



(86) Date de dépôt PCT/PCT Filing Date: 2009/06/19  
 (87) Date publication PCT/PCT Publication Date: 2009/12/30  
 (85) Entrée phase nationale/National Entry: 2011/12/23  
 (86) N° demande PCT/PCT Application No.: US 2009/047963  
 (87) N° publication PCT/PCT Publication No.: 2009/158280  
 (30) Priorité/Priority: 2008/06/25 (US61/075,618)

(51) Cl.Int./Int.Cl. *G06F 13/00* (2006.01)  
 (71) Demandeur/Applicant:  
 VELOXUM LLC, US  
 (72) Inventeur/Inventor:  
 LAZAR, MICHAEL A., US  
 (74) Agent: GOWLING LAFLEUR HENDERSON LLP

(54) Titre : SYSTEMES ET PROCEDES POUR REGLER UN SYSTEME D'EXPLOITATION, UNE APPLICATION, OU UN COMPOSANT RESEAU  
 (54) Title: SYSTEMS AND METHODS FOR TUNING AN OPERATING SYSTEM, APPLICATION, OR NETWORK COMPONENT

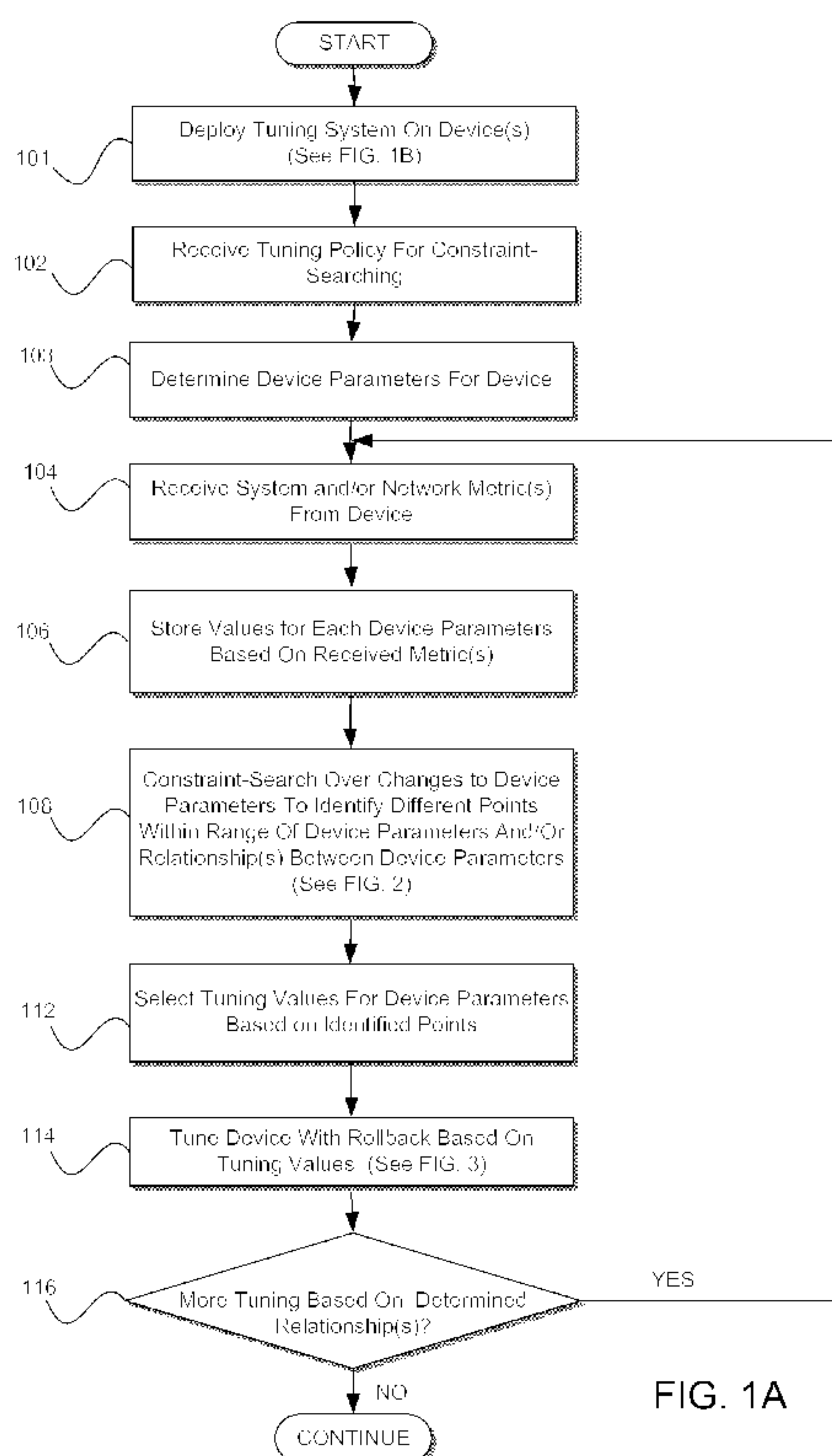


FIG. 1A

(57) **Abrégé/Abstract:**

A method, system, apparatus, and media are directed to tuning a device. Values are stored for each of a plurality of device parameters based on device metrics. The device parameters are constraint-matched over changes to settings of the device parameters to identify at least one point within each range of the device parameters that optimizes a performance measure for the device. Tuning values are selected for each device parameter based on the identified point. The tuning values are provided for tuning the device. The device is tuned with roll-back based on the running values.

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

(19) World Intellectual Property Organization  
International Bureau(43) International Publication Date  
30 December 2009 (30.12.2009)(10) International Publication Number  
**WO 2009/158280 A1**(51) International Patent Classification:  
*G06F 13/00* (2006.01)

(21) International Application Number:

PCT/US2009/047963

(22) International Filing Date:

19 June 2009 (19.06.2009)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

61/075,618 25 June 2008 (25.06.2008) US

(71) Applicant (for all designated States except US):  
**VELOXUM LLC** [US/US]; 2 Victoria Pines Lane,  
Lafayette, New Jersey 07848 (US).

(72) Inventor; and

(75) Inventor/Applicant (for US only): **LAZAR, Michael**  
[US/US]; 2 Victoria Pines Lane, Lafayette, New Jersey  
07848 (US).(74) Agents: **SHARIFI, Pejman** et al.; Winston & Strawn  
LLP, Patent Department, 1700 K Street, N.W., Washing-  
ton, District of Columbia 20006-3817 (US).(81) Designated States (unless otherwise indicated, for every  
kind of national protection available): AE, AG, AL, AM,  
AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ,  
CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO,  
DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT,  
HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP,  
KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD,  
ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI,  
NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD,  
SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT,  
TZ, UA, UG, US, UZ, VC, VN, 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, HR, HU, IE, IS, IT, LT, LU, LV,  
MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR),

[Continued on next page]

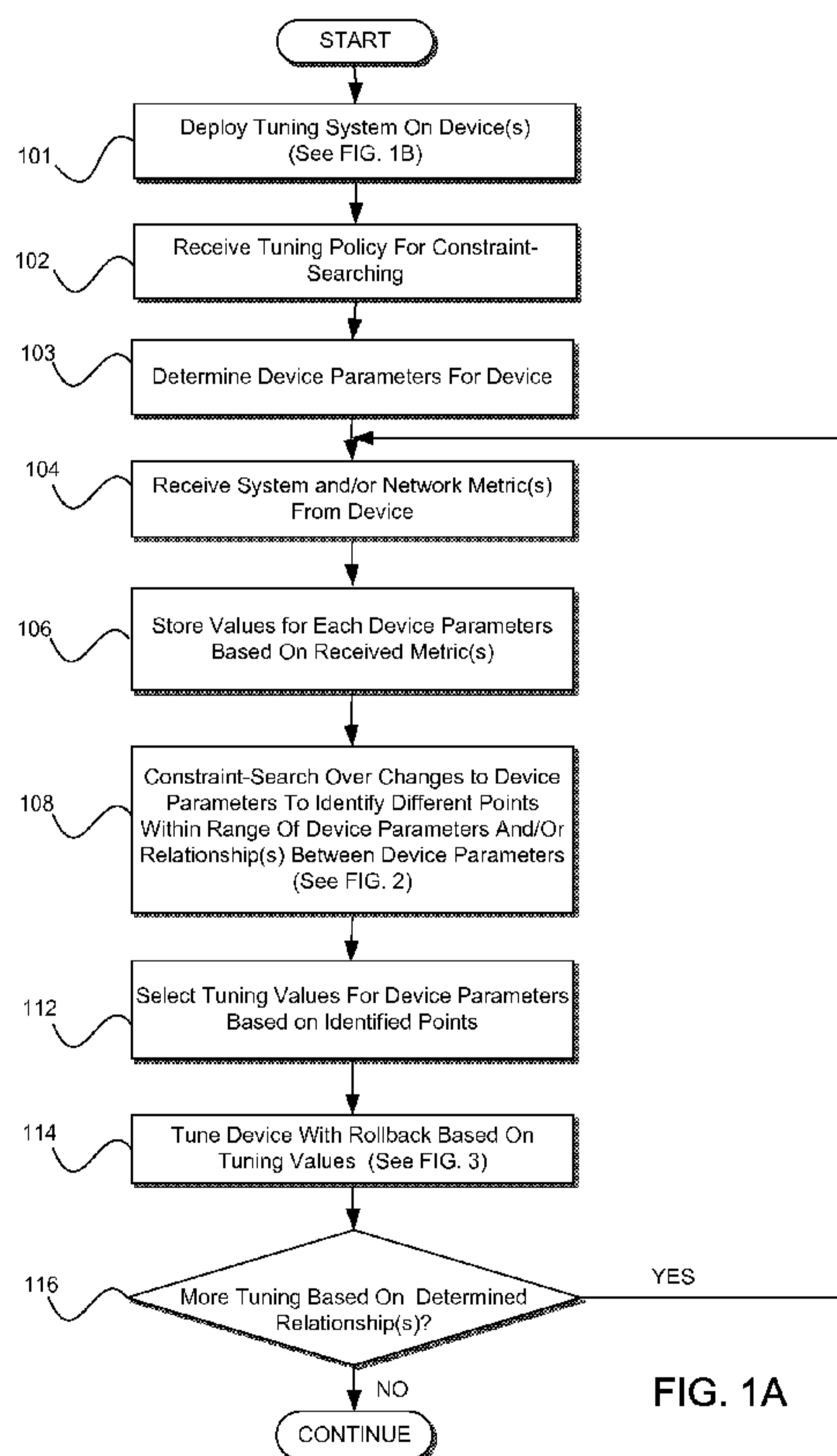
(54) Title: SYSTEMS AND METHODS FOR TUNING AN OPERATING SYSTEM, APPLICATION, OR NETWORK COM-  
PONENT

FIG. 1A

(57) Abstract: A method, system, apparatus, and media are directed to tuning a device. Values are stored for each of a plurality of device parameters based on device metrics. The device parameters are constraint-matched over changes to settings of the device parameters to identify at least one point within each range of the device parameters that optimizes a performance measure for the device. Tuning values are selected for each device parameter based on the identified point. The tuning values are provided for tuning the device. The device is tuned with roll-back based on the running values.

