2024 7th International Conference on Information and Computer Technologies (ICICT) | 979-8-3503-8562-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICICT62343.2024.00084

Development of Dialogue Feature between Participants and ChatGPT in Network Security Exercise System

1st Yuichiro Tateiwa
Graduate School of Engineering
Nagoya Institute of Technology
Nagoya, Japan
tateiwa@nitech.ac.jp

***Abstract*—The author confirmed that ChatGPT can provide advice including specific implementation examples on computer network management by dialogue with ChatGPT. According to this confirmation, it is expected that in network security exercises, ChatGPT will be able to resolve participants' network troubles in place of instructors or teaching assistants. In the network security exercises provided by the author, participants perform various communication experiments in a virtual network consisting of virtual machines as nodes. This paper describes a method for collecting network configuration information and conveying it to ChatGPT based on a custom YANG model, as well as a user interface for facilitating dialogue between participants and ChatGPT.**

***Index Terms*—ChatGPT, Network, Security, e-learning, Exercise, YANG model, Large Language Models, Virtual Machine, Dialogue**

I. Introduction

ChatGPT [1] possesses extensive knowledge of computer science and has demonstrated the ability to answer questions related to the field. In practice, the author confirmed this through dialogue with ChatGPT that it can provide advice on the construction of basic networks and the operation of secure networks, including specific implementation examples. According to this confirmation, it is expected that ChatGPT will assist in resolving issues encountered by participants in designing networks and implementing attacks and defenses in network security exercises.

In the network security exercises provided by the author, participants build networks consisting of Linux devices (e.g. servers and firewalls) and conduct attacks and defenses within these networks. The exercise environment, LiNeS Cloud [2], realizes virtual networks with virtual machines as nodes on a server, and participants manipulate these networks through a web browser. Therefore, collecting network configuration information on the server is not difficult. Additionally, the author proposed a notation for sharing network configuration information between applications and ChatGPT [3].

This paper proposes a feature for participants in LiNeS Cloud to dialogue with ChatGPT about their own networks. Specifically, it describes methods for collecting network configuration information, details a notation for sharing network configuration information with ChatGPT [3], and discusses the structure of dialogues with ChatGPT as well as the user interface used for these interactions.

II. LiNeS Cloud

User-mode Linux [4] (hereafter referred to as UML) is a process that runs on Linux and virtually realizes a Linux computer. When the UML program is executed with a distribution image as an argument, a process that realizes the Linux of that distribution is launched. Even users without administrative rights on the UML host OS can have administrative rights within UML, allowing them to install software and provide services. By connecting UML's network interface with Linux's Bridge, a virtual network can be realized on that host computer.

The author has previously proposed LiNeS Cloud, a web-based exercise system for network security exercises using UML, which aims for intuitive and seamless operation and lightweight responsiveness, as detailed in past research [2]. In LiNeS Cloud, virtual devices such as servers, clients, routers, and firewalls are realized with UML, and switching hubs are realized with Linux's Bridge.

In the system configuration shown in Fig. 1, squares represent computers, and solid lines represent communication

