Journal of Systemics, Cybernetics and Informatics (2022) 20(3), 55-63
https://doi.org/10.54808/JSCI.20.03.55

Design and Basic Evaluation of Virtual IPv4-based CYPHONIC adapter

Ren Goto
Faculty of Information Science, Aichi Institute of Technology,
1247 Yachigusa, Yakusa, Toyota, Aichi 470-0392, Japan

Taiki Yoshikawa
Graduate School of Business Administration and Computer Science, Aichi Institute of Technology,
1247 Yachigusa, Yakusa, Toyota, Aichi 470-0392, Japan

Hijiri Komura
Graduate School of Business Administration and Computer Science, Aichi Institute of Technology,
1247 Yachigusa, Yakusa, Toyota, Aichi 470-0392, Japan

Kazushige Matama
Faculty of Information Science, Aichi Institute of Technology,
1247 Yachigusa, Yakusa, Toyota, Aichi 470-0392, Japan

Chihiro Nishiwaki
Graduate School of Business Administration and Computer Science, Aichi Institute of Technology,
1247 Yachigusa, Yakusa, Toyota, Aichi 470-0392, Japan

Katsuhiro Naito
Faculty of Information Science, Aichi Institute of Technology,
1247 Yachigusa, Yakusa, Toyota, Aichi 470-0392, Japan

ABSTRACT [1]

The rapid spread of cloud services and the Internet of Things (IoT) leads to a request for secure communication between devices, called zero-trust security. However, security tends to be a low priority compared to the designated service because zero-trust security requires security knowledge. Therefore, a secure communication framework for developers' service development process is essential as the security measures. The authors have developed CYber PHysical Overlay Network over Internet Communication (CYPHONIC) for secure end-to-end communication between devices. Since the CYPHONIC provides secure communication, it also performs as the secure communication framework. The current implementation requires installing the device program into the end devices to join our overlay network. However, it should support general devices such as embedded devices or dedicated service servers even if they refuse to install the additional program. This paper proposes a new technology to support these general devices without installing the device program into the devices. We developed a CYPHONIC adapter to provide secure communication to general devices. It shows that general devices can communicate over the overlay network through the proposed CYPHONIC adapter.

Keywords: Internet of Things, Zero-trust security, End-to-End communications, Overlay network protocol, CYPHONIC

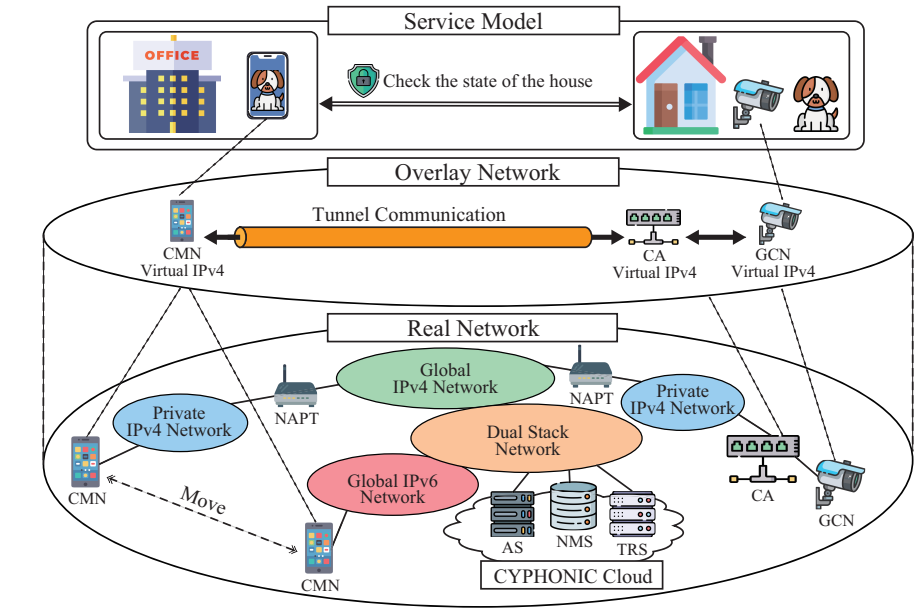
[1] In preparing this paper, the final version of this paper was peer-reviewed by Hidekazu Suzuki, Ph.D. He is an associate professor at Meijo University in Japan. The author wishes to acknowledge and thank Hidekazu Suzuki, Ph.D. for his invaluable assistance in peer reviewing and proofreading this manuscript.

1. INTRODUCTION

The zero-trust security model is the latest secure architecture to realize a more secure system by authorizing each endpoint. It requires all endpoints to be authenticated, authorized, and validated for security configuration, whether in or outside the organization's network [1], [2]. Since many organizations use cloud systems to process data, a typical zero-trust security product considers that cloud systems manage each endpoint. As a result, almost all products rely on cloud systems to realize the security architecture [3], [4].

The number of IoT devices is increasing rapidly. The resource of the IoT devices is extensive computational resources. Therefore, cooperated IoT devices can realize a single service by performing distributed computation [5], [6]. The zero-trust security model is also essential to realize the service because each IoT device typically exists in different areas or networks [7], [8]. It requires IoT devices to realize secure end-to-end communication even if they connect to any network systems. On the contrary, typical developers of IoT devices concentrate on their service, not the security model. As a result, they may often handle security as a low-priority function compared to the designated service functions [9]. Therefore, a secure communication framework is essential for developers to help their service development process without any knowledge of the zero-trust security model.

Since each IoT device exists in distributed networks, the framework should support several network types such as IPv4 global addresses, IPv4 private addresses, and IPv6 addresses. Typically, these techniques are supposed to be network acces-



AS: Authentication Service NMS: Node Management Service TRS: Tunnel Relay Service
CA: CYPHONIC Adapter CMN: CYPHONIC Mobile Node GCN: General Correspondent Node

Fig. 1. Overview of CYPHONIC

sibility issues [10], [11]. As a solution, several technologies have been proposed [12]–[15]. However, conventional methods require developers to understand the details of the technologies.

