IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT NO.: 7,124,289

ISSUED: October 17, 2006

FOR: AUTOMATED PROVISIONING FRAMEWORK FOR INTERNET SITE SERVERS

ATTACHMENT TO FORM PTO-1465
REQUEST FOR *EX PARTE* REEXAMINATION

SIR:

The Public Patent Foundation ("PUBPAT"), a not-for-profit public service organization that works to protect the public from the harms caused by wrongly issued patents and unsound patent policy, respectfully requests ex parte reexamination under 35 U.S.C. §§ 302 – 307 and 37 C.F.R. § 1.510 of every claim of United States Patent No. 7,124,289 issued October 17, 2006 to Raymond E. Suorsa ("the '289 patent") and assigned to Opsware Inc. ("Opsware") because they are all invalid under 35 U.S.C. § 102 and § 103 and their existence is causing significant public harm. Appendix A contains a copy of the '289 patent, and the other documents referred to below.

THE '289 PATENT IS CAUSING SIGNIFICANT PUBLIC HARM

The '289 patent claims methods for automatically configuring or installing software on a plurality of computing devices having different respective sets of software and/or configurations of operating parameters. When Opsware announced the issuance of their patent, Ben Horowitz, CEO of Opsware, was quoted as commenting, with respect to the '289 patent: "This patent

solidifies and protects the work we've done to greatly advance the automation of data centers." (See http://home.businesswire.com/portal/site/opsware/index.jsp?ndmViewId=news_view&newsId=20061113005296).

The work that Opsware claims to have done, however, may be found in publicly available references, including open source software projects that are directed to configuration and installing software on computers, and that were available and in public use in the United States more than a year before the filing date of the '289 patent. Moreover, the '289 patent may impact the developers and users of open source projects, including: Cfengine (available at http://www.gnu.org/software/cfengine/); LCFG (available at http://www.lcfg.org); Quattor (www.quattor.org); and BCFG (available at http://trac.mcs.anl.gov/projects/bcfg2). Individuals, non-profit organizations, educational institutions, and businesses throughout the United States rely on these open source software tools to manage their computers. The technology claimed in the '289 is not the work of the inventor named in the '289 patent, but rather, is the work of open source developers, who have provided their code for the benefit of the public good.

Although this issue is not grounds to grant this request for reexamination, PUBPAT respectfully requests that it be considered when determining whether the validity of the '289 patent as issued merits review by your office.

THE SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

The first substantial new question of patentability raised by this request is whether claims 1 through 28 of the '289 patent were anticipated under 35 U.S.C. § 102 by or were obvious under 35 U.S.C. § 103 in light of a paper published in September 1994, in the USENIX LISA conference proceedings, entitled "Towards a High-Level Machine Configuration System," by Paul Anderson, available from http://www.usenix.org/publications/library/proceedings/lisa94/anderson.html ("Anderson").

A detailed explanation of the pertinency and manner of applying Anderson to each of the claims 1 through 28 of the '289 patent is set forth below. A copy of Anderson is attached. With respect to some of the dependent claims of the '289 patent, reference is made to three additional

documents that demonstrate that certain features are inherent in or obvious from Anderson, specifically, a June 1999 description of the XML-RPC protocol, entitled "XML-RPC Specification," by Dave Winer, available from http://www.xmlrpc.com/spec ("Winer"), and a 1992 paper that describes the NIS protocol, Hess et al., "A Unix Network Protocol Security Study: Network Information Service," Technical Report, A&M University, 1992, available from http://citeseer.ist.psu.edu/12427.html ("Hess"), as well as a 1994 paper that was presented at the same USENIX LISA conference as Anderson, describing the OMNICONF system: Imazu Hideo, "OMNICONF- Making OS Upgrades and Disk Crash Recovery Easier," LISA VIII Proceedings, 1994 (http://www.usenix.org/publications/library/proceedings/lisa94/hideyo.html).

A second substantial new question of patentability raised by this request is whether claims 1 through 28 of the '289 patent were anticipated under 35 U.S.C. § 102 by or were obvious under 35 U.S.C. § 103 in light of a paper published in September 1996, in the USENIX LISA conference proceedings, entitled "Automating the Administration of Heterogeneous LANs," by Michael Fisk ("Fisk"). A detailed explanation of the pertinency and manner of applying Fisk to each of the claims 1 through 28 of the '289 patent is set forth below. A copy of Fisk is attached. With respect to some of the dependent claims of the '289 patent, reference is made to two additional documents that demonstrate that certain features are inherent in or obvious from Fisk, specifically, Winer and Hideo.

A third substantial new question of patentability raised by this request is whether claims 1-28 of the '289 patent were rendered obvious under 35 U.S.C. § 103 by U.S. Patent No. 6,009,274 to Fletcher et al. ("Fletcher") and U.S. Patent No. 6,138,153 to Collins, III et al. ("Collins") in light of Anderson. While Fletcher and Collins were of record during prosecution of the '289 patent, this is a new question of patentability because Anderson was not of record when Fletcher and Collins were considered. Anderson demonstrates the claim elements that the Examiner apparently believed were not shown in Fletcher.

FIRST SUBSTANTIAL NEW QUESTION OF PATENTABILITY:

ANDERSON ANTICIPATES THE '289 PATENT

The '289 patent's application date is October 31, 2000. Anderson was published in 1994, more than one year before the '289 patent's application date. Anderson is prior art to the '289 patent under 35 U.S.C. § 102 (b). The Anderson reference describes the open source lcfg configuration system, which Anderson developed. The chart below sets forth an element-by-element comparison of claims 1-28 of the '289 patent to Anderson. In essence, every element of each claim of the '289 patent was expressly taught or obvious in light of Anderson. As such, each claim of the '289 patent is invalid and should be canceled.

'289 PATENT	**ANDERSON**
1. A method for automatically configuring software on a plurality of computing devices having different respective sets of software and/or configurations of operating parameters, to enable said devices to perform predetermined operations, comprising the steps of:	Anderson was directed to automatically configuring software on a number of different devices having different sets of software and configurations of operating parameters. The Abstract describes configuration of multiple machines: "This paper presents a machine configuration system which stores all configuration parameters in a central ''database''…. A permanent record of every machine configuration is always available …." P. 19. The Anderson Introduction section describes setting configuration parameters: "When a new machine is installed, it will rarely be used with the default configuration supplied by the vendor of the operating system. The partitioning and allocation of space on the disks, the software packages to be carried, and the network name and address are typical *configuration parameters* that will be set differently by different sites and for different machines at the same site." P.19 Anderson teaches the use of "lcfg" on different platforms (from Sun, DEC, HP, and SGI, which would each have different sets of software and/or configurations of operating parameters): "At present, these machines are mostly Suns (currently being upgraded to Solaris 2) and X terminals, but the ability to integrate systems from different vendors is considered very important and DEC, HP, and SGI systems have all previously been integrated into the network…. Only Suns are currently being configured with lcfg, but is intended that the system be portable, presenting a uniform interface to the configuration process across different platforms." PP. 20-21.
storing in a database a model for each different type of device having a different respective set of software and/or configuration of operating parameters,	Anderson describes storing machine configuration information in a central database. This configuration information is the "model" for each different type of device: "All information that is necessary to distinguish one machine from another is contained in the central database." P. 21.

