一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
SeMI2023-38(2023-07)

CYPHONICの端末機能をWindows OSで実現する基礎開発

ウジイエ ギレルメ セイジ† 後藤 廉† 眞玉 和茂†
鈴木 誉写† 内藤 克浩†† 鈴木 秀和†††

† 愛知工業大学大学院経営情報科学研究科 〒470–0356 愛知県豊田市八草町八千草 1247
†† 愛知工業大学情報科学部 〒470–0356 愛知県豊田市八草町八千草 1247
††† 名城大学情報工学部 〒468–8502 名古屋市天白区塩釜口一丁目 501 番地
E-mail: †{useidy,r0719en,matama,homare448}@pluslab.org, ††naito@pluslab.org, †††hsuzuki@meijo-u.ac.jp

あらまし 著者らは，通信接続性と移動透過性を実現するセキュアな通信技術として，CYber PHysical Overlay Network over Internet Communication（CYPHONIC）の研究・開発を進めている．既存の CYPHONIC は Linux OS 端末での利用を想定した開発が進んでおり，他 OS 上で動作する端末機能が未実装である．他 OS としては，Windows OS は市場普及率が高いこともあり，端末機能の実装が将来的に必須と考えられる．本稿では，Windows OS を想定した CYPHONIC の端末機能の設計と実装を行う．Linux OS を想定した既存実装は Berkeley System Design （BSD）ソケットを用いた実装になっている上，アプリケーションが利用する仮想インターフェイスも Linux OS 専用の Tunnel / Terminal Access Point（TUN/TAP）を利用したものである．本稿では，Windows OS で利用可能な仮想インターフェイスを Windows OS 専用の TUN/TAP を利用した上で作成し，Windows OS のネットワークソケットを利用する端末機能の設計と実装を行い，CYPHONIC ノードを Windows OS 上で利用可能にする．プロトタイプ実装を用いた通信性能評価の結果，Linux OS の端末機能と同等の通信性能を実現可能であることを確認した．
キーワード 通信接続性，移動透過性，ゼロトラストネットワーク，オーバーレイネットワーク，Windows OS

Initial implementation of CYPHONIC end-device functions on Windows OS

Guilherme SEIDY UJIIE†, Ren GOTO†, Kazushige MATAMA†,
Yoshiya SUZUKI†, Katsuhiro NAITO††, and Hidekazu SUZUKI†††

† Graduate School of Business Administration and Computer Science，Aichi Institute of Technology
1247 Yachigusa，Yakusa Cho，Toyota City，Aichi Prefecture 470–0392 Japan
†† Faculty of Information Science，Aichi Institute of Technology
1247 Yachigusa，Yakusa Cho，Toyota City，Aichi Prefecture 470–0392 Japan
††† Faculty of Information Engineering，Meijo University
1–501 Shiogamaguchi，Tempaku-ku，Nagoya 468–8502，Japan
E-mail: †{useidy,r0719en,matama,homare448}@pluslab.org, ††naito@pluslab.org, †††hsuzuki@meijo-u.ac.jp

Abstract The authors have been researching and developing CYber PHysical Overlay Network over Internet Communication (CYPHONIC)，a secure communication technology that enables communication connectivity and transparent mobility．The current implementation of CYPHONIC is being developed specifically for Linux OS terminals，and the terminal functionality for other operating systems has not yet been implemented．Considering the widespread adoption of Windows OS in the market，implementing terminal functionality for Windows OS will become essential in the future．This paper focuses on designing and implementing CYPHONIC's terminal functionality for Windows OS．While the existing implementation for Linux OS relies on BSD sockets and utilizes the Tunnel / Terminal Access Point（TUN/TAP）functionality available only in the Linux kernel，we propose a design that leverages virtual interfaces compatible with Windows OS and network sockets provided by the Windows OS．The performance evaluation confirms that it is possible to achieve communication performance equivalent to that of the terminal functionality in the Linux OS.
Key words NAPT traversal，Mobility，Zero-trust network，Overlay network，Windows OS