Cellular systems are also spreading globally and are the main access network for IoT devices. As a result, IoT devices may change access networks according to the service condition of the cellular or Wi-Fi service [16]. Therefore, the framework should handle the change of IP address due to the change in access networks. IP mobility technologies are well-known solutions for this situation [17], [18]. However, conventional technologies focus on IPv6 networks because IPv6 is designed to support mobility. Finally, developers require a more packaged solution to realize their service on the zero-trust security model.

The authors have proposed and developed CYber PHysical Overlay Network over Internet Communication (CYPHONIC) as a technology that can comprehensively solve these issues [19]–[21]. CYPHONIC provides secure end-to-end communication for any application by inserting the new overlay network layer between the network and transport layers. As a result, developers can use the zero-trust security model by focusing on their services.

The current CYPHONIC system requires a device to install the program for the CYPHONIC communication function. On the contrary, some conventional devices (General nodes), such as IoT devices, embedded devices, and dedicated servers, tend to avoid the additional installation of the device program due to the limitation of system and hardware resources or the effect on the system reliability [22].

This paper proposes an adapter device called CYPHONIC adapter to provide a function to join the CYPHONIC network for general nodes. The CYPHONIC adapter is a new technology to support these general nodes without installing the device program into the devices. We developed a CYPHONIC adapter to provide secure communication over our overlay network to the general nodes. It shows that general devices can communicate over the overlay network through the proposed CYPHONIC adapter. Finally, as an essential evaluation, we measured the processing time, communication throughput, and network performance of the CYPHONIC adapter.

2. CYPHONIC

2.1. Overview of CYPHONIC

The communication framework needs to provide secure communication between end devices without being affected by Network Address Port Translation (NAPT) or differences in IP versions. Additionally, it is necessary to achieve continuous communication regardless of switching the network to which the device is connected. CYPHONIC is a new Internet protocol that simultaneously supports network connectivity and seamless mobility and enables secure end-to-end communication. CYPHONIC uses virtual IP addresses to enable secure end-to-end communication over an overlay network unaffected by the physical environment. Therefore, CYPHONIC can establish communication available even when there is a NAPT mechanism on the communication path, or both nodes use different protocol versions such as IPv4 and IPv6. Additionally, CYPHONIC can provide continuous connections even if the terminal changes access networks during communication. All devices communicating over CYPHONIC authenticate at the cloud service and exchange encrypted data based on encryption keys between devices. As a result, since the encryption key is not known to other devices, secure end-to-end communication can be achieved.

2.2. Components of CYPHONIC

Fig. 1 shows the overview of CYPHONIC. CYPHONIC comprises a cloud service and CYPHONIC nodes, end devices equipped with CYPHONIC. CYPHONIC nodes enable secure end-to-end communication between devices in cooperation with cloud services. The cloud service consists of three types of services: Authentication Service (AS), Node Management Service (NMS), and Tunnel Relay Service (TRS). AS performs the authentication process to verify that the CYPHONIC node is a valid user. This service assigns a unique virtual IP address to the CYPHONIC node, which remains even when the node changes an access network. Additionally, it assigns a Fully Qualified Domain Name (FQDN) as the identifier of the CYPHONIC node. It distributes an encryption key to encrypt the NMS and the CYPHONIC node communication. NMS manages network information such as the real IP addresses of CYPHONIC nodes and the existence of NAPT. It also selects an optimal route according to the network environment of both CYPHONIC nodes and instructs the construction procedure to establish secure communication. TRS relays the data between devices when direct communication is difficult due to the different protocol versions between IPv4/IPv6 networks or the rejection of incoming packets due to NAPT mechanisms.

The CYPHONIC node authenticates to AS at the service startup and registers its network location information to NMS. AS assigns a virtual IP address to the CYPHONIC node as the response message. NMS instructs the appropriate communication path for both CYPHONIC nodes so that bidirectional communication is always available even when there is a NAPT mechanism on the communication path or both nodes use different protocol versions such as IPv4 and IPv6. CYPHONIC node identifies a peer node by its FQDN and communicates with the peer node using a virtual IP address. At this time, the CYPHONIC node encapsulates virtual IP packets into real IP addresses and User Datagram Protocol (UDP) datagrams to enable secure communication. Additionally, CYPHONIC nodes exchange an encryption key to secure communication by encrypting all packets in UDP datagrams. Finally, the real IP packets contained in the virtual IP packets are sent to the peer node through the constructed UDP tunnel. Therefore, CYPHONIC can provide a secure communication mechanism for developers without special security knowledge.

2.3. Implementation of CYPHONIC node

Any device can communicate using CYPHONIC by installing the CYPHONIC daemon, the special end-node program. Fig. 2 shows the overview of the CYPHONIC daemon. The CYPHONIC daemon prepares the virtual interfaces for general applications. The applications can communicate through the virtual interface with virtual IP addresses. The CYPHONIC daemon handles the packets through the virtual interface to access the overlay network.