**WO 2009/158280 A1** 

---

OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG). **Published:**

— *with international search report (Art. 21(3))*

**Declarations under Rule 4.17:**

— *of inventorship (Rule 4.17(iv))*

**SYSTEMS AND METHODS FOR TUNING AN OPERATING SYSTEM,  
APPLICATION, OR NETWORK COMPONENT**

**TECHNICAL FIELD**

5           This invention relates to tuning a device, including operating systems, applications, and network components, and more specifically, but not exclusively to tuning based on constraint-searching of device parameters

**BACKGROUND OF THE INVENTION**

10           Computing devices often have a variety of different configurations and settings that require proper tuning and monitoring to ensure that the devices perform optimally. However, in an enterprise, the number of devices, applications, configurations, and settings may overwhelm the abilities of an administrator. Adding to this complexity, performance may also be based on the interrelationships between components within a device. Moreover, in a  
15 computer networking environment, performance of the devices may be based on the interrelationships between the networked components. Automated monitoring tools may provide some insight into trouble spots and possible optimizations. Possible manual optimizations may involve using rules to determine parameter tuning based on network conditions. However, manual optimization remains problematic. In particular, a system  
20 becomes chaotic over time and may produce different results from initially identical conditions. It is with regards to this issue and others that the present invention is directed.

**SUMMARY**

25           Generally, the present invention relates to a processor-implemented method for tuning a computing device. In one embodiment, the method may include storing values for each of a plurality of device parameters based on device metrics; constraint-searching over changes settings to the device parameters to identify at least one point within each range of the device parameters that optimizes an improvement function of a performance measure for the device; selecting tuning values for each device parameter based on the identified point; and providing  
30 the tuning values for tuning the device.

          In a preferred embodiment, constraint-searching includes modifying, on the device, each device parameters within a subset of the device parameters; in response to modifying, calculating the performance measure of the device; and recursively performing the steps of modifying and calculating until the improvement function of the performance measure is

determined to reach an optimal point. The optimal point may comprise a local minimum or maximum, a global minimum or maximum for a number of iterations of the recursive performance of the steps of modifying and calculating, or the like.

5 In one embodiment, the method may include determining a relationship between first device parameter and a second device parameter based on whether changing the first and second device parameters relative to each other changes the calculated performance measure. In yet another embodiment, the method may include, in response to determining whether the relationship, further constraint-searching based on the determined relationship, wherein the device parameters within the subset are related to each other by the determined relationship of  
10 related or unknown.

In a preferred embodiment, the step of modifying, on the device, the each device parameters within the subset of the device parameters comprises setting the device parameters that are within the subset, wherein these device parameters are related to each other or are unknown to be related to each other, while keeping a device parameter outside the subset  
15 constant. The step of modifying, on the device, the each device parameters within the subset of the device parameters may also comprise performing on the device parameters within the subset at least one of: a binary search; a random walk; a simulated annealing; a gradient descent; a change based on a schedule; a min-max search; or the like.

In a preferred embodiment, the plurality of device parameters includes a different  
20 device parameter for a plurality of types for each of a plurality of services, wherein the plurality of types is representative of device configurations for performing for performing a system operation, the Transmission Control Protocol (TCP)/Internet Protocol (IP), or other networking within a plurality of Open Systems Interconnection (OSI) layers. In yet other embodiments, the services comprise a use of an application, a use of a type of application, a  
25 type of user, or system usage.

In one embodiment, the method may include filtering the metric based on a service related to the device parameter before storing the values for each of a plurality of device parameters.

In one embodiment, the method may include saving a state of the device; tuning the  
30 device based on the determined tuning values; and if a self-test on the device fails for at least one of the tuning values, rolling back to the saved state.

Another aspect of the invention is directed to a system for tuning at least one device over a network. In one embodiment, the system includes a server configured to perform

actions comprising: storing tuning values for each of a plurality of device parameters and for each service of the device based on device metrics; constraint-searching over changes to settings of the device parameters to identify a point within each range of the device parameters, wherein the identified point optimizes a performance measure for the device; and  
5 providing tuning values for each device parameter based on the identified point. The system may also include another device configured to perform actions comprising: self-tuning based on the provided tuning values; and rolling-back to a saved state for at least one of the tuning values if a self-test for the at least one tuning values fails.

10 In one embodiment, constraint-searching may comprise performing an intersection of points for each of the ranges of the device parameters and for each service, performing a statistical analysis of changes in the device parameters, or the like.

In yet another embodiment, actions of the other device further comprises generating information usable to determine a metric of the other device; and sending the metric to the server for generating the points within the ranges.

15 In a preferred embodiment, an administrator interface configured to perform actions comprising: receiving a tuning policy usable in constraining the provided tuning values to satisfy the tuning policy.

Another aspect of the invention is directed to a network device for tuning another device over a network, including a processor configured to perform actions comprising storing  
20 values for each of a plurality of device parameters and for each service of the other device based on device metrics; constraint-searching over changes to settings of the device parameters to identify a point within each range of the device parameters, wherein the identified point optimizes a performance measure for the other device; and providing tuning values for each device parameter based on the identified point. The performance measure  
25 may comprise at least one of CPU usage, GPU (Graphical Processing Unit) usage, memory usage, input/output usage, network usage, or the like.

In a preferred embodiment, the actions further comprise receiving a tuning policy; and ensuring that the determined tuning values satisfies the tuning policy. The actions may further comprise receiving, from an agent on the device, a plurality of test messages over the network;  
30 gathering a plurality of network metrics, based on the received plurality of test messages; receiving a system metric based on a performance of the device; and combining the plurality of gathered network metrics and the received system metric into the device metrics.

Another aspect of the invention is directed to an apparatus for self-tuning. The apparatus may include a processor for performing actions comprising: generating information about the apparatus usable to determine a metric of the apparatus; providing the metric for generating a range of constraint-matched values for a device parameter and a service of the device based on the determined metric; receiving at least one tuning values for the device parameter based on the constraint-matched values; and self-tuning the apparatus based on the received tuning value. In a preferred embodiment, at least one of the device parameters comprise a network parameter for an interface for communicating over the network, an operating system parameter of an operating system operating on the apparatus, a registry setting, a group policy object, or a configuration file.

Yet another aspect of the invention is processor-readable medium comprising instructions that when executed by a processor causes the processor to perform actions. The actions may include determining a value for a device parameter and a service of the device based on a device metric and a filtering policy; constraint-searching over changes to settings of the device parameter to identify a point within a range of the device parameter, wherein the identified point optimizes a performance measure for the device; selecting a tuning value the device parameter based on the identified point; and providing the tuning value for tuning the device.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

The nature and various advantages of the present invention will become more apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, in which like reference characters refer to like parts throughout, and in which:

FIG. 1A shows an example of a process for tuning a device.

FIG. 1B shows an example of a process for deploying a tuning service;

FIG. 2 shows an example of a process for constraint-searching over changes to settings of device parameters;

FIG. 3 shows an example of a process for tuning with roll-back of a device;

FIG. 4A shows an example of a table for mapping device parameters to device and operating system specific configurations;

FIG. 4B shows an example of a table for mapping a device parameter to a service;

FIG. 5A shows an example of a table for processing different constraint-searching results;

FIG. 5B shows an example of a table for processing determined relationships between parameters; and

FIGs. 6A-6B show examples of systems and apparatuses for tuning a device.

5

### **DETAILED DESCRIPTION**

As used herein, the term "client" and "server" refers to computing devices that perform the roles of receiving a service (client) or providing a service (server), but the terms do not limit the roles of the devices. For example, a client may also be a server and vice-versa.

Moreover, in some embodiments, the devices may be configured in a peer-to-peer fashion, a load-balanced fashion (e.g., multiple servers providing load-balanced services), or the like.

As used herein, the term "parameter" refers to configuration of a computing device that is modifiable either through hardware or software, wherein the setting of the configuration affects the performance of the device. As used herein, the phrase "setting a parameter" refers to setting the value for the parameter.

As used herein, the term "point" refers to an ordered collection of information (e.g.,  $\langle X, Y, Z \rangle$ ), wherein each of the information in the collection represents a value within a range of values. The information may be, but need not be numeric.

As used herein, the terms "local minimum" or "local maximum" refers to an inflection point within a function within a particular dimension. The function may be an implicit function (e.g., a function determined by empirical measure and that may not be known). As used herein, the terms "global minimum" or "local maximum" refers to a highest or lowest point of a function over a range.

As used herein, the term "metric" refers to any measurable data for a particular type of data.

FIGs. 1A-1B, 2, and 3 show processes for tuning a device. In some embodiments, the processes may be performed over a network, but in other embodiments, the processes may be performed on a single device. The tuning may be performed on devices in an enterprise, on a single network, on devices sitting in different networks, or the like. In another embodiment, the tuning may be performed by a service provider providing the tuning service to a user device that downloads appropriate tuning software. While particular steps are shown, more or fewer steps may be performed without departing from the scope of the invention. In some embodiments, certain steps may be optional. In one embodiment, the processes may be performed on the devices of FIGs. 6A-6B.

FIG. 1A shows an example of a process for tuning a device. The process of FIG. 1A begins at step 101 where a tuning service is deployed on at least one device. In one embodiment, the process of step 101 may be performed by, for example, the process of FIG. 1B. Briefly, tuning agents may be installed, configured, and validated on at least one client  
5 device. Monitoring and tuning managers may be installed, configured, and validated on at least another device. In one embodiment, the tuning agents and monitoring and tuning managers may be installed on the same device. In one embodiment, step 101 may be optional and may not be performed. In some embodiments, devices may be pre-installed with tuning services.

10 At step 102, a tuning policy for defining a constraint-searching of device parameters is received. In one embodiment, the tuning policy may be received from an administrative component. In one embodiment, the tuning policy may comprise constraints of the type of processed device parameters, services, performance measures, an improvement function for a particular type of performance measure, a function to determine whether a performance  
15 measure or improvement function has reached an optimal point, or the like. The tuning policy may be configured as processor executable instructions or data structures, such as the tables of FIGs. 4A-4B and 5B. In one embodiment, the tuning policy constrains the type of tuning and optimization performed by the processes described herein. In one embodiment, the tuning policy constrains the tuning and optimization to particular combinations of Services, operating  
20 systems, platforms, performance property of a device, or the like. This tuning policy may represent a desire for a device or system to be tuned or optimized for a particular configuration or use. For example, the tuning policy may constrain tuning to combinations of (CPU for mySQL), (CPU for Graphical Artists), (Network Operations for Windows), (Cache Size for Email Programs and for Administrators), or the like.