1. はじめに

インターネットの通信は様々な用途で利用されており，セキュリティアプローチも用途に応じて変化している [1]．ネットワークに内外の境界を設けてセキュリティを担保するセキュリティソリューションを一般に境界型セキュリティモデルと呼び，通信が確立された LAN 内を安全な領域と見なし，外部からの通信を遮断することで安全性を担保している．しかしながら境界型セキュリティモデルでは，LAN に侵入した端末からの攻撃に対応することが困難であり，脆弱性が課題となっている [2]．また，ネットワークの境界を設けないセキュリティモデルとしてクラウド中継型のセキュリティモデルが挙げられる．この手法では，ネットワークの境界にセキュリティソリューションを設けずにクラウドを介したセキュアな通信を確立することが可能となる．一方でクラウド中継型セキュリティモデルではクラウドがクライアントの送受信データを直接扱うため，クラウドからの情報漏洩のリスクが考えられる [3]．

一方，リモートワークを始めとする広域ネットワーク網からの接続では Virtual Private Network（VPN）を利用することで Local Area Network（LAN）内の通信を確立する手法が用いられてきた [4]．そのため現代ではネットワークの多種多様な利用形態が存在する中，ネットワークの内外に関わらずあらゆる方向からの通信の正当性を保証することが求められる．これを実現する新たなセキュリティソリューションとして，近年ゼロトラストモデルというセキュリティモデルが提案されている．ゼロトラストモデルとは，ネットワークの内外に関わらず，全ての端末の安全性を検証するセキュリティモデルである [5]．全ての端末の安全性を検証することで，内部ネットワークでの不正な通信においても安全の確保を可能にする．また，先述のリモートワークや Internet of Things（IoT）システムなどの，インターネット上に分散された端末が行う通信の安全性確保も実現可能である [6] [7]．これらは，従来の境界型セキュリティモデル及びクラウド中継型セキュリティモデルには存在しない特徴である [8]．

ゼロトラストセキュリティモデルを採用にするにあたり，インターネット上のいかなる場所においても通信が可能であり，ネットワーク変更時にも端末の同一性を保証する必要があることから，通信接続性と移動透過性の 2 種類の問題を解決することが求められる．通信接続性とは，IP バージョンの差異や NAPT に影響されず通信ができる性質である [9] [10]．NAPT とは，IP アドレスおよびポート番号を相互変換する技術である [11] [12]．NAPT は IP アドレスをグローバル IP アドレスとプライベート IP アドレスへと分類する．しかし，NAPT は，外部に対して内部ネットワークに存在する端末を隠蔽する．結果として，NAPT 配下の端末に対して NAPT 外の端末から通信を開始することが困難となる．また IPv6 は，IPv4 が割り当てられる IP アドレスが，2^{32} 個であった点を解決し，2^{128} 個の割り当てを可能にする．これにより，全ての端末に対して一意なアドレスの割り当てが可能となる [13] [14]．一方，IPv4 と IPv6 はパケット構造が異なるため互換性がなく，相互に通信を行うことが困難である．加えて現代では，IPv4 アドレスを用いる端末も残っており，IPv4 と IPv6 が混在した環境となっている [15]．

移動透過性とは，IP アドレスの変更に対応し継続した通信を提供する性質である．現代のモバイル端末は複数の無線インターフェイスを備えており，アクセスするネットワークに応じて，インターフェイスを切り替えることが可能である [16] [17]．一方で IP は通信時のアドレス変更を考慮していない．そのため，移動に伴いネットワークが切り替わることで通信が切断される問題が存在する．

著者らはゼロトラストに基づくセキュアな通信を実現し，通信接続性と移動透過性を備えた通信技術として CYber PHysical Overlay Network over Internet Communication（CYPHONIC）の研究・開発を行なってきた [18]．CYPHONIC は，仮想 IP アドレスで構築されたオーバーレイネットワーク上で，認証された端末のみによる暗号化された通信を行うことで，実ネットワーク環境の影響を受けないセキュアな通信を実現する [19] [20]．CYPHONIC は，CYPHONIC クラウドと CYPHONIC ノードから構成される．CYPHONIC クラウドは，CYPHONIC ノードの認証や仮想 IP アドレスの割り当て，必要に応じて通信の中継等，端末間の通信をサポートする．CYPHONIC ノードは，専用常駐型プログラム（以下，CYPHONIC Daemon）を搭載した端末である．

