US009270983B1

(12) **United States Patent**
Hare, Jr.

(10) **Patent No.: US 9,270,983 B1**
(45) **Date of Patent: Feb. 23, 2016**

(54) **QUICKLY DIAGNOSE SERVICE AND COMPONENT RELATED ISSUES ON A CABLE MODEM, MULTIMEDIA TERMINAL ADAPTER, OR GATEWAY**

(75) Inventor: **William Charles Hare, Jr.**, Cumming, GA (US)

(73) Assignee: **ARRIS Enterprises, Inc.**, Suwanee, GA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **13/525,777**

(22) Filed: **Jun. 18, 2012**

(51) **Int. Cl.**
H04N 7/173 (2011.01)
H04N 17/00 (2006.01)
H04N 17/04 (2006.01)

(52) **U.S. Cl.**
CPC ***H04N 17/00*** (2013.01); *H04N 17/045* (2013.01)

(58) **Field of Classification Search**
USPC 725/107, 111, 114, 116, 117, 121, 124; 370/241, 242; 455/423
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,868,508	B2 *	3/2005	Grey	G06F 11/263 702/120
7,739,717	B1 *	6/2010	Kuether	H04N 7/17336 725/107
8,209,732	B2 *	6/2012	Le	H04N 17/00 348/180
8,544,051	B1 *	9/2013	Ramakrishnan	H04J 14/0298 725/111
2002/0019983	A1 *	2/2002	Emsley	H04N 7/10 725/107
2006/0184988	A1 *	8/2006	Skalina	H04N 17/004 725/107
2007/0076616	A1 *	4/2007	Ngo	H04L 12/2697 370/241
2008/0059838	A1 *	3/2008	Melman	G01R 31/318314 714/25

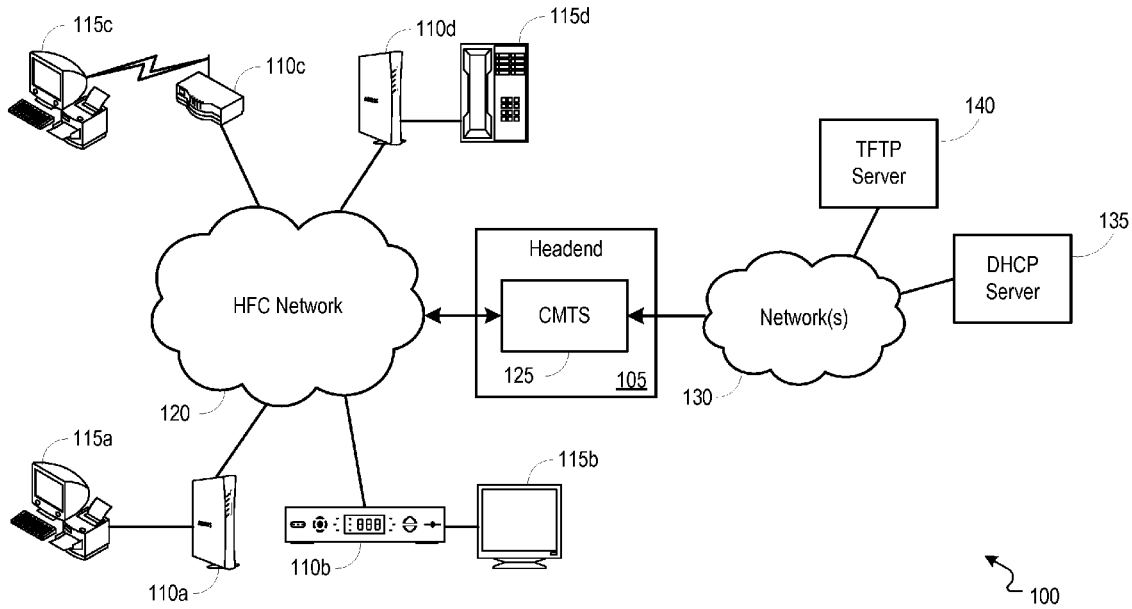
* cited by examiner

Primary Examiner — Brian T Pendleton
Assistant Examiner — Alexander Gee
(74) *Attorney, Agent, or Firm* — Bart A. Perkins

(57) **ABSTRACT**

Methods, systems, and computer readable media can provide diagnosis of service-affecting issues in CPE devices. The diagnostic process can include retrieving a testing hierarchy associated with a received diagnostic command, executing the lowest-level diagnostic in the testing hierarchy, successively executing the remaining diagnostics in the testing hierarchy in the order implicated by the hierarchy until the commanded diagnostic is executed, and identifying the service-affecting issues found.