25 In yet another example, an administrator may desire to tune a system or device to perform well for another type of use and a type of user. For example, the administrator may specify the tuning policy to tune a workstation on an enterprise to perform well for (Lawyers using Microsoft Word and Outlook) as well as for (Database Administrators using Database tools). In another example, the administrator may tune a mobile device to perform well for  
30 (Networking Operations and the Safari Browser) as well as for (Excel and Salespeople). As can be seen, the performance for a device may be multi-dimensional.

At step 103, at least one device parameter for a client device is determined. In one embodiment, an agent may examine a system configuration, registry, or the like, and may

determine whether the configuration matches a parameter type. In one embodiment, the parameter type may be pre-configured, configured by a tuning policy, determined automatically by clustering device behavior and configuration settings, or the like. In one embodiment, the determination may be performed on the device (e.g., by the agent), on a server device in communication with the device, or the like.

At step 104, a system and/or network metric is received from the device. In one embodiment, the metric is received over a network, through an interface such as an Application Programming Interface (API), through a web service, or the like. In one embodiment, the metric may be polled, pushed, or pulled from the client device, based on a schedule, a determined frequency, or the like. The system and/or network metric may be further processed, filtered, categorized, combined, or the like, into other metric(s), using, for example, filters, clustering, an expert system, or the like. In one embodiment, the modified/combined device metric may be used in subsequent processing. In one embodiment, device parameters may be represented by each row of FIG. 4A.

At step 106, values for each determined device parameter may be retrieved, calculated, and/or stored based on the received metric. In one embodiment, the device parameter values may simply be the received metric. In another embodiment, further calculations, filtering, and/or monitoring may be performed to calculate the device parameter value. In one embodiment, the metric may be filtered based on the Services 426 specified for the device parameters. For example, for services related to Financial Users, the metric related to Financial Users (e.g., database usage, CPU usage, GPU usage, chip or processor usage) are filtered and selected and converted into device parameters.

In other embodiments, further processing based on the Device Parameter Type is performed to determine the device parameters. For example, an agent on a device may send a first metric, such as a send time of a message to a server device. The server device may receive the message and may calculate the latency based on the send time and the time of receipt of the message and may store the calculated value as the device parameter value. In one embodiment, a plurality or a range of values may be retrieved, calculated, and/or stored for a particular device parameter and/or for a combination of device parameters.

At step 108, constraint-searching is performed over at least one change of the values of the device parameter(s) to, among other things, identify a point or different points within the range of the device parameters and/or to identify the relationships between the device parameters. In one embodiment, the process of step 108 may be performed by the process of

FIG. 2. Briefly, constraint-searching may comprise modifying, on the device, each device parameters within a subset of the device parameters, in response to modifying, calculating at least one performance measure of the device or an improvement function of the at least one performance measure, recursively performing the steps of modifying and calculating until the performance measure or a function thereof is determined to reach an optimal point, or the like. In one embodiment, constraint-searching may involve performing an intersection of points for each of the ranges of the device parameters for each service, performing a statistical analysis of changes in the values of the device parameters, modifying, on the device, the settings of at least some device parameters (and in some cases, not all the device parameters) and empirically measuring and calculating a performance measure, or the like. In one embodiment, the identified point may comprise the device parameters, a subset of the device parameters, a function of the device parameters, or the like.

At step 112, at least one tuning value for at least one device parameter may be selected based on the identified point(s). In one embodiment, the tuning values may be selected from the values within the identified point(s), may be based on a mapping between tuning values and the identified point(s), or the like. The selected tuning values may be provided for tuning the device. The tuning values may be provided over a network, for example, as an Extensible Markup Language (XML) file, a network message, or the like.

At step 114, the device is tuned with roll-back based on the selected tuning values. In one embodiment, the process of step 114 may be performed by the process of FIG. 3. Briefly, tuning with rollback may comprise saving a state of the device; tuning the device based on the determined tuning values; and if a self-test on the device fails for at least one of the tuning values, rolling back to the saved state.

At step 116, it is determined if more tuning should be performed. In one embodiment, further tuning may be configured, requested, or required by an administrator, and administrative module, a heuristic, a number of iterations, a schedule, or the like. In one embodiment, further tuning may be performed until the identified optimal point stabilizes (e.g., remains the same point, or within a range of the same point between iterations of the process of FIG. 1). In one embodiment, the further tuning is based on the relationship(s) between the device parameter(s) identified in step 108. In one embodiment, further execution of step 108 and/or step 202 of FIG. 2 (described below) may be based on the identified relationship(s). That is, further execution of constraint-searching will generate a possibly smaller subset for searching based on the previously determined relationship, thus

constraining the combinatorial explosion of the number of possible changes to settings of device parameters. In any case, at step 116, if it is determined that more tuning should be performed, processing loops back to step 104. Otherwise, processing continues to other processing.

5           FIG. 1B shows an example of a process for deploying a tuning service. The process of FIG. 1B begins at steps 120-123 where a plurality of information related to device is gathered. At step 120, an IP address and/or TCP ports of a server device to send a device metric and the polling frequency at which to send the metric to the server device are determined. At step 121, it is determined if unicast flood monitoring will be active (e.g., a particular type of  
10 networking process will be monitored by the server device to determine device metrics). At step 122, the operating system type and/or platform (e.g., hardware/mobile/non-mobile) is determined. At step 123, the location(s) of devices on which to deploy the tuning agents are determined. The information from steps 120-123 is provided to step 124, where the pre-deployment information is stored and/or combined.

15           At step 132, the pre-deployment information is used to install and/or configure tuning software on the server device(s). The license for the use of the tuning software may also be used to authorize the use of the software at step 130.

          At step 134, the available software distribution method for a particular operating system or platform is determined. At steps 135-138, the particular operating systems and  
20 platforms are specifically determined (e.g., OSX is determined at step 135, Solaris at step 136, WinTel at step 137, Linux at step 138). The determined distribution method is used to deploy the agent(s) at step 140 on the determined client devices.

          At step 150, the initial client check-in process begins. The process may comprise validating the license for the client device, collecting initial system parameters (e.g., used to  
25 build the table of FIGs. 4A-4B, 5A-5B), running initial tests (e.g., to see if any parameters fail self-tests), and uploading collected data to the server. In one embodiment, the process may be performed by an agent on the client device.

          FIG. 2 shows an example of a process for constraint-searching over changes in the values to device parameters. The process begins at step 202 where a subset of the device  
30 parameters may be generated, wherein the subset includes device parameters that are determined to be related to each other, or that have an unknown relationship to each other. In one embodiment, the subset may include not all of the device parameters. In one embodiment, generating the subset may comprise selecting a seed device parameter, and

finding one or more other device parameters related to the seed device parameter, or finding device parameters with an unknown relationship to the seed device parameter. In one embodiment, selecting the seed may be random, may be based on a heuristic, a schedule, or the like. The relationship between the device parameters may be stored and/or retrieved from a table, such as the table of FIG. 5B.

At step 204, the value settings for the device parameters within the subset are modified on the device. In one embodiment, the modification of a particular device parameter may be constrained by a Value Range 424 of possible values, a range of desired values, a range of pre-configured values by, for example, and administrator, or the like. In one embodiment, modifying the device parameters may include setting the value within a particular location (e.g., as shown in the Location 425 column of the table of FIG. 4A), or the like.

In another embodiment, the device parameters' settings may be set with a fail-safe rollback as described in FIG. 3. In one embodiment, each parameter may be set individually, while in another embodiment, some or all of the device parameters may be set at once. In one embodiment, if setting a particular device parameter to a particular value fails, the device parameter's range may be modified to exclude the problematic value. In another embodiment, failure may cause the device parameter to be removed from the subset, removed from further use as a device parameter within the process of FIGs. 1 and/or 2, or the like. In one embodiment, modifying, on the device, the each device parameters within the subset of the device parameters comprises performing on the device parameters within the subset at least one of: a binary search; a random walk; a simulated annealing; a gradient descent; a change based on a schedule; a min-max search; and A\* or heuristic search, or the like.

At step 206, the settings of the device parameters that are outside the subset may be maintained constant on the device. In one embodiment, no action may be needed, or a no-op operation may be performed. In another embodiment, the maintained value setting may be set on the device within the particular device parameter's location.

At step 207, at least one performance measure is calculated based on received information (empirical results) from the device. Thereby, the device parameters in the subset is tested on the device to provide the empirical results. In one embodiment, the performance measure comprises at least one of CPU/GPU usage, memory usage, input/output usage, network usage, or the like. As shown in FIG. 5A, the performance measure of Throughput 520 may be calculated based on received networking information. In one embodiment, an improvement function of the performance measure, such as a Relative Performance 520 (as

shown in FIG. 5A) may also be calculated. In one embodiment, the function may be representative of an improvement in the performance measure.

At step 208, it is determined if a first change of a first device parameter relative to a second device parameter correlates with a second change in a performance measure. If it is  
5 determined that the first change correlates with the second change, processing continues to step 210. Otherwise, processing continues to step 212.

At step 210, the relationship between the first device parameter and the second device parameter is set to be "related." Referring to table FIG 5A row 503, the performance measure changes from 65 (from row 502) to 20 for a relative change between MTU and  
10 TCPWindowSizeREAD (e.g., TCPWindowSizeREAD changes from 65535 to 87238). For this example, it may be determined that MTU and TCPWindowSizeREAD are related. The "related" relationship between the device parameters may be stored, updated, or corrected in a field such as Relationship Update 523 or a table, such as the table of FIG. 5B.

At step 212, the relationship between the first device parameter and the second device  
15 parameter is set to be "un-related." Referring to the table of FIG. 5A row 506, the performance measure (95) does not change between changes of first device parameter Max Simultaneous Connection (ID = 7) and any of the other parameters. Thus, it is determined that device parameter 7 is unrelated to device parameters 1, 2, 3, 4, 5, 6, 8. In row 507, the performance measure (95) does not change between changes to settings of the first device  
20 parameter Max Simultaneous Connection 1\_0 and any of the other parameters. Thus, the device parameter 8 is determined to be unrelated to device parameters 1, 2, 3, 4, 5, 6, 7. The "un-related" relationship between the device parameters may be stored, updated, or corrected in a field such as Relationship Update 523 or a table, such as the table of FIG. 5B.