現在の CYPHONIC Daemon は，Linux OS を搭載している端末のみに導入可能である．しかしながら，CYPHONIC は汎用的な通信技術として活用されることが望ましい．したがって，CYPHONIC は様々なプラットフォームへの対応が必要である．

そこで本研究では，CYPHONIC のより汎用的な利用を実現するために新たなプラットフォームで動作する CYPHONIC Daemon を実現する．本研究では特に普及率の高い Windows OS に向けた CYPHONIC Daemon の拡張手法を提案する [21]．また Windows OS 用の CYPHONIC Daemon の提供に加え，非情報従事者が容易に CYPHONIC を利用可能となるためのインストーラーを導入する．本論文では，Windwos OS 上で CYPHONIC Daemon を実装するための設計，実装及びインストーラーの導入について詳述する．また，既存の CYPHONIC ノードとの性能比較を行い提案システムの実用性を検証する．

2. CYPHONIC

図 1 に CYPHONIC の概要図を示す．CYPHONIC は，CYPHONIC クラウドと CYPHONIC ノードから構成される．CYPHONIC クラウドは，認証を行う Authentication Service（AS），経路情報の管理及び CYPHONIC ノードへの通信経路指示を行う Node Management System（NMS），直接通信が困難であると判断された場合に中継処理を行う Tunnel Relay Service（TRS）から構成される．以下に CYPHONIC クラウドの各構成要素について詳述する．

- Authentication Service（AS）

AS は，CYPHONIC ノードの認証を行うサービスである．AS は，認証を経て正規の端末であると判断した CYPHONIC ノードに対して，仮想 IP アドレスと FQDN を割り当てる．また，

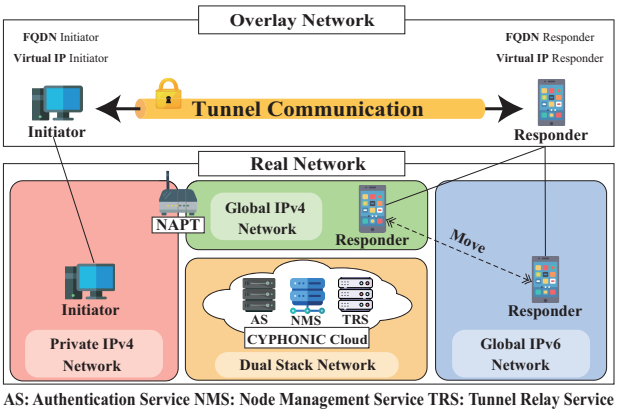

図 1　CYPHONIC の概要

認証処理と同時に，CYPHONIC ノードに対して，NMS との通信で使用する暗号鍵を配布する

- Node Management Service（NMS）

NMS は，認証を経た CYOHONIC ノードのネットワーク情報を管理する．ネットワーク情報は，CYPHONIC ノードの実 IP アドレスや NAPT の有無を指す．これらは，CYPHONIC ノードから送信されるパケットに含まれる情報から取得する．

- Tunnel Relay Service（TRS）

TRS は，CYPHONIC ノードが異なる NAPT 配下に存在する時や通信相手との実 IP アドレスのバージョンが異なるといった理由で，直接通信を行うことが困難であると判断された場合に，中継処理を行うサービスである．

CYPHONIC ノードは，CYPHONIC で利用する独自のドメイン（以下，CYPHONIC ドメイン）を用いた FQDN によって通信相手を指定し，AS から割り当てられた仮想 IP アドレスを用いることで，実ネットワークの影響を受けないオーバーレイネットワーク上の通信を実現する．CYPHONIC ノードが通信を行う際には，CYPHONIC ノード間で暗号鍵を直接交換することによりセキュアな End-to-End（E2E）通信を実現する．また，CYPHONIC ノードは移動に伴い実 IP アドレスが変更された場合にも，NMS に対して再度ネットワーク情報を提供することで，NMS が管理しているネットワーク情報を更新する．この時，AS から割り当てられた仮想 IP アドレスは影響されず，一意なままとなるため，アプリケーションは仮想 IP アドレスを指定したシームレスな通信が可能となる．