24 Claims, 4 Drawing Sheets



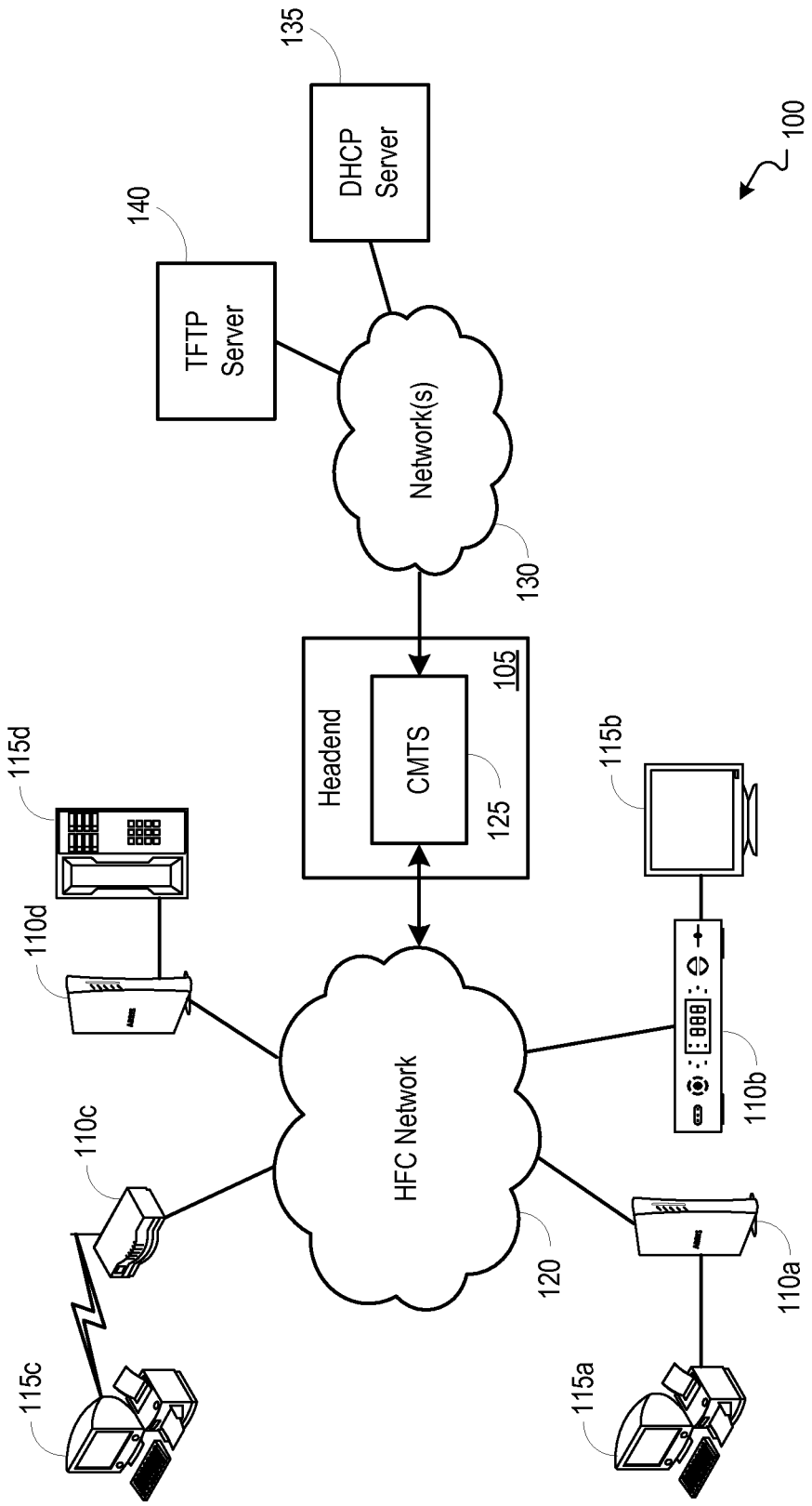

100
135
DHCP Server
140
TFTP Server
Network(s)
130
Headend
105
CMTS
125
115d
110d
115b
110b
HFC Network
120
110c
110a
115c
115a

FIG. 1

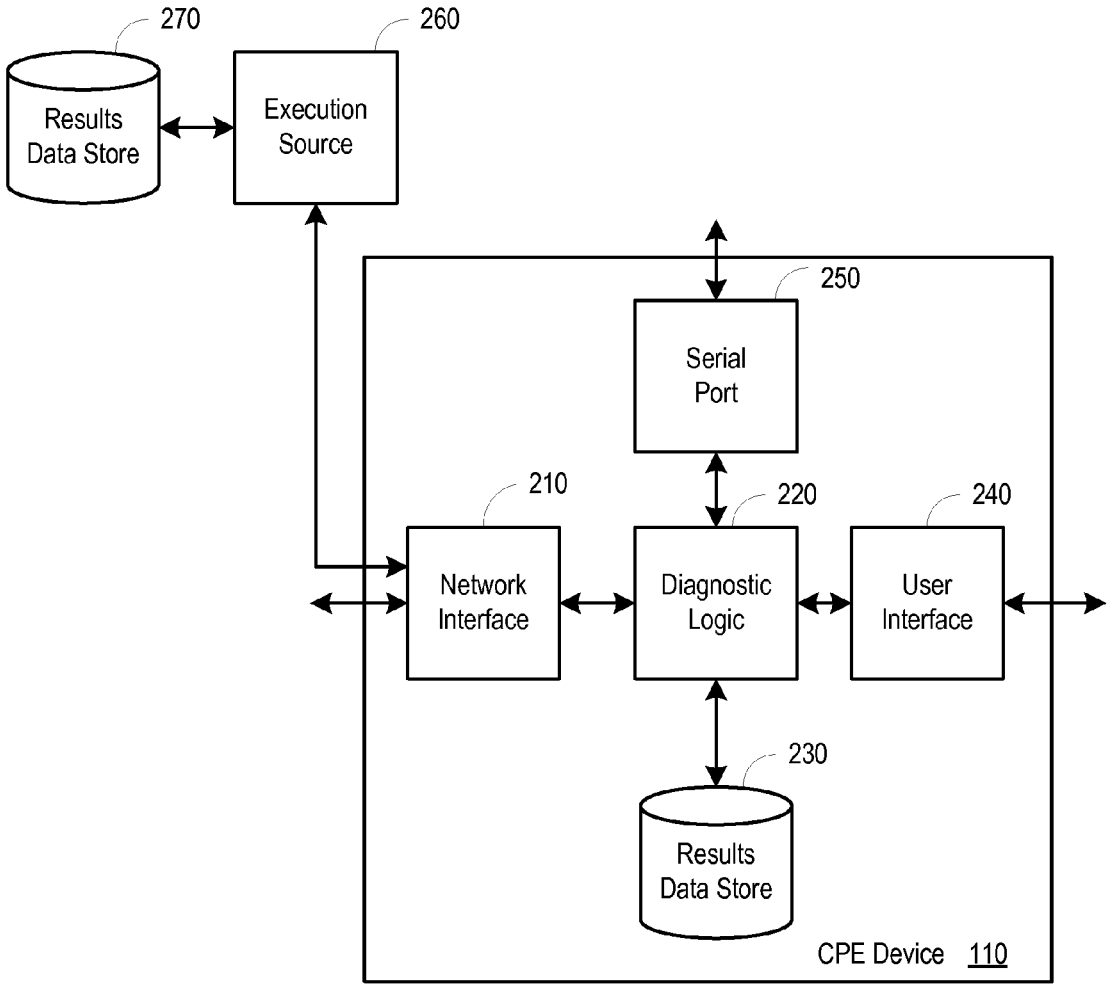

270
260
Results Data Store
Execution Source
250
Serial Port
210
220
240
Network Interface
Diagnostic Logic
User Interface
230
Results Data Store
CPE Device 110

