November 2019

Rapid troubleshooting and management of a network using a chatbot

n/a

Rapid troubleshooting and management of a network using a chatbot

ABSTRACT

Managing a large and complex network is currently tedious and error-prone: administrators log in via lengthy authentication processes; look up and access a diversity of network devices, services, tools and other entities with differing communication protocols; issue relatively obscure, device-specific commands; etc. These problems of network management become especially manifest while troubleshooting critical issues under time constraints. This disclosure describes a fast, simple, and unified approach to network management via a chat interface. At the back end of the chat interface is a bot that is in communication with a variety of network entities in their native protocols. The bot receives aliased, nearly natural-language commands from the administrator via the chat interface, acts on such commands by communicating with relevant network entities, and returns responses via the chat interface. Per the disclosed techniques, the network can be managed over simple, widely available interfaces such as chat over mobile device, and without lengthy or complicated procedures.

KEYWORDS

- Network management
- Network troubleshooting
- Chat interface
- Chatbot
- Software as a service (SaaS)
- Authentication
- Command-line interface (CLI)

BACKGROUND

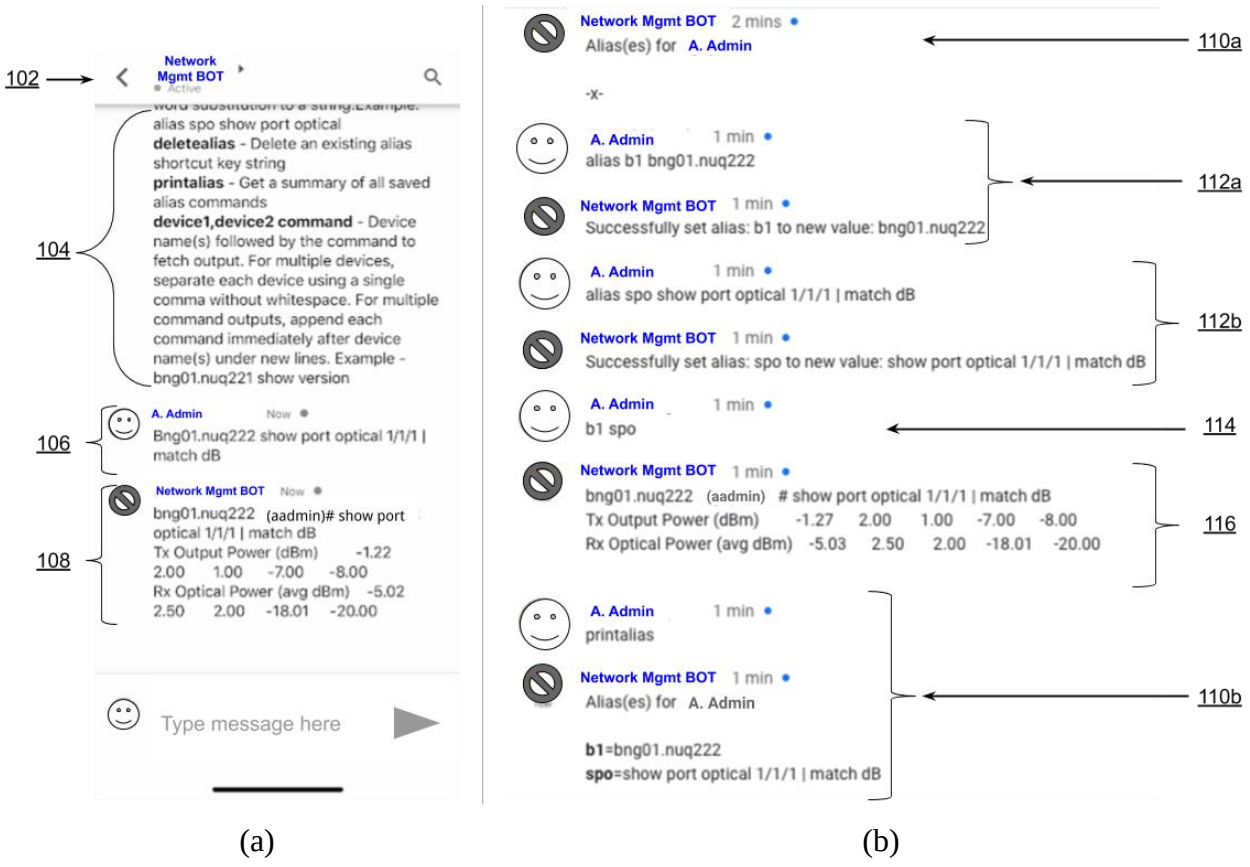
Managing a large and complex network is currently tedious and error-prone. Some examples of procedural frictions associated with network management are below:

