愛知工業大学 情報科学部 情報科学科
コンピュータシステム専攻

令和6年度 卒業論文

オーバーレイネットワークプロトコルにおける簡易パケット認証に関する研究

2025年2月

研究者

K21071 須田 倫代

指導教員 内藤克浩 教授

目 次

図 目 次

表 目 次

第1章

序論

1.1 背景

Information Technology (IT) 技術の革新的進展により，ネットワークおよび社会の分散化・オープン化が進行し，中央集権的なシステムが解体される過程にある．この結果，個々の主体が情報発信の役割を担う状況が徐々に形成されている．デジタル機器の普及に伴い，個人がデジタル情報を扱う機会が増加したことで，個人間の情報交換の手段として Peer to Peer (P2P) 通信を活用したサービスの利用が拡大している [1,2].

P2P とは，ネットワークに接続されたコンピュータ同士がサーバを介さずに直接通信する方式である．P2P 方式に対し，コンピュータ同士がサーバを介して通信する方式はクライアント・サーバ方式と称される．クライアント・サーバ方式は，アクセス集中による高負荷，経路冗長による通信遅延，サーバが単一障害点となり得る側面から懸念が存在する [3]．一方，P2P 方式は，サーバを介さない通信をするため，負荷集中や経路冗長問題の回避が可能となる [4]．P2P 通信を利用した身近なサービス例として，Soccial Networking Service (SNS) アプリケーションの LINE やファイル共有ソフトの Winny などがある．

現在のインターネットでは，Internet Protocol (IP) という規格に従って通信が行われるため，インターネット上で P2P 通信を実現する際には，IP 通信を前提とする必要がある．IP 通信には，端末間直接通信を阻害する通信接続性の課題や，モバイル端末の移動により通信が切断される移動透過性の課題が存在する．

通信接続性とは，端末のネットワーク環境に影響されることなく，相互に通信が可能

な状態のことである．コンピュータがインターネットにアクセスする際には，Internet Protocol (IP) アドレスを使用する [5]．インターネットに接続するコンピュータの増加に伴い，IP アドレスが枯渇したことで対応策が導入されている [6]．代表的な対応策として，Network Address Translation (NAT) 及び Network Address Port Translation (NAPT) と Internet Protocol version 6 (IPv6) がある [7–9]．しかし，これらの対応策を導入すると，NAT/NAPT 超え問題や Internet Protocol version 4 (IPv4)-IPv6 非互換性問題によって，通信が困難な場合が発生する [10,11]．NAT/NAPT 超え問題や IPv4-IPv6 非互換性問題を解決するために，多くの技術が提案されている [12–16]．

また，移動透過性とは，通信端末が移動しても通信が切断されない状態のことである．スマートフォンなどのモバイル端末の普及により，移動しながら通話をしたり，インターネットを利用したりすることが増加している．モバイル端末は複数の通信規格を備えており，利用するネットワーク環境に応じて通信規格を切り替えて通信を行う [17,18]．そのため，移動によって接続するネットワークが切り替わると通信が切断されてしまう [19]．移動によって通信が切断されてしまう問題に対しても，様々な技術が提案されている [20–22]．

一方で，安全な通信を実現するためには、ゼロトラストに基づくセキュアな通信が求められる．ゼロトラストとは，従来の境界型セキュリティモデルのような境界内の端末は安全と捉える考え方に対し，境界の概念を撤廃し，全ての端末を信頼しないという考え方に基づくセキュリティモデルである [23]．そのため，セキュアな端末間通信を実現するには，端末認証と通信内容の暗号化が必要である．

著者らの研究グループでは，これらの課題を解決する P2P 通信技術の一つとして，CYber PHysical Overlay Network over Internet Communication (CYPHONIC) を提案している [24–28]．CYPHONIC は，物理的なネットワークの上に仮想的なネットワークであるオーバーレイネットワークを構築する．オーバーレイネットワークの構築により，通信接続性や移動透過性の課題を解決し，端末のネットワーク環境に関わらず，セキュアな端末間通信を提供する．

また，CYPHONIC は，CYPHONIC ノードと CYPHONIC クラウドから構成される．CYPHONIC ノードは，CYPHONIC のクライアントプログラムを搭載した端末である．一方で，CYPHONIC クラウドは，CYPHONIC ノードや通信経路の管理を行うサーバ群である．CYPHONIC ノードは，通信開始前に CYPHONIC クラウドと連携をとるこ

とで，その後の端末同士の直接通信が可能となる．このように，P2P でも P2P ネットワークにサーバを用いる通信方式は，ハイブリッド P2P と呼ばれる [29].