FIG. 2

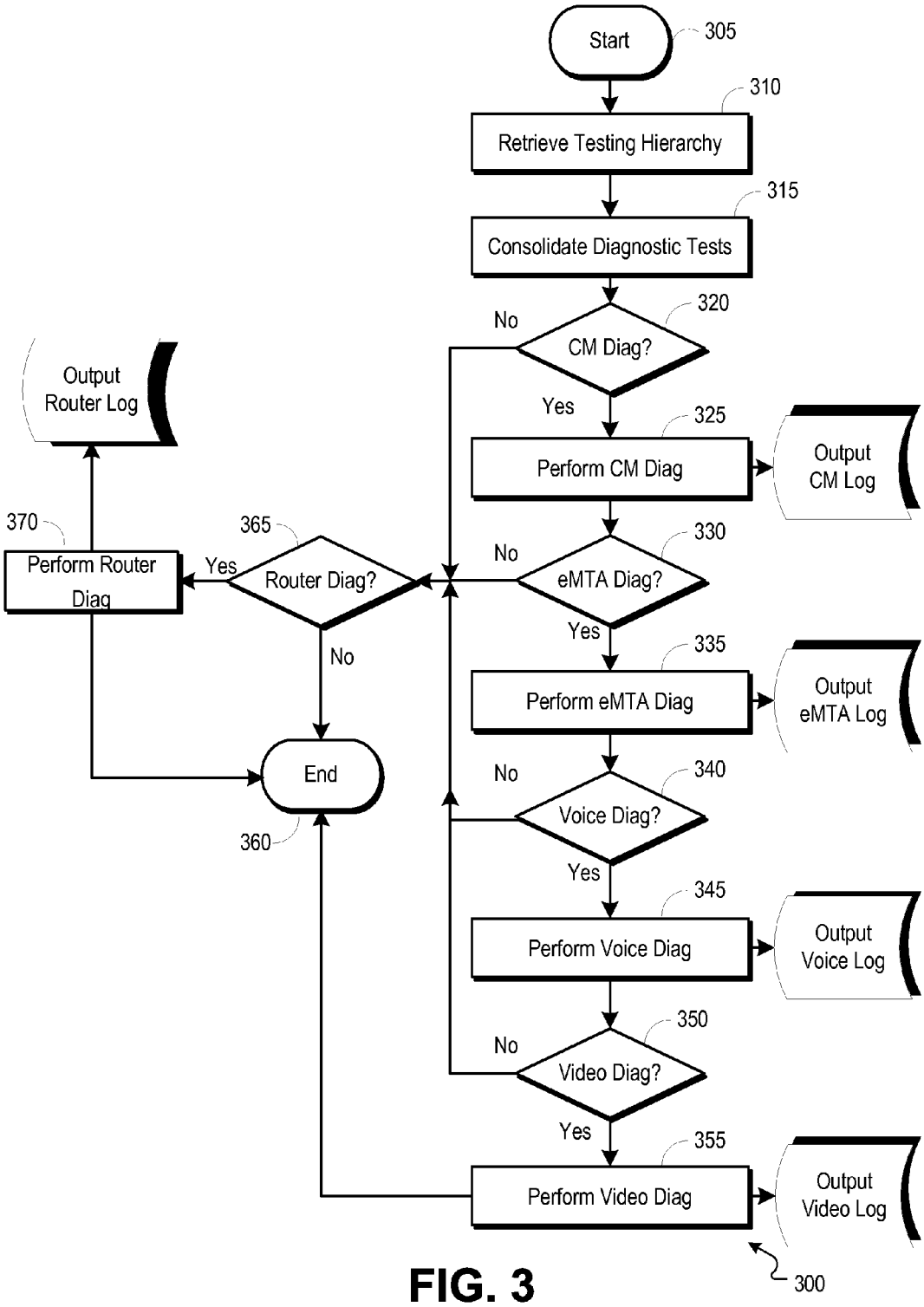

Start
305
310
Retrieve Testing Hierarchy
315
Consolidate Diagnostic Tests
320
No
CM Diag?
Yes
325
Perform CM Diag
Output CM Log
330
No
eMTA Diag?
Yes
335
Perform eMTA Diag
Output eMTA Log
No
340
Voice Diag?
Yes
345
Perform Voice Diag
Output Voice Log
350
No
Video Diag?
Yes
355
Perform Video Diag
Output Video Log
365
Router Diag?
Yes
No
370
Perform Router Diag
Output Router Log
End
360
300