- *Authentication*: The authentication procedures for an administrator to log in to a network can be lengthy, complicated, and involve multiple points of failure, e.g., failure of a USB security tokens, failures arising out of network latency or connectivity issuers, etc. Further, authorization for issuing specific commands is often sought manually.
- *A diversity of network entities*: The network typically comprises a large diversity of devices, services, sites, tools, and other entities. Network entities often have unique and obscure commands, and differ in communication, secure-login, or remote procedure protocol. Their addresses are not often easily available. It is not reasonable to expect network administrators to have mastery over the syntax of the plethora of network entities and access protocols. Network administrators often look up manuals or share jotted-down, troubleshooting steps, both of which are stressful during a crisis.
- *Sub-optimal interfaces*: A network administrator that manages several network entities simultaneously has multiple windows open on a small screen, and types multiple similar commands in different windows, thereby risking human error. Simplifying aliases for commands cannot be used, as these are not generally accepted by network entities, which have their own native command-line interfaces.
- *Difficulties of off-site or remote network management*: A field-operations network administrator may not easily be able to access the network from an off-site location due to the lack of availability of a personal computer, laptop, or other client with VPN or other company-authorized secure shell.

- *Managing lengthy logs generated by network entities*: Outputs created by network devices are typically large and detailed and are not amenable to reading by scrolling. To effectively use them to point to network issues, the outputs need to be frictionlessly collated, and their essential information extracted, stored, and shared between members of the network management team.
- *Workflow management*: Configuration changes that are pushed to network entities currently require detailed knowledge of the entity in order to avoid missing commits or change-contexts, and be generally effective. Troubleshooting dashboards are not generally available or shareable across peers in the form of a unified user interface. Workflows with variations or deviations from standard architecture are not easy to automate.

DESCRIPTION

This disclosure describes a fast, simple, and unified approach to network management via a chat interface. At the back end of the chat interface is a bot, e.g., a software-as-a-service (SaaS), that is in communication with a diversity of network entities in their native protocols. The bot receives aliased, nearly natural-language commands from the administrator via the chat interface, acts on such commands by communicating with relevant network entities, and returns responses via the chat interface.



(a) (b)

Fig. 1: Network management via a chat interface: (a) Sending a command and receiving a response over a mobile-device chat; (b) Working with aliased commands over a laptop chat

Fig. 1 illustrates examples of network management via chat interface, per techniques of this disclosure. In Fig 1(a), a network administrator ("A. Admin") manages the network by chatting with a network management bot ("Network Mgmt BOT") over a mobile-device chat interface (102). Upon the administrator's request, the network management bot provides a command manual (104). The administrator issues a command (106). The bot executes the command, communicating with or configuring network entities as necessary. The bot returns the results (108) and/or log-messages, both of which are stored in a searchable repository accessible by members of the network management team. If the results or log messages are lengthy, a link to the results or log messages is posted to the chat interface. Native features of the chat interface such as threads, chat-groups, user groups, etc. can be used to separate contexts, to view multiple

sessions from different network entities, to view image-based or card-based responses to chat commands, etc.