At step 214, it is determined if the performance measure and/or the improvement  
25 function of the performance measure for the device is optimized based on the updated values of the device parameters. In one embodiment, the performance measure/function is optimized when the performance measure/function reaches an optimal point. In one embodiment, the optimal point comprises a local minimum or maximum, or a global minimum or maximum for a number of iterations of the recursive performance of the step of modifying and calculating,  
30 or the like. Referring to FIG. 5A, in one embodiment, a Probability 521 is calculated to determine the likelihood that the performance measure or function has reached an optimal point. As shown, at row 503, the Probability 521 is measured to be 0.0 which represents the likelihood of the information at that row representing a an optimal point is 0%. At row 504,

the Probability 521 is measured to be 1.0 which represents the likelihood of the information at that row representing a an optimal point is 100%. In one embodiment, the Probability 521 may be determined to be 1.0 if the current values of Relative Performance 522 (e.g., 0.6) is the current local maximum. In other embodiments, Probability 521 may be measured by  
5 historical information such as conditional probabilities of changes to the performance measures, heuristics, or any other function that increases/decreases as the likelihood of an optimal point increases/decreases.

In one embodiment, the current local maximum may be a local optimal (minimum or maximum) point. In an alternate embodiment, an optimal calculation (e.g., Probability 521)  
10 may be performed for each of the different types of performance measures (or improvement function thereof) such as a measure for CPU/GPU, Memory, Input/Output (e.g., Hard Disk Access), Network Operations, or the like. In performing the process of FIG. 2, the result of the optimal calculations may yield a plurality of indications that a current iteration represents the overall optimal. In one embodiment, a weighted sum or other function may be calculated  
15 to bias one or more of the associated results of the optimal calculations. For example, a tuning policy may specify that for an iteration, a performance measure may be determined for a CPU for use with the Internet Explorer Application (Service is IE), another performance measure may be determined for Network Operations and a type of Accountant users (Service = Accountant User). The tuning policy may also specify that the performance measure for  
20 CPU-IE should be weighed more heavily (e.g., 0.8) and the performance measure for Network Operations-Accountant should be weighted less heavily (e.g., 0.2). In this example, the Probability 521, the tuning policy may specify a desire for the tuning to be biased towards performance for good CPU usage for IE and good networking performance for uses associated with Accountants. As another example a tuning policy may specify  $\text{Probability 521} = (0.2 * \text{Probability (CPU)} + 0.6 * \text{Probability (Memory)} + 0.1 * \text{Probability (Input/Output)} + 0.1 * \text{Probability (Network Operations)})$ . In this example, Memory and CPU are biased more than the other measures. In general, the probability measures may be based on functions of any combinations of services and performance property of a device. Any weights may also be  
25 used for any combinations of performance measures.

30 In any case, if the performance measure/function is not optimized or otherwise non-optimized, processing continues to step 202. Subsequent iterations of the process of FIG. 2 is performed based on updated relationships between device parameters and updated relative

performance measures. As can be seen, the system self-corrects for errors in these and other values.

If, at step 214, the performance measure/function is not optimized or otherwise non-optimized, processing continues to step 202 processing returns to other processing. In one  
5 embodiment, the identified device parameter point(s) that are related to measure/function's optimal point are provided to step 112 of FIG. 1 for further processing.

FIG. 3 shows an example of a process for tuning with roll-back of a device. The process begins at step 302, where a state of the device is saved. In one embodiment, the parameters, configurations, or the like, of the device are saved, for example, in a state file,  
10 within a memory device, a storage medium, or the like.

At step 304, at least one setting of one device parameter may be modified on the device. In one embodiment, the device parameter's location may be retrieved for the particular device parameter from a table such as the table of FIG. 4A. In one embodiment, at  
least one parameter is changed at a time.

At step 306, a test of the device is determined to fail. In one embodiment, the test may  
15 be performed by the device (e.g., a self-test). In one embodiment, the test may be to determine if a performance of the device satisfies at least some or all pre-determined values or settings, or the like. If the test is determined to fail, processing continues to step 308, wherein the device parameter's setting are reset to the saved state. Otherwise, processing continues to  
20 other processing.

In an alternate embodiment, at least some subset of device parameters may be tuned by a rules based system while another subset(s) of device parameters may be tuned using the processes of FIGs. 1A-1B, 2-3. Briefly, a rule-based system may receive a plurality of device parameter values. The rule-based system may be a decision tree, an expert system, or the like.  
25 Based on logic and at least one comparison of the device parameter value as inputs to threshold values (pre-configured or adaptable), a rule fires and causes an action to occur. The action may be to cause other rules to fire and/or to indicate that at least one device parameter on the device should to be set/configured to a particular value. The particular value may be a function of the device parameter value input. In one embodiment, the device parameter value  
30 inputs may be gathered by a network diagnostic tool such as the University of Chicago's Web100 Network Diagnostic Tool (NDT), however any type of system or network measuring component may be used.

In one embodiment, rule-based system's indications or suggestions for device parameters and their settings or configurations may be combined with the constraint-searching's settings of the identified device parameter point(s) associated with an optimal point. In one embodiment, the subset of device parameters managed by the rule-based system and the subset managed by the constraint-searching may not overlap. In this case, the device parameters may be set according to their respective settings. In another embodiment, the subset may overlap. In this case, the overlapping parameters' settings may be combined, for example, using either the constraint-searching or the rule-based systems' parameters as the overriding settings, combining the settings using an average, a weighted average, or the like. The weighted average may be based on the Probability 521 for the optimal point as at least a portion of the weight for a parameter's setting and/or other weights, such as certainty factors, for the weights of another parameter's setting.

FIGs. 4A-4B, 5A-5B represents examples of particular embodiments of data structures usable by the processes and systems described herein. Other fields and data structures, such as B-trees, XML files, text files, or the like may used without departing form the scope of the invention. The data structures may be embodied in memory such as memory 608 of FIG. 6.

FIG. 4A shows an example of a table for mapping device parameters to device and operating system, and/or platform specific configurations. In one embodiment, the operating system may include Windows, Linux, SunOS, Symbian, Palm OS, Android, or the like. In one embodiment, the platform may be hardware specific and/or may specify the device type (e.g., mobile/non-mobile, embedded device, networking device, workstation). As shown in FIG. 4A, device parameter rows 402 of a particular Device Parameter Type 421 (e.g., MTU, RFC1323, Receive, Send, TCPWindowSize READ and Write, Maximum Simultaneous Connection(s), Selective Acknowledgements, Timeout values, Retransmit values, etc.) may have a particular value of a Value Type 422. The Value Type 422 may be device or operating specific and/or may be a generic data type (e.g., text, float, int). Each device parameter may be specific to a particular Operating System (OS) 423 and may be known or may be desired to be constrained within a Value Range 422. Each device parameter may be stored and/or modified within a particular configuration Location 425. The Location 425 may be a physical location on a memory device, for example, or may be a logical location modifiable using, for example, operating system calls, such as a registry call, or the like. As such, the device parameter rows 402 define the constraints on the use of the device parameters by the processes and systems described herein. For example, the Value Range 422 defines the

constraint-searching range. A binary-search may divide the Value Range 422 in half and use the boundaries of the ranges as a particular candidate device parameter, and recursively divide that range in half to find an optimal device parameter. The binary-search may also modify the values based on the Value Type 422. For example, a floating point number may be divided in  
5 half, but a number on a logarithmic scale may use a non-linear division function.

As shown in FIG. 4B, the rows 404 of particular device parameters of particular Device Parameter Types 421 may be related to a plurality of Services 426 (e.g., TCP, Internet Explorer, Citrix, SystemBuffer, etc.). The Services 426, and relationships between device parameter (types) and Services 426 may be pre-configured, configured by a tuning policy,  
10 determined automatically by clustering device behavior and configuration settings, or the like. In one embodiment, the Services 426 may be directed to improving performance for an application (e.g., Internet Explorer, MySQL) or application type (e.g., Web Browser, Database), directed to improving performance for a type of user (e.g., Power User, Financial User, Developer), directed to improving networking performance (e.g., TCP, Hyper Text  
15 Transfer Protocol (HTTP)), directed to improving general system performance (e.g., System Buffer, CPU usage, memory usage, input/output activity), or the like. Using the tables of FIGs. 4A and 4B, a mapping between a particular device parameter represented by rows from FIG. 4A a service may be determined.

FIG. 5A shows an example of a table for processing different constraint-searching  
20 results. As shown, each row 502-507 of the table represents one iteration of a process for constraint-searching over changes to device parameters, such as the process of FIG. 2.

As shown, each iteration row 502-507 may include a plurality of values of device parameters, e.g., MTU 524, TCPWindowSizeREAD 525, TCPWindowSizeWRITE 526, RFC1323 527, Receive 528, Send 529, Max Simultaneous Connection 530, Max  
25 Simultaneous Connection 1\_0 531, or the like. The values may be gathered initially when the values are sent from an agent on a device. In one embodiment, an initial testing may be performed on the device to create a baseline of performance characteristics. The baseline may be set as the first row of the table of FIG. 5a. The values may also be set for each iteration of the constraint-searching process, thereby testing the space embodied by the ranges of device  
30 parameters. The data set of the values may form a point within the ranges of the device parameters.

In one embodiment, a performance value, such as Throughput 520, is determined based on information received from the device in response to setting the device parameters on

the device. In one embodiment, the information received may include measured values. The performance value may be received directly from the device or may be calculated based on information received from the device.

As shown, the Relative Performance 520 may be initially set at an initial value such as 0.5. In one embodiment, the Relative Performance 520, for subsequent iteration, is calculated based on whether the current performance measure improves over the previous performance measure. In one embodiment, the Relative Performance 520 may be increased by an increment (e.g., 0.1) if the performance measure improves. In one embodiment, the improvement of the performance measure may be specific to the type of the performance measure. For example, Throughput 520 improves if the value of the Throughput 520 increases. In one embodiment, there may be a plurality of performance measures, and the Relative Performance 520 measure may be increased if at least some or all of the performance measures improve. For example, the performance measures may be Throughput 520 and CPU time (not shown in FIG. 5A). The Relative Performance 520 measure increases if Throughput 520 increases and CPU time decreases.

FIG. 5B shows an example of a table for processing determined relationships between parameters. As shown in FIG 5B, each row 508 of device parameter types may be identified by an ID 540. A Relationship Type 542 between the Device Parameter Type 421 and other device parameter types (stored in column Relationship 541) may be determined as: Known Related, Known Un-related, or Unknown. The relationships of FIG. 4B accordingly constrain the relationships between particular device parameters, such as those shown in FIG. 4A.

FIGs. 6A-6B show examples of systems and apparatuses for tuning a device. While particular components and/or interconnections are shown, other components and/or interconnections may be used without departing the scope of the invention.