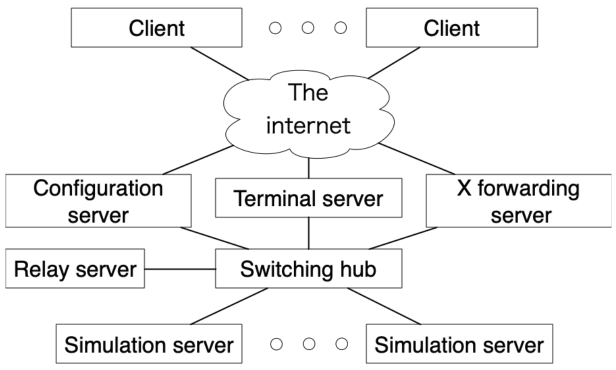
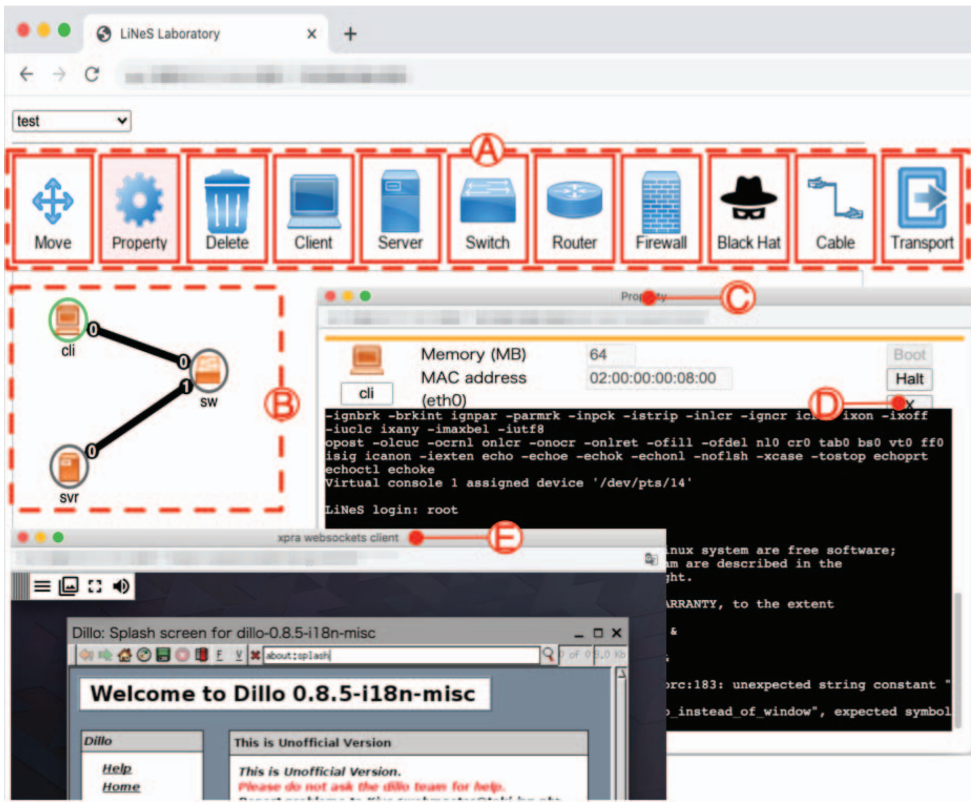


Fig. 1. System overview of LiNeS Cloud.



Memory (MB) 64
MAC address (eth0) 02:00:00:00:08:00

```
-ignbrk -brkint ignpar -parmrk -inpck -istrip -inlcr -igncr ic    ixon -ixoff
-iuclc ixany -imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon -iexten echo -echoe -echok -echonl -noflsh -xcase -tostop echoprt
echoctl echoke
Virtual console 1 assigned device '/dev/pts/14'

LiNeS login: root

                                    inux system are free software;
                                    am are described in the
                                    ght.

                                    ARRANTY, to the extent

                                    &


                                    rc:183: unexpected string constant "

                                    _instead_of_window", expected symbol
```

This is Unofficial Version.
Please do not ask the dillo team for help.

Fig. 2. Example of web pages for managing networks.

paths. The Simulation server accepts remote operations from the Configuration server and operates virtual devices to form a network. The Configuration server provides web pages for editing the connections between virtual devices, for input and output of UML terminals, and for input and output of UML's X clients. The Relay server connects the tunnel endpoints of the Ethernet tunnel created between itself and the Simulation server with Linux's Bridge, enabling the exchange of Ethernet frames between the tunnel endpoints of the two Simulation servers. The Terminal server relays input and output between the terminal page and UML's terminal. The X forwarding server acts as an X server, accepting rendering requests from UML's X clients and rendering their output onto an X page. It also accepts user input from the X page and transmits it to the X client. Through a web browser on the Client, a participant manages the network.