上記の端末機能は，CYPHONIC Daemon と呼ばれる専用の端末プログラムにより実現される．CYPHONIC Daemon は CYPHONIC ノード上のアプリケーションが送受信する仮想 IP を用いた IP データグラムを仮想インターフェイス経由で取り扱い，暗号化/復号，カプセル化/デカプセル化することにより，セキュアな E2E を実現する．

3.　提案する WindowsOS 用の端末機能について

3.1　Windows OS 版の端末機能の概要

CYPHONIC ノードには，CYPHONIC Daemon と呼ばれる端末機能を導入することにより，オーバーレイネットワーク上での通信を実現する．CYPHONIC Daemon は，インターフェイスの操作やデータの暗号化，通信シグナリングなど，役割を細分化した複数のモジュールから構成されている．これにより，個々のモジュールの影響範囲を最小化し，カーネル空間を操作する必要がある動作と，ユーザ空間に対する操作のみが必要な動作を分離している．そのため，他の OS に向けた CYPHONIC Daemon の開発に際して変更の必要があるモジュールは，仮想インターフェイスの操作等，カーネル空間への処理を必要とするもののみである．本章では，Windows OS 用にカーネル空間に影響を与えるモジュールを対応させる，Windows OS 用の CYPHONIC Daemon の提案及び実装を行う．加えて，非情報従事者による CYPHONIC の諸機能の導入を簡易化するインストーラーの提案および実装を行う．

3.2　導入した新機能

既存の CYPHONIC ノードは，Linux OS を想定した実装となっている．また，Linux OS と Windows OS では OS の設計が異なるため，別の手法でカーネルの操作を行うことが求められる．以下に，新たに導入した要素について詳述する．

- Dynamic Link Library(DLL）

Windows OS では，OS の機能の多くが DLL によって提供される．DLL を利用することで，コードのモジュール化やコードの再利用，メモリ使用率の向上及びディスク領域の削減が可能となる．また，Windows OS アーキテクチャにおいて，カーネル空間への操作は，DLL を介して行われる．仮想インターフェイスを作成可能な DLL は初期段階で存在しないため，新たに DLL を追加し作成する．

- ネットワークソケット

既存の CYPHONIC の端末機能は，Linux OS 上の動作を想定しているため BSD ソケットを用いた実装が施されている．本実装では，Windows OS 上での通信を実現するネットワークソケットを用いて，実装を行う．Linux OS を想定した既存実装は BSD ソケットの利用を前提とした実装になっている上，アプリケーションが利用する仮想インターフェイスも Linux OS を想定したものである．本稿では，Windows OS で利用可能な仮想インターフェイスを想定した上で，Windows OS のネットワークソケットを利用する端末機能の設計と実装を行う．

- Domain Name System（DNS）Message Handler

CYPHONIC を用いた通信に利用する FQDN には CYPHONIC ドメインが含まれている．CYPHONIC ドメインが含まれた FQDN の名前解決は，CYPHONIC クラウドとの連携によって行われる．そのため，DNS クエリを CYPHONIC クラウドへの問い合わせにするために，DNS パケットを CYPHONIC Daemon に転送する機能が必要である．

3.3　Windows OS を想定した CYPHONIC Daemon の概要

図 2 に提案する Windows OS 用の CYPHONIC ノードのシステムモデル図を示す．CYPHONIC ノードは，CYPHONIC Daemon の導入と仮想インターフェイスの作成によってオーバーレイネットワーク上での通信を実現することが可能となる．CYPHONIC Daemon は以下のモジュールから構成されている．

- System Setting

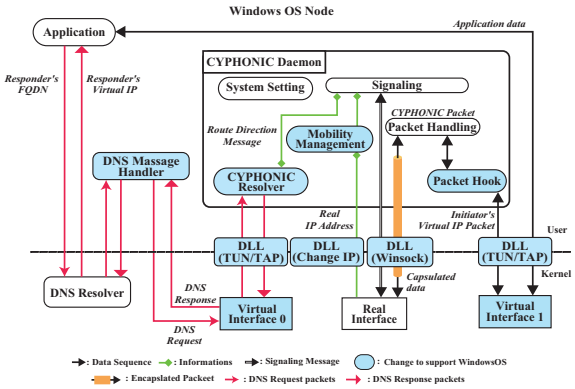

図 2　Windows OS 用の CYPHONIC Node

設定ファイルを読み込み CYPHONIC の通信に必要な諸情報の取得処理を行う． 設定ファイルには，認証方法や，仮想インターフェイスの設定，クラウドサービスとの通信に用いるポート番号の情報が記述されている．