CYPHONIC daemon works in the user space of the CYPHONIC node as a background process and provides the functions necessary to communicate on CYPHONIC. The signaling module performs signaling with each cloud service. The CYPHONIC resolver module is a DNS resolver for resolving virtual

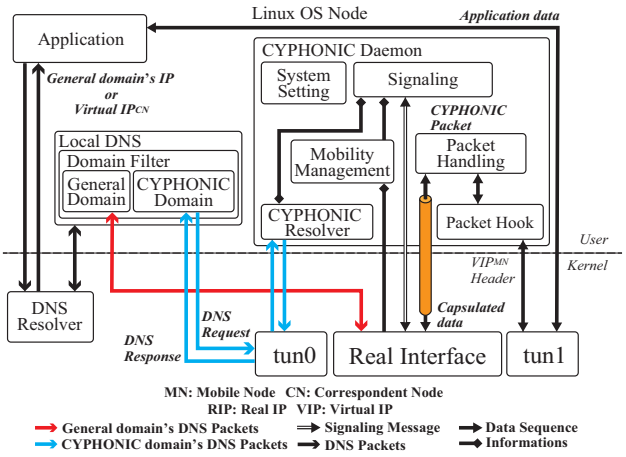


Fig. 2. System model of CYPHONIC daemon

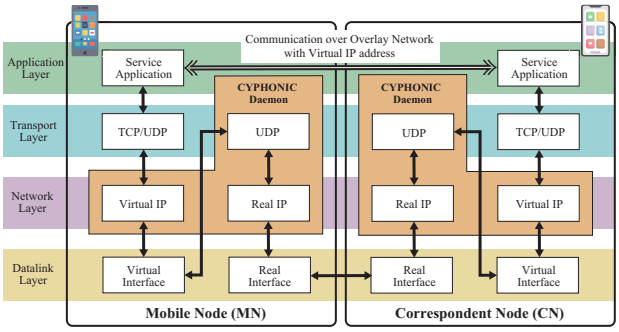


Fig. 3. Overview of Packet capsulation flow

IP addresses from the peer node's FQDN. The packet hook module hooks application data from the virtual interface. The packet handling module performs encapsulation/decapsulation and encryption/decryption processing on packets with virtual IP addresses. The CYPHONIC daemon implements these modules to send and receive capsuled messages to and from the peer node via the real network.

Also, the virtual interface is implemented as a TUNnel (TUN) / Terminal Access Point (TAP) device that allows applications to exchange virtual IP packets with the CYPHONIC daemon and assign virtual IP addresses. At this time, the default route of the application is set to the virtual interface. Therefore, an application can communicate using virtual IP addresses. The virtual interface encapsulates the data received from the application with a virtual IP address. Finally, the CYPHONIC daemon generates CYPHONIC's signaling messages according to the communication process and communicates with each CYPHONIC cloud and peer node.

Fig. 3 shows the encapsulation flow of CYPHONIC. The application sends application data to the virtual interface. First, the CYPHONIC daemon encapsulates the received data into virtual IP packets and encrypts it. Then, it adds a UDP header to virtual IP packets and encapsulates the packet with the real IP address of the CYPHONIC node. Finally, it sends the encapsulated packets to the peer node via the real interface.

When the peer node receives the packet, it decapsulates

it by removing the real IP and UDP headers and obtaining virtual IP packets. Then, the peer node takes data from the virtual IP packet and forwards it to the application. As a result, CYPHONIC can establish communication through an overlay network.

3. CYPHONIC ADAPTER

3.1. Concept design

Since an end node must install the device program to support secure communication, it also requires flexible specifications to accept the additional installation into the end node. On the contrary, some end nodes for IoT or embedded devices limit device resources. Therefore, it may be difficult for these devices to install the CYPHONIC daemon. Additionally, some particular server tends to refuse the additional installation because they are concerned about the stability and reliability of their service. As a result, CYPHONIC should support these general nodes without installing the CYPHONIC daemon.

This paper proposes an adapter device called CYPHONIC adapter to provide a function to join the CYPHONIC network for general nodes. General nodes can connect to adjacent CYPHONIC adapters to communicate over our overlay network without installing the CYPHONIC daemon. Since the CYPHONIC adapter performs processing for CYPHONIC communication instead of the general nodes, it interconnects the general nodes and CYPHONIC end nodes. We also develop a prototype implementation of the CYPHONIC adapter to evaluate the fundamental evaluation for secure communication.

3.2. System model

Fig. 4 shows the system model of the proposed CYPHONIC adapter. The CYPHONIC adapter needs to have the ability to manage general nodes and the CYPHONIC communication function. The CYPHONIC daemon provides the necessary functionality to communicate over our overlay network. Therefore, the CYPHONIC adapter communicates using the functions of the conventional CYPHONIC daemon. Then, it is implemented as the adapter daemon to manage the information for general nodes and handle the communication between general nodes and CYPHONIC nodes. The management function obtains the pair of a virtual IP address and an FQDN for each general node because CYPHONIC identifies an end node based on FQDN. Additionally, it also assigns the virtual IP address to the general nodes. The communication handling function acquires packets from general nodes, performs signaling with cloud services, and encrypts and decrypts messages.

3.3. Component functions

The CYPHONIC adapter extends the implementation of the conventional CYPHONIC daemon to support management and transportation functions for general nodes. It has two physical Network Interface Cards (NICs), one is bridged to the general node, and the other is connected to a network. The CYPHONIC adapter utilizes three modules of the conventional CYPHONIC daemon: signaling module, CYPHONIC resolver module, and packet handling module.