Fig. 2 shows examples of clients in Google Chrome for Mac, namely a topology page, a terminal page, and an X page sorted from the top. After selecting a device from the available icons Ⓐ, clicking on the area Ⓑ will draw the selected device. The window Ⓒ displays the property of the device cli and the terminal area at the bottom draws a UML terminal running on a simulation server and accepts keyboard input. Clicking the button Ⓓ displays the window Ⓔ that shows the X window manager of the device cli. In the window Ⓔ, the browser (X client) started in the device cli is displayed. Furthermore, the icons representing the devices in the area Ⓑ can be moved by mouse dragging.

III. Implementation Method

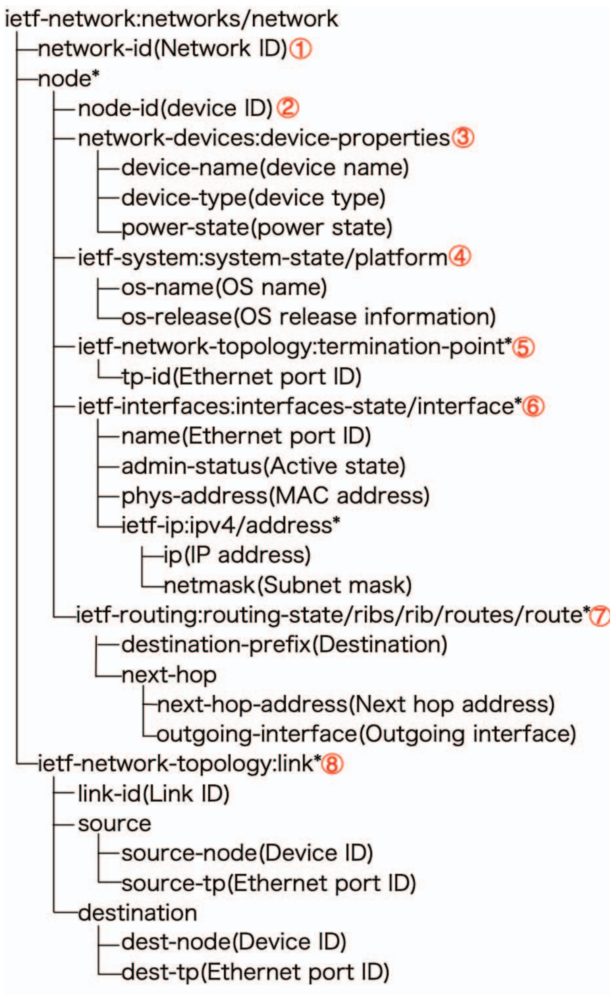
A. Describing Networks based on a Custom YANG Model

When participants dialogue with ChatGPT about their network, it is necessary to inform ChatGPT about the network at the beginning of the conversation. However, having participants do this themselves can lead to reduced exercise efficiency. To prevent this, the author adopts an approach where LiNeS Cloud communicates the participant's network to ChatGPT.

The author proposed a notation for describing network configuration information shared between applications and ChatGPT in the paper [3]. This notation is useful not only for transmitting network configuration information from applications to ChatGPT but also in the reverse direction. The reason for this is that while ChatGPT excels in natural language processing, many applications are better suited for processing artificial languages. In other words, when ChatGPT expresses network configuration information using this notation, it becomes easier for applications to interpret and utilize this information. Not only for the purpose of LiNeS Cloud (i.e., the application) conveying the participant's network to ChatGPT, but also considering future uses such as participants

```
module network-devices {
  import ietf-network { prefix inet; }
  augment "/inet:networks/inet:network/inet:node" {
    container device-properties {
      leaf device-name {
        type string;
        description "The name of the device.";
      }
      leaf device-type {
        type enumeration {
          enum "L2-switch";
          enum "linux-server";
          enum "linux-client";
          enum "linux-firewall";
          enum "linux-router";
        }
        description "The type of the device.";
      }
      leaf power-state {
        type enumeration {
          enum "on";
          enum "off";
        }
        description "The power state of the device.";
      }
    }
  }
}
```