- Signaling

上述の CYPHONIC クラウドである，AS との認証処理や，NMS との経路構築処理，TRS との中継処理など一連のシグナリングを行う．また，AS から通知された仮想 IP アドレスを割り当てた仮想インターフェイスを作成する．

- CYPHONIC Resolver

CYPHONIC ドメインの FQDN は，通常の DNS サーバによる解決が困難である．故に，FQDN から仮想 IP アドレスへの解決は，NMS とのシグナリングで実行される．そのため，CYPHONIC Resolver は，DNS パケットを検知し，Signaling モジュールへと受け渡す．その後，シグナリングを経て相手 CYPHONIC ノードの仮想 IP アドレスを取得した CYPHONIC Resolver は，DNS Response を生成し，アプリケーションに対して仮想 IP アドレスを通知する．

- Packet Hook

CYPHONIC クラウドや CYPHONIC ノードと通信を行う際に，仮想インターフェイスを介して仮想 IP アドレス宛てのパケットを取得する． また，Packet Handling モジュールからデカプセル化されたパケットを受信した場合には，仮想インターフェイスを介してアプリケーションへ転送する．

- Packet Handling

CYPHONIC は定義されたパケットに従い，通信を行う．Packet Handling モジュールでは，パケットの生成，暗号化及び復号化，カプセル化及びデカプセル化の処理を行う．CYPHONIC パケット生成後，末端に改ざん検出を行うための Hash-based Message Authentication Code（HMAC）を付与する．これにより，受信側は受信したパケットの HMAC を計算することで，改ざん検知が可能となりデータの完全性の保証が可能となる．

- Mobility Management

ネットワークを利用した通信を行う端末は，移動に伴い別のネットワーク領域に変化することで実 IP アドレスが変更される．Mobility Management モジュールでは，実 IP アドレスの変更を検知し，NMS に対して再度ネットワーク情報を登録する．これにより，実 IP アドレスが変更された後でも再び通信相手とトンネルを確立することが可能となり，継続的な通信が実現される．

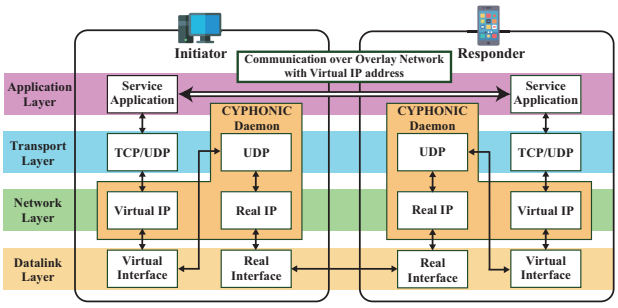

図 3　WindowsOS 用の CYPHONIC Node におけるデータパケットフロー

3.4　想定システムモデルを用いた処理概要

3.4.1　DNS パケットの流れ

オーバーレイネットワークでの通信を実現するためには，アプリケーションが発出する DNS の問い合わせに対して，仮想 IP アドレスを含む応答を返す必要がある． CYPHONIC 独自の FQDN と仮想 IP アドレスの対応付けを把握しているサービスは NMS であるため，CYPHONIC の通信における FQDN の名前解決は，NMS が担う．DNS Message Handler は，CYPHONIC で用いる FQDN の DNS クエリのみをフィルタリングし，CYPHONIC Resolver モジュールへ受け渡す．CYPHONIC Resolver モジュールは，DNS クエリを受信すると経路選択処理を開始するため，Signaling モジュールへ DNS Request を生成して転送する．Signaling モジュールは，NMS に対して相手ノードのネットワーク情報を取得するため，Direction Request メッセージを生成して送信する．その後，Signaling モジュールを通じて，NMS から Route Direction to Initiator メッセージによって相手ノードの仮想 IP アドレスを取得すると，CYPHONIC Resolver モジュールへ受け渡す． 最後に，CYPHONIC Resolver モジュールは，相手ノードの仮想 IP アドレスを含む DNS Response を生成してアプリケーションへ返答する．