As shown, the systems of FIGs. 6A-6B include administrative component 620, server device 602, client devices 630-634. The devices of the system may be in communication with each other over at least one network, such as network 640. The system may also include network devices 635-638 for managing network traffic such as routers, firewalls, or the like. The networks may be a wired or wireless network, a Local Area Network (LAN), Wide Area Network (WAN), or the like.

The devices of the system of FIGs. 6A-6B may be implemented in hardware, software, or the like. In one embodiment, the device may be at least one physical device, may be mapped as a virtual device across several devices or on one device (e.g., virtual server 631), or

the like. In one embodiment, the client device may be an enterprise server, a database server, a workstation, or the like. In one embodiment, the devices may be mobile devices.

As shown, the client devices 630-634 may be in communication with server device 602 over a network. In other embodiments, the components of client and server may be  
5 combined in one device. In one embodiment, at least some client devices 630-631 may be in a different sub-net than server device 602.

In one embodiment, an agent may be deployed on at least one of the devices 630-638, using for example, the process of FIG. 1B. In one embodiment, the agent, such as agents 621-622, may be a software component sent from server device 602 to at least one of client  
10 devices 630-634 over a network.

In one embodiment, server device 602 may include tuning manager 604, device parameter gathering manager 606, administrative manager 608, memory 610, and processor 612. The components may be software and/or hardware components and may be in communication with each other. Other components may be included without departing from  
15 the scope of the invention. In one embodiment, server device 602 may perform the processes of FIGs. 1A-1B, 2, and 3. In one embodiment, device parameter gathering manager 606 receives and/or determines device parameters based on received device metrics from devices 630-638.

In one embodiment, tuning manager 604 may be configured to perform constraint-  
20 searching over changes to settings of device parameters. In one embodiment, tuning manager 604 may include a logic and database of remedial actions that it can intelligently apply. Tuning manager 604 may establish test points to test the device's parameters. Example of test points include rules such as: based on X and Y Network/System tests, request agents 621-622 to perform Z actions. Tuning manager 604 and/or agents 621-622 may use a decision tree  
25 stored for performing such tests. The Z actions may include further sending data to the server for determining network and system metrics. X, Y tests may be based on registry settings, group policy object, user permissions, configuration files, or the like.

Tuning manager 604 may also specify a remedial action plan to be executed by agents 621-622 based on constraint-searching. The remedial action plan may include tuning with  
30 rollback based on tuning values, but may also comprise setting or sending alerts, turning on or off services including backup services, privacy protection services, network protection services, or the like. The remedial action plan may be selected based on the selected tuning values. In one embodiment, an expert system may select the remedial action plan based on

the tuning values and/or other information, e.g., time, historical activities, permissions of users, or the like.

Device parameter gathering manager 606 may receive information from devices 630-638 over a network. Device parameter gathering manager 606 may include a polling  
5 mechanism to check for various device parameters. Device parameter gathering manager 606 may determine the services available on devices 630-638 based on the received information (services include database services, types of database services (Oracle, SQL Server), applications (accounting, graphics, internet usage, etc), or the like.

In one embodiment, administrative manager 608 may include an interface for  
10 managing the components of server 602. The communication between administrative manager 608 and server device 602 may be over a networking protocol such as HTTP, or the like. In one embodiment, administrative manager 608 comprises an administrator interface configured to perform actions comprising receiving a tuning policy usable in constraining the provided tuning values to satisfy the tuning policy. In one embodiment, the tuning policy may  
15 be a software code or a data structure such as tables of FIGs. 4A-4B and 5B. In one embodiment, the administrative manager 608 may be configured to receive a tuning policy; ensure that the determined tuning values satisfies the tuning policy, or the like.

Administrative component 620 may communicate with server device 602 using administrative  
manger 608 to manage the tuning as described herein. In one embodiment, the types of device  
20 parameters, the constraint of ranges for the device parameters, an initial relationship between device parameters, or the like may be entered using administrative manager 608. In one embodiment, the policy configuration may be per service type, device parameter type, or the like. For example, the Value Range 422 for MTU for a first Service 426 may be a first range, while the Value Range 422 for MTU for a second Service 426 may be a second range. In this  
25 embodiment, constraint-searching is configured to search within the overlapping ranges of the first and second ranges.

Memory 610 may include any component for storing data, including the data of tables  
of FIGs. 4A-4B, and 5A-5B. Memory 610 includes disk drives, EEPROM, flash memory,  
RAM, ROM, secondary storage devices, Serial ATA, ATA, IDE, EIDE, SCSI, Fibre Channel  
30 attached SCSI, iSCSI, or the like. In one embodiment, memory 610 and/or other memory components may be configured to store processor executable instructions that when executed causes a processor, such as processor 612, to perform the processes of FIGs. 1A-1B, 2, and 3.

In one embodiment, devices 630-638 may include components such as agents 621-622, self-tuners 624-625, and services 626-629. The components may be software and/or hardware components and may be in communication with each other. In one embodiment, devices 630-638 may perform the processes of FIGs. 1A-1B, 2, and 3. Agents 621-622 may gather device parameters from devices 630-638. In one embodiment, agents 621-622 may collect information about the system, applications installed, what devices are talking to devices 630-638, communications protocols (TCP/IP), or the like. Agents 621-622 may send the gathered information to server 602 for further processing.

In one embodiment, agents 621-622 may act as an active test object. Agents 621-622 may communicate with server 602 over specific communications ports. Server 602 may send an instruction set based on which the agents reports. Agents 621-622 may perform the instruction set (e.g., perform Z action based on X/Y tests). In one embodiment, agents 621-622 acts as a meta processing layer. Agents 621-622 may also execute remedial action plans including setting the values of device parameters on devices 630-638.

The services 626-629 may be configured to perform a plurality of functions, including word processing, data processing, or the like. The services may be configured to perform an activity for a particular type of user. Self tuners 624-625 perform the process of FIG. 3 to change device parameters and/or roll-back the device's state if a self-test fails. In one embodiment, self-tuners 624-625 may be configured to optimize services 626-629.

The foregoing is merely illustrative of the principles of this invention and various modifications can be made by those skilled in the art without departing from the scope and spirit of the invention.

**CLAIMS**

What is claimed is:

- 5 1. A processor-implemented method for tuning a device, comprising:  
storing, on a computer memory, values for each of a plurality of device parameters  
based on device metrics;  
constraint-searching, by a processor, over changes to settings of the device parameters  
to identify at least one point within each range of the device parameters that optimizes an  
10 improvement function of a performance measure for the device;  
selecting tuning values for each device parameter based on the identified point;  
providing the tuning values for tuning the device; and  
tuning the device based on the provided tuning values.
- 15 2. The method of claim 1, wherein constraint-searching comprises:  
modifying, on the device, each device parameters within a subset of the device  
parameters;  
in response to modifying, calculating the performance measure of the device; and  
recursively performing the steps of modifying and calculating until the improvement  
20 function of the performance measure is determined to reach an optimal point.
3. The method of claim 2, wherein the optimal point comprises a local minimum or  
maximum, or a global minimum or maximum for a number of iterations of the recursive  
performance of the step of modifying and calculating.
- 25 4. The method of claim 2, further comprising:  
determining a relationship between first device parameter and a second device  
parameter based on whether changing the first and second device parameters relative to each  
other changes the calculated performance measure.
- 30

5. The method of claim 4, further comprising:

in response to determining whether the relationship, further constraint-searching based on the determined relationship, wherein the device parameters within the subset are related to each other by the determined relationship of related or unknown.

5

6. The method of claim 2, wherein modifying, on the device, the each device parameters within the subset of the device parameters comprises:

changing the device parameters that are within the subset, wherein these device parameters are related to each other or are unknown to be related to each other, while keeping a device parameter outside the subset constant.

10

7. The method of claim 2, wherein modifying, on the device, the each device parameters within the subset of the device parameters comprises performing on the device parameters within the subset at least one of:

15

- a binary search;
- a random walk;
- a simulated annealing;
- a gradient descent;
- a change based on a schedule; or

20

- a min-max search.

8. The method of claim 1, wherein the plurality of device parameters comprises a different device parameter for a plurality of types for each of a plurality of services, wherein the plurality of types is representative of device configurations for performing a system operation or networking within a plurality of Open Systems Interconnection (OSI) layers, and wherein the services comprise a use of an application, a use of a type of application, a type of user, or system usage.

25

9. The method of claim 1, further comprising:

filtering the metric based on a service related to the device parameter before storing the values for each of a plurality of device parameters.

30

10. The method of claim 1, further comprising:  
saving a state of the device;  
tuning the device based on the determined tuning values; and  
if a self-test on the device fails for at least one of the tuning values, rolling back to the  
5 saved state.
11. A system for tuning over a network, comprising:  
a server configured to perform actions comprising:  
storing tuning values for each of a plurality of device parameters and for each service  
10 of the device based on device metrics;  
constraint-searching over changes to settings of the device parameters to identify a  
point within each range of the device parameters, wherein the identified point optimizes a  
performance measure for the device; and  
providing tuning values for each device parameter based on the identified point; and  
15 another device configured to perform actions comprising:  
self-tuning based on the provided tuning values; and  
rolling-back to a saved state for at least one of the tuning values if a self-test for the at  
least one tuning values fails.
- 20 12. The system of claim 11, wherein constraint-searching comprises:  
performing an intersection of points for each of the ranges of the device parameters  
and for each service.
13. The system of claim 11, wherein constraint-searching comprises:  
25 performing a statistical analysis of changes in the device parameters.
14. The system of claim 11, wherein the actions of the other device further comprises:  
generating information usable to determine a metric of the other device; and  
sending the metric to the server for generating the points within the ranges.

30

15. The system of claim 11, further comprising:

an administrator interface configured to perform actions comprising:

receiving a tuning policy usable in constraining the provided tuning values to satisfy the tuning policy.

5

16. A network device for tuning a another device over a network, including a processor configured to perform actions comprising:

storing values for each of a plurality of device parameters and for each service of the other device based on device metrics;

10 constraint-searching over changes to settings of the device parameters to identify a point within each range of the device parameters, wherein the identified point optimizes a performance measure for the other device; and

providing tuning values for each device parameter based on the identified point.

15 17. The network device of claim 16, wherein the actions further comprise:

receiving a tuning policy; and

ensuring that the determined tuning values satisfies the tuning policy.

18. The network device of claim 16, wherein the performance measure comprises at least one  
20 of CPU usage, memory usage, input/output usage, or network usage.

19. The network device of claim 16, wherein the actions further comprise:

receiving, from an agent on the device, a plurality of test messages over the network;

gathering a plurality of network metrics, based on the received plurality of test

25 messages;

receiving a system metric based on a performance of the device; and

combining the plurality of gathered network metrics and the received system metric into the device metrics.