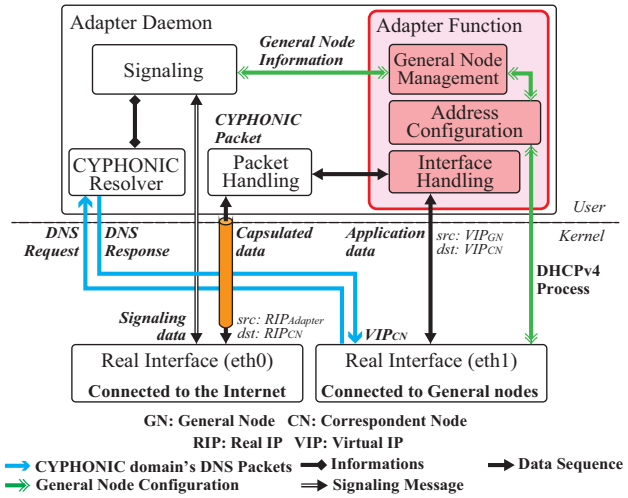


Fig. 4. System model of CYPHONIC adapter

- Signaling Module
  The signaling module performs the communication signaling to and from each cloud service. The CYPHONIC adapter obtains the information of the connecting general nodes from the CYPHONIC cloud through the get general node information process (see below). It also executes a series of signaling processes with each cloud service to perform the authentication and location information registration processes for the CYPHONIC adapter.
- CYPHONIC Resolver Module
  The CYPHONIC resolver module processes DNS packets for FQDN queries because CYPHONIC uses FQDN as the identifier of each node. It generates a DNS request based on the FQDN query obtained from the general node and forwards it to the signaling module. It also generates a DNS response based on the virtual IP address obtained from the signaling module and sends it to the general node.
- Packet Handling Module
  The packet handling module realizes secure communication. It performs encryption, capsulation, decapsulation, and decryption of virtual IP packets. The CYPHONIC adapter performs these processes on packets obtained from the general node. Therefore, the general node does not encrypt the virtual IP packets.

Additionally, the CYPHONIC adapter also should manage the connected general nodes. Therefore, the CYPHONIC adapter implements its additional adapter function to manage the general nodes.

- General Node Management Module
  The general node management module manages the virtual IP addresses, MAC addresses, and FQDNs of the connected general nodes. Additionally, it handles encryption keys generated for each communication process of general nodes to achieve secure communication. The encryption key is transferred to the packet handling module when the general node communicates.
- Address Configuration Module

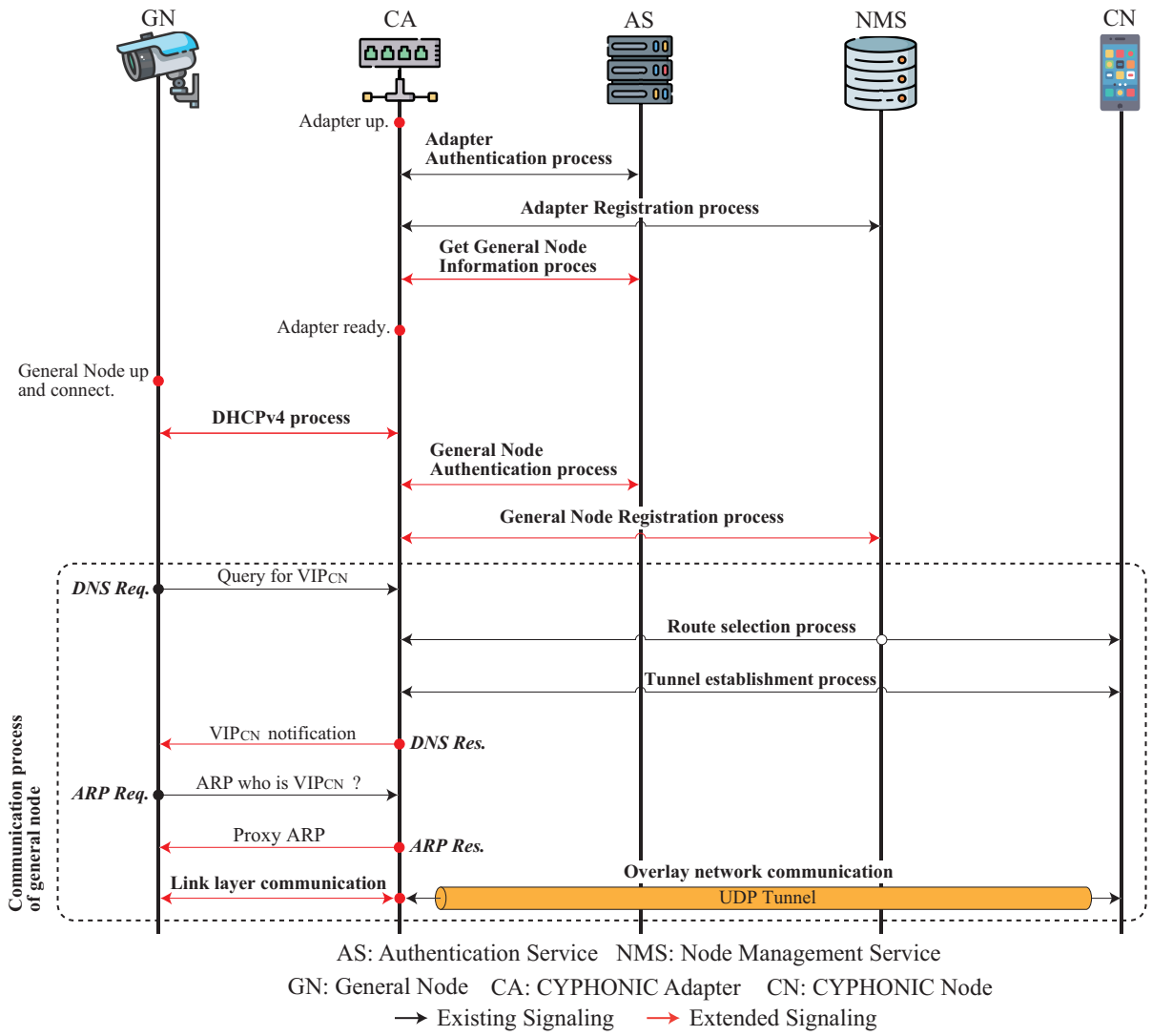


Fig. 5. Communication sequence using CYPHONIC adapter

The address configuration module provides functions to assign virtual IP addresses to general nodes. It assigns virtual IP addresses by the Dynamic Host Configuration Protocol (DHCP) mechanism based on the MAC address of a general node. It also notifies the virtual IP address, the default gateway address, and the IP address of the DNS server simultaneously. This paper assumes that the address configuration module assigns a virtual IPv4 address to a general node because the mainstream IP protocol is still IPv4.

