February 12, 2015

COMMAND VERIFICATION

Roman Shuvaev

Anton Muhin

Andrew Warren

Gene Gutnik

Pavel Podivilov

See next page for additional authors

Inventor(s)
Roman Shuvaev, Anton Muhin, Andrew Warren, Gene Gutnik, Pavel Podivilov, Robert Toscano, and Yuri Dolgov

COMMAND VERIFICATION

ABSTRACT

A command verification system verifies validity of a command before execution of the command at a connected device. The system receives the command to be executed at the connected device. The connected device can be any electronic device that is communicatively coupled to the client device that can receive commands from the client device, e.g., a scanner, printer, media streaming device, or storage device. The connected device may also simultaneously receive the command or the system may forward the command to the connected device. The system then receives a verification request from the connected device to verify the validity of the command. On receiving the request, the system determines the validity of the command. If the command is valid, the system verifies the command and transmits instructions to the connected device to execute the command. If the command is not valid, the system does not verify the command and transmits instructions to the connected device to not execute the command.

PROBLEM STATEMENT

Users interact with consumer electronic peripherals, e.g., scanners and printers, that are in physical proximity on local networks. As users increasingly interact with peripherals remotely from non-local networks, reliable command delivery becomes important from a usability perspective. Unreliable command delivery can cause expired or cancelled commands to be executed by the peripheral, resulting in a waste of resources or privacy/security issues. For example, consider a print command that is sent to an offline printer, which later executes the

command when the printer comes online, even after the user has canceled the print command and sent the print command to a different printer. This results in an undesired second printing that the user may not want or realize. A method and system for verifying whether a command is valid before executing the command is described.

COMMAND VERIFICATION SYSTEM

Introduction: The system and techniques described in this disclosure relate to a command verification system that verifies the validity of a command for a connected device before the connected device executes the command. The command verification system can be or include program instructions implemented locally on a client device or implemented across a network, e.g., an Internet, an intranet, cloud infrastructure, or another client device and server environment. The client device can be any electronic device capable of communicating with the system or network, e.g., a laptop, phone, computer, tablet, wearable, printer, scanner, cameras, smart thermostat, smart oven, smart television, etc. A connected device is sometimes referred to herein as a cloud-enabled device or a cloud device.

In one implementation, the system provides a service, which may be referred to herein as a Cloud Device service, that acts as a communicator between clients (e.g., client application or client devices) and devices (e.g. cloud-enabled devices). The Cloud Device service maintains shared command states and allows clients and devices to update the command state atomically. A protocol used by the service ensures that a device, e.g., a cloud-enabled printer, does not execute a command before explicitly permitted by the service. When the device is ready to execute a command, the device sends a special request, e.g., an ACK, with a command ID and

new state (for example, “ready” or “in progress”) to the service. In one embodiment, the Cloud Device service atomically checks that command has not expired and was not canceled by the user, updates the state and gives a green light to the device. If command execution is not desired, the Cloud Device service responds accordingly, e.g., with an error, and the cloud device does not execute the command. Accordingly, the service and/or protocol can ensure that, even with delayed delivery of a command, the device will not execute an expired command or command that was canceled by user. From one perspective, use of the Cloud Device service also allows execution of commands even if a device and user/client are not online at the same time. A user can enqueue a command and go offline. A Cloud Device service client/user can be confident that a submitted command is executed on time, or properly canceled and never executed.

In some implementations, the system may also provide an optimization for light commands that may be executed without acknowledgment. In this case, a device may check, for example, that expiration timestamp associated with the command does not exceed current timestamp. The optimization can be applied to commands that are considered to be safe, e.g., “directory list command,” “read attributes,” etc.

Additional Description: Fig. 1 illustrates an example method 100 for verifying the validity of a command before the command is executed at a connected device. Method 100 can be performed by a system that verifies the validity of the command, e.g., a command verification system.