Fig. 3. Custom YANG module.



```
ietf-network:networks/network
 ├network-id(Network ID)①
 └node*
   ├node-id(device ID)②
   ├network-devices:device-properties③
   │ ├device-name(device name)
   │ ├device-type(device type)
   │ └power-state(power state)
   ├ietf-system:system-state/platform④
   │ ├os-name(OS name)
   │ └os-release(OS release information)
   ├ietf-network-topology:termination-point*⑤
   │ └tp-id(Ethernet port ID)
   ├ietf-interfaces:interfaces-state/interface*⑥
   │ ├name(Ethernet port ID)
   │ ├admin-status(Active state)
   │ ├phys-address(MAC address)
   │ └ietf-ip:ipv4/address*
   │   ├ip(IP address)
   │   └netmask(Subnet mask)
   └ietf-routing:routing-state/ribs/rib/routes/route*⑦
     ├destination-prefix(Destination)
     └next-hop
       ├next-hop-address(Next hop address)
       └outgoing-interface(Outgoing interface)
 └ietf-network-topology:link*⑧
   ├link-id(Link ID)
   ├source
   │ ├source-node(Device ID)
   │ └source-tp(Ethernet port ID)
   └destination
     ├dest-node(Device ID)
     └dest-tp(Ethernet port ID)
```

Fig. 4. Custom YANG model.

having ChatGPT design the network, this study implements the dialogue function based on the notation [3].

The notation is a custom YANG model based on the YANG language [5]. YANG is an artificial language suitable for describing network configurations, but its specifications are described in natural language and widely published on the Internet. The author confirmed that ChatGPT can describe basic network settings with reasonable accuracy based on existing YANG models. However, in the YANG models published in RFC [6], there was no single model found that could describe the entire configuration of a network for the exercises.Therefore, the author extended the ietf-network [7] with other YANG modules and further addressed the deficiencies by expanding it with custom YANG modules that the author newly defined for this purpose. However, although the former extension does not follow the rules of YANG modeling, ChatGPT seemed to understand this extension with reasonable accuracy.

Fig. 3 shows the custom YANG module with the additionally defined deficiencies, and due to space constraints, some parts have been omitted. The node container in the ietf-network module now includes a new container element named device-properties. The device-properties container consists of the leaf device-name that stores the device name as a string, the leaf device-type that stores the type of device as an enumerator, and the leaf power-state that stores the power state of the device as an enumerator.

The proposed YANG model is shown in Fig. 4, based on the tree diagram [8]. The strings in parentheses following the elements represent the components of the exercise network to be conveyed to ChatGPT. ① and ② are elements of the *ietf-network* module. ③ is an element added by the extension in the custom YANG module shown in Fig. 3, and ⑤ and ⑧ are elements added by the extension in the *ietf-network-topology* module [7]. ④, ⑥, ⑦ are elements defined in ietf-system [9], ietf-interfaces [10], and ietf-routing [11], respectively, but the extensions of ietf-network are not declared in each module's definition. In other words, treating ④, ⑥, ⑦ as elements of ietf-network does not follow the rules of YANG. However, due to the concept of YANG and the flexibility of ChatGPT's interpretation of YANG, it is expected that ChatGPT can interpret this custom YANG model as intended.

B. Collection of Data for Transmission to ChatGPT

Consider collecting the network configuration information shown in Fig. 4. The data required for ① to ⑤ and ⑧ in the figure are stored in the database of LiNeS Cloud. The data needed for ⑥ and ⑦ are managed by UML and can be collected by executing Linux commands on the UML console.

When collecting the necessary data for ⑥ and ⑦ from UML, the following challenges can be identified:

1) Not using the participant's console: The participant's terminal client is connected to the UML console via the Terminal server. One straightforward approach to consider for the data collection involves disconnecting the terminal client from the Terminal server and subsequently allowing the data collection application to utilize the UML console through the Terminal server. However, this approach may interrupt the exercise as participants cannot use UML while the application is collecting data.
2) Not polluting the shell command history: During exercises, participants may refer to their past commands. If the commands used for collection remain in the history, it could decrease the participants' work efficiency or cause confusion.