30 20. An apparatus for self-tuning, comprising:

a processor for performing actions comprising:

generating information about the apparatus usable to determine a metric of the apparatus;

providing the metric for generating a range of constraint-matched values for a device parameter and a service of the device based on the determined metric;

receiving at least one tuning values for the device parameter based on the constraint-matched values; and

5 self-tuning the apparatus based on the received tuning value.

21. The apparatus of claim 20, wherein at least one of the device parameters comprise a network parameter for an interface for communicating over the network, an operating system parameter of an operating system operating on the apparatus, a registry setting, a group policy  
10 object, or a configuration file.

22. A processor-readable medium comprising instructions that when executed by a processor causes the processor to perform actions comprising:

15 determining a value for a device parameter and a service of the device based on a device metric and a filtering policy;

constraint-searching over changes to settings of the device parameter to identify a point within a range of the device parameter, wherein the identified point optimizes a performance measure for the device;

20 selecting a tuning value the device parameter based on the identified point;

providing the tuning value for tuning the device; and

tuning the device based on the tuning value.

1/8

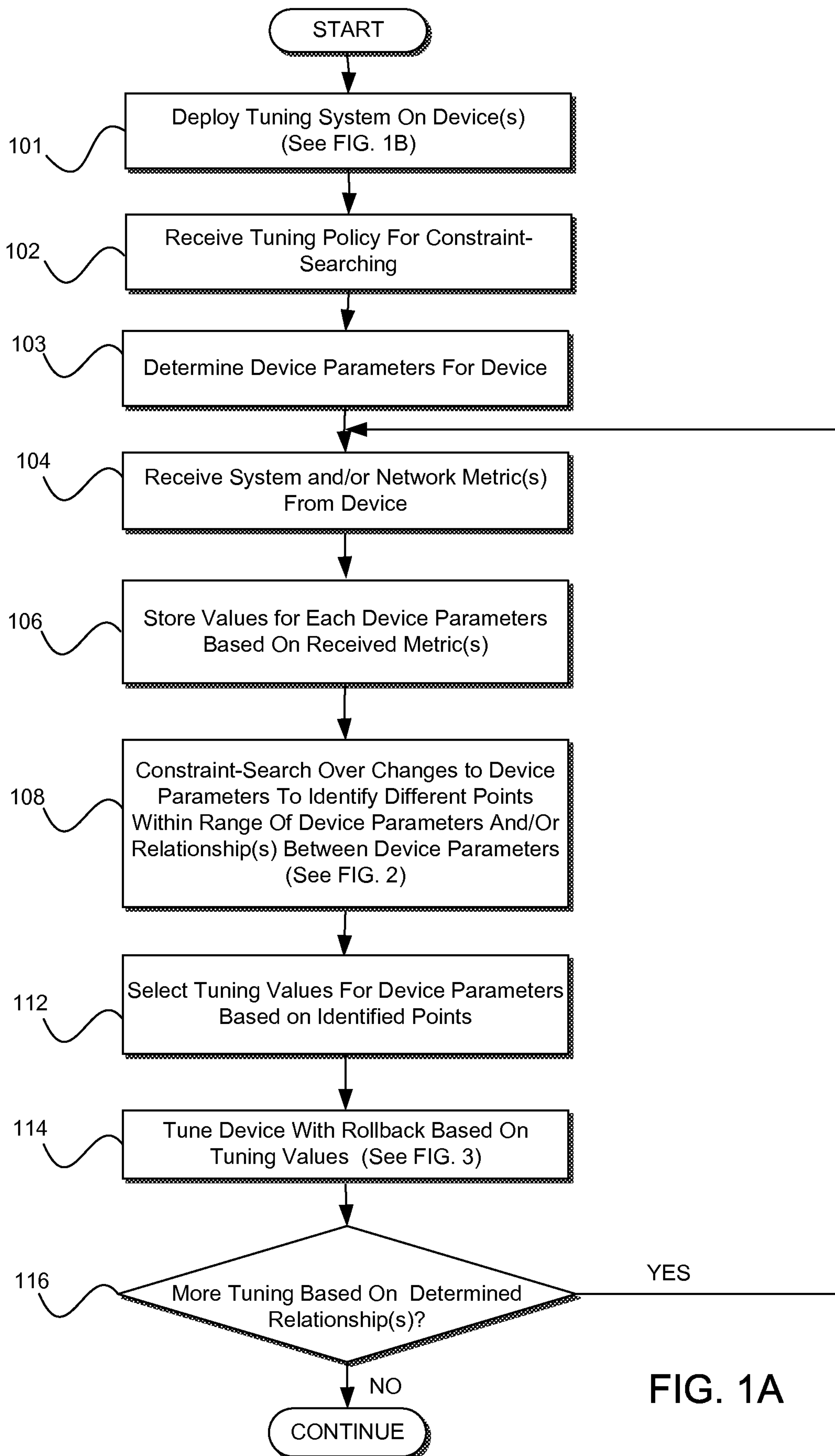


FIG. 1A

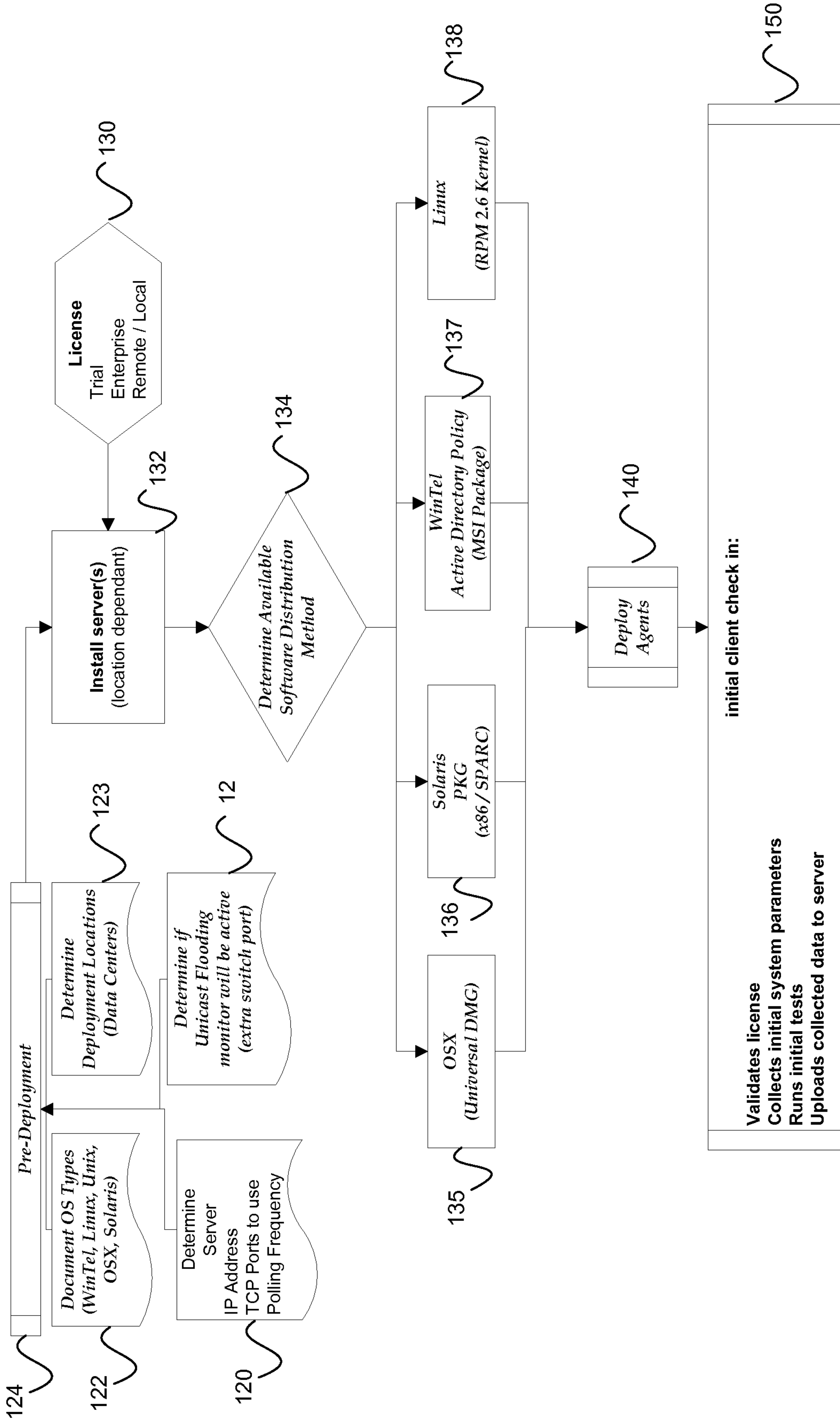


FIG. 1B

3/8

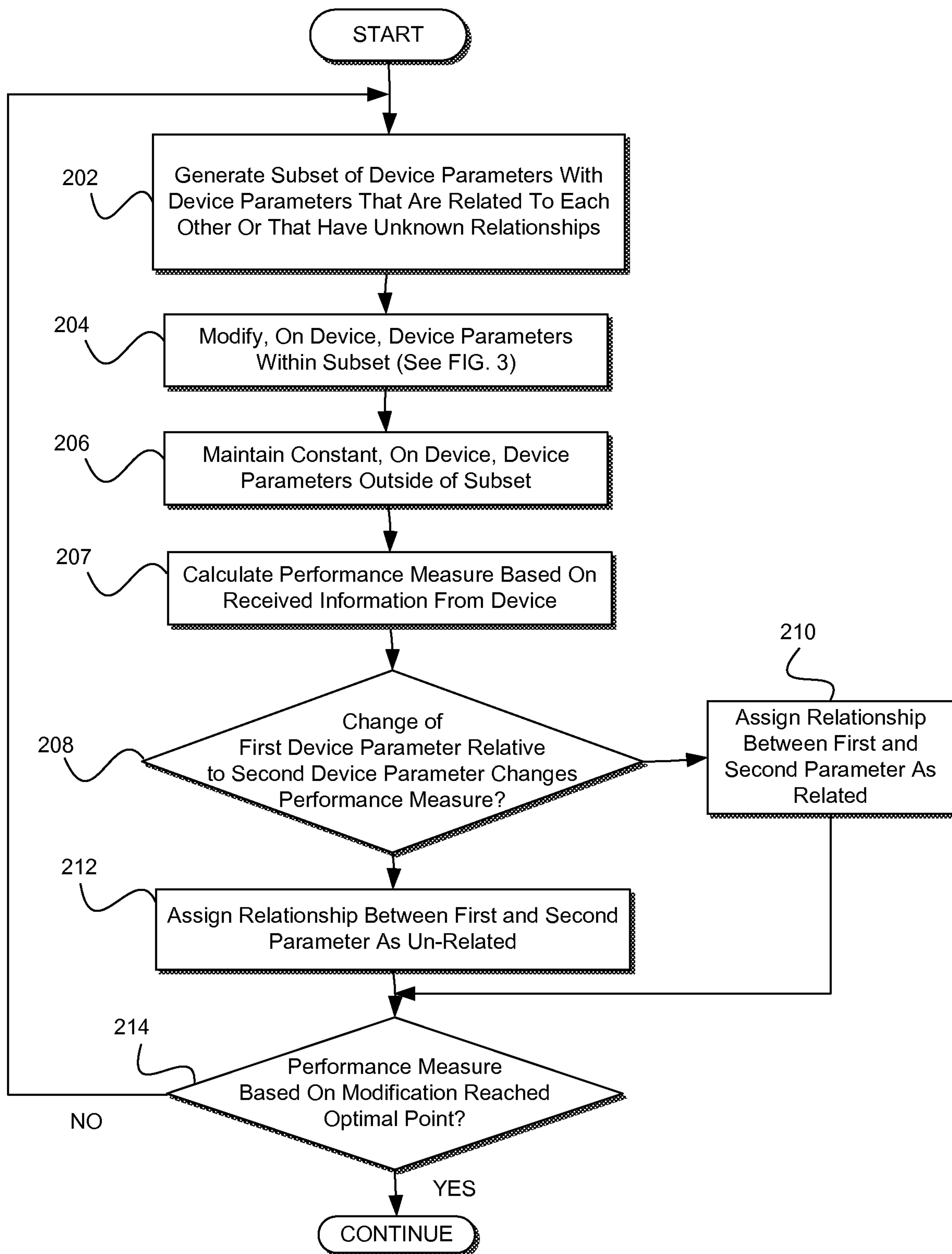


FIG. 2

4/8

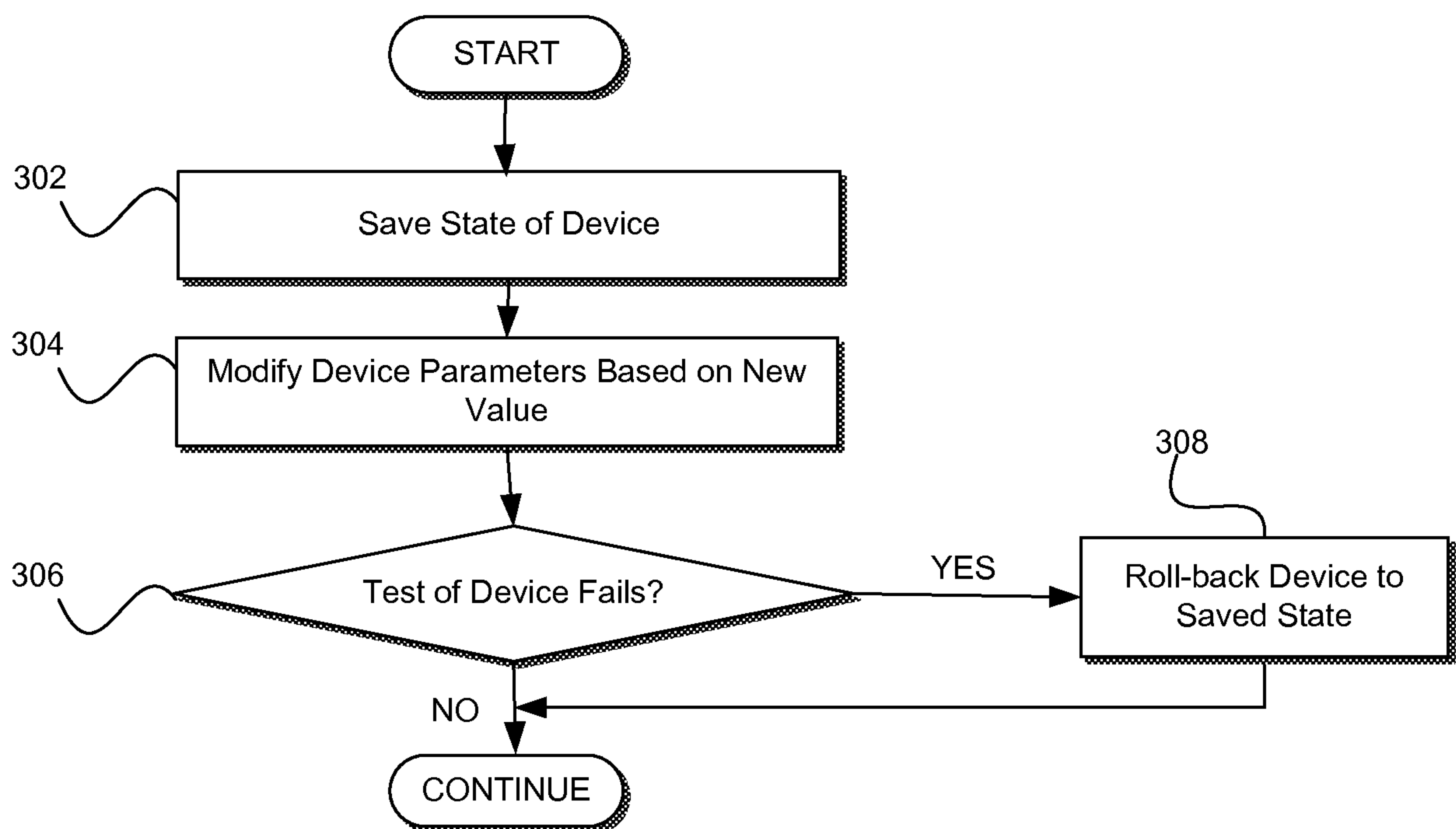


FIG. 3

Device Parameter (Type)	Value Type	OS	Value Range	Location
MTU	MTU	Windows	88-9000	HKLM\CurrentControlSet\Tcpip\Parameters
TCPWindowSizeREAD	TcpWindowSize	Windows	0-1973741823	HKLM\CurrentControlSet\Tcpip\Parameters
TCPWindowSizeWRITE	TcpWindowSize	Windows	0-1973741823	HKLM\CurrentControlSet\Tcpip\Parameters
RFC1323	Tcp1323Opts	Windows	0-3	HKLM\CurrentControlSet\Tcpip\Parameters
Receive	DefaultReceiveWindow	Windows	0-1973741823	HKLM\CurrentControlSet\AFD\Parameters