3.4.2　データパケットの流れ

図 3 に，オーバーレイネットワークを介した通信のパケットフローを示す． 仮想 IP アドレス宛のパケットは仮想インターフェイスを介すようルーティング設定を行う． アプリケーションが，仮想インターフェイスへアプリケーションデータを送信すると，Packet Hook モジュールによって CYPHONIC Daemon が取得し，Packet Handling モジュールに受け渡し，暗号化等の処理を行う．その際，取得した仮想 IP パケットに対して CYPHONIC の通信を実現するための CYPHONIC ヘッダを付与し，CYPHONIC パケットを生成する． その後，UDP ヘッダを付与し，実 IP アドレスでカプセル化を行う．これにより，実 IP パケットが生成され，実インターフェイスを介した通信が可能となり，構築した UDP トンネルにしたがって相

手ノードへ送信される．宛先ノードがパケットを受信すると，デカプセル化によって実 IP ヘッダと UDP ヘッダが取り除かれ，CYPHONIC パケットを取得し，CYPHONIC Daemon へ受け渡す．CYPHONIC パケットは，Packet Handling モジュールによって復号及びデカプセル化され仮想 IP パケットとなる．そして，仮想 IP パケットを仮想インターフェイスへルーティングする．最後に，仮想 IP パケットからアプリケーションデータを取り出し，アプリケーションへ転送する．

3.5 インストーラー

CYPHONIC を用いた通信を行うためには，上述したシステムの設定や CYPHONIC Daemon の導入が必須となる．そこで，非情報従事者が容易に CYPHONIC を導入することが可能となるインストーラーを作成する．インストーラーを用いてインストールされるモジュールは，CYPHONIC EXE，Setup Package，CYPHNIC Starter Application の 3 つで構成される．CYPHONIC EXE は上述した CYPHONIC Daemon がバイナリ形式でインストールされる．Setup Packege は，CYPHONIC EXE の起動に際して TUN/TAP Device に用いる DLL や，ドメイン名を識別するフィルター機能の設定などが事前に行うモジュールが組み込まれている．CYPHONIC Starter Application は，ユーザによるデバイス ID およびパスワードの入力を受け付けるソフトウェアである．ここで入力された情報を元に，CYPHONIC EXE が実行される．なお, インストーラーを用いた CYPHONIC の導入は, 仮想インターフェイスの作成等のカーネル空間の操作が必須であるため, 管理者権限でのみ実行可能となる．

4. 検 証 評 価

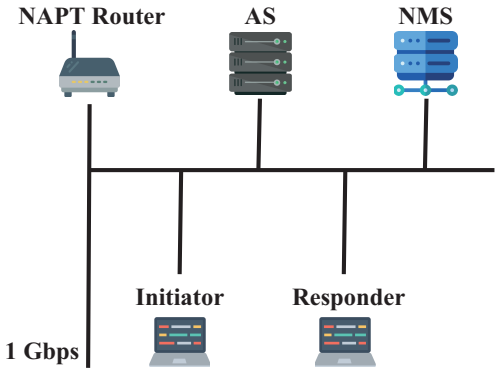

図 4　検 証 環 境

動作検証および性能評価を行うにあたり，図 4 に示すネットワーク構成を準備した．また，性能評価に利用した端末の情報を表 1 に示す．動作検証と性能評価において，既存の Linux OS 用の端末機能を用いて CYPHONIC を利用した時の通信性能を評価基準とする．

4.1 動 作 検 証

動作検証は，Windows OS 端末 2 台による CYPHONIC を用いた通信を行う場合と，Linux OS 端末 2 台による CYPHONIC を用いた通信を行う場合の 2 つのケースで実施した．表 2 に，2 つのケースにおいて ping を用いて通信を行なった際の Round Trip Time（RTT）を示す．検証結果より，Windows OS 端末を用いて Responder の FQDN を元に仮想 IP アドレスを取得し，仮想 IP アドレスを用いた疎通が観測できた点からオーバーレイネットワーク上の通信が可能であることを確認した．また，RTT の結果から Windows OS において通信に大きなオーバーヘッドが生じていないことを確認した．