A solution for Challenge 1 is that a pseudo-terminal is connected to a different console of UML (one that is not being used by the participants), and then the data collection application executes the necessary commands for collection through the pseudo-terminal on that console. This can be achieved through the following steps:

1) The Terminal server starts a UML with the added boot option "con1=pty".
2) The UML executes a shell program with tty1 as the controlling terminal.
3) The device file for the slave side of the pseudo-terminal is extracted from the UML's boot log.
4) A session of screen [13] is created with the device file as an argument.
5) By connecting to the session, the application can execute Linux commands in the UML and retrieve the necessary data (as shown in ⑥ and ⑦ of Fig. 4).

A solution for Challenge 2 is to set the default shell in UML to Bash [14]. Bash has frequently been chosen as the default shell in many Linux distributions. When executing commands in Bash, setting the environment variable "HISTCONTROL" to "ignorespace" and then prefixing the command with a space ensures that the command is not recorded in the execution history. Therefore, by setting the shell in UML to Bash and using it as described, Challenge 2 can be resolved.

IV. Prototype System

Fig. 5 shows the main data flow used when a participant interacts with ChatGPT. When a participant enters questions for ChatGPT or commands for LiNeS Cloud in the Web browser, the Web browser sends them (the Request) to the Node.js [15]. The Node.js's behavior changes according to the content of the Request. When conveying the participant's network to ChatGPT, the Node.js retrieves data for ① to ⑤ and ⑧ from the Database and collects data for ⑥ and ⑦ from the UML. Then, it converts the data from ① to ⑧ into prompts and sends them to ChatGPT, and sends the results to the Web browser. When conveying the participant's message to ChatGPT, the Node.js sends the received message (the Request) to ChatGPT and sends the obtained response (the Reply) to the Web browser.

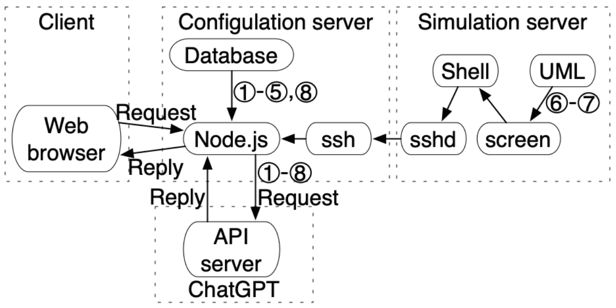
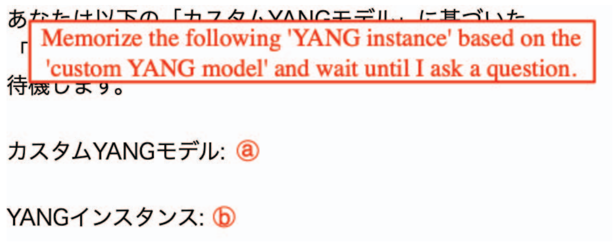


Fig. 5. Data flow during dialogue.



あなたは以下の「カスタムYANGモデル」に基づいた
「
待機します。

Memorize the following 'YANG instance' based on the 'custom YANG model' and wait until I ask a question.