	Anderson further teaches storing roles for machines that will have a different configuration (e.g., name server, member of research group), and performing "high-level" configuration, in which the roles determine the configuration: "Ideally, we would like to describe the relationship between machines at a much higher level and have the low level configuration information generated automatically. For example: • Machine A is the name server for the research group. • Machine B is a member of the research group. • Machine C is a member of the research group. From the above specification, it is possible to generate all the necessary low level configuration information to load the name-server software, and start the name server subsystem, on machine A, and configure the other machines to act as clients of this machine." PP. 22-23.
said model including a description of software components installed on a device and operating parameter values for the software components;	Anderson's stored information includes a description of software components installed on a device: "Storing the machine-specific configuration information explicitly in some external database (for example, `sad[6])` is a major improvement, since the configuration of a particular machine is always clear and the information is always accessible, even when the machine is down." P.20 The Anderson model includes all information that is used to configure a device: "All information that is necessary to distinguish one machine from another is contained in the central database and every machine can be rebuild or duplicated using just the information from the database together with the generic system software." P. 21.
installing an agent on each device that has the ability to manipulate software components installed on the device;	Anderson describes a "script" installed on the device that reads a configuration database and configures the device: "Every time the machine boots, a script reads the configuration database to determine the *subsystem* that should be configured on that machine. This executes a script for each subsystem (for example, DNS or xntp) which consults the database for relevant parameters and dynamically configures the subsystem accordingly." P. 21. The Anderson scripts are later described as running at boot time, manually, or at regular intervals: "Provision is also made to execute these scripts manually, or at regular intervals." P. 22.
and transmitting messages, which contain data from a given one of said models, from said database to agents on only those devices which are associated with said given model, to cause said agents to manipulate operating parameters of software components on said devices in accordance with said data.	Anderson states that a configuration file with the configuration for a machine is provided to that machine using the NIS protocol: "[A]t present, a simple flat file is used for each machine. The resources are distributed and supplied to the client machines using NIS[8]. NIS is not ideal for this purpose, since it involves propagation of the entire database every time a single change is made, and all system software below the level of NIS must be statically configured. We hope to eventually develop a special protocol that operates at a lower level, but NIS is currently proving adequate as a resilient method of supplying machines with the necessary resources." P.21 Anderson also teaches an alternative implementation using a relational database to extract information about groups of machines, and with data only for a particular machine, although he describes it as unnecessary in this implementation: "A large relational database might be a useful tool for extracting information about machine configurations, and making complicated changes to groups of machines…." P. 21

	Based on the configuration file, the devices can then update software and apply patches for the software described in the configuration file for that machine. The processes that are run manipulate the parameters of software components on the devices: "A group of processes run every night to perform any necessary updates to the local file-systems: **Updatelf** uses `lfu` [9] to update the local filesystems with any changes that have been made to the master copies of locally maintained software. The configuration of this subsystem determines the software packages that are to be carried by the machine. **patch** applies any new systems patches that have been installed which are relevant to the machine. **update** makes any necessary modifications to files in the root file-system to track the latest static configuration. Most class scripts also accept additional arguments to stop and restart the subsystem, and to display logging and status information." P.22
2. The method of claim 1, further including the step of modifying a model stored in said database,	Anderson contemplates modifications to the configuration information stored in the central database, for example, adding a new subsystem to the configuration: "New subsystems can therefore be incorporated in the configuration process simply by adding their names to the database entry for a specified machine". P.21.
and sending a message to all devices associated with said model to cause said agents to reconfigure software components in accordance with the change in the model.	Anderson contemplates communicating changes to machines that have been reconfigured: "The dynamic configuration allows machines to be reconfigured very quickly to adapt to changing requirements, or work around failed hardware". P.21. By changing the configuration in the central database, the machines are configured: "The ease with which configurations can be changed, and machines can be completely rebuilt, means that machine configurations do not ''rot'' and are always up-to-date." P. 23.
3. The method of claim 1, wherein said messages are transmitted by means of a gateway that provides an interface between the database and the devices,	Anderson describes the use of NIS protocol, which as a TCP/IP-based protocol, could use a gateway to provide an interface between the database and the devices: "The resources are distributed and supplied to the client machines using NIS[8]." P. 21. NIS (formerly known as Yellow Pages or YP) was developed by Sun Microsystems primarily to reduce the effort required to setup and maintain a network of Unix workstations. The purpose of NIS is to provide a distributed network database. See Hess.
and further including the step of converting messages in said gateway from a first protocol associated with the database to a second protocol employed by said devices.	Anderson also teaches an alternative implementation using a relational database to extract information about groups of machines, and with data only for a particular machine, although he describes it as unnecessary in this implementation: "A large relational database might be a useful tool for extracting information about machine configurations, and making complicated changes to groups of machines…." P. 21. It is inherent in an implementation with a relational database to convert relational database requests (e.g., SQL) to another protocol (e.g., NIS).

	In addition, the conversion of messages to a first protocol to a second protocol within a gateway is inherent in the operation of TCP/IP networks, upon which NIS protocol runs. Moreover, the claim does not state that the first protocol and the second protocol are different protocols. Thus, both the first protocol and the second protocol could be NIS protocol. In addition, Anderson itself teaches the use of a second, low-level protocol: "We hope to eventually develop a special protocol that operates at a lower level…" P. 21.
4. The method of claim 3, wherein said second protocol includes remote procedure calls.	The use of remote procedure calls is inherent in Anderson, which describes the use of NIS: "The resources are distributed and supplied to the client machines using NIS[8]." P. 21. NIS is based upon the Remote Procedure Call (RPC) protocol which uses the External Data Representation (XDR) standard. See Hess. Also, see Winer.
5. The method of claim 4, wherein said second protocol comprises XML-RPC.	XML-RPC had not been developed at the time of Anderson. The use of XML-RPC is a design choice, and it would have been obvious to select XML-RPC as a communications protocol. XML-RPC was published in 1999 as a way to encapsulate RPC in HTTP protocol requests. See Winer. Anderson describes use of NIS (which as described above and in Hess uses RPC/XDR) and the use of a relational database (which would typically communicate using SQL protocol). Once XML-RPC was announced, it would have been obvious use XML-RPC to transmit NIS messages. Also, see Winer.
6. The method of claim 1, further including the step of recognizing a change in configuration in one of said devices, and modifying said model in accordance with the change in configuration.	Anderson describes recognizing installation and configuration of machines: "The static part of the configuration occurs when a machine is installed. Information is read from the database and used to construct auto-install configuration files determining the type of machine, the layout of the disks, the base software configuration, and other static parameters." P. 21. Anderson also describes validating the data on machines: "The ability to validate and examine explicit machine configurations from the database has reduced the number of errors that are caused…" P.23. Anderson also describes modifying the model in accordance with the change in configuration: "When the machine reboots for the first time after an installation, a further script performs any remaining static configuration." P.21. "Every time the machine boots, a script reads the configuration database to determine the subsystems that should be configured on that machine. This executes a script for each subsystem (for example, DNS or xntp) which consults the database for relevant parameters and dynamically configures the subsystem accordingly. New subsystems can therefore be incorporated into the configuration process simply by adding their names to the database entry for a specified machine." P. 21. In addition, Hideo, which was presented at the same conference as Anderson, describes recognizing a change in configuration, the capturing of configuration

	into a model (the Hideo repository), and moving the configuration from one machine to another: "The repository can be placed on a different machine, which means OMNICONF can save and restore a configuration of a machine to and from another machine." P. 30.
7. The method of claim 6, further including the step of sending a message to all other devices of the same type as said one device,	Devices that are similarly configured have the same subsystems, and each subsystem is updated using a class script: "Each configurable subsystem on a machine (for example, a printer) is a member of a particular *class* and the configuration for all subsystems in a class is performed by the same *class script*." P. 22.
which causes the agents in said other devices to reconfigure software components in accordance with the change in the model.	Changes in configuration are then made on the systems: "[A] group of processes runs every night to perform any necessary updates to the local file-systems: **Updatelf** uses `lfu` [9] to update the local filesystems with any changes that have been made to the master copies of locally maintained software. The configuration of this subsystem determines the software packages that are to be carried by the machine. **patch** applies any new systems patches that have been installed which are relevant to the machine. **update** makes any necessary modifications to files in the root file-system to track the latest static configuration.
8. The method of claim 1, further including the step of sending messages from said database to said devices	The messages sent to the devices update the configurations, which are used to update the systems: "The resources are distributed and supplied to the client machines using NIS[8]." P. 21.
which cause said agents in said devices to retrieve software components from a source external to said devices and install said software components on the devices.	Changes in configuration are then made on the systems: "[A] group of processes runs every night to perform any necessary updates to the local file-systems: **Updatelf** uses `lfu` [9] to update the local filesystems with any changes that have been made to the master copies of locally maintained software. The configuration of this subsystem determines the software packages that are to be carried by the machine. **patch** applies any new systems patches that have been installed which are relevant to the machine. **update** makes any necessary modifications to files in the root file-system to track the latest static configuration.
9. The method of claim 8, further including the step of storing said software components in a file system, wherein said components are classified into multiple roles which respectively contain different categories of software.	Anderson describes different roles for each machine (and its associated software): "One of the most important aspects of machine configuration is to specify the role of a machine within the network….Typically these will include file services of various types (home directories, program binaries), name service (DNS), time synchronization (xntp), font service, and others." P. 22. In Anderson, the different machine roles will be implmented by subsystems of software components: "Each configurable subsystem on a machine (for example, a printer) is a member of a particular *class* and the configuration for all subsystems in a class is performed by the same *class script*." P. 22. Subsystems on different systems can be controlled together: "This provides a