しかし，サーバを基盤としたサービスは，サイバー攻撃に対して脆弱である．近年，サイバー攻撃の脅威は増大しており，インターネット上の様々な種類のサイバー攻撃が存在することを考慮する必要がある [30]．インターネット上で行われるサイバー攻撃の種類には，主に不正アクセス，なりすまし，脆弱性を狙った攻撃，サービス妨害，盗聴がある [31,32]．不正アクセスには，ブルートフォース攻撃を例とした可能性のある全ての組み合わせを試行する攻撃，なりすましには，マルウェア感染，リプレイ攻撃などがある [33–35]．また，脆弱性を狙った攻撃には，SQL インジェクションやクロスサイトスクリプティング，サービス妨害攻撃には，DoS 攻撃や DDoS 攻撃が挙げられる [36,37]．一般的に不正アクセスやなりすまし，脆弱性を狙った攻撃，サービス妨害攻撃に対しては，ファイアウォールや IDS (Intrusion Detection System)/ IPS (Intrusion Prevention System) の使用によって不正な通信を遮断する対策が講じられている [38]．特に，Web アプリケーションへの通信に対しては，WAF (Web Application Firewall) の使用が有効である [39]．盗聴に対しては，通信を暗号化することにより，外部からの盗み見を防止する．加えて，現在では，侵入検知，マルウェア分析，脆弱性評価などに機械学習や人工知能 (AI) を用いて自動化を図っている [40].

先述した攻撃のうち，サービス妨害攻撃によって莫大な損失が発生する恐れがある [41,42]．サービス提供を行うサーバが DoS 攻撃の被害を被ると，サーバに負荷が掛かり，最悪の場合サーバがダウンする．サーバを攻撃されることは，機密情報や個人情報の漏洩やサービスの利用停止が発生する恐れがあるため，金銭的な損失や社会的な信頼を失うことに繋がる [43]．また，リプレイ攻撃によって，サーバ内の情報に対して不正な操作を行うと，情報の整合性が損なわれる [44]．そのため，サーバのセキュリティ対策が必須となる．

CYPHONIC において，CYPHONIC クラウドはサーバの働きをするため，セキュリティ対策を講じる必要がある．CYPHONIC は，CYPHONIC ノードと CYPHONIC クラウドが連携することで通信が実現される．CYPHONIC ノードは，端末間直接通信を開始する前に，CYPHONIC クラウドと通信を行い準備プロセスを実行する．しかし，攻撃者によって CYPHONIC クラウドがリプレイ攻撃や DoS 攻撃の被害を被り，サービス提供不能に陥ると端末間直接通信が実現不可能となる．

CYPHONIC クラウドがリプレイ攻撃や DoS 攻撃の被害を被る場合として，ネットワーク上を流れる CYPHONIC 独自のパケットを再利用した不正な遅延パケットを処理した場合が考えられる．CYPHONIC では設計上，同じ形式のパケットを通信中のあらゆる場面で送受信する．そのため，攻撃者がネットワーク上に流れている CYPHONIC のパケットをキャプチャし，CYPHONIC クラウドに送信すると，CYPHONIC クラウドはそのパケットを処理してしまう可能性がある．処理する不正パケットが増加すれば，CYPHONIC クラウドに負荷が集中し，サービス提供が停止する恐れがある．また，キャプチャした CYPHONIC パケットを改ざんし，CYPHONIC クラウドに対して再送信すると，CYPHONIC クラウドに存在する端末の不当な情報更新が発生するリスクがある．端末の不当な情報更新が発生すると，実際の端末情報との齟齬が生じ，その端末は相手端末との通信が不可能となる．