A user can transmit a command from a client device, e.g., a computer, laptop, tablet, phone, camera, or remote control, to a connected device. The connected device can be any electronic device that is communicatively coupled to the client device and that can receive

commands from the client device, e.g., a scanner, printer, media streaming device, television, thermostat, oven, or storage device. Example commands include a print command from a laptop to printer, a file save or delete command from a phone to a storage device, and a media play command from a camera or remote control application to a media streaming device. As shown in Fig. 1, the system receives the command to be executed at the connected device (block 110). The system can be implemented on a combination of one or more of: the client device, the connected device, and any other devices.

In one embodiment, both the client device and connected device are connected to the command verification system, and the system and the connected device both receive the client device's command that is to be executed at the connected device.

Alternatively, in another embodiment, the system acts as an intermediary, receiving the command and transmitting the command to the connected device. In this manner, the intermediary may manage delivery of one or more commands from the client application to one or more devices, and may manage the execution of commands by one or more devices. According to an example implementation, the intermediary may receive a message from a client application, with the message including a command that is supported and to be performed by the cloud-enabled device. The intermediary forwards the command to be performed to the cloud-enabled device. In another example implementation, the intermediary may determine whether any received commands have been subsequently cancelled, before the device is activated. The intermediary may decide not to forward a received command for a cloud-enabled device to the cloud-enabled device if a command cancellation request or cancel notification for the command has been received from the client application/client device.

One or more devices may be registered with the system, e.g., registered with a Cloud Device service provided by the system. The registered device may identify one or more commands supported by the device and one or more transports supported by the device.

The system receives a request to verify the command from the connected device (block 120). The connected device transmits a verification request to the system that may include a command identifier for the command to be verified. Additionally, the verification request may also include a current state of the connected device.

In some implementations, the connected device may not query the system to verify all received commands. The connected device can determine commands to execute without requesting verification from the system. For example, the connected device may determine to execute commands that do not change the state of the connected device without verification from the system. Examples of commands that do not change the state of the connected device may include retrieving data from a storage device, retrieving photos stored in memory of a digital camera, identifying a current volume setting of a media device or requesting status information of the connected device, e.g., printer, thermostat, oven or other connected household device. Requesting status information of a connected printer may indicate, for example, that the printer is out of paper. Examples of commands that change the state of the connected device may include changing an existing file, creating a new file, or deleting an existing file in a storage device; printing a document at a printer; capturing and storing a photo at a digital camera; changing the volume of a media device; and changing the temperature setting of a thermostat or connected oven.

Additionally or alternatively, the connected device may execute commands that satisfy a threshold command execution time without verification from the system. The connected device can determine whether the execution of the command will take longer than a pre-determined threshold time. If the amount of time is greater than the pre-determined threshold time, the connected device can transmit a command verification request to the system. If the amount of time is less than the pre-determined threshold time, the connected device may directly execute the command. For example, a connected printer can determine that it will take 20 seconds to execute a command to print a document. If the pre-determined threshold time is 60 seconds, the connected printer may execute the command without requesting a command verification from the system. If the connected printer determines that it will take 100 seconds to start executing the print command the connected printer may request a command verification, e.g., immediately with a corresponding status or after the pre-determined threshold time.

On receiving the request to verify the command, the system determines whether the command is valid (block 130). The system may determine a command is valid if the command has not expired, e.g., if a command is associated with an expiration time. The expiration time can identify a period of time after the command is generated for which the command is still valid. The expiration time can also identify a date and time before which the command is valid. For example, a user may issue a print command to a printer that is offline. Ten hours later, the printer may come online and request the system verify whether the command is valid. The system may determine that the command had an expiration time of five minutes, and therefore, the command has expired and is not valid. Additionally or alternatively, the system may determine a command is no longer valid if it has received subsequent contradictory commands. For example, after the

user issues the initial command, the user can issue a subsequent command to cancel the command or transmit the same command to a different connected device. For example, a user may issue a print command to a printer and subsequently issues a command to cancel the print command. The printer requests the system to verify whether the print command is valid. The system determines that because a contradictory command was subsequently issued, the earlier print command is no longer valid. As another example, a user may issue a print command to a printer and subsequently issues a command to another printer on the same floor as the first printer. In some embodiments, the system may have determined that the second printer is commonly used as an alternative to the first printer. When the first printer requests verification from the system that the print command is valid, the system may determine that because of the subsequent command to the second printer, the earlier print command is no longer valid.