	central configuration database and a ‘‘framework’’ into which objects can be slotted to control the various subsystems in a uniform way.” P.20. In Anderson, master copies of software are stored on another machine, and accessed using updatelf: **Updatelf** uses `lfu` [9] to update the local filesystems with any changes that have been made to the master copies of locally maintained software. The configuration of this subsystem determines the software packages that are to be carried by the machine. P. 21.
10. The method of claim 9, wherein the categories of software are determined in accordance with the probable frequency with which their respective components are likely to be changed during the service lifetime of a device.	By dividing the software components into subsystems, Anderson inherently stores the components in accordance with the probably frequency with which their respective components are likely to be changed during the service lifetime of the device. In addition, Anderson describes different update timing frequency for different subsystems: “The above subsystems run only when the machine boots, and any change in the database resources is not reflected in the corresponding subsystem until the machine is rebooted (or the subsystem is manually restarted). These are mostly one-off configurations (such as auth) or daemons which start once and run continuously (such as www and xdm). Some subsystems need to be run at regular intervals (for example, backups) and the boot subsystem can arrange to schedule these to run from cron. In particular, a group of processes runs every night to perform any necessary updates to the local file system[.]” P. 22.
11. The method of claim 9, wherein the model of a device is stored in said database as one set of software components from each of said multiple roles.	The different subsystems in Anderson will include the operating system software for a particular device, the various application system software, and data content (e.g., machine name, and other configuration information.) For example, the subsystems, which each have associated classes in the boot script, include “auth,” and “amd,” which perform operating system configuration, “dns” which includes application configuration and data content, and “www,” which is an application: “**auth** configures al the authorisation of access to the machine. This controls, for example, the groups of users that are permitted to log in, and the machines to be included in hosts.equiv file. **amd** controls the `amd` automounter, specifying the cluster that is to be used and hence determining the servers from which the various file-systems will be mounted. **dns** controls the type of DNS service to be provided and (where appropriate) specifies the servers to be used. **www** controls the World Wide Web server.” P. 22.
12. The method of claim 11, wherein one of said roles includes operating system software for the devices.	See the discussion of claim 11. For example, the “auth” class is for operating system configuration.
13. The method of claim 12, wherein another of said roles includes application programs for said devices.	See the discussion of claim 11 above. For example, the “www” class is for configuring an application program.
14. The method of claim 12, wherein another of said roles	See the discussion of claim 11 above. For example, the “dns” class will use data content such as the dns servers to be used.

includes data content associated with the devices.	
15. The method of claim 1, wherein the step of transmitting messages comprises the steps of storing commands in a queue in said database, sending a first message containing the first command in said queue, awaiting a report from a device that the first message has been executed, and sending the next command in the queue in response to said report.	These steps are inherent in the transmission of messages in Anderson, because sending a first message, awaiting an acknowledgement, sending the next command, would be implemented by any TCP-based protocol, and in particular, any RPC protocol, including NIS: "The resources are distributed and supplied to the client machines using NIS[8]." P. 21.
16. The method of claim 1, wherein said agents have a level of authority that enables them to manipulate operating system software installed on said devices.	Anderson teaches that the configuration scripts can access the root file system, which is the level of authority required to manipulate operating system software: "**update** makes any necessary modifications to files in the root file-system to track the latest static configuration." P. 22.
17. A method for automatically installing software components on a plurality of computing devices having different respective sets of software, comprising the steps of:	Anderson was directed to automatically configuring software on a number of different devices having different sets of software and configurations of operating parameters. The Abstract describes configuration of multiple machines: "This paper presents a machine configuration system which stores all configuration parameters in a central ''database''…. A permanent record of every machine configuration is always available …." P. 19. The Anderson Introduction section describes setting configuration parameters: "When a new machine is installed, it will rarely be used with the default configuration supplied by the vendor of the operating system. The partitioning and allocation of space on the disks, the software packages to be carried, and the network name and address are typical *configuration parameters* that will be set differently by different sites and for different machines at the same site." P.19 Anderson teaches the use of "lcfg" on different platforms (from Sun, DEC, HP, and SGI, which would each have different sets of software and/or configurations of operating parameters): "At present, these machines are mostly Suns (currently being upgraded to Solaris 2) and X terminals, but the ability to integrate systems from different vendors is considered very important and DEC, HP, and SGI systems have all previously been integrated into the network…. Only Suns are currently being configured with lcfg, but is intended that the system be portable, presenting a uniform interface to the configuration process across different platforms." PP. 20-21.
storing in a database a model for each different type of device having a different respective set of software,	Anderson describes storing machine configuration information in a central database. This configuration information is the "model" for each different type of device: "All information that is necessary to distinguish one machine from another is contained in the central database." P. 21

	Anderson further teaches storing roles for machines that will have a different configuration (e.g., name server, member of research group), and performing "high-level" configuration, in which the roles determine the configuration: "Ideally, we would like to describe the relationship between machines at a much higher level and have the low level configuration information generated automatically. For example: • Machine A is the name server for the research group. • Machine B is a member of the research group. • Machine C is a member of the research group. From the above specification, it is possible to generate all the necessary low level configuration information to load the name-server software, and start the name server subsystem, on machine A, and configure the other machines to act as clients of this machine." PP. 22-23.
said model including a description of software components installed on a device;	Anderson's stored information includes a description of software components installed on a device: "Storing the machine-specific configuration information explicitly in some external database (for example, `sad[6])` is a major improvement, since the configuration of a particular machine is always clear and the information is always accessible, even when the machine is down." P.20 The Anderson model includes all information that is used to configuration a device: "All information that is necessary to distinguish one machine from another is contained in the central database and every machine can be rebuild or duplicated using just the information from the database together with the generic system software." P. 21
installing an agent on each device that has the ability to install and delete other software components on said device;	Anderson describes a "script" installed on the device that reads a configuration database and configures the device: "Every time the machine boots, a script reads the configuration database to determine the *subsystem* that should be configured on that machine. This executes a script for each subsystem (for example, DNS or xntp) which consults the database for relevant parameters and dynamically configures the subsystem accordingly." P. 21 The Anderson scripts are later described as running at boot time, manually, or at regular intervals: "Provision is also made to execute these scripts manually, or at regular intervals." P. 22
and transmitting messages, which contain data from a given one of said models, from said database to agents on only those devices which are associated with said given model, to cause said agents to retrieve software components from a source external to said devices and install said software components on the devices.	Anderson states that a configuration file with the configuration for a machine is provided to that machine using the NIS protocol: "[A]t present, a simple flat file is used for each machine. The resources are distributed and supplied to the client machines using NIS[8]. NIS is not ideal for this purpose, since it involves propagation of the entire database every time a single change is made, and all system software below the level of NIS must be statically configured. We hope to eventually develop a special protocol that operates at a lower level, but NIS is currently proving adequate as a resilient method of supplying machines with the necessary resources." P.21

	Anderson also teaches an alternative implementation using a relational database to extract information about groups of machines, and with data only for a particular machine, although he describes it as unnecessary in this implementation: "A large relational database might be a useful tool for extracting information about machine configurations, and making complicated changes to groups of machines…." P. 21 Based on the configuration file, the devices can then update software and apply patches for the software described in the configuration file for that machine. The "master copy" of locally maintained software are external to the devices: "A group of processes run every night to perform any necessary updates to the local file-systems: **Updatelf** uses `lfu` [9] to update the local filesystems with any changes that have been made to the master copies of locally maintained software. The configuration of this subsystem determines the software packages that are to be carried by the machine. **patch** applies any new systems patches that have been installed which are relevant to the machine. **update** makes any necessary modifications to files in the root file-system to track the latest static configuration. Most class scripts also accept additional arguments to stop and restart the subsystem, and to display logging and status information." P.22
18. The method of claim 17, wherein said messages are transmitted by means of a gateway that provides an interface between the database and the devices,	Claim 18 is identical to claim 3. See the discussion of claim 3 above.
and further including the step of converting messages in said gateway from a first protocol associated with the database to a second protocol employed by said devices.	Claim 18 is identical to claim 3. See the discussion of claim 3 above.
19. The method of claim 18, wherein said second protocol includes remote procedure calls.	Claim 19 is identical to claim 4. See the discussion of claim 4 above.
20. The method of claim 19, wherein said second protocol comprises XML-RPC.	Claim 20 is identical to claim 5. See the discussion of claim 5 above.
21. The method of claim 17, further including the step of storing said software components in a file system,	Claim 21 is identical to claim 9. See the discussion of claim 9 above.