表 1　検証端末の詳細

CYPHONIC Cloud	
OS	Ubuntu 21.10
CPU	11th Gen Intel Core i9-11900K @ 3.50GHz 16cores
Memory	128GB
CYPHONIC Node（WindowsOS）	
OS	Windows10
CPU	11th Gen Initel Core i5-1135G7@2.40Ghz
Memory	8GB RAM
CYPHONIC Node（LinuxOS）	
OS	Ubuntu 22.04.1 LTS
CPU	11th Gen Initel Core i5-1135G7@2.40Ghz
Memory	8GB RAM

表 2　ICMP を用いた通信性能

		Windows OS	Linux OS
RTT	min	8.0 ms	4.9 ms
	max	132.5 ms	227.6 ms
	avg	22.5 ms	43.5 ms

4.2 性 能 評 価

表 3　TCP を用いた通信性能

Windows OS		Linux OS	
Transfer	Throughput	Transfer	Throughput
36.0 MBytes	30.2 Mbps	42.3 MBytes	35.4 Mbps

表 4　UDP を用いた通信性能

	Windows OS		Linux OS	
Traffic	Throughput	Jitter	Throughput	Jitter
10Mbps	9.9 Mbps	1.09 ms	9.9 Mbps	1.523 ms
20Mbps	19.8 Mbps	0.917 ms	19.7 Mbps	0.794 ms
30Mbps	29.7 Mbps	1.001 ms	28.7 Mbps	2.184 ms
40Mbps	39.6 Mbps	1.496 ms	38.2 Mbps	0.554 ms

性能評価では，iperf3 を用いて，TCP および UDP の通信スループットを測定した．測定値は，10 秒間の測定を 10 セット行い，取得した値の平均値とする．表 3 に，iperf3 による TCP パフォーマンスの測定結果を示す．測定結果より，WindowsOS 端末を用いた CYPHONIC を利用した通信と LinuxOS 端末を用いた CYPHONIC を利用した通信において，同等の通信性能が観測されている点から，TCP において十分な通信パフォーマンスが確認された．

表 4 に，iperf3 による UDP パフォーマンスの測定結果を示

す．TCP と同様に，Windows OS 端末を用いた CYPHONIC を利用した通信と Linux OS 端末を用いた CYPHONIC を利用した通信において， 10 Mbps， 20 Mbps， 30 Mbps， 40 Mbps のいずれの通信性能も同等の性能が確認された．また， Jitter においても通信に影響を及ぼさない値が観測されたため， UDP においても十分な通信パフォーマンスが実現されていることを確認した．

4.3 インストーラーの動作検証

図 5　インストーラーの動作検証

Windows OS 端末上で提案したインストーラーを実行した．図 5 にインストーラーの実行後の画面を示す．

実行後，Windows OS 端末上のローカル内に CYPHONIC EXE，Setup Packege，CYPHNIC Starter Application がインストールされていることを確認した． また， CYPHNIC Starter Application を実行し，その後適切な入力を行なった場合，CYPHONIC Daemon が起動することを確認した．これにより，GUI アプリケーションの操作のみで，CYPHONIC を用いた通信を Windows OS 端末で利用可能であると結論付けた．

4.4 結果および考察

動作検証より，Linux OS と WIndows OS で CYPHONIC を用いた通信性能を比較したとき，RTT，TCP，UDP 共に大きな性能差がないことを確認した．さらにインストーラーの動作検証より，インストーラーが想定される挙動を取ることを確認した．

5. ま と め

本論文では，CYPHONIC のより汎用的な利用を実現するために Windows OS 用の端末機能を提案した．実装では， Windows OS のカーネル空間を操作する DLL を加え， 仮想インターフェイスを作成し， ユーザ空間の操作のみを行うモジュールに変更を加えることなく Windows OS 用の端末機能を実現した．加えて， インストーラーを作成することで， ユーザが容易に CYPHONIC を導入することが可能なツールを提供した．検証評価より， Windows OS 用の端末機能と Linux OS 用の端末機能を比較し， 通信パフォーマンスに大きな差が観測されなかったことから，実用性を確認した．さらに，インストーラーが想定通りの動作をすることを確認した．以上のことから Linux OS 以外の OS に対応したことでより CYPHONIC における汎用性を実現した．

文　　献

[1] Y.Zou，J.Zhu，X.Wang and .Hanzo，“A Survey on Wireless Security: Technical Challenges，Recent Advances，and Future Trends，” 10.1109/JPROC.2016.2558521，September 2016.