- Interface Handling Module
  The interface handling module performs the same function as the packet hook module in the CYPHONIC daemon. This module hooks packets from general nodes and forwards them to the packet handling module. The CYPHONIC adapter also decapsulates and decrypts packets by the packet handling module in the case of backward communication. Then, it forwards the virtual IP packets to the general node by the interface handling module. As a result, the CYPHONIC adapter can interconnect between the general node and correspondent nodes.

We extend the conventional CYPHONIC daemon module for designing the adapter daemon by linking the adapter function to manage general nodes. The extended module supports secure communication over the overlay network without installing device programs into general nodes.

3.4. Communication sequence

Fig. 5 shows the communication sequence of the CYPHONIC adapter. CYPHONIC executes five processes for end-to-end communication using overlay networks: authentication, registration, route selection, tunnel establishment, and data communication. AS performs the authentication process of end nodes to join the overlay network and assigns a virtual IP address. NMS handles the registration process from end nodes to obtain access network environment. It also manages the route selection process to select an adequate signaling process to

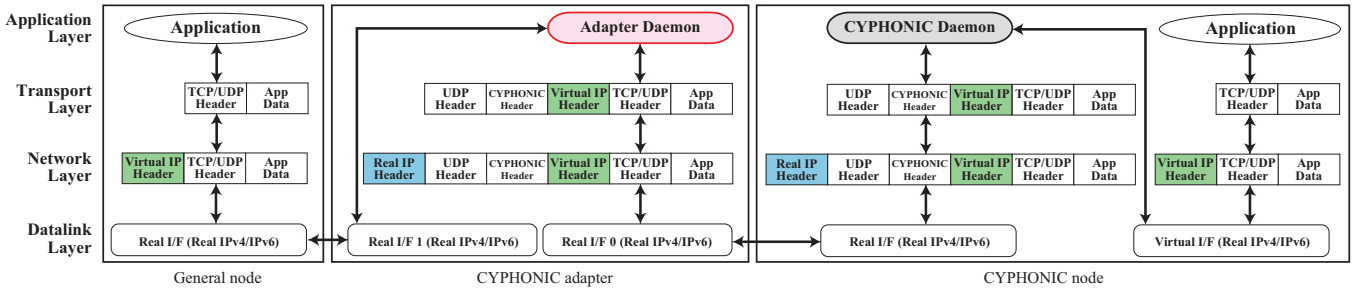


Fig. 6. Packet flow of general node

establish secure communication according to the access network environment for both end nodes. Both end nodes perform the tunnel establishment process to establish secure communication by exchanging an encryption key directly.

Since the CYPHONIC adapter obtains the information of general nodes from the cloud service, it performs the authentication and registration processes instead of the general nodes. Then, it executes the get general node information process as signaling and manages information on connected general nodes. The first process for a general node is to assign the virtual IP address through the DHCP process according to the information of general nodes because general nodes may be offline. After that, the CYPHONIC adapter performs the authentication process and registration process of the general node to AS.

The general node also identifies each end node by FQDN. Therefore, the initial communication trigger is a DNS query message, including FQDN. Since the CYPHONIC adapter provides a DNS function to general nodes, it can receive DNS query messages. As a result, it starts the tunnel establishment process when the DNS query message arrives at the CYPHONIC adapter. NMS selects a communication pattern to establish a tunnel according to the network information of each end node. It also notifies the selected pattern to both end nodes. As CYPHONIC supports secure communication, each end node should exchange an encryption key. Therefore, the CYPHONIC adapter also exchanges an encryption key for each tunnel. Finally, it generates a DNS response message, including the correspondent node's virtual IP address. The general node starts communication to the correspondent node's virtual address as the destination. Therefore, the CYPHONIC adapter should receive packets to the correspondent node's virtual address even if the virtual address is not the same as the IP address for the network interface. We use the Proxy Address Resolution Protocol (ARP) mechanism to transmit an ARP reply message to the ARP request for the virtual address. As a result, the CYPHONIC adapter receives the packets instead of the correspondent node and performs the CYPHONIC communication.

3.5. Packet flow

Fig. 6 shows the packet flow when the general node communicates with a CYPHONIC node. Since the CYPHONIC adapter has already performed Proxy ARP to the general node, it can obtain virtual IP packets to the peer node via link layer communication. First, when the general node receives a DNS response

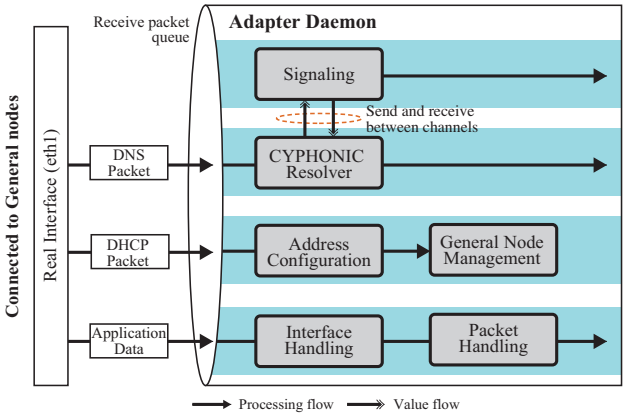


Fig. 7. Overview of the processing flow

from the CYPHONIC adapter, it initiates communication to the virtual IP address of the peer node. At this time, the IP header of packets from general nodes includes a virtual IP address because general nodes use virtual IP addresses for standard IP-based communication. Then, the CYPHONIC adapter encrypts and encapsulates the incoming packets, including the virtual IP addresses, for secure communication. Finally, the CYPHONIC adapter encapsulates the virtual IP packets received from the general node into real IP packets with the real IP address for the network interface. It sends encapsulated packets to the peer node in a constructed UDP tunnel.