wherein said components are classified into multiple roles which respectively contain different categories of software.	
22. The method of claim 21, wherein the categories of software are determined in accordance with the probable frequency with which their respective components are likely to be changed during the service lifetime of a device.	Claim 22 is identical to claim 10. See the discussion of claim 10 above.
23. The method of claim 21, wherein the model of a device is stored in said database as one set of software components from each of said multiple roles.	Claim 23 is identical to claim 11. See the discussion of claim 11 above.
24. The method of claim 23, wherein one of said roles includes operating system software for the devices.	Claim 24 is identical to claim 12. See the discussion of claim 12 above.
25. The method of claim 24, wherein another of said roles includes application programs for said devices.	Claim 25 is identical to claim 13. See the discussion of claim 13 above.
26. The method of claim 24, wherein another of said roles includes data content associated with the devices.	Claim 26 is identical to claim 14. See the discussion of claim 14 above.
27. The method of claim 17, wherein the step of transmitting messages comprises the steps of storing commands in a queue in said database, sending a first message containing the first command in said queue, awaiting a report from a device that the first message has been executed, and sending the next command in the queue in response to said report.	Claim 27 is identical to claim 15. See the discussion of claim 15 above.
28. The method of claim 17, wherein each agent has a level of authority that enables it to manipulate operating system software installed on said devices.	Claim 28 is identical to claim 16. See the discussion of claim 16 above.

SECOND SUBSTANTIAL NEW QUESTION OF PATENTABILITY: FISK ANTICIPATES THE '289 PATENT

The '289 patent's application date is October 31, 2000. Fisk was published in 1996, more than one year before the '289 patent's application date. Fisk is prior art to the '289 patent under 35 U.S.C. § 102 (b). The Fisk reference describes the system that Michael Fisk developed to automate the administration of computers on heterogeneous networks. The chart below sets forth an element-by-element comparison of claims 1-28 of the '289 patent to Fisk. In essence, every element of each claim of the '289 patent was expressly taught or obvious in light of Fisk. As such, each claim of the '289 patent is invalid and should be canceled.

'289 PATENT	**FISK**
1. A method for automatically configuring software on a plurality of computing devices having different respective sets of software and/or configurations of operating parameters, to enable said devices to perform predetermined operations, comprising the steps of:	Fisk was directed to automatically configuring software on a number of different devices having different sets of software and configurations of operating parameters. In the Abstract, Fisk describes configuration and software installation on multiple machines: "The areas of machine configuration and software package installation and maintenance have been frequent areas of work in recent years. This paper describes a hybrid system developed to address both problems and more. The resulting system is designed to reduce the complexity of the administration of a large network of computers down to that of the administration of a few heterogeneous systems." P. 181.
storing in a database a model for each different type of device having a different respective set of software and/or configuration of operating parameters,	Fisk has a machine database called machdb, which "defines all machine-specific variables." P. 183. "Each object can contain any number of of variable/value pairs and can inherit from any number of other objects, recursively. This technique allows a definition for a machine to consist of only a few unique values such as IP and MAC address. All other values can be inherited from other objects." P. 183.
said model including a description of software components installed on a device and operating parameter values for the software components;	"Each object can contain any number of of variable/value pairs and can inherit from any number of other objects, recursively. This technique allows a definition for a machine to consist of only a few unique values such as IP and MAC address. All other values can be inherited from other objects." P. 183.
installing an agent on each device that has the ability to manipulate software components installed on the device;	The agent software is called gutinteg: "The main program, affectionately called *gutinteg*, is also written in Perl." P. 184. "The program probes a machine for its MAC addresses using `ifconfig` and `dmesg`. The *Machdb* entry for that MAC address is then loaded. *Packagelink* is then called to install software on the machine." P. 184
and transmitting messages, which contain data from a given one of said models, from said database to agents on only those	Fisk describes propagating the configuration data to the client devices: "Changes to configuration information should be made on a central server and then propagated to the client. This is most robustly done by pushing information to all reachable clients and by having all clients check the servers

devices which are associated with said given model, to cause said agents to manipulate operating parameters of software components on said devices in accordance with said data.	periodically or at boot time." P. 182 Fisk describes devices making the changes based on the configuration data: "The program configures an already running machine by performing a partial install…. The program probes a machine for its MAC addresses using `ifconfig` and `dmesg`. The *Machdb* entry for that MAC address is then loaded. *Packagelink* is then called to install software on the machine." P. 184.
2. The method of claim 1, further including the step of modifying a model stored in said database,	Fisk describes making changes to the configuration information: "Changes to configuration information should be made on a central server and then propagated to the client." P. 182
and sending a message to all devices associated with said model to cause said agents to reconfigure software components in accordance with the change in the model.	Fisk describes pushing information to the devices: "This is most robustly done by pushing information to all reachable clients and by having all clients check the servers periodically or at boot time." P. 182.
3. The method of claim 1, wherein said messages are transmitted by means of a gateway that provides an interface between the database and the devices,	Fisk describes the use of TCP/IP protocol, for example in the appendix, where the systems are described as having TCP/IP addresses. All TCP/IP-based protocols could use a gateway to provide an interface between the database and the devices. Thus, the possible use of a gateway is implicit in Fisk.
and further including the step of converting messages in said gateway from a first protocol associated with the database to a second protocol employed by said devices.	All gateways between two devices would have a first protocol associated with a first device and a second protocol associated with a second device.
4. The method of claim 3, wherein said second protocol includes remote procedure calls.	The use of remote procedure calls is a design choice, which would be obvious from the discussion in Fisk. P. 182 The '289 patent does not claim any particular benefits from the use of RPC. Also see Winer.
5. The method of claim 4, wherein said second protocol comprises XML-RPC.	The use of XML-RPC is a design choice, which would be obvious from the discussion in Fisk. P. 182 The '289 patent does not claim any particular benefits from the use of RPC. Also see Winer.
6. The method of claim 1, further including the step of recognizing a change in configuration in one of said devices, and modifying said model in accordance with the change in configuration.	Fisk defines objects for different machines: "Objects can be defined for any logical group of machines that share some configuration information." P. 183. Fisk describes making changes to the configuration information: "Changes to configuration information should be made on a central server and then propagated to the client." P. 182 In addition, Fisk refers to Hideo's OMNICONF, in which "changes are supposed to be made on a sample workstation and the differences stored for applications to other machines." P. 182. Thus, it would have been obvious from Fisk to add the functionality of Hideo, in order to recognize a change in configuration in one of said devices, and include the change in the Fisk model. Also see Hideo.
7. The method of claim 6,	Fisk describes propagating the configuration data to the client devices:

further including the step of sending a message to all other devices of the same type as said one device,	"Changes to configuration information should be made on a central server and then propagated to the client. This is most robustly done by pushing information to all reachable clients and by having all clients check the servers periodically or at boot time." P. 182
which causes the agents in said other devices to reconfigure software components in accordance with the change in the model.	Fisk describes devices making the changes based on the configuration data: "The program configures an already running machine by performing a partial install…. The program probes a machine for its MAC addresses using `ifconfig` and `dmesg`. The *Machdb* entry for that MAC address is then loaded. *Packagelink* is then called to install software on the machine." P. 184.
8. The method of claim 1, further including the step of sending messages from said database to said devices	Fisk describes devices making the changes based on the configuration data: "The program configures an already running machine by performing a partial install…. The program probes a machine for its MAC addresses using `ifconfig` and `dmesg`. The *Machdb* entry for that MAC address is then loaded. *Packagelink* is then called to install software on the machine." P. 184.
which cause said agents in said devices to retrieve software components from a source external to said devices and install said software components on the devices.	Fisk describes the Packagelink program, which copies files from a remote filesystem located an installation server that is external to the client: "In the last two years, [Packagelink] has been expanded to build mirror filesystems by building the target filesystem on a different machine than the packages reside on and copying files instead of linking them….This method of operation is used to build / on a machine from a set of packages on the installation server." P. 183
9. The method of claim 8, further including the step of storing said software components in a file system, wherein said components are classified into multiple roles which respectively contain different categories of software.	Fisk describes the storage of the files in packages: "Therefore, the system is built around a set of separate packages. Each package is the files necessary for some logically discreet functionality." P. 182.
10. The method of claim 9, wherein the categories of software are determined in accordance with the probable frequency with which their respective components are likely to be changed during the service lifetime of a device.	By organizing the software components into packages, Anderson inherently stores the components in accordance with the probably frequency with which their respective components are likely to be changed during the service lifetime of the device. Fisk's examples of packages are "X11R6, Emacs, and INN." P. 182.
11. The method of claim 9, wherein the model of a device is stored in said database as one set of software components from each of said multiple roles.	In Fisk, each of the packages for a device are found on the installation server: "This method of operation is used to build / on a machine from a set of packages on the installation server." P. 183.
12. The method of claim 11, wherein one of said roles includes operating system software for the devices.	Fisk's examples of packages are "X11R6, Emacs, and INN." P. 182. X11R6 is an operating system component. Fisk also states: "we have two different versions of the Slackware package." P. 183. Slackware is a distribution of the Linux operating system.
13. The method of claim 12, wherein another of said roles includes application programs for said devices.	Fisk's examples of packages are "X11R6, Emacs, and INN." P. 182. Emacs is an example of an application program.
14. The method of claim 12,	According to Fisk, "a machine's configuration can be completely represented