[2] A.Deshpande，“A Study on Rapid Adoption of Zero Trust Network Architectures by Global Organizations Due to COVID-19 Pandemic，” 10.9734/bpi/nvst/v1/3640F，August 2021.

[3] Kandukuri，B.Reddy，V.，R.Paturi，Rakshit and Atau，“Cloud Security Issues，” 10.1109/SCC.2009.84，September，2009.

[4] Y.Rekhter and Eric C.Rosen，“BGP/MPLS IP Virtual Private Networks(VPNs)，” 10.17487/RFC4364，February 2006.

[5] Alawneh，Muntaha，Abbadi，M.Imad，“Integrating Trusted Computing Mechanisms with Trust Models to Achieve Zero Trust Principles，” 10.1109/IOTSMS58070.2022.10062269，December 2022.

[6] Campbell and Mark，“Beyond Zero Trust: Trust Is a Vulnerability，” 10.1109/MC.2020.3011081，October，2020.

[7] Miller，Courtney，Rodeghero，Paige，Storey，Margaret-Anne，Ford，Denae，Zimmermann and Thomas，“Survey Instruments for ”How Was Your Weekend?” Software Development Teams Working from Home During COVID-19，” 10.1109/ICSE-Companion52605.2021.00101，May，2021.

[8] Syed，N.Firdous，Shah，Syed W.，Shaghaghi，Arash，Anwar，Adnan，Baig，Zubair，Doss and Robin，“Zero Trust Architecture(ZTA): A Comprehensive Survey，” 10.1109/ACCESS.2022.3174679，May，2022.

[9] Müller，Andreas，Carle，Georg，Klenk and Andreas，“Behavior and classification of NAT devices and implications for NAT traversal，” 10.1109/MNET.2008.4626227，September，2008.

[10] Charles E. Perkins，“IP Mobility Support，”，10.17487/RFC2002，October 1996.

[11] T.Eckert and D.Wing “IP Multicast Requirements for a Network Address Translator(NAT)and a Network Address Port Translator(NAPT)，” 10.17487/RFC5135，February，2008.

[12] Ferdous，Md.Sadek，Chowdhury，Farida，Acharjee and J.Chandra，“An Extended Algorithm to Enhance the Performance of the Current NAPT，”，10.1109/ICICT.2007.375401，March，2007.

[13] “Internet Protocol，” 10.17487/RFC0791，September 1981.

[14] S.Kawamura and M.Kawashima“A Recommendation for IPv6 Address Text Representation，” 10.17487/RFC5952，August 2010.

[15] M.A.Hossain，D.Podder，S.Jahan and M.Hussain，“Performance Analysis of Three Transition Mechanisms between IPv6 Network and IPv4 Network: Dual Stack，Tunneling and Translation，” March 2016.

[16] Ferrari，Paolo，Flammini，Alessandra，Marioli，Daniele，Taroni and Andrea，“IEEE802. 11 sensor networking，”，10.1109/TIM.2006.870105，2006.

[17] Saliba，Danielle，Imad，Rodrigue，Houcke，Sebastien，Hassan，B.El，“WiFi Dimensioning to offload LTE in 5G Networks，”，10.1109/CCWC.2019.8666585，2019.

[18] S.Isomura，T.Yoshikawa，H.Komura，S.Kubota，C.Nishiwaki and K.Naito，“Prototyping evaluation of CYPHONIC: Overlay network technology for Cyber-Physical communication，”，10.1109/CCNC49032.2021.9369500，2021.

[19] 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.

[20] 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.

[21] Luo，Lan，Zou，Cliff，Narain，Sashan，Fu，Xinwen，“On Teaching Malware Analysis on Latest Windows，” 2022.