したがって，CYPHONIC クラウドは受信するパケットが正規のユーザからの正当なパケットであるか否かを確認する仕組みが必要である．CYPHONIC クラウドに対するリプレイ攻撃や DoS 攻撃は，CYPHONIC が持つ直接通信の意義を大いに損なうことに繋がるため，CYPHONIC クラウドを攻撃から保護する対策が必要となる．

1.2　本研究の目的

本研究では，CYPHONIC クラウドをリプレイ攻撃や DoS 攻撃から保護する簡易パケット認証を提案する．リプレイ攻撃や DoS 攻撃に利用される不正なパケットは，ネットワーク上を流れる CYPHONIC 独自のパケットをキャプチャし，再送信した遅延パケットが想定される．

従来の CYPHONIC では，リプレイ攻撃対策として，Sequence Number と Hash-Based Message Authentication Code(HMAC) を用いたパケット検証を提案している．この方法では，メッセージを送信毎に Sequence Number を増加させ，HMAC でメッセージの改ざん検知を行うことによって，正当なパケットであるか否かのパケット検証をしている．

しかし，従来のパケット検証のみのリプレイ攻撃対策では，パケット検証における CYPHONIC クラウドの処理負荷や処理時間の増加が課題となる．CYPHONIC は設計上，CYPHONIC クラウドの負荷に応じてスケールアウトし，負荷分散する仕様になっている．CYPHONIC クラウドがスケールアウトすると，受信したパケットがリプレ

イされたものであるかの判断をスケールアウトした CYPHONIC クラウド同士がデータベースを介して共有する必要が発生してしまい，CYPHONIC クラウドの処理負荷や処理時間が増加する．加えて，HMAC の処理には時間がかかるため，膨大なトラフィックを処理することを想定している CYPHONIC において，多くの不正なパケットを処理することは CYPHONIC クラウドにとって負荷となり，通信パフォーマンスを損なう原因となる．

また，CYPHONIC パケットを用いた DoS 攻撃を CYPHONIC クラウドに仕掛けられると，CYPHONIC クラウドが機能停止してしまうことも考えられる．そのため，不正パケットは可能な限り，CYPHONIC クラウドで処理することなく破棄し，認証済端末からの正規のパケットのみを処理する仕組みを実現する．また，検証および評価として，動作検証と性能検証を行う．動作検証では，簡易パケット認証機能を実装した CYPHONIC ノードと CYPHONIC クラウドで正常に簡易パケット認証が実施されていることを確認する．また，性能検証では，簡易パケット認証のパケット遅延時間に対するパケット破棄率を評価する．

1.3　本論文の構成

本論文は，次の章立てで構成される．第 2 章では，本研究に関係する既存技術を取り上げて説明する．第 3 章では，著者らの研究グループが提案する CYPHONIC について説明する．第 4 章では，CYPHONIC クラウドをリプレイ攻撃や DoS 攻撃から保護するための簡易パケット認証手法を提案する．第 5 章では，提案手法の実装の詳細について説明する．第 6 章では，簡易パケット認証の動作検証と性能評価を行う．第 7 章では，本論文のまとめ及び今後の課題について述べる．