If the command is valid, the system verifies the command (block 140). For example, the system may transmit instructions to the connected device to execute the command. If the command is not valid, the system does not verify the command (block 150). For example, the system may transmit instructions to the connected device to not execute the command. The connected device accordingly performs or does not perform the command based on the system's command verification. The method 100 ensures that the connected device does not execute commands before it is verified by the system. This method prevents the connected device from executing expired or cancelled commands.

Fig. 2 illustrates an example scenario of different stages of operation of a command verification system. A system receives a command as shown in step 210 of Fig. 2. The command may be transmitted by a user via a client application/device (not shown in Fig. 2). The client

application/device may also transmit the command to a connected device (step 220). The connected device can be any electronic device that is communicatively coupled to the client device that can receive commands from the client device, e.g., a scanner, printer, digital photo frame, media streaming device (e.g., smart television), or storage device. Upon receiving the command, the connected device queries the system to verify the command. The system receives the request to verify the validity of the command (step 230). The verification request can include a command identifier for the command to be verified and a current state of the connected device.

On receiving the request to verify the command, the system determines whether the command is valid. If the command is valid, the system verifies the command and transmits instructions to the connected device to execute the command (step 240). If the command is not valid, the system does not verify the command and transmits instructions to the connected device to not execute the command (step 240). The system is shown as a stand alone system in Fig. 2; however, the system can be a part of the client device, part of the connected device, or part of a remote server associated with the client device.

A command may have one of several states (command states), including, for example: 1) queued (e.g., command received by the system/service but the cloud device has not started performance or execution of the command); 2) in-progress (command has started execution by cloud device); 3) error (cloud device has reported an error that occurred, e.g., error that occurred while attempting performance of the command); 4) done (cloud device has performed or executed command); 5) paused (performance at the cloud device has been paused); 6) canceled (performance of the command has been cancelled); 7) expired (a timer, or time to live value

associated with the command has expired, indicating that the command should not be performed or executed).

The system and/or service may issue one or more command notifications that instructs a connected device to perform an action with respect to a command, and which may also change a state of a command. For example, a client application may issue, via the system/service, one or more of the following command notifications to the device (e.g., the Cloud Device service receives and forwards the command notification to the device): 1) command created (instructing the cloud device to perform the command, which may change the command state to a state of queued); 2) command expired (indicating to the cloud device that the identified command has expired and should not be performed, and which changes a state of the command to expired); 3) command cancelled (indicating that the command has been cancelled, and if not already started execution/performance, the cloud device should not perform the command, and which causes the state of the command to change to cancelled); 4) command paused (which causes the cloud device to pause or stop the execution or performance of the command, and which changes a state of the command to paused); 5) command resumed (instructing the cloud device to resume a command that was earlier paused, and which changes a state of the command to, e.g., in-progress).

Also, commands submitted or sent to a cloud-enabled device may be stored by the system/service, e.g., for a predetermined period of time or until it is cleared by the user. By storing commands, the system/service can provide a list of all commands to a client application/client device or to an owner of a cloud device, so that the owner can view the commands that have been sent to a cloud device. In such an implementation in which commands

are stored by the system/service, when a cloud-enabled device is activated or connects to the system/service, the cloud-enabled device may send a request for any commands or notifications for that device. The system/service may then send all pending commands (e.g., commands having a command state of queued) to the device. The device may then follow the above-noted procedure (e.g., based on whether the command would change a device state, or would take longer than a threshold period of time to perform) to determine whether the device can execute the command immediately, or should first obtain a command confirmation from the client application via the system/service.