wherein another of said roles includes data content associated with the devices.	as the union of the correct set of files." P. 182. Fisk provides data content associated with devices, such as DNS files: "in addition, we sensed the opportunity to use the same infrastructure to provide the following functions that had also been identified as tasks that were overly troublesome or redundant. "Maintain legal DNS files." P. 181.
15. The method of claim 1, wherein the step of transmitting messages comprises the steps of storing commands in a queue in said database, sending a first message containing the first command in said queue, awaiting a report from a device that the first message has been executed, and sending the next command in the queue in response to said report.	These steps are inherent in the transmission of messages in Fisk, because sending a first message, awaiting an acknowledgement, sending the next command, would be implemented by any TCP-based protocol. In addition, Fisk explicitly describes prior art systems in which, "All operations must be saved and applied to machines to insure proper configuration." P. 182.
16. The method of claim 1, wherein said agents have a level of authority that enables them to manipulate operating system software installed on said devices.	Fisk describes using the system to install and upgrade operating system software: "After partitioning and formatting disks, Gutinteg installs the appropriate operating system package(s) and proceeds with the steps of a partial installation." P. 184. Fisk's examples of packages are "X11R6, Emacs, and INN." P. 182. X11R6 is an operating system component. Fisk also states: "we have two different versions of the Slackware package." P. 183. Slackware is a distribution of the Linux operating system.
17. A method for automatically installing software components on a plurality of computing devices having different respective sets of software, comprising the steps of:	Fisk was directed to automatically configuring software on a number of different devices having different sets of software and configurations of operating parameters. In the Abstract, Fisk describes configuration and software installation on multiple machines: "The areas of machine configuration and software package installation and maintenance have been frequent areas of work in recent years. This paper describes a hybrid system developed to address both problems and more. The resulting system is designed to reduce the complexity of the administration of a large network of computers down to that of the administration of a few heterogeneous systems." P. 181.
storing in a database a model for each different type of device having a different respective set of software,	Fisk has a machine database called machdb, which "defines all machine-specific variables." P. 183. "Each object can contain any number of of variable/value pairs and can inherit from any number of other objects, recursively. This technique allows a definition for a machine to consist of only a few unique values such as IP and MAC address. All other values can be inherited from other objects." P. 183.
said model including a description of software components installed on a device;	The database includes a description of the software components associated with a device: "Each object can contain any number of of variable/value pairs and can inherit from any number of other objects, recursively. This technique allows a definition for a machine to consist of only a few unique values such as IP and MAC address. All other values can be inherited from other objects." P. 183.

installing an agent on each device that has the ability to install and delete other software components on said device;	The agent software is called gutinteg: "The main program, affectionately called *gutinteg*, is also written in Perl." P. 184. "The program probes a machine for its MAC addresses using `ifconfig` and `dmesg`. The *Machdb* entry for that MAC address is then loaded. *Packagelink* is then called to install software on the machine." P. 184
and transmitting messages, which contain data from a given one of said models, from said database to agents on only those devices which are associated with said given model, to cause said agents to retrieve software components from a source external to said devices and install said software components on the devices.	Fisk describes propagating the configuration data to the client devices: "Changes to configuration information should be made on a central server and then propagated to the client. This is most robustly done by pushing information to all reachable clients and by having all clients check the servers periodically or at boot time." P. 182 Fisk describes devices making the changes based on the configuration data: "The program configures an already running machine by performing a partial install…. The program probes a machine for its MAC addresses using `ifconfig` and `dmesg`. The *Machdb* entry for that MAC address is then loaded. *Packagelink* is then called to install software on the machine." P. 184.
18. The method of claim 17, wherein said messages are transmitted by means of a gateway that provides an interface between the database and the devices,	Claim 18 is identical to claim 3. See the discussion of claim 3 above.
and further including the step of converting messages in said gateway from a first protocol associated with the database to a second protocol employed by said devices.	Claim 18 is identical to claim 3. See the discussion of claim 3 above.
19. The method of claim 18, wherein said second protocol includes remote procedure calls.	Claim 19 is identical to claim 4. See the discussion of claim 4 above.
20. The method of claim 19, wherein said second protocol comprises XML-RPC.	Claim 20 is identical to claim 5. See the discussion of claim 5 above.
21. The method of claim 17, further including the step of storing said software components in a file system, wherein said components are classified into multiple roles which respectively contain different categories of software.	Claim 21 is identical to claim 9. See the discussion of claim 9 above.
22. The method of claim 21, wherein the categories of software are determined in accordance with the probable frequency with which their respective components are likely to be changed during the service lifetime of a device.	Claim 22 is identical to claim 10. See the discussion of claim 10 above.
23. The method of claim 21,	Claim 23 is identical to claim 11. See the discussion of claim 11 above.

wherein the model of a device is stored in said database as one set of software components from each of said multiple roles.	
24. The method of claim 23, wherein one of said roles includes operating system software for the devices.	Claim 24 is identical to claim 12. See the discussion of claim 12 above.
25. The method of claim 24, wherein another of said roles includes application programs for said devices.	Claim 25 is identical to claim 13. See the discussion of claim 13 above.
26. The method of claim 24, wherein another of said roles includes data content associated with the devices.	Claim 26 is identical to claim 14. See the discussion of claim 14 above.
27. The method of claim 17, wherein the step of transmitting messages comprises the steps of storing commands in a queue in said database, sending a first message containing the first command in said queue, awaiting a report from a device that the first message has been executed, and sending the next command in the queue in response to said report.	Claim 27 is identical to claim 15. See the discussion of claim 15 above.
28. The method of claim 17, wherein each agent has a level of authority that enables it to manipulate operating system software installed on said devices.	Claim 28 is identical to claim 16. See the discussion of claim 16 above.

THIRD SUBSTANTIAL NEW QUESTION OF PATENTABILITY:
THE '289 PATENT IS OBVIOUS OVER
FLETCHER AND COLLINS IN LIGHT OF ANDERSON

The '289 patent's application date is October 31, 2000. Anderson was published in 1994, more than one year before the '289 patent's application date. Anderson is prior art to the '289 patent under 35 U.S.C. § 102(b). Fletcher was filed on June 24, 1997, prior to the filing date for the '289 patent. Thus, Fletcher is prior art under 35 U.S.C. § 102(e). Collins was filed on December 18, 1997, prior to the filing date for the '289 patent. Thus, Collins is prior art under 35 U.S.C. § 102(e).

On December 8, 2004, in the course of prosecution of the '289 patent, an Office Action was mailed from the USPTO in which claims 1-14, 16-26, and 28 were rejected as anticipated by Fletcher and claims 15 and 27 were rejected as obvious over Fletcher in view of Collins. In response, in the Amendment mailed on May 9, 2005, the applicant amended claim 1 to recite that the computing devices "hav[e] different respective sets of software and/or configurations of operating parameters," that a model is stored "for each different type of device having a different set of software and/or configuration of operating parameters," and transmitting messages "which contain data from a given one of the models" to agents on "only those devices which are associated with said given model." Likewise, claim 17 was amended to recite computing devices "having different respective sets of software," storing "in a database" a model for each "different" type of device "having a different respective set of software," and transmitting messages "which contain data from a given one of said models," from said database to said agents "on only those devices which are associated with said given model."