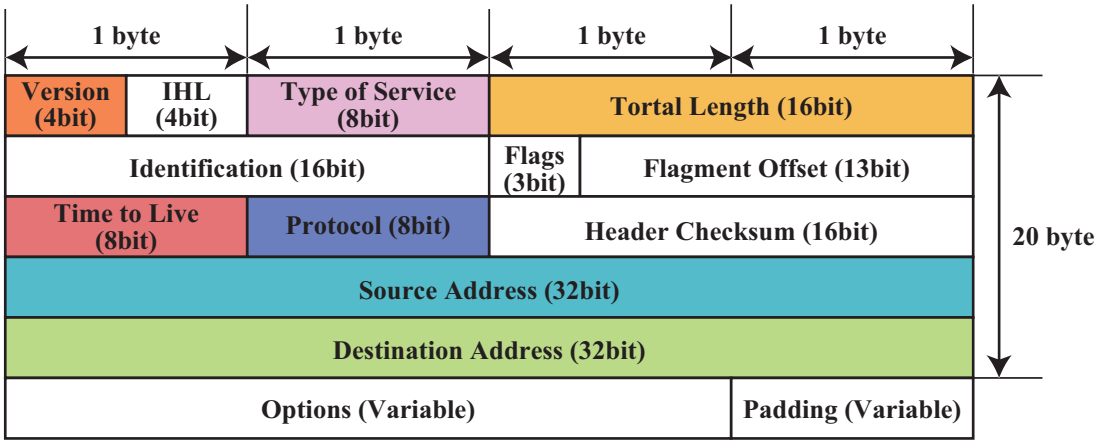
第2章

関連技術

2.1 Internet Protocol

Internet Protocol (IP) とは，インターネット上のコンピュータ通信におけるデータの送受信方法を規定した通信プロトコルである．IP は，OSI 参照モデルでは第 3 層のネットワーク層，Transmission Control Protocol (TCP)/IP 階層モデルでは第 2 層のインターネット層に位置付けられる．IP による通信では，通信データをパケットと呼ばれる単位に分割し，個々に宛先や送信元などの制御情報を付加して送受信をするパケット交換方式で通信を行う．また，IP は，事前にコンピュータ間でコネクションを確立せずにデータ転送を行うコネクションレス型のプロトコルである．コネクションを確立しないため，オーバーヘッドが少ない一方で，宛先へ確実にデータが到達する保証はない．そのため，IP は宛先に到達不可能となったデータを再送しない．さらに，ネットワーク通信において，実現し得る最大限の通信速度を確保する一方で，十分な品質を保証しないベストエフォートの特性を持つ．従って，IP は信頼性の低いプロトコルであるため，確実なデータの送受信を行うには，TCP などのトランスポート層のプロトコルによって通信を制御する必要がある．

また，IP の通信では IP アドレスが使用される．IP アドレスとは，インターネット上に存在する端末を一意に特定するための識別子であり，ネットワーク上の位置情報と通信識別子を同時に表現している．元来から現在に至るまで，インターネットでは Internet Protocol version 4 (IPv4) が広く利用されてきた．IPv4 アドレスは 32bit の 2 進数で表現されるデータであるため，割り当て可能なアドレス数は 2^{32} 個である．即ち，