Fig. 3 is a block diagram of an exemplary environment that shows components of a system for implementing one or more of the techniques described in this disclosure. The environment includes client devices 310, servers 330, and network 340. Network 340 connects client devices 310 to servers 330. Client device 310 is an electronic device. Client device 310 may be capable of requesting and receiving data/communications over network 340. Client devices 310 may be the client device and/or connected device described above, depending on the specific device purpose or capability. Example client devices 310 are personal computers (e.g., laptops), mobile communication devices, (e.g. smartphones, tablet computing devices), set-top boxes, game-consoles, embedded systems, and other devices 310' that can send and receive data/communications over network 340. Client device 310 may execute an application, such as a web browser 312 or 314 or a native application 316. Web applications 313 and 315 may be displayed via a web browser 312 or 314. Server 330 may be a web server capable of sending, receiving and storing web pages 332. Web page(s) 332 may be stored on or accessible via server 330. Web page(s) 332 may be associated with web application 313 or 315 and accessed using a

web browser, e.g., 312. When accessed, webpage(s) 332 may be transmitted and displayed on a client device, e.g., 310 or 310'. Resources 318 and 318' are resources available to the client device 310 and/or applications thereon, or server(s) 330 and/or web pages(s) accessible therefrom, respectively. Resources 318' may be, for example, memory or storage resources; a text, image, video, audio, JavaScript, CSS, or other file or object; or other relevant resources. Network 340 may be any network or combination of networks that can carry data communication.

The subject matter described in this disclosure can be implemented in software and/or hardware (for example, computers, circuits, or processors). The subject matter can be implemented on a single device or across multiple devices (for example, a client device and a server device, or multiple peer devices). Devices implementing the subject matter can be connected through a wired and/or wireless network. Such devices can receive inputs from a user (for example, from a mouse, keyboard, or touchscreen) and produce an output to a user (for example, through a display). Specific examples disclosed are provided for illustrative purposes and do not limit the scope of the disclosure.