The peer CYPHONIC node receives the incoming capsulated packets from the CYPHONIC adapter. Then, it decrypts and decapsulates the received packets to retrieve the inner IP packets, including the virtual IP addresses. Since it also has a virtual network interface, the application receives the decapsulated packets through the virtual network interface. As a result, the peer CYPHONIC node can uniquely identify and communicate with the general node behind the CYPHONIC adapter.

4. IMPLEMENTATION

In this section, we explain the implementation of the CYPHONIC adapter supporting general nodes. Currently, CYPHONIC is implemented using the Go language, a compiled language that is expected to be fast and capable of OS-specific processing. Therefore, the CYPHONIC adapter is also implemented in the Go language. In this paper, we have developed a

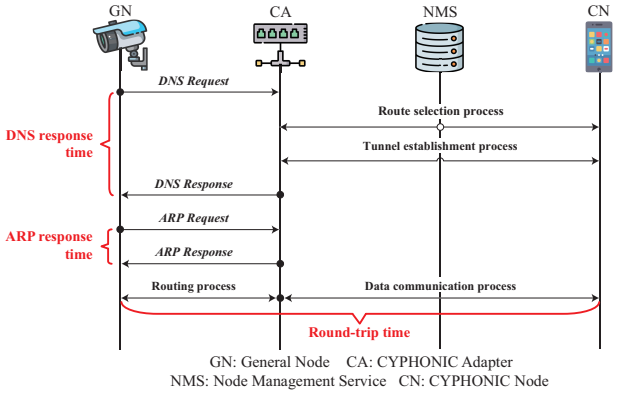


Fig. 8. Evaluation period in the communication process

prototype implementation of the CYPHONIC adapter on Linux OS.

The implementation of each module is described in detail below.

- Signaling Module
  The signaling module implements the OS standard socket Application Programming Interface (API) to manage socket communications. It uses Secure Sockets Layer (SSL) / Transport Layer Security (TLS) communication for the signaling process to AS. It performs its own encrypted UDP communication for the signaling process to NMS and TRS. As the secure communication library, we employ OpenSSL. Since the signaling module identifies the packet types of incoming packets, it manages the communication process according to the message types.
- CYPHONIC Resolver Module
  Since CYPHONIC uses FQDN as the identifier of each node, the CYPHONIC adapter should handle DNS packets from general nodes to identify the correspondent node. The CYPHONIC resolver module receives the DNS query from general nodes and generates a DNS response, including a virtual IP address of the correspondent node.
- Packet Handling Module
  The packet handling module generates a capsule message using the gopacket library according to the defined CYPHONIC packet format [23]. It also uses OpenSSL to encrypt CYPHONIC packets and hash calculation of Hash-based Message Authentication Code (HMAC). This module supports AES-256-CBC and HMAC-MD5 for the common key cryptography and packet authentication algorithms. It also adds HMAC to the end of the packet after the encryption part. Therefore, the peer node can perform tamper detection by performing HMAC calculation when it decapsulates the received CYPHONIC packets.
- General Node Management Module
  The general node management module stores information on general nodes in an in-memory cache. When a general node communicates, the CYPHONIC adapter refers to the cache information and gives necessary information to other modules. Also, when the CYPHONIC adapter receives a request from the peer node to start communication with the general node behind the adapter, it responds with the virtual IP address associated with the FQDN by referring to its cache table. It also clears the cache table entry when it detects no communication for a certain period about the entry. As a result, the CYPHONIC adapter reduces unnecessary memory consumption.
- Address Configuration Module
  The address configuration module implements DHCP functions. The CYPHONIC adapter refers to the cache table when it assigns addresses and obtains the managed virtual IP address from the MAC address of the general node. At this time, it assigns a virtual IP address to a general node by a static address configuration method. Therefore, the same virtual IP address will be assigned again when the lease period expires.
- Interface Handling Module
  The interface handling module implements raw socket and promiscuous mode because it needs to receive and process all packets from general nodes. Then, it performs processing according to packets received from general nodes. Additionally, the CYPHONIC adapter notifies its own MAC address to hook the virtual IP packets sent from the general node to the peer node. Typically, this mechanism is known as Proxy ARP. Through this mechanism, general nodes can map the virtual IP address of the peer node to the MAC address of the CYPHONIC adapter in the ARP table. As a result, the CYPHONIC adapter will be able to receive virtual IP packets from general nodes and process them with the adapter daemon. On the contrary, when it receives a decapsulated packet from the adapter daemon, it generates an ethernet frame and sends it to the general node.