Fig. 1(b) illustrates examples of working with aliased commands over a laptop chat interface. Aliases are convenient short-forms, similar to those found in a .cshrc file in Unix, for lengthy CLI commands that may have complex arguments. Aliases are not typically enabled natively by network devices; the techniques herein leverage the bot to expand an alias into language recognized by the network device. The network administrator requests a screen-print of known aliases (110a-b). The administrator defines new aliases (112a-b). The administrator issues aliased commands (114). The network management bot responds to aliased commands (116). Aliases can be authored by a network administrator and are maintained in a searchable repository (alias dictionary) that is accessible by members of the network management team. The alias dictionary can be loaded by a member of the network management team, thereby enabling team members to build on each others' dictionary entries. Aside from parsing aliased commands, e.g., performing tabular short-to-long-form translations, the bot can also partially parse natural-language commands and translate them to the language of the network device, service, site, tool, or other entity.

Fig. 2: The button feature (a) Pushing configuration to network entities with a commit/cancel button (b) A button that encapsulates a lengthy command sequence

Fig. 2 illustrates efficient encapsulation of lengthy command sequences via clickable (or touch-to-click) boxes or buttons, per techniques of this disclosure. In Fig. 2(a), the administrator issues a command to push configuration to a network entity (202), without needing to know the details of the network entity. The chatbot responds with a button that offers the administrator options to confirm or to cancel the changes (204). The confirm comments are tagged and logged. In Fig. 2(b), a button (206) encapsulates a lengthy command sequence. Buttons can be authored by a network administrator and shared with team members in a searchable repository.

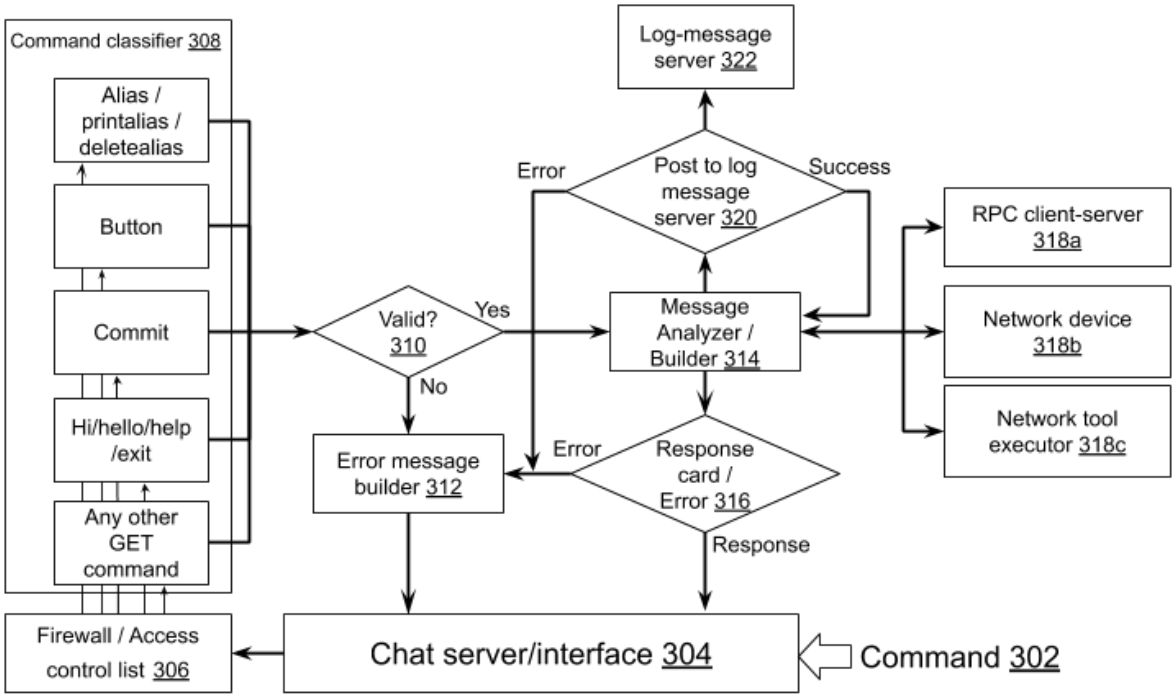


Fig. 3: Lifecycle of a command issued over chat interface

Fig. 3 illustrates the lifecycle of a network management command, e.g., the processing of a command by the chatbot, per techniques of this disclosure. A network administrator issues a command (302) using the chat interface (304). The command passes a firewall and an access control list (306), where the administrator is authenticated and checked for having the authority to issue the command. A command classifier (308) classifies the command into categories such as alias, print-alias, delete-alias, button, commit, hi/hello/help/exit, other get command, etc.