FIG. 3

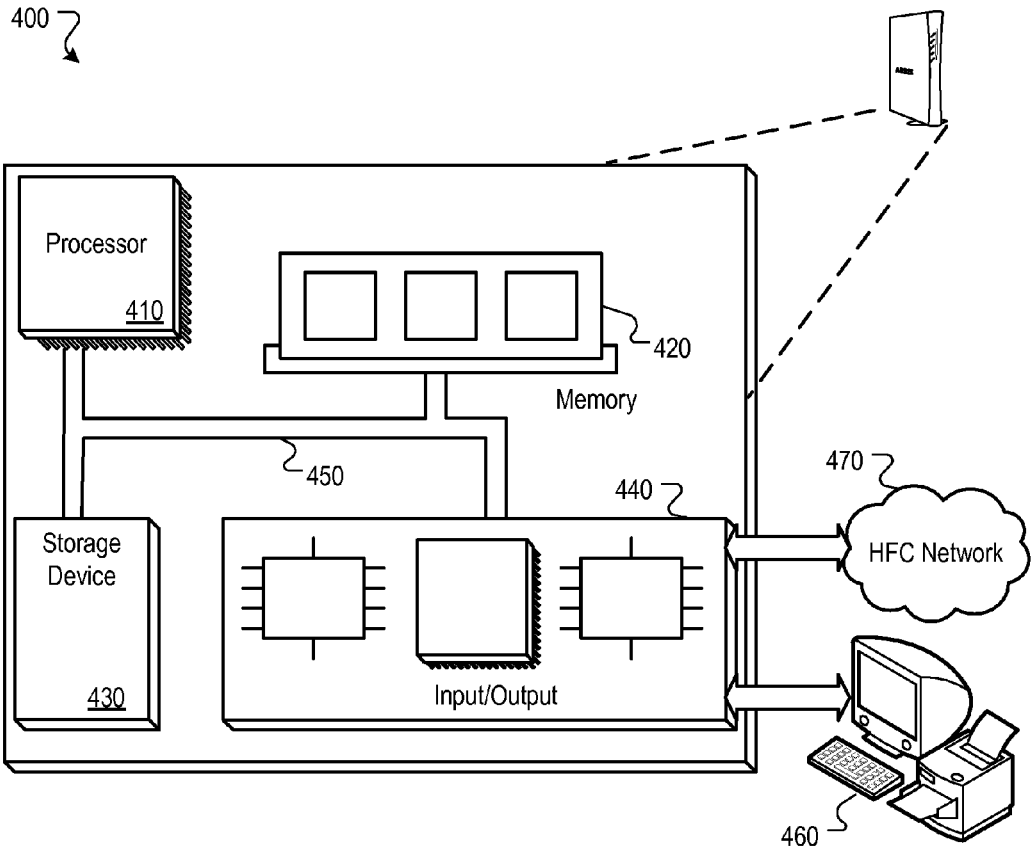

400
Processor
410
420
Memory
450
440
470
HFC Network
Storage
Device
430
Input/Output
460

FIG. 4

QUICKLY DIAGNOSE SERVICE AND COMPONENT RELATED ISSUES ON A CABLE MODEM, MULTIMEDIA TERMINAL ADAPTER, OR GATEWAY

TECHNICAL FIELD

This disclosure relates to diagnosing service-affecting issues in customer premise equipment.

BACKGROUND

The Data-Over-Cable Service Interface Specification (DOCSIS) was established by cable television network operators to facilitate transporting data traffic, primarily Internet traffic, over existing community antenna television (CATV) networks. In addition to transporting data traffic, as well as television content signals over a CATV network, multiple services operators (MSO) also use their CATV network infrastructure for carrying voice, video on demand (VoD) and video conferencing traffic signals, among other types.

When a service-affecting issue occurs in a customer premise equipment (CPE) device, there are few tools available to quickly diagnose the problem without having extensive knowledge of the interworking of the modem. Rather, current diagnostic tools capture traces and use one or more command line interfaces (CLI) to sift through large amounts of data. This results in multiple iterations and additional CLI commands being needlessly executed. Using current diagnostic tools, diagnosing even the simplest issues can take more time than is necessary.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram illustrating an example network environment operable to facilitate diagnostics for service-affecting issues in a CPE device quickly using a single command.

FIG. **2** is a block diagram illustrating an exemplary CPE device operable to diagnose service-affecting issues in a CPE device.

FIG. **3** is a flowchart illustrating an example process operable to provide a quick diagnosis for service-affecting issues in a CPE device.

FIG. **4** is a block diagram of a hardware configuration operable to provide a single command diagnosis for service-affecting issues in a CPE device.

Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

In some implementations of this disclosure, systems and methods can operate to diagnose service-affecting issues in a CPE device quickly by using a single command. When service-affecting issues occur in a CPE device, there are few tools available to quickly diagnose the problem. Diagnosing the problem typically requires knowledge of the inner-workings of the device, along with the execution of multiple CLI commands and knowledge of a nominal value range for responses to the CLI commands.

FIG. **1** is a block diagram illustrating an example network environment **100** operable to facilitate diagnostics for service-affecting issues in CPE devices quickly using a single command. In some implementations, a headend **105** can provide video, data and/or voice service(s) to customer premise equipment (CPE) devices **110***a-d* in one or more subscriber groups (e.g., service group(s)). The CPE devices can include, for example, a cable modem **110***a*, a set top box **110***b*, a wireless router including an embedded cable modem **110***c*, or a media terminal adapter (MTA) **110***d*, among many others (e.g., digital subscriber line (DSL) modem, voice over internet protocol (VoIP) terminal adapter, video game console, digital versatile disc (DVD) player, communications device, etc.). A cable modem **110***a* can facilitate communications between the headend **105** and a computer **115***a*. A set top box **110***b* can facilitate communications from the headend **105** to a television or a digital video recorder. A wireless router **110***c* can facilitate wireless communications between a computer **115***c* and a headend **105**. An MTA **110***d* can facilitate communications between a telephone **115***d* and a headend **105**.

The CPE devices **110***a-d* can communicate with the headend **105** via a hybrid fiber-coax (HFC) network **120**. The headend **105** can include devices such as a cable modem termination system (CMTS) **125** and/or an edge quadrature amplitude modulation (EQAM) device (not shown), or a combined or converged device (not shown) including multiple edge and/or video or data processing functionalities. Such devices can operate to facilitate communications between a network **130** and the CPE devices **110***a-d*. In various implementations, the network **130** can include one or more networks internal to the headend and/or one or more networks external to the headend (e.g., one or more extranets, the Internet, etc.).

Data services can be handled by the headend **105** through a CMTS **125**. The CMTS **125** can receive data signals from external device(s) or nodes through network(s) **130**. The network(s) **130**, for example, can operate using internet protocol (IP), sending data packets to and receiving data packets from the headend **105**. In some examples, the CMTS **125** can be paired with a SIP proxy server (not shown) operable to provide voice over internet protocol (VoIP) services with voice connectivity to other subscribers or voice connectivity to a public switched telephone network (PSTN) (not shown). In still further examples, one or more video sources may provide streaming data through the network(s) **130** to the CMTS **125**.