In the Amendment mailed on May 9, 2005, applicant's counsel argued that Fletcher "describes a system that is designed for a homogeneous, or nearly homogeneous set of end systems," and that because Fletcher "uses multicast advertisements to identify the latest version of available software components," that Fletcher is not practical "in a heterogeneous environment where different devices may be provisioned with entirely different sets of software." Amendment, page 12. Applicant's counsel further argued that Fletcher had "no disclosure of different devices having different respective configurations for software and/or operating parameters, let alone the

storage of a respective model for reach in a database." Amendment, P. 13 Applicants counsel further argued, with reference to claim 1, that Fletcher "does not disclose the storage of a model containing operating parameter values for software components, nor the use of such data by the agents to manipulate operating parameters of software components on the devices." Amendment, P. 13. Following the Amendment, the Examiner withdrew the rejection, and cited other references.

The Examiner, however, did not have the benefit of Anderson. Anderson, as described above, describes a system that implements both software provisioning and operating parameter value configuration in a heterogeneous environment. Anderson communicates different model information to different devices. Where Fletcher took a multicast approach, Anderson uses the NIS protocol to share model data with individual devices. At the time of the '289 patent, it would have been trivial to one skilled in the art to modify Fletcher so as to individually communicate a different model to different devices. The cost/benefit tradeoff of multicasting versus transmitting individually to each device would have been obvious to one skilled in the art who was aware of the Anderson approach, and no other changes in the operation of Fletcher's system would have been required to reach the invention claimed in the '289 patent.

Fletcher mentions, in its discussion of prior art update systems, products like Intel's LANdesk, which Fletcher states "use a 'push' technology for updating that forces updating at the end system, and which requires large database management resources to keep track of update statistics such as which end-station received which update, and which end-station still needs to be updated with which components, for example." Fletcher, col. 4, lines 35-41. Thus, one skilled in the art may have understood from Fletcher alone that it would be possible to modify Fletcher to operate in a fashion that had a centralized database and that still had the other benefits of Fletcher, albeit that Fletcher thought that the multicast approach was better. In view of Anderson, which describes a system that takes the individual communication approach, and discusses use of a relational database, it would have been clear to one skilled in the art that Fletcher could be modified in such a manner.

It also should be noted, for the record, that Fletcher does contemplate a heterogeneous network. The network described in Fletcher includes a variety of end system (ES) devices: "The

ESs may be familiar end-user data processing equipment such as personal computers, workstations, and printers and additionally may be digital devices such as digital telephones or real-time video displays. Different types of ESs can operate together on the same LAN." Fletcher, col. 1, lines 54-59. Fletcher also specifically states: "the method and apparatus of the present invention may operate with a wide variety of types of network devices including networks dramatically different from the specific examples illustrated in FIG. 1 and described below." Fletcher, col. 4, lines 65-66. The difference alleged by the applicant of the '289 patent during prosecution was that Fletcher does not communicate different model information to different types of devices. This alleged shortcoming, to the extent it wasn't already obvious from Fletcher itself, is shown in Anderson, as described above. The remaining rejections in the Office Action mailed December 8, 2004 ("OA-2004") therefore should stand in light of Anderson.

The chart below sets forth an element-by-element comparison of claims 1-14, 16-26, and 28 of the '289 patent to Fletcher in light of Anderson, and claims 15 and 17 of the '289 patent to Fletcher in light of Anderson and in light of Collins. Every element of each claim of the '289 patent was expressly taught by or is obvious in light of these references. As such, each claim of the '289 patent is invalid and should be canceled.

'289 PATENT	**FLETCHER, ANDERSON, AND COLLINS**
1. A method for automatically configuring software on a plurality of computing devices having different respective sets of software and/or configurations of operating parameters, to enable said devices to perform predetermined operations, comprising the steps of:	The Dec. 2004 Office Action stated that Fletcher discloses a method and apparatus for automatically configuring software to enable said devices to perform predetermined operations. This is correct. Fletcher states: "The present invention is a method and apparatus for automatic software updating (ASU) in a LAN." Col. 5, Lines 6-8. Fletcher also states: "the method and apparatus of the present invention may operate with a wide variety of types of network devices." Col. 4, lines 65-66. As described above, Anderson was directed to automatically configuring software on a number of different devices having different sets of software and configurations of operating parameters.
storing in a database a model for each different type of device having a different respective set of software and/or configuration of operating parameters,	The Dec. 2004 Office Action stated that Fletcher discloses storing a model for each type of device in a database [ASU server]. Fletcher describes each agent (which may be on each different type of device) periodically sending its current version information to the ASU server: "on an intermittent basis, possibly initiated by a polling packet from the ASU server, the ASU agents forward current version information regarding a subset or all of their software components to an ASU server…." Col. 5, lines 17-19. "According to an embodiment of the invention, an agent response to an ASU

	server request is defined to indicate the current version level of all software components in the ES." Col. 9, lines 17-19. The ASU server stores the responses as a model of what the device has and needs in a database: "The ASU server receives update requests from the agents and sorts and aggregates that information into a cohesive database." Col. 7, lines 18-20. Fletcher updates operating parameters at the time that updates are installed: "For ASU components, proper Windows 95 registry entries are modified to reflect Auto update status at the time that files are copied." Col. 12, lines 62-64. As mentioned above, Anderson describes storing machine configuration information in a central database. This configuration information is the "model" for each different type of device: "All information that is necessary to distinguish one machine from another is contained in the central database." P. 21 This information includes machine-specific configuration as well as software installation and update information.
said model including a description of software components installed on a device and operating parameter values for the software components;	Fletcher stores the versions that are available, and what is needed by each device: "the ASU server receives requests from each agent and stores the requests in table form, for example, with each file defining a column entry and each requesting agent defining a row entry." Col. 11, lines 48-51. As mentioned above, Anderson's stored information includes a description of software components installed on a device: "Storing the machine-specific configuration information explicitly in some external database (for example, `sad[6])` is a major improvement, since the configuration of a particular machine is always clear and the information is always accessible, even when the machine is down." P.20
installing an agent on each device that has the ability to manipulate software components installed on the device;	Fletcher has an agent installed on the device: "[t]he invention includes two types of primary components, the agents that reside in ESs [end systems] and the ASU server…" Col. 7, lines 1-3. The agents update the ES devices: "new files received by an ASU agent (at an end system) are stored in one or more special update directories. These files are copied to their respective directories when all requested files have been received." Col. 12, lines 58-62. As mentioned above, Anderson describes a "script" running on the device that reads a configuration database and configures the device. "Every time the machine boots, a script reads the configuration database to determine the *subsystem* that should be configured on that machine. This executes a script for each subsystem (for example, DNS or xntp) which consults the database for relevant parameters and dynamically configures the subsystem accordingly." P. 21 The Anderson scripts are later described as running at boot time, manually, or at regular intervals: "Provision is also made to execute these scripts manually, or at regular intervals." P. 22
and transmitting messages, which contain data from a given one of said models, from said database to agents on only those devices which are associated with said given model, to cause said agents to manipulate	Fletcher describes sending the update files to each agent that needs a particular update on an agent-by-agent basis, or concurrently to multiple agents that need the same files, but in any event to only those devices that need the update: "For example, the ASU server accesses the first column (file) of the request table and sends that file in a point-to-point manner to each agent requesting that file. The ASU server then proceeds to the next column (file) that has at least one agent requesting that file and sends out that file to the