The command is tested for validity (310). If found to be invalid, an error-message builder (312) builds an error message and posts it on the chat interface. If the command is found to be valid, a message analyzer/builder (314) de-aliases the command, parses natural-language components of the command, and translates the command to the native language of the targeted

network service, tool, device, site, or other entity. To parse the natural-language components of the command, the message analyzer/builder can use machine learning models.

After processing by the message analyzer/builder, the command is transmitted to one or more of the following targets.

- Network services (318a) such as intent-based networking services, etc. for purposes such as onward-and-return communication with running network services, etc. using RPC or other protocols. The response to generic RPC commands can be sent back to the administrator as a bot-initiated message. Generic RPC client-server can be used to follow a workflow, whereby chained operations are performed in order per directions of the administrator. The administrator is thereby enabled to perform controlled workflows per the inputs provided to the message analyzer/builder through chat messages.
- Network devices (318b) such as routers, switches, etc. for purposes such as configuring a network device, changing the service-level output of a network device, troubleshooting a network device, interrogating a network device, onward-and-return communication with a network device, etc.
- Network tools (318c), e.g., binaries, the execution of which result in running diagnostic tests on network devices, testing and reporting on the status of network sites or services, configuring or bringing up of one or more devices, etc.

The network service, device, tool, or other entity returns a response or log-message to the message analyzer upon the completion of execution of the command. The message builder posts the log-message to a log-message server (322). If the posting of the message results in an error (320), then the error-message builder is invoked to post an error message on the chat interface. If the posting of the message to the log-message server is successful, then a response card builder

(316) creates a response card with a link to the log message and posts the response card to the chat interface. Alternatively, the response card builder can also create a button or a text message to post over the chat interface. The log-message server enables lengthy log messages to be read under the limited character-length constraints of a chat interface.

The techniques of this disclosure enable many use-cases that were previously difficult to achieve, as illustrated in the following examples.

- A remote field operations technician can debug problems with just a mobile phone, without having to rely on the network operations center. A central operations engineer can store a button or alias that can be sent to the field technician by chat message, and used by the field technician using the same chat interface.
- Staging, canary, or nightly network tests that are accessed within chats give valuable hands-on experience with troubleshooting. A newly on-boarded engineer can quickly come to speed with troubleshooting content.
- Content can be quickly uploaded to or downloaded from multiple devices by asynchronously interacting with the bot. By separating incoming data using native chat functionality such as threads, groups, and users, an administrator can quickly handle multiple requests, rather than having to access devices individually using traditional methods.

In this manner, network management and troubleshooting can be carried out over a standard, widely available, widely understood chat interface in a variety of popular form-factors such as mobile device, laptop, etc. Authentication to perform network management activities is handled by the authentication procedures of the chat application itself. Alternatively, a one-time key-based authentication can be obtained via the chat interface.

Thus, in time-critical troubleshooting episodes, a network administrator can quickly target the problem without lengthy authentication procedures. The techniques offer a unified command interface to the network that abstracts the large diversity of network devices, sites, tools, services, and other entities. Since the network administrator works within the chat interface, a chat message format can be as simple as `command device`, with no further identifying or authenticating parameters, Additional provisions are provided for further simplification using buttons and aliasing. The maintenance of a large and complex network is transitioned from manual to nearly zero-touch operation.

CONCLUSION

This disclosure describes a fast, simple, and unified approach to network management via a chat interface. At the back end of the chat interface is a bot that is in communication with a variety of network entities in their native protocols. The bot receives aliased, nearly natural-language commands from the administrator via the chat interface, acts on such commands by communicating with relevant network entities, and returns responses via the chat interface. Per the disclosed techniques, the network can be managed over simple, widely available interfaces such as chat over mobile device, and without lengthy or complicated procedures.