In some implementations, the CMTS **125** can forward packets destined for subscribers to an EQAM device used to apply the signal to a carrier waveform. In some implementations, the carrier waveform can include either or both data and video streams, in either or both multicast and unicast (e.g., point-to-point) formats for transmission to a combiner, which can combine multiple signals onto a single fiber for transmission to one or more CPE devices **110***a-d* via the hybrid fiber-coax (HFC) network **120**. In other implementations, the CMTS **125** can apply a baseband signal to a carrier wave and transmit the signal to a combiner for upconversion to a transmission frequency.

When a CPE device **110***a-d* initially attempts to connect to the headend **105**, the device **110***a-d* goes through a ranging and registration process with the headend **105**. Ranging typically involves finding and locking onto a signal and determining a timing offset for the device **110***a-d*. The registration process typically includes retrieval of a configuration filename from a dynamic host control protocol (DHCP) server **130** through the network **125**. Upon receipt of the configuration filename, the CPE device **110***a-d* identifies a trivial file transfer protocol (TFTP) server **135** where the configuration file is stored. The CPE device **110***a-d* then requests the configuration file from the TFTP server **135** using the filename provided by the DHCP server. Upon receiving the configuration file, the CPE device **110***a-d* can register with the CMTS **120**.

At times, the CPE device will be unable to connect to the headend. This can occur for various reasons, and diagnosis of the issue can involve initiating several complicated commands with results that may not clearly identify whether there is a problem and can waste valuable time. The inability to connect to the headend can occur for a variety of reasons, e.g., including errors in a cable modem, an embedded multimedia terminal adapter (eMTA), a CMTS, a router, or faults in the HFC network itself.

FIG. **2** is a block diagram illustrating an exemplary CPE device **110** operable to diagnose service-affecting issues in a CPE device quickly by using a single command. The CPE device **110** can include a network interface **210**, a diagnostic logic module **220**, a device results data store **230**, a client interface **240**, and a serial port **250**. The network interface **210** (e.g., an HFC interface) can be used to provide an interface to a network (e.g., an HFC network **120** of FIG. **1**). It should be understood that in other devices, the network interface **210** can be a generic network interface to a local area network (LAN) or wide area network (WAN).

In some implementations, the diagnostics logic module **220** can be used to retrieve a testing hierarchy from the HFC network **120**, or other generic network, execute diagnostics commands associated with the testing hierarchy, compare the results from those commands to sets of nominal results associated with the executed diagnostics commands, and print information found on service-affecting issues. In some implementations, the device results data store **230** can be operable to store information found on service-affecting errors. The client interface **240** is operable to provide a client interface, for example, to a host computer in the case of a cable modem or MTA device, or to a television for a set top box, etc. In some implementations, the serial port **250** can be operable to print or communicate data to a device external to the CPE device.

Execution of diagnostic testing can be started by entering a diagnostic command through an execution source **260** such as command line interface (CLI), hypertext transfer protocol (HTTP) or simple network management protocol (SNMP). In some implementations, when initiating the diagnostic process via HTTP or SNMP, entering a diagnostic command may result in the performance of a deep check of the service that can report the first found issue based upon a testing hierarchy and the particular diagnostic command entered.

In some implementations, when executing the diagnostic commands via CLI, the user can be given two standard command options: diagnostic level and intrusive action query. For example, in response to the diagnostic level option, the user may be able to choose between four different diagnostic levels. The first level may print only information on the lowest-level service-affecting error found. The second level may print information on all service-affecting errors. The third level may print information on all service-affecting errors as well as potential service-affecting warnings. The fourth level may print information on all data examined during execution of the command, including all errors and warnings. The user can choose which diagnostic level to run based on the user's experience with the system or simply the level of thoroughness sought by the user. The user can also be provided the option of allowing the system to run intrusive actions during execution of the command.

In some implementations, the diagnostic command can be service-type commands, for example "cm_diag," "mta_diag," "voice_diag," "router_diag" and "video_diag." The service-type command can be used to perform a status check of the corresponding CPE device/service component by verifying that everything needed for a selected service-type is functioning properly within the component. In this example, the header of each command name corresponds to the type of component to be checked. For example, the "cm" command will check the status of a cable modem, the "mta" command will check the status of the embedded multimedia terminal adapter (eMTA), the "voice" command will check the status of the voice line, and the "router" command will check the status of the router.

In some implementations, the service-type commands can include service diagnostic sub-commands such as "rf_diag," "cmDhcp_diag," "mtaDhcp_diag," among others. These sub-commands can check the status of individual sub-components of the component/service corresponding to the service-type command. When a service-type command is executed, all of the sub-commands associated with that service-type command can be executed. In additional implementations, the sub-commands can be executed separately and individually.

When a diagnostic command is entered via CLI, HTTP, or SNMP, the diagnostic logic module **220** can retrieve a testing hierarchy from the network. Alternatively, the testing hierarchy can be retrieved from a local data store. The testing hierarchy can identify the service-type commands and their corresponding sub-commands in a prioritized order. In some implementations, the service-type commands can build upon one another. For example, when a main service command is entered, the diagnostic logic module **220** can first execute the service-type commands associated with the service-type diagnostic command initiated by the user based upon the testing hierarchy associated with that service-type command. Thus, for example, because voice service can be disrupted by a connection error in the cable modem, when the command corresponding to the voice service is executed, the status of the cable modem will first be checked by invoking the service-type command corresponding to the cable modem service. In some implementations, service-type commands and sub-commands can be added to the testing hierarchy or have their priority within the testing hierarchy altered.

When a command is entered via CLI, HTTP, or SNMP, the diagnostic logic module **220** can execute the specified command, and any other commands implicated by the testing hierarchy. The diagnostic logic can store information found on any service-affecting issues in the device results data store **230** and in some implementations, in an external results data store **270**. The device results data store **230** can be located in the CPE device's non-volatile memory or volatile memory. The external results data store **270** can be located in memory external to the CPE device and can be accessed via the CPE device's network interface **210** or serial port **250**.