DRAWINGS

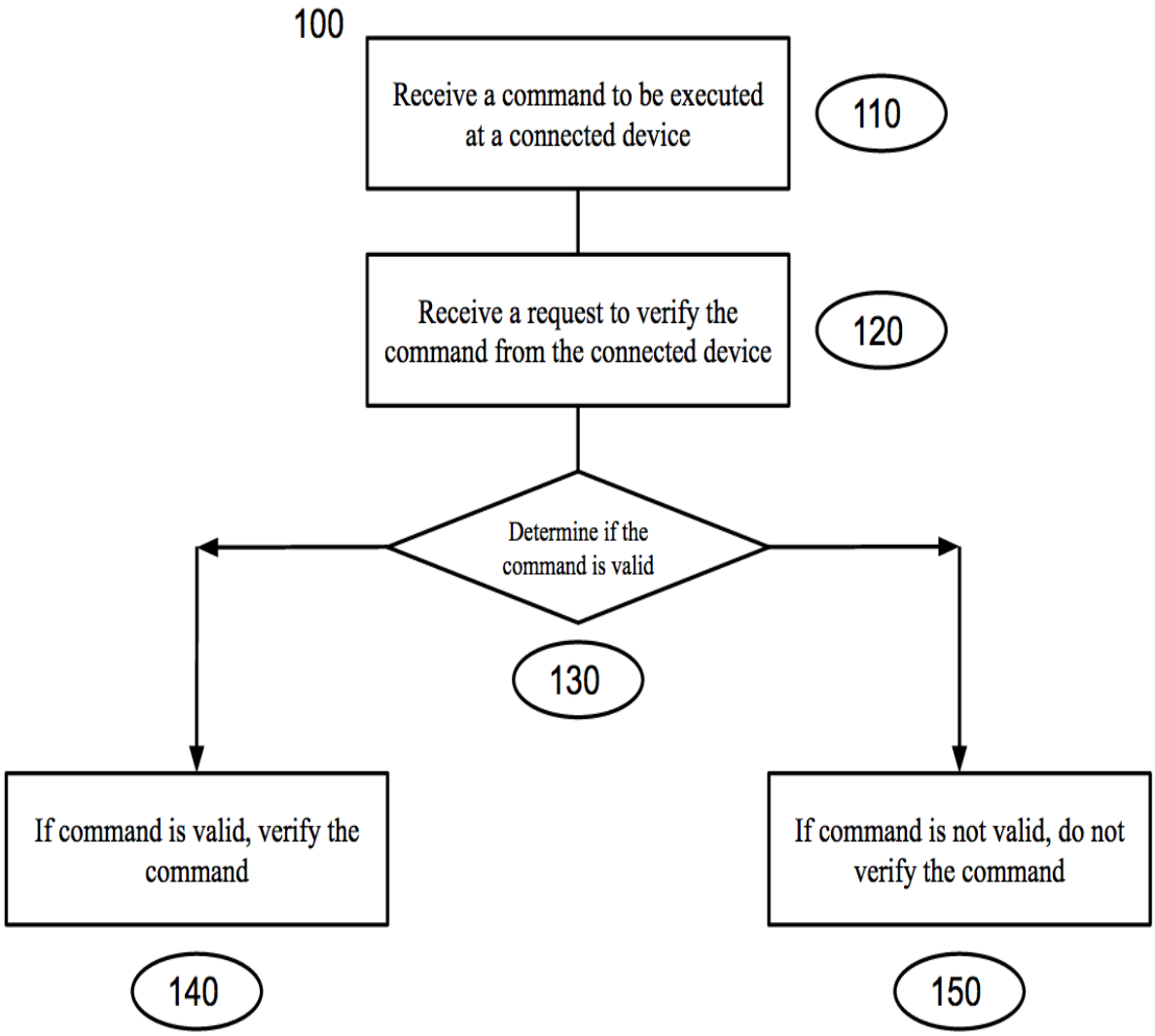

Fig. 1

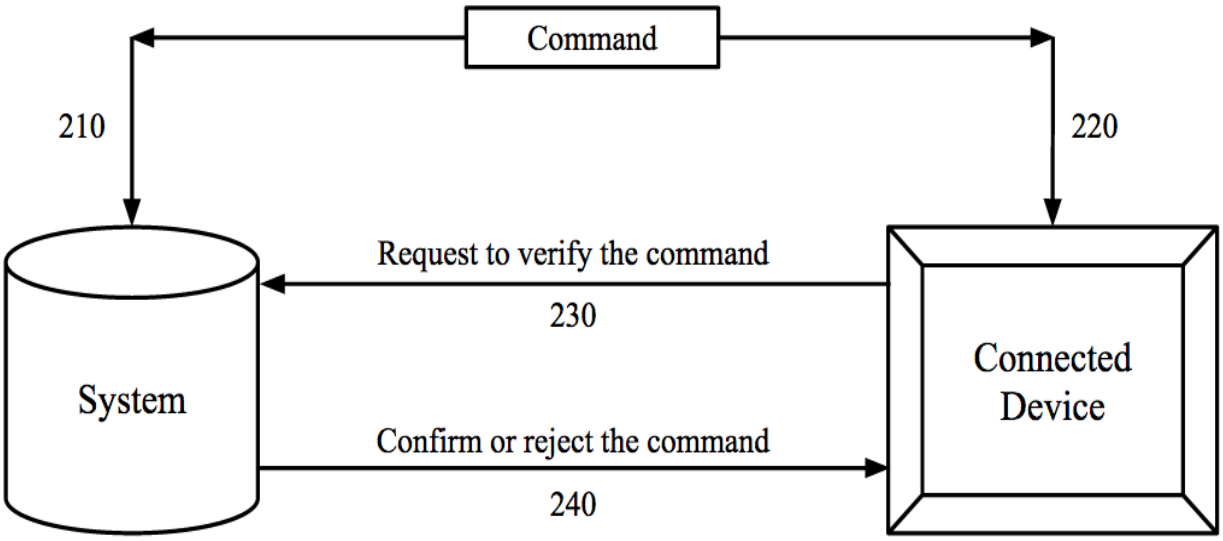
Command
210
220
Request to verify the command
230
System
Connected Device
Confirm or reject the command
240