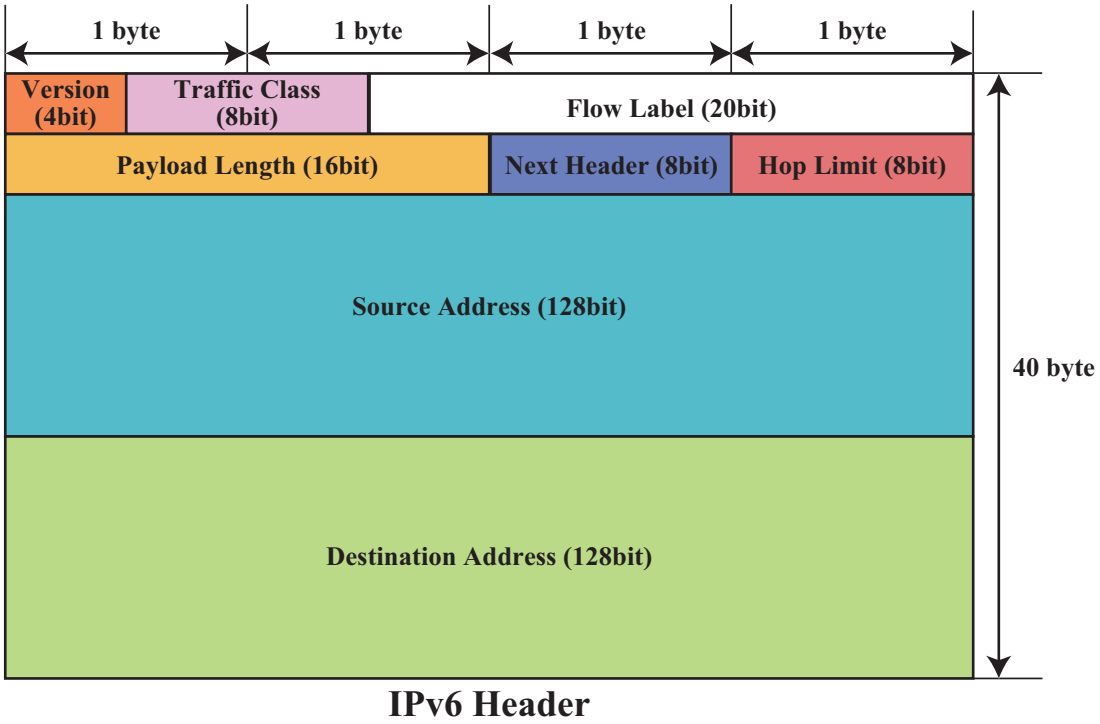


図 2.1 Protocol header:IPv4 and IPv6

2024 年の世界人口約 82 億人に対して, IPv4 のアドレス領域は不足している. IPv4 の枯渇問題を解消する対策の 1 つとして，Internet Protocol version 6 (IPv6) が登場した. IPv6 アドレスは 128bit の 2 進数で表現されるデータであるため，割り当て可能アドレス数は 2^{128} 個である. 故に，世界の全ての端末に対して一意にアドレスを割り振ることが可能となる. しかし，IPv4 から IPv6 への完全な移行には時間を要するため，当面の間，IPv4 と IPv6 が混在した環境が利用される. IPv4 と IPv6 が混在した環境をデュアルスタック環境と呼ぶ. デュアルスタック環境では IPv4 を使用する端末と IPv6 を使用する端末が通信を行う場合が考えられる. しかし，図 2.1 に示すように，IPv4 と IPv6 はパケットの構造が異なる. そのため，IPv4 と IPv6 には互換性がなく互いに通信が困難であることに起因して，デュアルスタック環境では，IPv4 と IPv6 の相互運用上の問題が存在する.

2.2　アドレス変換技術

従来，IP のバージョンとして IPv4 が広く普及してきた. しかし，IPv4 アドレスは数に限りがあるため，現在においてインターネットに接続する全ての端末に一意なアドレスを割り当てることは不可能である. そのため，1990 年代後半から 2000 年代初頭にかけて，IPv4 アドレスを節約するために，Network Address Translation (NAT) 及び Network Address and Port Translation (NAPT) が提案された. 以下のセクションで NAT 及び NAPT について詳細に説明する.

2.2.1　Network Address Translation

Network Address Translation (NAT) とは，パケットがルータを通過する際に，プライベート IP アドレスとグローバル IP アドレスを相互に変換する技術である. 図 2.2 に NAT の概要を示す. NAT による IP アドレスの変換により，インターネットをグローバルネットワークと複数のプライベートネットワークに分割可能となる. グローバル IP アドレスは，グローバルネットワークで使用される IP アドレスであり，世界で重複が許されないアドレスである. プライベート IP アドレスは Local Area Network (LAN) などのプライベートネットワークでのみ使用されるアドレスであり，同一プライベート

