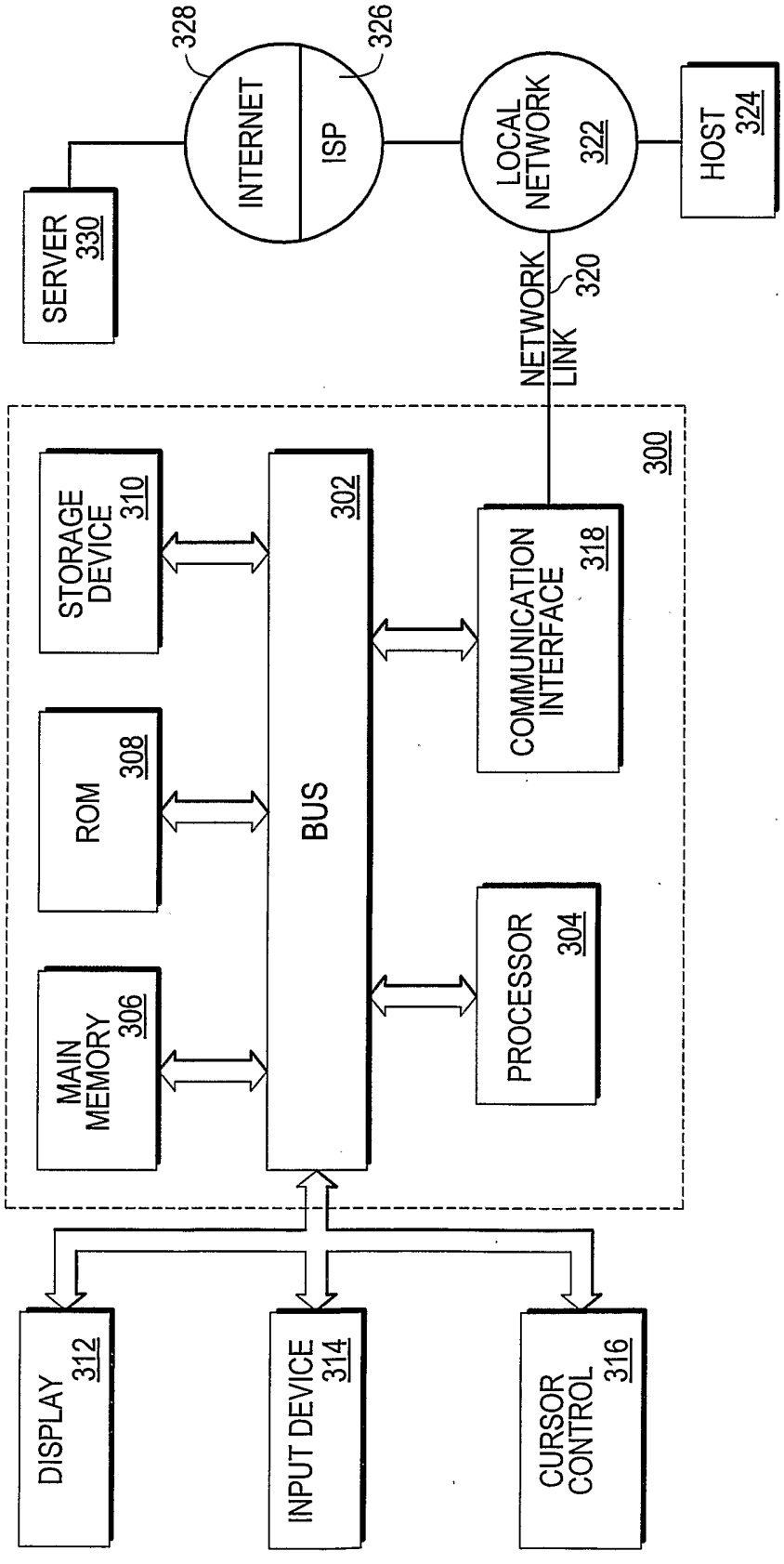

328
INTERNET
ISP
326
LOCAL NETWORK 322
HOST 324
SERVER 330
NETWORK LINK 320
300
STORAGE DEVICE 310
ROM 308
MAIN MEMORY 306
BUS 302
COMMUNICATION INTERFACE 318
PROCESSOR 304
DISPLAY 312
INPUT DEVICE 314
CURSOR CONTROL 316

Fig. 3