Fig. 2

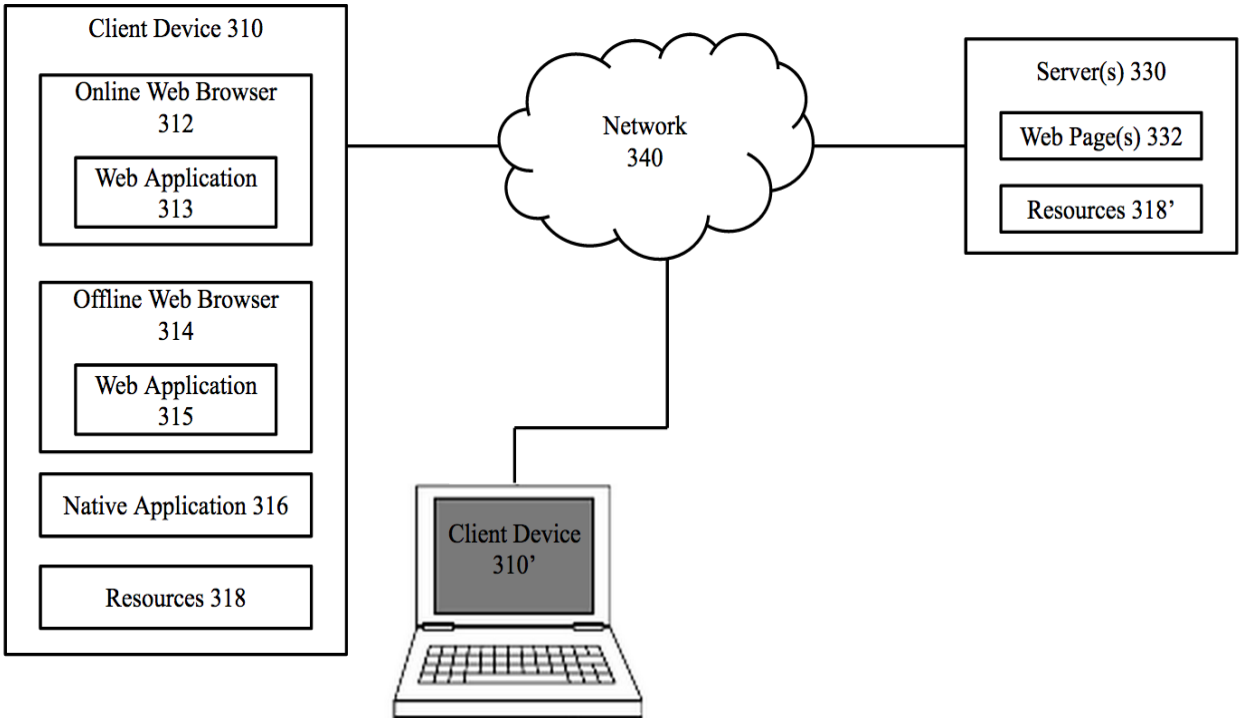
Client Device 310
Online Web Browser 312
Web Application 313
Offline Web Browser 314
Web Application 315
Native Application 316
Resources 318
Network 340
Server(s) 330
Web Page(s) 332
Resources 318'
Client Device 310'

Fig. 3