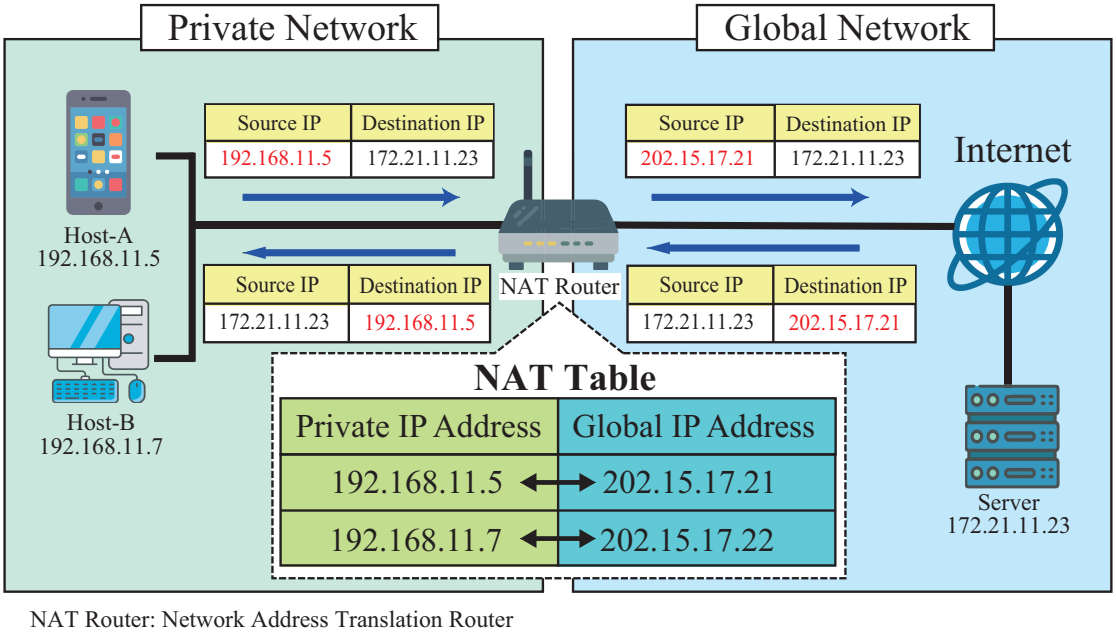


図 2.2 Overview of Network Address Translation

ネットワーク内でのみ一意なアドレスである．これにより，異なるプライベートネットワークでは，重複したプライベート IP アドレスを割り当て可能であるため，IP アドレス枯渇問題の対応策となる．

NAT は，プライベートネットワークとグローバルネットワークの境界に設置される．NAT の技術を使用してグローバルネットワークに接続する端末は，プライベートネットワークに所属し，プライベート IP アドレスが設定される．端末がグローバルネットワークにアクセスする際，NAT はパケットに付加されたプライベート IP アドレスを一意なグローバル IP アドレスへと変換する．これにより，端末はインターネットへの接続が可能となる．この際，NAT の機能を有したルータは，プライベート IP アドレスとグローバル IP アドレスの対応情報をマッピングテーブルに保持する．グローバルネットワークからの返信パケットは，宛先 IP アドレスがグローバル IP アドレスとなっている．返信パケットを受信した NAT ルータは，自身のマッピングテーブルを確認して，対応するプライベート IP アドレスに変換し，プライベートネットワークの

端末宛にパケットを送る．マッピングテーブルに保持された情報は，使用されなかった時間が一定時間経過した際に消去される．

一方で，NAT 配下の同一なプライベートネットワークに属する複数の端末が，1 つのグローバル IP アドレスに変換して通信を行う環境では課題が存在する．この際，プライベート IP アドレスを持つ複数の端末は，全て同一のグローバル IP アドレスに変換されて通信を行うこととなる．しかし，複数台の端末が同時に通信を行った場合，返信パケットに含まれるグローバル IP アドレスから対応したプライベート IP アドレスを一意に変換することが不可能である．このように，NAT 配下のプライベートネットワークに存在する複数の端末によるグローバルネットワークへの同時接続は不可能である．この問題を解消するには，グローバルネットワークに接続する端末分のグローバル IP アドレスが必要になるが，使用する IP アドレスを削減するという効果を失ってしまう．

また，NAT を使用している環境では，グローバルネットワークに存在する端末に対して，変換前のプライベート IP アドレスが隠蔽される．この特性は，ファイアウォールとして機能する反面，端末間の直接通信の実現を困難にさせる要因でもある．このように，NAT によって端末間の直接通信が妨げられる問題を NAT 超え問題と呼ぶ．

2.2.2　Network Address Port Translation

Network Address Port Translation (NAPT) は，グローバル IP アドレスとプライベート IP アドレスの相互変換の際に，同時にポート番号も変換する技術である．図 2.3 に NAPT の概要を示す．前述の NAT の説明では，NAT のアドレス変換が一対一対応であるため，同一のプライベートネットワークに接続された複数の端末が，グローバルネットワークに存在する端末と同時通信が不可能であるという問題があった．この問題を解決するために考案された技術が NAPT である．NAPT は，IP アドレスの変換に加え，ポート番号も変換する．ポート番号とは，TCP や User Datagram Protocol (UDP) で規定されたプログラムを識別する番号である．NAPT では，IP アドレスとポート番号を変換することにより，1 つのグローバル IP アドレスに対応する，複数のプライベート IP アドレスとそのポート番号を識別可能な一対多対応を実現している．故に，プライベートネットワークに存在する複数端末は，1 つのグローバル IP ア