operating parameters of software components on said devices in accordance with said data.	requesting agents." Col. 11, line 67 – Col. 12, line 2. The Fletcher agents update the ES devices: "new files received by an ASU agent (at an end system) are stored in one or more special update directories. These files are copied to their respective directories when all requested files have been received." Col. 12, lines 58-62. Fletcher updates operating parameters at the time that updates are installed: "For ASU components, proper Windows 95 registry entries are modified to reflect Auto update status at the time that files are copied." Col. 12, lines 62-64. As mentioned above, Anderson states that a configuration file with the configuration for a machine is provided to that machine using the NIS protocol: "The resources are distributed and supplied to the client machines using NIS[8]." P.21. Based on the configuration file, the devices can then update software and apply patches for the software described in the configuration file for that machine, as described with respect to updatelf, patch, and update. P. 22.
2. The method of claim 1, further including the step of modifying a model stored in said database,	Updated versions of files are provided to the ASU Server by the ASU Manager: " ASU Mgr. allows a user to input and control the files to be updated from an ASU server to the ASU agents. The files are provided to the ASU Manager by a user, and the ASU manager, in turn, uses FTP (or some other file transfer protocol) to transfer these files to ASU servers." Col. 9, line 67 - Col. 10, line 4.
and sending a message to all devices associated with said model to cause said agents to reconfigure software components in accordance with the change in the model.	Fletcher states that once the ASU Server has the updates, they are communicated to the agents in due course: "Once the ASU server receives the files, the ASU server may transfer the files to the ASU agents." Col. 10, lines 5-6.
3. The method of claim 1, wherein said messages are transmitted by means of a gateway that provides an interface between the database and the devices,	In the Dec. 2004 Office Action, the Examiner stated that Fletcher discloses messages transmission from server to devices in LAN and WAN networking applications "by means of a gateway [63, bridge] that provides an interface between the database [64, server] and the devices [50-52 a to d]." Para. 8. These are further described in Fletcher as intermediate systems: "LAN intermediate systems 60-63 are referred to as bridges or switches or hubs…" Col. 1, lines 59-62.
and further including the step of converting messages in said gateway from a first protocol associated with the database to a second protocol employed by said devices.	In the Dec. 2004 Office Action, the Examiner stated that Fletcher "discloses the step of converting messages in said gateway from a first protocol associated with the database to a second protocol employed by said devices." Para. 8. The specification describes the various protocols that are used on a network at Col. 1, line 65 – Col. 2, line 15. In general, it is inherent for gateways in a network such as shown in Fletcher FIG. 1 to convert packets from one protocol to another.
4. The method of claim 3, wherein said second protocol includes remote procedure calls.	With respect to claims 4 and 5, the Dec. 2004 Office Action, the Examiner stated that Fletcher "discloses the use of standardized communication protocols [SNMP] for messaging between a first process means [ASU server] and second process means [ES] with remote procedure calls [RMON, remote monitoring and managing]." Para. 9.
5. The method of claim 4, wherein said second protocol comprises XML-RPC.	XML-RPC had not been developed at the time of Fletcher. It would have been obvious, however, when XML-RPC was published, to use it with or instead of the standard protocols described in Fletcher.

6. The method of claim 1, further including the step of recognizing a change in configuration in one of said devices,	Updated versions of files are provided to the ASU Server by the ASU Manager: " ASU Mgr. allows a user to input and control the files to be updated from an ASU server to the ASU agents. The files are provided to the ASU Manager by a user, and the ASU manager, in turn, uses FTP (or some other file transfer protocol) to transfer these files to ASU servers." Col. 9, line 67 - Col. 10, line 4.
and modifying said model in accordance with the change in configuration.	Fletcher states that once the ASU Server has the updates, they are communicated to the agents in due course: "Once the ASU server receives the files, the ASU server may transfer the files to the ASU agents." Col. 10, lines 5-6.
7. The method of claim 6, further including the step of sending a message to all other devices of the same type as said one device,	Fletcher describes sending the update files to each agent that needs a particular update on an agent-by-agent basis, or concurrently to multiple agents that need the same files, but in any event to only those devices that need the update: "For example, the ASU server accesses the first column (file) of the request table and sends that file in a point-to-point manner to each agent requesting that file. The ASU server then proceeds to the next column (file) that has at least one agent requesting that file and sends out that file to the requesting agents." Col. 11, line 67 – Col. 12, line 2. Devices of the same type will have the same files, and so will be updated.
which causes the agents in said other devices to reconfigure software components in accordance with the change in the model.	Fletcher states that once the ASU Server has the updates, they are communicated to the agents in due course: "Once the ASU server receives the files, the ASU server may transfer the files to the ASU agents." Col. 10, lines 5-6.
8. The method of claim 1, further including the step of sending messages from said database to said devices	The Dec. 2004 Office Action stated that "As to claim 8, Fletcher teaches the step of sending messages [multicast requests, advertisemnt, broadcast] from said database [ASU server] to said devices [50, 51, 52] which cause said agents in said devices to retrieve software components [request upgrade software component] from a source external [ASU server] to said devices and install said software components on the devices [by ASU agent] [col. 9, lines 47-52, col. 10, lines 19-25, fig. 1]" Para. 12. The Fletcher ASU server sends out messages to the devices: "The ASU server uses multicast requests, or advertisements, to identify the latest version levels for one or more components that are currently available." Col. 10, lines 19-21.
which cause said agents in said devices to retrieve software components from a source external to said devices and install said software components on the devices.	The Fletcher agents send messages which result in the receipt of components from the ASU server: "an agent receives the broadcast information and responds with the agent's current version level of one or more software components that were advertised by the ASU server. The ASU server then compares the agent's current version levels with the latest version levels of the software components." Col. 11, lines 1-6. The ASU server then sends out the needed files: "once all components in the update directory have been advertised and once all responses (version requests) have been received from the agents, the ASU server sends out files…." Col. 11, lines 63-67.
9. The method of claim 8, further including the step of storing said software components in a file system, wherein said components are classified into multiple roles which respectively contain	Fletcher describes storing the components in a file system, for example, when Fletcher talks about using FTP protocol to transfer files to ASU servers: "The files are provided to the ASU Mgr. by a user, and the ASU Mgr. in turn uses FTP (or some other file transfer protocol) to transfer these files to ASU servers. Col. 10, lines 2-4. Fletcher classifies the components into categories (e.g., ASU agent

different categories of software.	components, OS components and NIC drivers), which categories are used to determine the components that will be broadcast for updates: "The ASU server, in one embodiment, broadcasts out the latest version [of] one component (e.g. from an update list of available ASU agent components, OS components and NIC drivers received from the ASU manager) at a time with unique ids. Col. 10, lines 22-25.
10. The method of claim 9, wherein the categories of software are determined in accordance with the probable frequency with which their respective components are likely to be changed during the service lifetime of a device.	It is implicit in Fletcher that the categories described are each going to have a different probably frequency with which their respective components are likely to be changed: "The ASU server, in one embodiment, broadcasts out the latest version [of] one component (e.g. from an update list of available ASU agent components, OS components and NIC drivers received from the ASU manager) at a time with unique ids. Col. 10, lines 22-25. In addition, Anderson describes different update timing frequency for different subsystems: "The above subsystems run only when the machine boots, and any change in the database resources is not reflected in the corresponding subsystem until the machine is rebooted (or the subsystem is manually restarted). These are mostly one-off configurations (such as auth) or daemons which start once and run continuously (such as www and xdm). Some subsystems need to be run at regular intervals (for example, backups) and the boot subsystem can arrange to schedule these to run from cron. In particular, a group of processes runs every night to perform any necessary updates to the local file system[.]" P. 22.
11. The method of claim 9, wherein the model of a device is stored in said database as one set of software components from each of said multiple roles.	The files on the ASU server determine the software components to be provided to the devices: "ASU Mgr. allows a user to input and control the files to be updated from an ASU server to the ASU agents." Col. 9, line 67 – Col. 10, line 2.
12. The method of claim 11, wherein one of said roles includes operating system software for the devices.	Fletcher describes operating system components: "The ASU server, in one embodiment, broadcasts out the latest version [of] one component (e.g. from an update list of available ASU agent components, OS components and NIC drivers received from the ASU manager) at a time with unique ids. Col. 10, lines 22-25.
13. The method of claim 12, wherein another of said roles includes application programs for said devices.	Fletcher describes application program components (e.g., ASU agent components): "The ASU server, in one embodiment, broadcasts out the latest version [of] one component (e.g. from an update list of available ASU agent components, OS components and NIC drivers received from the ASU manager) at a time with unique ids. Col. 10, lines 22-25.
14. The method of claim 12, wherein another of said roles includes data content associated with the devices.	Fletcher describes data content associated with the devices (e.g., NIC driver components): "The ASU server, in one embodiment, broadcasts out the latest version [of] one component (e.g. from an update list of available ASU agent components, OS components and NIC drivers received from the ASU manager) at a time with unique ids. Col. 10, lines 22-25.
15. The method of claim 1, wherein the step of transmitting messages comprises the steps of storing commands in a queue in said database, sending a first message containing the first command in said queue,	As described in the Dec. 2004 Office Action, "Collins discloses a system and method for transferring software and data from one computer to one or more computer through network with installation agent established at each target machine and software package that includes the software and installation commands for installing the software is transferred to each target machine….It would have been obvious to one of ordinary skill in art, having the teaching of Fletcher and Collins before him at the time of invention was made, to modify the packets of messages for automatically updating software components on