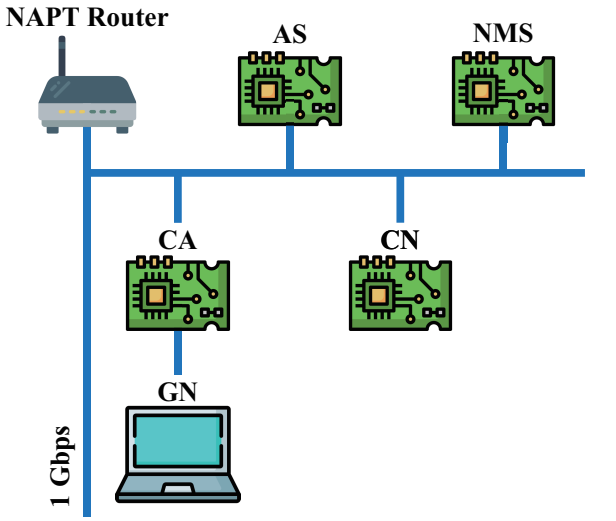


Fig. 9. Evaluation environment

Fig. 7 shows the processing flow of each module of the

TABLE I
SPECIFICATIONS OF THE MEASURING DEVICES

CYPHONIC cloud	
Machine	Raspberry Pi 4 Model B
OS	Raspbian GNU/Linux 10.0 (Buster)
CPU	Quad Core 1.5GHz Broadcom BCM2711 64bit
Memory	4GB RAM
CYPHONIC adapter / CYPHONIC node	
Machine	Raspberry Pi 4 Model B
OS	Raspbian GNU/Linux 10.0 (Buster)
CPU	Quad Core 1.5GHz Broadcom BCM2711 64bit
Memory	8GB RAM
General node	
OS	macOS Monterey Version 12.2
CPU	Dual Core 2.20GHz Intel(R) Core i7-5650U 64bit
Memory	8GB RAM

TABLE II
IMPLEMENTATION ENVIRONMENT

	CYPHONIC adapter	CYPHONIC node
runtime	1.17.2	1.17.2
syscall	1.17.2	-
net	v0.0.0-20220127	v0.0.0-20220127
google/gopacket	v1.1.19	v1.1.19
songgao/water	-	v0.0.0-20200317
crypto	v0.0.0-20220516	v0.0.0-20220516
miekg/dns	1.1.49	1.1.49

adapter daemon. Each module in the adapter daemon performs concurrent processing by using thread mechanisms. Therefore, they can execute a process independently without waiting for the results. The adapter daemon also provides an information-sharing function based on Go language among threads. As a result, each module can execute asynchronously and concurrently to handle multiple processes simultaneously and flexibly.

5. BASIC EVALUATION

In this section, we conduct a basic evaluation of the proposed system model. The first evaluation measures the signaling processing time and packet processing time of the CYPHONIC adapter.

Fig. 8 shows the evaluation period in the communication process. We have evaluated the signaling, including route selection, tunnel establishment, and data communication processes. The route selection process includes the processing time of DNS

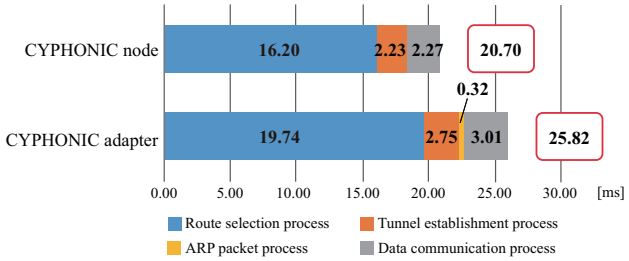


Fig. 10. Results of processing time

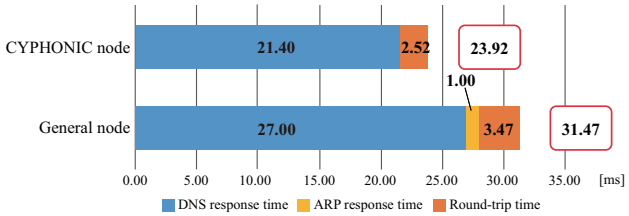


Fig. 11. Results of delay time

packets. The packet processing time of the CYPHONIC adapter also includes the ARP packet processing time.

The second evaluation is measuring the communication delay time of a general node. This evaluation includes the response time of DNS and ARP packets. We also measure the round-trip time to evaluate the communication performance. Additionally, in recent years, many streaming services adopt UDP. Since CYPHONIC is UDP-based, we also measure UDP throughput in this evaluation.

Fig. 9 shows the verification environment in this evaluation. Table I shows the device specification for the verification. Table II shows the specific version of the software to implement the CYPHONIC adapter. The CYPHONIC adapter and the conventional CYPHONIC node communicate over the proposed overlay network in the evaluation environments.

Fig. 10 compares the communication processing time of the CYPHONIC adapter and the CYPHONIC node. The CYPHONIC adapter can process from route selection to data communication in about 25 [ms]. On the contrary, the CYPHONIC node can process it in about 20 [ms]. As a reason, although the CYPHONIC node has two virtual interfaces, the CYPHONIC adapter uses only one real interface to determine packet types. Therefore, comparing the adapter daemon with the conventional system, the time required for the route establishment process is slightly larger. Additionally, the CYPHONIC adapter must also process ARP packets. However, the measurement results show that the processing of ARP packets does not incur significant overhead.

Fig. 11 compares the delay time for initial communication between the General node and the CYPHONIC node. The general node experiences a difference of about 8 [ms] compared to the CYPHONIC node. The response time of the DNS response is due to the processing time of the CYPHONIC adapter. The response time of the ARP response is 1 [ms]. We used the ping tool to check the reachability and measured the round-trip time. The measured RTT results show a difference of about 1 [ms] in

TABLE III
UDP THROUGHPUT PERFORMANCE

	CYPHONIC adapter		CYPHONIC node	
UDP traffic	Throughput	Jitter	Throughput	Jitter
10 Mbps	10.0 Mbps	0.40 ms	10.0 Mbps	0.52 ms
20 Mbps	20.0 Mbps	0.42 ms	20.0 Mbps	0.66 ms
30 Mbps	29.7 Mbps	0.40 ms	30.0 Mbps	0.55 ms

both systems. As a reason, it is assumed to be a communication delay between the general node and the CYPHONIC adapter.

Table III compares the UDP throughput of the CYPHONIC adapter and the CYPHONIC node. We have used the iperf tool to measure the throughput. The measured UDP throughput was 29.7 Mbit/sec, which was confirmed to be comparable to the performance of the conventional CYPHONIC node. Typically, HD quality requires a throughput of 5 Mbit/sec. The measurement results show that the throughput of the CYPHONIC adapter is slightly lower than that of the CYPHONIC node. However, the prototype has enough throughput performance required by high throughput applications such as streaming, etc [24], [25].