After the diagnostic process has been executed, the diagnostic logic module **220** can access the information stored in the device results data store **230** and print or display the information requested by the command to a location external to the device. In some implementations, the information requested by the command can include any of: only the first service-affecting issue found, all service-affecting issues found, or all service-affecting issues and potential service-affecting warnings found during the diagnostic process.

FIG. **3** is a flowchart illustrating an example process **300** operable to diagnose service-affecting issues in a CPE device. The process **300** can start at stage **305** when a diagnostic command is entered, for example, via CLI, HTTP, or SNMP. The command can be entered, for example, using a service interface (e.g., an execution source **260** or serial port **250** of FIG. **2**). The diagnostic command can be a service-type diagnostic command including, for example, "cm_diag," "mta_diag," "voice_diag," "router_diag" or "video_diag." Alternatively, the diagnostic command can be a service-type

diagnostic sub-command including, for example, "rf_diag," "cmDhcp_diag," "mtaDhcp_diag," etc.

At stage **310**, the CPE device can retrieve a testing hierarchy through the HFC network. The testing hierarchy can be retrieved, for example, from a test source (e.g., execution source **260** of FIG. **2** or diagnostics logic **220** of FIG. **2**). In some implementations, the testing hierarchy can list the service-type diagnostic commands and the service-type diagnostic sub-commands in order starting with the lowest-level diagnostic. For example, the service-type diagnostic commands and the service diagnostic sub-commands can be listed in order from the lowest-level diagnostic to the highest-level diagnostic.

At stage **315**, a determination can be made as to what diagnostic tests should be conducted based on the received diagnostic command and those diagnostic tests can be consolidated. The determination can be made, for example, by a diagnostic logic (e.g., diagnostic logic **220** of FIG. **2**). For example, the diagnostics logic **220** can remove any tests from the testing hierarchy that are unnecessary based on the particular diagnostic command received. In some implementations, the diagnostics logic **220** can add to the testing hierarchy diagnostic tests implicated by a received diagnostic command.

At stage **320**, the diagnostic logic can determine whether the diagnostic command initiated is a cable modem service diagnostic command. The determination can be made, for example, by a diagnostics logic (e.g., diagnostics logic **220** of FIG. **2**). In some implementations, the name of the command can be compared with known commands to determine what type of service is associated with the diagnostic command initiated by the user.

If the diagnostic command is a cable modem diagnostic command or is a lower listed diagnostic than the diagnostic command initiated by the user, then the process **300** can proceed to stage **325** and execute the cable modem diagnostic. The cable modem diagnostic can be executed, for example, by a diagnostic logic (e.g., diagnostic logic **220** of FIG. **2**). In some implementations, after executing the diagnostic command, the diagnostic logic can print or write the results of the diagnostic test to a results data store (e.g., an output log).

After executing the cable modem diagnostic, the diagnostic logic can then proceed to stage **330**. At stage **330**, the diagnostic logic can determine whether the diagnostic command initiated by the user is an embedded multimedia terminal adapter (eMTA) diagnostic command or a lower-level diagnostic command. The determination can be made, for example, by a diagnostic logic (e.g., diagnostic logic **220** of FIG. **2**). If the eMTA diagnostic is the commanded diagnostic or is a lower listed diagnostic than the commanded diagnostic, then the diagnostic logic can proceed to stage **335** and execute the eMTA diagnostic. In some implementations, the diagnostic logic can then print or write the information found during the execution of the eMTA diagnostic to the results data store (e.g., an output log).

After executing the eMTA diagnostic, the process **300** can proceed to stage **340**. At stage **340**, the diagnostic logic can determine whether the service-type command initiated by the user is a voice diagnostic command or a lower-level diagnostic. The determination can be made, for example, by a diagnostic logic (e.g., diagnostic logic **220** of FIG. **2**). If the voice diagnostic is the commanded diagnostic or is a lower listed diagnostic than the commanded diagnostic, then the diagnostic logic can proceed to stage **345** and execute the voice diagnostic. In some implementations, the diagnostic logic can then print or write the information found during the execution of the voice diagnostic to the results data store (e.g., an output log).

After executing the voice diagnostic, the process **300** can then proceed to stage **350**. At stage **350**, the diagnostic logic can determine whether the service-type diagnostic command received from the user is a video diagnostic or a lower-listed diagnostic. The determination can be made, for example, by a diagnostic logic (e.g., diagnostic logic **220** of FIG. **2**).

If the video diagnostic is the commanded diagnostic or is a lower listed diagnostic than the commanded diagnostic, then the diagnostic logic can proceed to stage **355** and execute the video diagnostic. The video diagnostic can be executed, for example, by a diagnostic logic (e.g., diagnostic logic **220** of FIG. **2**). In some implementations, the diagnostic logic can then print or write the information found during the execution of the video diagnostic to the results data store (e.g., an output log). After executing the video diagnostic, the diagnostic logic can then proceed to stage **360** where the process **300** ends.

At each of stages **320**, **330**, **340** and **350**, if the associated type of diagnostic command is not implicated by the received service-type diagnostic command, the provisioning logic can proceed to stage **365**.

At stage **365**, the diagnostic logic can determine whether the router diagnostic is the commanded diagnostic or is a lower-listed diagnostic than the commanded diagnostic. This determination can be made, for example, by a diagnostic logic (e.g., diagnostic logic **220** of FIG. **2**).

If the router diagnostic is the commanded diagnostic or is a lower listed diagnostic than the commanded diagnostic, then the process can proceed to stage **370** and execute the router diagnostic. The router diagnostic can be executed, for example, by a diagnostic logic (e.g., diagnostic logic **220** of FIG. **2**). In some implementations, the diagnostic logic can then print or write the information found during the execution of the router diagnostic to the results data store (e.g., an output log). After executing the router diagnostic, the diagnostic logic can then proceed to stage **360** where the process **300** ends. Returning to stage **365**, if the received diagnostic command is not a router diagnostic command or a lower-level diagnostic, the provisioning logic can proceed to the end, stage **360**.