Send	DefaultSendWindow	Windows	0-1973741823	HKLM\CurrentControlSet\AFD\Parameters
MTU	ifconfig eth	Linux2.6	88-9000	ifconfig eth
TCPWindowSizeREAD	net.ipv4.tcp_rmem	Linux2.6	0-1973741823	net.ipv4.tcp_rmem
TCPWindowSizeWRITE	net.ipv4.tcp_wmem	Linux2.6	0-1973741823	net.ipv4.tcp_wmem
RFC1323	timestamps	Linux2.6	0-1	/proc/sys/net/ipv4/tcp_timestamps
RFC1323	tcp_window_scaling	Linux2.6	0-1	/proc/sys/net/ipv4/tcp_window_scaling
RFC1323	tcp_sack	Linux2.6	0-1	/proc/sys/net/ipv4/tcp_sack
Receive	net.core.rmem_max	Linux2.6	0-4194304	net.core.rmem_max
Send	net.core.wmem_max	Linux2.6	0-4194304	net.core.wmem_max
Max Simultaneous Connections	MaxConnectionsPer1_0Server	Windows	0-20	HKCU\Software\Microsoft\Windows\CurrentVersion\InternetSetting
Max Simultaneous Connections1_0	MaxConnectionsPer_Server	Windows	0-20	HKCU\Software\Microsoft\Windows\CurrentVersion\InternetSetting
Max Simultaneous Connections	N/A	Linux		
Max Simultaneous Connections1_0	N/A	Linux		

FIG. 4A

Device Parameter (Type)	Service
MTU	TCP
TCPWindowSizeREAD	TCP
TCPWindowSizeWRITE	TCP
RFC1323	TCP
Max Simultaneous Connections	Internet Explorer
Max Simultaneous Connections1_0	Internet Explorer
Receive	SystemBuffer
Send	SystemBuffer

FIG. 4B

502	520	521	522	523	524	525	526	527	528	529	530	531
Throughput	Probability	Relative Performance	Relationship Update	MTU	TCPWindowSize READ	TCPWindowSize WRITE	RFC 1323	Receive	Send	Max Simultaneous Connection	Max Simultaneous Connection	1 0
65	0	0.5		576	65535	65535	0	N/A	N/A	4	4	
20	0	0.4		576	87238	87238	0	87238	87238	4	4	
85	1	0.6		1500	87238	87238	0	87238	87238	4	4	
95	1	0.6		1500	121000	121000	0	121000	121000	4	4	
95	1	0.6	7 Unrelated to 1,2,3,4,5,6, 8	1500	121000	121000	0	121000	121000	8	4	
95	1	0.6	8 Unrelated to 1,2,3,4,5,6, 7	1500	121000	121000	0	121000	121000	4	4	

FIG. 5A

540	421	541	542
ID	Device Parameter (Type)	Relationship	Relationship Type
1	MTU	2,3,4,5,6	Known Related
2	TCPWindowSizeREAD	1,3,4,5,6	Known Related
3	TCPWindowSizeWRITE	1,2,4,5,6	Known Related
4	RFC1323	1,2,3,5,6	Known Related
5	Receive	1,2,3,4,6	Known Related
6	Send	1,2,3,4,5	Known Related
7	Max Simultaneous Connections	1,2,3,4,5,6	Unknown
8	Max Simultaneous Connections1_0	1,2,3,4,5,6	Unknown