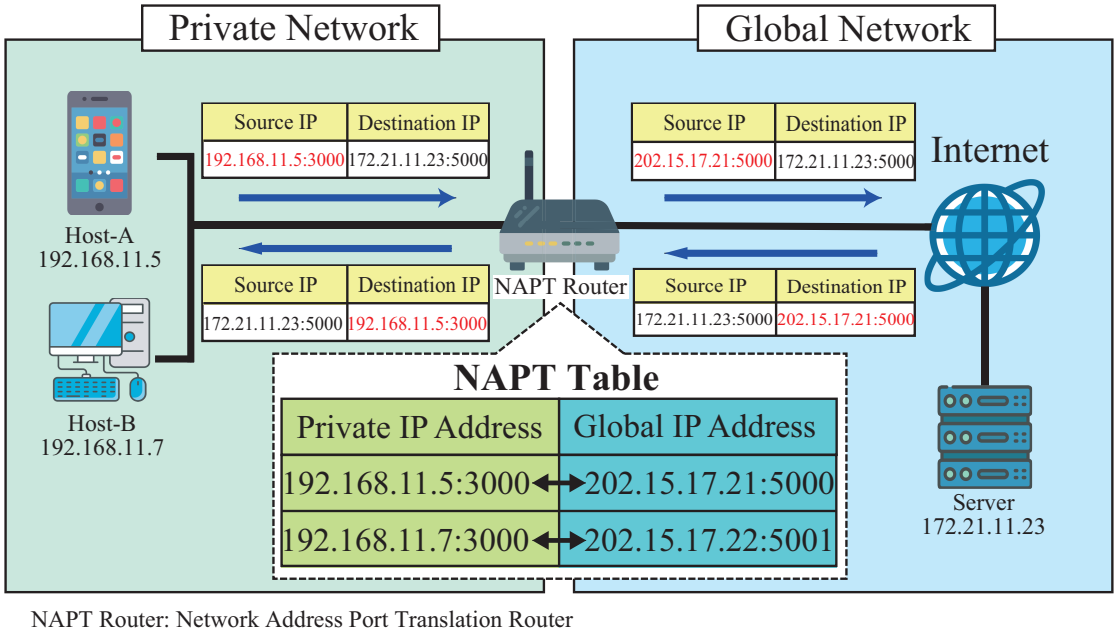


図 2.3 Overview of Network Address Port Translation

ドレスを用いて，グローバルネットワークに向けた同時通信を可能にし，NAT の課題を解消している．さらに，この方法であれば，IP アドレスの消費を抑えることが可能なため，IPv4 アドレス枯渇問題の具体的な対応策として，現在のインターネット環境では NAPT が広く利用されている．

プライベートネットワークに存在する端末が，グローバルネットワークに存在する端末に向けて通信を開始すると，NAPT 機能を有したルータは，送信元のパケットに付加されたプライベート IP アドレスをグローバル IP アドレスに変換する．同時に，送信元パケットに付加された送信元ポート番号を，その時点でマッピングテーブルに存在しないエフェメラルポート番号に変換する．エフェメラルポート番号とは，一時的な用途のために動的に使用されるポート番号である．NAPT の場合，マッピングテーブルでの情報保持が一時的であることから，エフェメラルポートが使用される．端末からのグローバルネットワークへの通信に対する返信パケットには宛先 IP アドレス

にグローバル IP アドレスが指定されている．返信パケットを受信した NAPT ルータは，自身のマッピングテーブルを参照し，グローバル IP アドレスを対応するプライベート IP アドレスに変換する．その際，同時に，宛先ポート番号として指定されたエフェメラルポート番号を対応する変換前のポート番号に変換し，プライベートネットワークの端末宛にパケットを送る．NAT 同様，マッピングテーブルに保持された情報は，使用されなかった時間が一定時間経過した際に消去される．