FIG. **4** is a block diagram of a hardware configuration **400** operable to diagnose service-affecting issues in a CPE device quickly by using a single command. However, it should be understood that many different kinds of network devices (e.g., including network hubs, bridges, routers, edge termination devices, etc.) can implement a service-affecting diagnostic mechanism. The hardware configuration **400** can include a processor **410**, a memory **420**, a storage device **430**, and an input/output device **440**. Each of the components **410**, **420**, **430**, and **440** can, for example, be interconnected using a system bus **450**. The processor **410** is capable of processing instructions for execution within the system **400**. In one implementation, the processor **410** is a single-threaded processor. In another implementation, the processor **410** is a multi-threaded processor. The processor **410** is capable of processing instructions stored in the memory **420** or on the storage device **430**.

The memory **420** stores information within the hardware configuration **400**. In one implementation, the memory **420** is a computer-readable medium. In one implementation, the memory **420** is a volatile memory unit. In another implementation, the memory **420** is a non-volatile memory unit.

In some implementations, the storage device **430** is capable of providing mass storage for the device **400**. In one imple-

mentation, the storage device **430** is a computer-readable medium. In various different implementations, the storage device **430** can, for example, include a hard disk device, an optical disk device, flash memory or some other large capacity storage device.

The input/output device **440** provides input/output operations for the hardware configuration **400**. In one implementation, the input/output device **440** can include one or more of a plain old telephone system (POTS) interface (e.g., an RJ11 connector), a network interface device, e.g., an Ethernet card, a serial communication device, e.g., and RS-232 port, and/or a wireless interface device, e.g., and 802.11 card. In another implementation, the input/output device can include driver devices configured to receive input data and send output data to other input/output devices, such as one or more subscriber devices **460** (e.g., set top box, cable modem, etc.), as well as sending communications to, and receiving communications from a network **470**. Other implementations, however, can also be used, such as mobile computing devices, mobile communication devices, set-top box television client devices, etc.

The subject matter of this disclosure, and components thereof, can be realized by instructions that upon execution cause one or more processing devices to carry out the processes and functions described above. Such instructions can, for example, comprise interpreted instructions, such as script instructions, e.g., JavaScript or ECMAScript instructions, or executable code, or other instructions stored in a computer readable medium.

Implementations of the subject matter and the functional operations described in this specification can be provided in digital electronic circuitry, or in computer software, firmware, or hardware, including the structures disclosed in this specification and their structural equivalents, or in combinations of one or more of them. Embodiments of the subject matter described in this specification can be implemented as one or more computer program products, i.e., one or more modules of computer program instructions encoded on a tangible program carrier for execution by, or to control the operation of, data processing apparatus. The tangible program carrier can be a propagated signal or a computer readable medium. The propagated signal is an artificially generated signal, e.g., a machine generated electrical, optical, or electromagnetic signal that is generated to encode information for transmission to suitable receiver apparatus for execution by a computer. The computer readable medium can be a machine readable storage device, a machine readable storage substrate, a memory device, a composition of matter effecting a machine readable propagated signal, or a combination of one or more of them.

The term "system processor" encompasses all apparatus, devices, and machines for processing data, including by way of example a programmable processor, a computer, or multiple processors or computers. The system processor can include, in addition to hardware, code that creates an execution environment for the computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of one or more of them.

A computer program (also known as a program, software, software application, script, or code) can be written in any form of programming language, including compiled or interpreted languages, or declarative or procedural languages, and it can be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or other unit suitable for use in a computing environment. A computer program does not necessarily correspond to a file in a file system. A program can be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A computer program can be deployed to be executed on one computer or on multiple computers that are located at one site or distributed across multiple sites and interconnected by a communication network.

The processes and logic flows described in this specification are performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating output thereby tying the process to a particular machine (e.g., a machine programmed to perform the processes described herein). The processes and logic flows can also be performed by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit).

Processors suitable for the execution of a computer program include, by way of example, both general and special purpose microprocessors, and any one or more processors of any kind of digital computer. Generally, a processor will receive instructions and data from a read only memory or a random access memory or both. The elements of a computer typically include a processor for performing instructions and one or more memory devices for storing instructions and data. Generally, a computer will also include, or be operatively coupled to receive data from or transfer data to, or both, one or more mass storage devices for storing data, e.g., magnetic, magneto optical disks, or optical disks. However, a computer need not have such devices. Moreover, a computer can be embedded in another device, e.g., a mobile communications device, a telephone, a cable modem, a set-top box, a mobile audio or video player, or a game console, to name just a few.

Computer readable media suitable for storing computer program instructions and data include all forms of non volatile memory, media and memory devices, including by way of example semiconductor memory devices, e.g., EPROM, EEPROM, and flash memory devices; magnetic disks, e.g., internal hard disks or removable disks; magneto optical disks; and CD ROM and DVD ROM disks. The processor and the memory can be supplemented by, or incorporated in, special purpose logic circuitry.

To provide for interaction with a user, embodiments of the subject matter described in this specification can be operable to interface with a computing device having a display, e.g., a CRT (cathode ray tube) or LCD (liquid crystal display) monitor, for displaying information to the user and a keyboard and a pointing device, e.g., a mouse or a trackball, by which the user can provide input to the computer. Other kinds of devices can be used to provide for interaction with a user as well; for example, feedback provided to the user can be any form of sensory feedback, e.g., visual feedback, auditory feedback, or tactile feedback; and input from the user can be received in any form, including acoustic, speech, or tactile input.

While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even

initially claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

Similarly, while operations are depicted in the drawings in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

Particular embodiments of the subject matter described in this specification have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results, unless expressly noted otherwise. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In some implementations, multitasking and parallel processing may be advantageous.