awaiting a report from a device that the first message has been executed, and sending the next command in the queue in response to said report.	end system over network as disclosed by Fletcher to include installation commands with software in the packages as taught by Collins in order to obtain improved in speed, reliable, accurate and efficient method for automatically updating electronic software and distribution [col. 1, lines 29-37]." Para. 19. In addition, these steps are inherent in the transmission of messages in Fletcher, because sending a first message, awaiting an acknowledgement, sending the next command, would be implemented by any TCP-based protocol, and in particular, any TCP/IP protocol, including SNMP: "." P. 21
16. The method of claim 1, wherein said agents have a level of authority that enables them to manipulate operating system software installed on said devices.	Fletcher states: "The mechanism is generalizable and may be used to automatically update any number of components, but network and non-network components, including system level (OS) software components." Col. 9, lines 7-11.
17. A method for automatically installing software components on a plurality of computing devices having different respective sets of software, comprising the steps of:	The Dec. 2004 Office Action stated that Fletcher discloses a method and apparatus for automatically configuring software to enable said devices to perform predetermined operations. This is correct. Fletcher states: "The present invention is a method and apparatus for automatic software updating (ASU) in a LAN." Col. 5, Lines 6-8. Fletcher also states: "the method and apparatus of the present invention may operate with a wide variety of types of network devices." Col. 4, lines 65-66. As described above, Anderson was directed to automatically configuring software on a number of different devices having different sets of software and configurations of operating parameters.
storing in a database a model for each different type of device having a different respective set of software,	The Dec. 2004 Office Action stated that Fletcher discloses storing a model for each type of device in a database [ASU server]. Fletcher describes each agent (which may be on each different type of device) periodically sending its current version information to the ASU server: "on an intermittent basis, possibly initiated by a polling packet from the ASU server, the ASU agents forward current version information regarding a subset or all of their software components to an ASU server…." Col. 5, lines 17-19. "According to an embodiment of the invention, an agent response to an ASU server request is defined to indicate the current version level of all software components in the ES." Col. 9, lines 17-19. The ASU server stores the responses as a model of what the device has and needs in a database: "The ASU server receives update requests from the agents and sorts and aggregates that information into a cohesive database." Col. 7, lines 18-20. Fletcher updates operating parameters at the time that updates are installed: "For ASU components, proper Windows 95 registry entries are modified to reflect Auto update status at the time that files are copied." Col. 12, lines 62-64. As mentioned above, Anderson describes storing machine configuration information in a central database. This configuration information is the "model" for each different type of device: "All information that is necessary to distinguish one machine from another is contained in the central database." P. 21. This information includes machine-specific configuration as well as software installation and update information.

said model including a description of software components installed on a device;	Fletcher stores the versions that are available, and what is needed by each device: "the ASU server receives requests from each agent and stores the requests in table form, for example, with each file defining a column entry and each requesting agent defining a row entry." Col. 11, lines 48-51. As mentioned above, Anderson's stored information includes a description of software components installed on a device: "Storing the machine-specific configuration information explicitly in some external database (for example, `sad[6])` is a major improvement, since the configuration of a particular machine is always clear and the information is always accessible, even when the machine is down." P.20
installing an agent on each device that has the ability to install and delete other software components on said device;	Fletcher has an agent installed on the device: "[t]he invention includes two types of primary components, the agents that reside in ESs [end systems] and the ASU server…" Col. 7, lines 1-3. The agents update the ES devices: "new files received by an ASU agent (at an end system) are stored in one or more special update directories. These files are copied to their respective directories when all requested files have been received." Col. 12, lines 58-62. As mentioned above, Anderson describes a "script" running on the device that reads a configuration database and configures the device. "Every time the machine boots, a script reads the configuration database to determine the *subsystem* that should be configured on that machine. This executes a script for each subsystem (for example, DNS or xntp) which consults the database for relevant parameters and dynamically configures the subsystem accordingly." P. 21 The Anderson scripts are later described as running at boot time, manually, or at regular intervals: "Provision is also made to execute these scripts manually, or at regular intervals." P. 22
and transmitting messages, which contain data from a given one of said models, from said database to agents on only those devices which are associated with said given model, to cause said agents to retrieve software components from a source external to said devices and install said software components on the devices.	Fletcher describes sending the update files to each agent that needs a particular update on an agent-by-agent basis, or concurrently to multiple agents that need the same files, but in any event to only those devices that need the update: "For example, the ASU server accesses the first column (file) of the request table and sends that file in a point-to-point manner to each agent requesting that file. The ASU server then proceeds to the next column (file) that has at least one agent requesting that file and sends out that file to the requesting agents." Col. 11, line 67 – Col. 12, line 2. The Fletcher agents update the ES devices: "new files received by an ASU agent (at an end system) are stored in one or more special update directories. These files are copied to their respective directories when all requested files have been received." Col. 12, lines 58-62. Fletcher updates operating parameters at the time that updates are installed: "For ASU components, proper Windows 95 registry entries are modified to reflect Auto update status at the time that files are copied." Col. 12, lines 62-64. As mentioned above, Anderson states that a configuration file with the configuration for a machine is provided to that machine using the NIS protocol: "The resources are distributed and supplied to the client machines using NIS[8]." P.21. Based on the configuration file, the devices can then update software and apply patches for the software described in the

	configuration file for that machine, as described with respect to updatelf, patch, and update. P. 22.
18. The method of claim 17, wherein said messages are transmitted by means of a gateway that provides an interface between the database and the devices,	Claim 18 is identical to claim 3. See the discussion of claim 3 above.
and further including the step of converting messages in said gateway from a first protocol associated with the database to a second protocol employed by said devices.	Claim 18 is identical to claim 3. See the discussion of claim 3 above.
19. The method of claim 18, wherein said second protocol includes remote procedure calls.	Claim 19 is identical to claim 4. See the discussion of claim 4 above.
20. The method of claim 19, wherein said second protocol comprises XML-RPC.	Claim 20 is identical to claim 5. See the discussion of claim 5 above.
21. The method of claim 17, further including the step of storing said software components in a file system, wherein said components are classified into multiple roles which respectively contain different categories of software.	Claim 21 is identical to claim 9. See the discussion of claim 9 above.
22. The method of claim 21, wherein the categories of software are determined in accordance with the probable frequency with which their respective components are likely to be changed during the service lifetime of a device.	Claim 22 is identical to claim 10. See the discussion of claim 10 above.
23. The method of claim 21, wherein the model of a device is stored in said database as one set of software components from each of said multiple roles.	Claim 23 is identical to claim 11. See the discussion of claim 11 above.
24. The method of claim 23, wherein one of said roles includes operating system software for the devices.	Claim 24 is identical to claim 12. See the discussion of claim 12 above.
25. The method of claim 24, wherein another of said roles includes application programs for said devices.	Claim 25 is identical to claim 13. See the discussion of claim 13 above.

26. The method of claim 24, wherein another of said roles includes data content associated with the devices.	Claim 26 is identical to claim 14. See the discussion of claim 14 above.
27. The method of claim 17, wherein the step of transmitting messages comprises the steps of storing commands in a queue in said database, sending a first message containing the first command in said queue, awaiting a report from a device that the first message has been executed, and sending the next command in the queue in response to said report.	Claim 27 is identical to claim 15. See the discussion of claim 15 above.
28. The method of claim 17, wherein each agent has a level of authority that enables it to manipulate operating system software installed on said devices.	Claim 28 is identical to claim 16. See the discussion of claim 16 above.

CONCLUSION

For the reasons set forth above, each claim of the '289 patent is invalid. PUBPAT respectfully requests that the patent be reexamined ex parte and ultimately canceled in its entirety.

January 12, 2007
Date

/s/ Daniel Ravicher

Daniel B. Ravicher
Reg. No. 47.015
PUBLIC PATENT FOUNDATION, INC.
1375 Broadway, Suite 600
New York, NY 10018
Tel: (212) 796-0570
Fax: (212) 591-6038
www.pubpat.org

CERTIFICATE OF SERVICE

The undersigned certifies that a copy of this Request for *Ex Parte* Reexamination in its entirety, including all accompanying documents, is being deposited with the U.S. Postal Service as Priority Mail with Delivery Confirmation on the date of the signature below in an envelope addressed to the attorney of record for the assignee of U.S. Patent No. 7,124,289 as provided for in 37 C.F.R. § 1.33(c):

Attn: Daniel C. Kloke, Esq.
CARR & FERRELL LLP
2200 Geng Road
Palo Alto, CA 94303

January 12, 2007
Date

/s/ Daniel Ravicher
Daniel B. Ravicher
U.S.P.T.O. Reg. No. 47,015
PUBLIC PATENT FOUNDATION, INC.
1375 Broadway, Suite 600
New York, NY 10018
Tel: (212) 796-0570
Fax: (212) 591-6038
www.pubpat.org