6. CONCLUSION

This paper has proposed the adapter device called CYPHONIC adapter to provide a function to join the CYPHONIC network for general nodes such as IoT devices, embedded devices, and dedicated servers. The CYPHONIC adapter is a new technology to support these general nodes without installing the device program into the devices. Therefore, the conventional general nodes can quickly join the secure overlay network. We have developed the prototype of the CYPHONIC adapter to provide secure communication over our overlay network to the general nodes. It shows that general devices can communicate over the overlay network through the proposed CYPHONIC adapter without significant overhead.

ACKNOWLEDGMENT

This work is supported in part by the collaborative research project with Mobiline Co., Ltd., Japan, and Grant-in-Aid for Scientific Research (C)(21K11877), Japan Society for the Promotion of Science (JSPS).

REFERENCES

[1] A. Iftekhar, N. Tahmin, U. Sultana, and T. Kazi, "Protection of Sensitive Data in Zero Trust Model," 10.1145/3377049.3377114, January 2020.

[2] Y. Qigui, W. Qi, Z. Xiaojian, and F. Jiaxuan, "Dynamic Access Control and Authorization System based on Zero-trust architecture," 10.1145/3437802.3437824, October 2020.

[3] N. Lahar, Akriti, P. Jaya, P. Roli, B. Sumitra, and K. Sarvesh, "Security, Privacy Issues and challenges In Cloud Computing: A Survey," 10.1145/2905055.2905253, March 2016.

[4] H. Sufian, K. Faraz, H. Bilal, and S. Djamel, "Understanding Security Requirements and Challenges in Internet of Things (IoT): A Review," 10.1155/2019/9629381, January 2019.

[5] M. Beevi, "A fair survey on Internet of Things (IoT)," 10.1109/ICETETS.2016.7603005, February 2016.

[6] S. Khaitan and J. McCalley, "Design Techniques and Applications of Cyberphysical Systems: A Survey," 10.1109/JSYST.2014.2322503, June 2015.

[7] L. Shancang, "Editorial: Zero Trust based Internet of Things," 10.4108/eai.5-6-2020.165168, June 2020.

[8] A. Mariam and T. Radwan, "Enabling Security as a Service for IoT Emerging Technologies: A Survey," 10.1016/j.comnet.2014.11.0088, November 2021.

[9] S. Sicari, A. Rizzardi, L.A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," 10.1016/j.comnet.2014.11.0088, January 2015.

[10] C. Huang and S. Hwang, "The asymmetric NAT and its traversal method," 10.5555/1689139.1689175, April 2009.

[11] P. Wu, Y. Cui, J. Wu, J. Liu, and C. Metz, "Transition from IPv4 to IPv6: A state-of-the-art survey," 10.1109/SURV.2012.110112.00200, January 2013.

[12] K. Navin, A. Onur, P. Ioannis, and P. George, "Rewarding relays for decentralised NAT traversal using smart contracts," 10.1145/3397166.3412799, October 2020.

[13] K. Hanna, P. Amir, M. Alberto, and D. Jim, "NATCloud: cloud-assisted NAT-traversal service," 10.1145/2851613.2851640, April 2016.

[14] A. Keränen, C. Holmberg, and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal," 10.17487/RFC8445, July 2018.

[15] W. Jianing, Z. Shilong, and J. Lurong, "Research and application of related technologies of new generation network communication protocol IPv6," 10.1109/ICMCCE51767.2020.00479, December 2020.

[16] B. Feng, C. Zhang, J. Liu, and Y. Fang, "D2D Communications-Assisted Traffic Offloading in Integrated Cellular-WiFi Networks," 10.1109/JIOT.2019.2922550, October 2019.

[17] S. Ghaleb, S. Subramaniam, Z. Zukarnain, and A. Muhammed1, "Mobility management for IoT: a survey," 10.1186/s13638-016-0659-4, July 2016.

[18] A. Hussain, S. Nazir, F. Khan, L. Nkenyereye, A. Ullah, S. Khan, S. Verma, and Kavita, "A Resource Efficient hybrid Proxy Mobile IPv6 extension for Next Generation IoT Networks," 10.1109/JIOT.2021.3058982, February 2021.

[19] T. Yoshikawa, S. Isomura, H. Komura, S. Kubota, C. Nishiwaki, and K. Naito, "Performance evaluation of shared library supporting multi-platform for overlay network protocol," 10.1109/GCCE50665.2020.9292015, October 2020.

[20] T. Yoshikawa, H. Komura, C. Nishiwaki, R. Goto, K. Matama, and K. Naito, "Evaluation of new CYPHONIC: Overlay network protocol based on Go language," 10.1109/ICCE53296.2022.9730323, January 2022.

[21] T. Yoshikawa, H. Komura, R. Goto, K. Matama, C. Nishiwaki, and K. Naito, "Demonstration of video conferencing tool with overlay network protocol," 10.1109/CCNC49033.2022.9700703, January 2022.

[22] A. Musaddiq, Y. Zikria, O. Hahm, H. Yu, A. Bashir, and S. Kim, "A Survey on Resource Management in IoT Operating Systems," 10.1109/ACCESS.2018.2808324, February 2018.

[23] google/gopacket, https://github.com/google/gopacket, May 2022.

[24] S. Tavakoli, "Subjective QoE Analysis of HTTP Adaptive Streaming Applications," December 2015.

[25] A. Biernacki and K. Tutschku, "Performance of HTTP video streaming under different network conditions," 10.1007/s11042-013-1424-x, September 2014.