What is claimed is:

1. A computer-implemented method, comprising:

receiving a diagnostic command at a device that is configured to provide multiple services, the multiple services comprising data services, video services, voice services and routing services;

retrieving a testing hierarchy, the testing hierarchy being comprised of a plurality of diagnostic tests, wherein each diagnostic test is associated with at least one of the multiple services provided by the device, and wherein the plurality of diagnostic tests are ordered within the testing hierarchy based upon dependencies existing between the diagnostic tests, and the diagnostic tests are further ordered within the testing hierarchy based upon dependencies existing between the multiple services provided by the device;

generating a modified testing hierarchy by removing from the retrieved testing hierarchy one or more diagnostic tests that are associated with a service upon which the service associated with the received diagnostic command is not dependent, such that the modified testing hierarchy includes only those diagnostic tests that are associated with a service upon which the service associated with the received diagnostic command is dependent;

executing a lowest-level diagnostic test below the received diagnostic command in the modified testing hierarchy;

successively executing a next highest-level diagnostic test in the modified testing hierarchy until the received diagnostic command has been executed; and

providing results from each executed diagnostic test.

2. The computer-implemented method of claim **1**, further comprising:

if a service error is identified during any diagnostic test, ending the method prior to completion of the successive execution of diagnostic tests implicated by the modified testing hierarchy; and

providing information associated with the service error found.

3. The computer-implemented method of claim **2**, wherein the service error makes further diagnostic tests associated with the modified testing hierarchy unnecessary.

4. The computer-implemented method of claim **1**, further comprising:

comparing the service test results to ranges of known nominal values; and

flagging results that lie outside the nominal values but within a second range as warnings, and values that lie outside the second range as service affecting errors.

5. The computer-implemented method of claim **1**, further comprising:

providing an output log comprising information on all service-affecting errors and all potential service-affecting warnings found.

6. The computer-implemented method of claim **1**, further comprising:

providing an output log comprising information on all data examined during the diagnostic process.

7. The computer-implemented method of claim **1**, further comprising:

executing intrusive actions during the diagnostic process based upon explicit instruction included in a received diagnostic command.

8. The computer-implemented method of claim **1**, wherein the diagnostic command is a service-type command or a service-type sub-command.

9. The computer-implemented method of claim **1**, further comprising:

consolidating diagnostic tests associated with the received diagnostic command in the testing hierarchy.

10. A system, comprising:

an interface operable to receive and transmit data and commands to or from external modules, through a hybrid fiber-coaxial network;

a data store operable to store computer program instructions and provide temporary storage for the system;

a processor operable to execute said computer program instructions, the computer program instructions being operable to cause the processor to:

receive a diagnostic command, wherein the diagnostic command is associated with a service provided by the system;

retrieve a testing hierarchy, the testing hierarchy comprising a plurality of diagnostic tests, wherein each diagnostic test is associated with at least one service type provided by the system, and the plurality of diagnostic tests are ordered within the testing hierarchy according to dependencies existing between one or more of the service types;

generate a modified testing hierarchy by removing from the retrieved testing hierarchy one or more diagnostic tests that are associated with a service upon which the service associated with the received diagnostic command is not dependent, such that the modified testing hierarchy includes only those diagnostic tests that are associated with a service upon which the service associated with the received diagnostic command is dependent;

execute a lowest-level diagnostic test below the received diagnostic command in the modified testing hierarchy;

successively execute a next highest-level diagnostic test in the modified testing hierarchy until the received diagnostic command has been executed; and

provide results from each executed diagnostic test.

11. The system of claim **10**, wherein said computer program instructions are further operable to cause the processor to:

end the method prior to completion of the successive execution of diagnostic tests implicated by the modified testing hierarchy if a service error is identified during any diagnostic test; and

provide information associated with the service error found.

12. The system of claim **11**, wherein the service error makes further diagnostic tests associated with the modified testing hierarchy unnecessary.

13. The system of claim **10**, wherein said computer program instructions are further operable to cause the processor to:

compare the service test results to ranges of known nominal values; and

flag results that lie outside the nominal values but within a second range as warnings, and values that lie outside the second range as service affecting errors.

14. The system of claim **10**, wherein said computer program instructions are further operable to cause the processor to:

provide an output log comprising information on all service-affecting errors and all potential service-affecting warnings found.

15. The system of claim **10**, wherein said computer program instructions are further operable to cause the processor to:

provide an output log comprising information on all data examined during the diagnostic process.

16. The system of claim **10**, wherein said computer program instructions are further operable to cause the processor to:

execute intrusive actions during the diagnostic process based upon explicit instruction included in a received diagnostic command.

17. The system of claim **10**, wherein the diagnostic command is a service-type command or a service-type sub-command.

18. One or more non-transitory computer readable media operable to execute on a processor, the computer readable being operable to cause the processor to perform the operations comprising:

receiving a diagnostic command at a device that is configured to provide data services, video services, voice services and routing services, the diagnostic command being associated with a service provided by the device;

dynamically constructing a testing hierarchy based upon a service associated with the received diagnostic command, the testing hierarchy being comprised of a plurality of diagnostic tests that are associated with the service associated with the diagnostic command or at least one service provided by the device upon which the service associated with the diagnostic command is dependent, wherein the first diagnostic test in the testing hierarchy is associated with the lowest-level component which can affect the service associated with the received diagnostic command;

executing the lowest-level diagnostic test in the testing hierarchy;

successively executing a next highest-level diagnostic test in the testing hierarchy; and

providing results from each executed diagnostic test.

19. The one or more non-transitory computer-readable media of claim **18**, further operable to cause the processor to perform the operations comprising:

comparing the results to ranges of known nominal values; and

flagging results that lie outside the nominal values but within a second range as warnings, and values that lie outside the second range as service affecting errors.

20. The one or more non-transitory computer-readable media of claim **18**, further operable to cause the processor to perform the operations comprising:

providing an output log comprising information on all data examined during the diagnostic process.

21. The one or more non-transitory computer-readable media of claim **18**, further operable to cause the processor to perform the operations comprising:

executing intrusive actions during the diagnostic process based upon explicit instruction included in a received diagnostic command.

22. The computer-implemented method of claim **1**, wherein the testing hierarchy is retrieved from an upstream network.

23. The one or more non-transitory computer-readable media of claim **18**, wherein the last diagnostic test in the testing hierarchy is associated with the received diagnostic command.

24. The one or more non-transitory computer-readable media of claim **18**, wherein a next highest-level diagnostic test in the testing hierarchy is successively executed until a diagnostic test returns a service-affecting error.

* * * * *