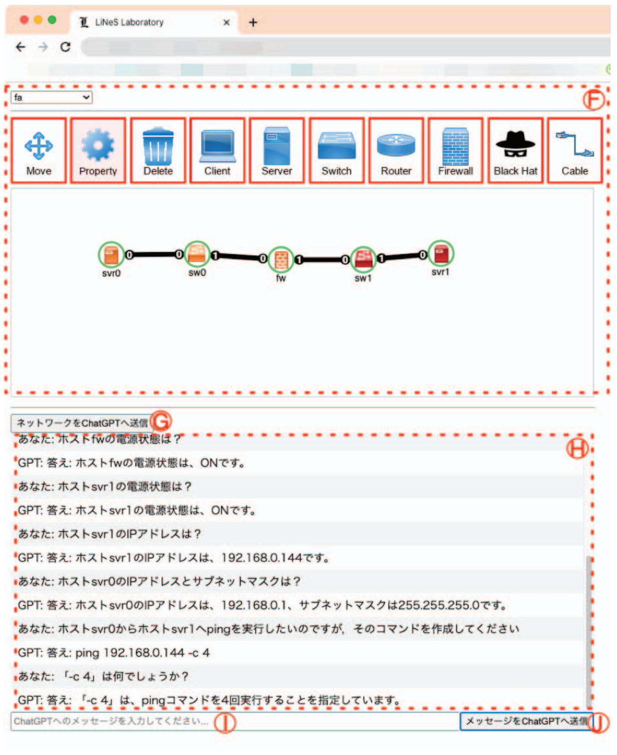
カスタムYANGモデル: ⓐ

YANGインスタンス: ⓑ

Fig. 6. Prompt to ChatGPT.

Fig. 6 shows the prompt used by the Node.js when sending ① to ⑧ to the API server in Fig. 5. The custom YANG model (Fig. 4) is described in ⓐ, and a YANG instance based on ① to ⑧ is described in ⓑ.

Fig. 7 shows an example of the dialogue function on a web browser launched on a client. A participant creates a network topology in the area Ⓕ and configures each device in separate windows. To initiate a dialogue with ChatGPT about the network, the participant first needs to press button Ⓖ to have the system convey this network information to ChatGPT. Then, the participant enters a message in the text box Ⓘ and press the button Ⓙ to have the system convey the message to ChatGPT. Finally, when the system receives a response from ChatGPT, the content is added to the area Ⓗ. The current state in the area Ⓗ shows inquiries about the power status of devices, IP addresses, and requests to generate ping commands. ChatGPT (GPT-4) is fulfilling these requests with reasonable accuracy.



```
あなた: ホストfwの電源状態は？
GPT: 答え: ホストfwの電源状態は、ONです。
あなた: ホストsvr1の電源状態は？
GPT: 答え: ホストsvr1の電源状態は、ONです。
あなた: ホストsvr1のIPアドレスは？
GPT: 答え: ホストsvr1のIPアドレスは、192.168.0.144です。
あなた: ホストsvr0のIPアドレスとサブネットマスクは？
GPT: 答え: ホストsvr0のIPアドレスは、192.168.0.1、サブネットマスクは255.255.255.0です。
あなた: ホストsvr0からホストsvr1へpingを実行したいのですが，そのコマンドを作成してください
GPT: 答え: ping 192.168.0.144 -c 4
あなた: 「-c 4」は何でしょうか？
GPT: 答え: 「-c 4」は、pingコマンドを4回実行することを指定しています。
ChatGPTへのメッセージを入力してください...
```

Fig. 7. Execution example of dialogue function.

V. Verification Experiment

The objective of this experiment is to verify whether the methods described in Section III have been correctly implemented in the prototype system, and whether their implementation, as interconnected like in Fig. 5, can facilitate dialogue between participants and ChatGPT. In the experiment, the following five points are verified for two basic networks in the exercise:

1) Can the necessary data be collected from the network?
2) Can the custom YANG model be correctly generated based on the collected data?
3) Can the generated custom YANG model be successfully transmitted to ChatGPT?
4) Is ChatGPT (GPT-4) able to interpret the custom YANG model as expected?
5) Does the user interface (as shown in Ⓖ-Ⓙ of Fig. 7) function correctly?

Figs. 8 and 9 illustrate the basic network topologies used in the exercise. The verifier collects the necessary data for validation by executing the following steps for each network. During this process, the Node.js displays ①-⑧, the custom YANG model, and responses from ChatGPT in the standard output.

1) Construct a network using the functions Ⓐ to Ⓒ in Fig. 2 to enable ICMP echo communication between the devices cli0 and svr0.

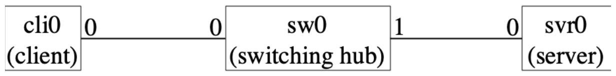


Fig. 8. Basic network topology (three devices).