FIG. 5B

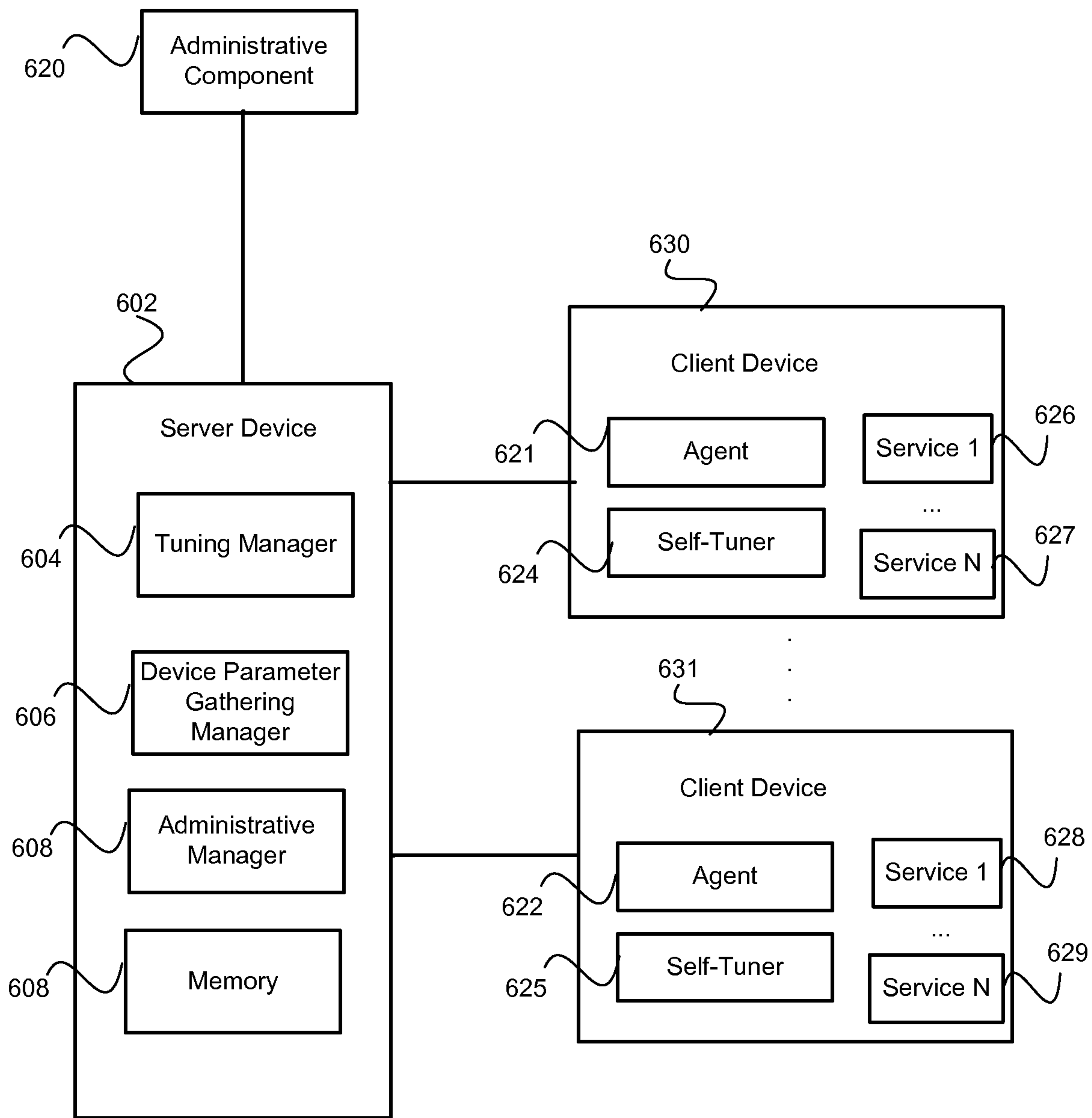


FIG. 6A

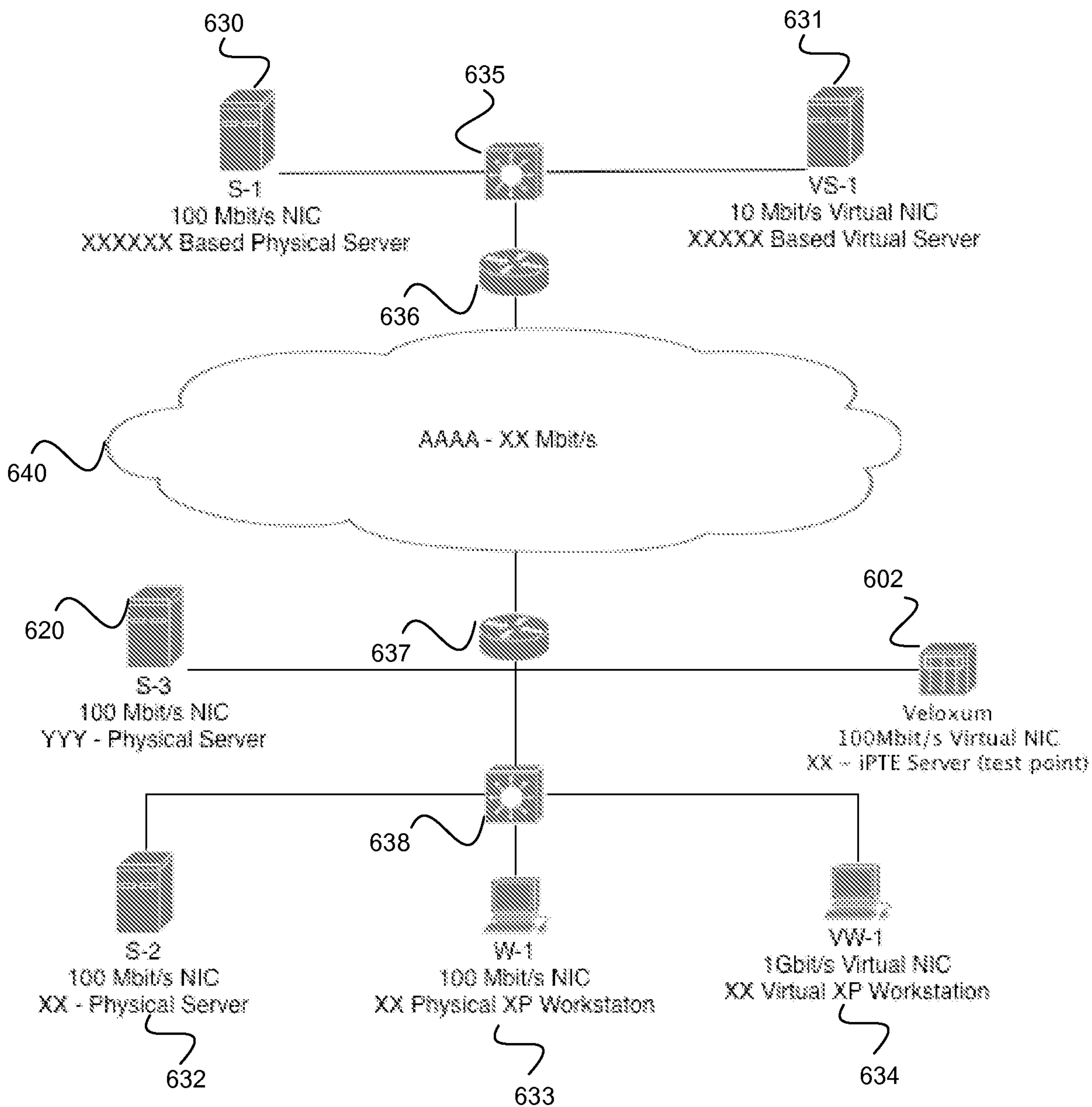


FIG. 6B

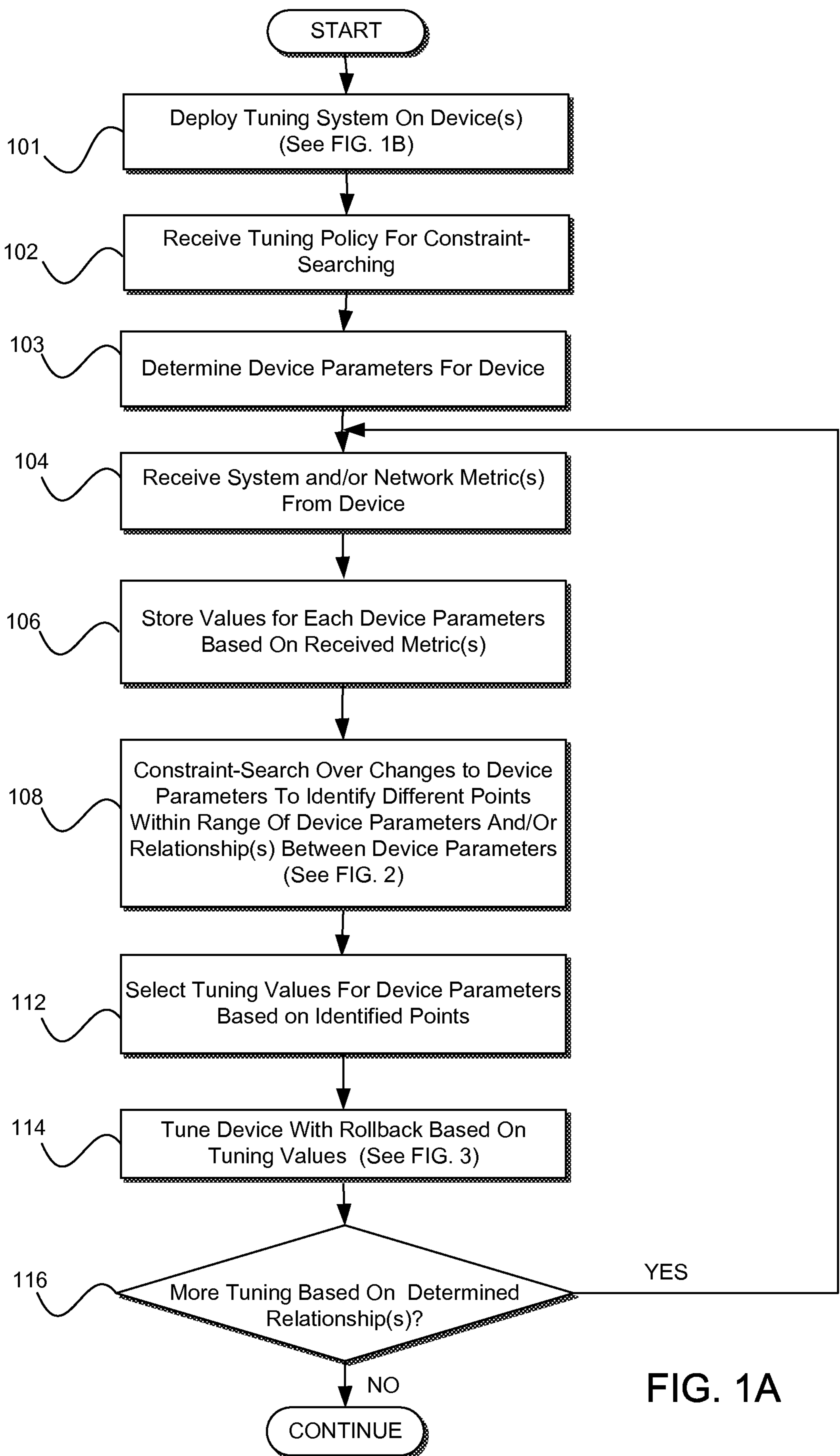


FIG. 1A