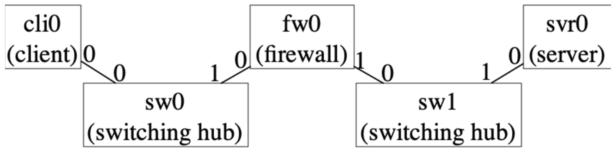


Fig. 9. Basic network topology (five devices).

2) Press the button Ⓖ in Fig. 7 to send the network to ChatGPT and receive a response.
3) Interact with ChatGPT using the functions Ⓗ-Ⓙ.

Based on the Node.js's standard output, the verifier verified the items 1 to 3 and found that all items were satisfied. The verifier then asked ChatGPT questions in Japanese about the items ① to ⑧ of the custom YANG model. For example, a question was, "What is the IP address of the host cli0?" Since the responses from ChatGPT were correct, it was confirmed that the item 4 was satisfied. Finally, the fact that the verifier could conduct the verification smoothly up to this point indicated that the item 5 was satisfied.

VI. Conclusion

This paper proposed a method for sharing network configuration information using a custom YANG model, a technique for collecting internal data from UML without affecting the participant's work, and the structure of prompts to ChatGPT, all to facilitate dialogue between participants and ChatGPT. Additionally, a prototype system was implemented, and it was confirmed that the participant's network could be transmitted to ChatGPT and that ChatGPT could provide answers to questions about that network.

As a future challenge, the performance evaluation of the dialogue function is identified. The response speed and answer accuracy of ChatGPT vary inversely with the model version; for example, GPT-3.5 has faster response times but lower accuracy compared to GPT-4, which tends to have the opposite characteristics. First, the time taken to collect network configuration information and the time taken for ChatGPT to respond will be measured. Then, based on actual exercise manuals, various network situations and troubles will be simulated in dialogues to evaluate how accurately ChatGPT understands the questions and generates responses.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 20K12108.

References

[1] Introduction - OpenAI API, https://platform.openai.com/docs/introduction, accessed Jan. 23, 2024.

[2] Yuichiro Tateiwa, “LiNeS Cloud: A Web-Based Hands-On System for Network Security Classes with Intuitive and Seamless Operability and Light-Weight Responsiveness,” IEICE Transactions on Information and Systems, Vol. E105.D, No. 9, pp.1557–1567, 2022.

[3] Yuichiro Tateiwa, “Proposal of Custom YANG Model for Sharing Network Configuration Information between Application and ChatGPT,” The 2023 IEICE Society Conference, p.256, 2023. (in Japanese)

[4] The User-mode Linux Kernel Home Page, http://user-mode-linux.sourceforge.net/, accessed Jan. 23, 2024.

[5] M. Bjorklund, Ed., “The YANG 1.1 Data Modeling Language,” RFC 7950, 2016.

[6] RFC INDEX, https://www.rfc-editor.org/rfc-index.html, accessed Jan. 23, 2024.

[7] A. Clemm et al., “A YANG Data Model for Network Topologies,” RFC 8345, 2018.

[8] M. Bjorklund et al, “YANG Tree Diagrams,” RFC 8340, 2018.

[9] A. Bierman, M. Bjorklund, “A YANG Data Model for System Management,” RFC 7317, 2014.

[10] M. Bjorklund, “A YANG Data Model for Interface Management,” RFC 8343, 2018.

[11] L. Lhotka et al., “A YANG Data Model for Routing Management (NMDA Version),” RFC 8349, 2018.

[12] M. Bjorklund, “A YANG Data Model for IP Management,” RFC 8344, 2018.

[13] Screen - GNU Project - Free Software Foundation, https://www.gnu.org/software/screen/, accessed Jan. 23, 2024.

[14] Bash - GNU Project - Free Software Foundation, https://www.gnu.org/software/bash/, accessed Jan. 23, 2024.

[15] Node.js, https://nodejs.org/en/, accessed Jan. 23, 2024.