NAPT も NAT と同様，グローバルネットワークに存在する端末に対してプライベートネットワークに存在する端末が隠蔽される．そのため，IP フィルタリングやポートフィルタリング等の設定が可能であり，ファイアウォールとしての機能も果たす．しかしながら，NAT 同様，グローバルネットワークに存在する端末は変換前のプライベート IP アドレスとポート番号を認識することが不可能である．したがって，プライベートネットワークに存在する端末に対して，グローバルネットワークまたは他のプライベートネットワークに存在する端末からの直接通信が極めて困難である．このような問題は NAPT 超え問題と呼ばれる．

NAPT は，変換の際に，マッピングテーブルに記録される変換情報の特性とパケットのフィルタリング規則から，Full Cone NAPT，Restricted Cone NAPT，Port Restricted Cone NAPT，Symmetric NAPT の 4 種類に分類される．フィルタリング規則とは，グローバルネットワークからプライベートネットワークに宛てられたパケットを選別し，中継の可否を判断する際の規則である．Full Cone NAPT は，パケットに付加される，宛先のグローバル IP アドレスとポート番号が正しければ NAPT 超えに成功する．たとえ，一度も通信をしたことがない端末からのパケットでも，宛先のグローバル IP アドレスとポート番号が合っていれば，NAPT 超えに成功する．Restricted Cone NAPT は，パケットに付加される宛先グローバル IP アドレスとポート番号が正しく，かつ，パケットを送信する端末がパケットを受信する端末と通信をしたことがある場合に NAPT 超えに成功する．この場合は，送信元グローバル IP アドレスが同じであればポート番号が異なっていても NAPT 超えが可能である．しかし，送信元の端末が宛先の端末と一度も通信をしたことがない場合は，NAPT 超えをすることは不可能である．Port Restricted Cone NAPT は，Restricted Cone NAPT と同様，パケットに付加される宛先グローバル IP アドレスとポート番号が正しく，かつ，パケットを送信する端末がパケットを受信する端末と通信をしたことがある場合に NAPT 超えに成功するが，

送信元グローバル IP アドレスが同じであっても，ポート番号が異なれば NAPT 超えをすることが不可能である．また，送信元の端末が宛先の端末と一度も通信をしたことがない場合も，NAPT 超えをすることは不可能である．Symmetric NAPT は，初めに通信をしたグローバル IP アドレスとポート番号のみに対して，専用ポートを用意し，その他のグローバル IP アドレスとポート番号からのパケットは全て破棄する．

2.3　通信接続技術

端末がどのようなネットワーク環境に接続されている場合でも通信が可能となる性質を，通信接続性と呼ぶ．通信接続性を脅かす要因は 2 つ存在する．ひとつは，NAT/NAPT 超え問題である．これは，NAT/NAPT が配下のプライベートネットワークで使用するプライベート IP アドレスが, 外部ネットワークから隠蔽されることにより, 直接通信を妨げてしまう問題である．もうひとつは，IPv4-IPv6 非互換性問題である．IPv4 と IPv6 が混在するデュアルスタック環境では，IPv4 と IPv6 のパケット構造が異なるため相互運用が困難であるという問題が存在している．これらによって端末間の直接通信が実現できない問題は，通信接続問題と呼ばれる．よって，通信接続性を確保するためには，NAT/NAPT 超え問題の解決や IPv4-IPv6 相互通信を実現する技術が必要となる．NAT/NAPT 超え問題を解決し，プライベートネットワークに存在する端末への直接通信を実現する技術を NAT Traversal 技術と呼ぶ．具体的な NAT Traversal 技術として，UDP Hole Punching，Session Traversal Utilities for NAPT (STUN)，Traversal Using Relays around NAPT (TURN)，Interactive Connectivity Establishment (ICE) が存在する．また，デュアルスタック環境において，IPv4-IPv6 の相互通信を実現する技術として，DualStack，Translator，Tunneling，が存在する．以下のセクションにて，これら通信接続性の詳細を説明する．

2.3.1　UDP Hole Punching

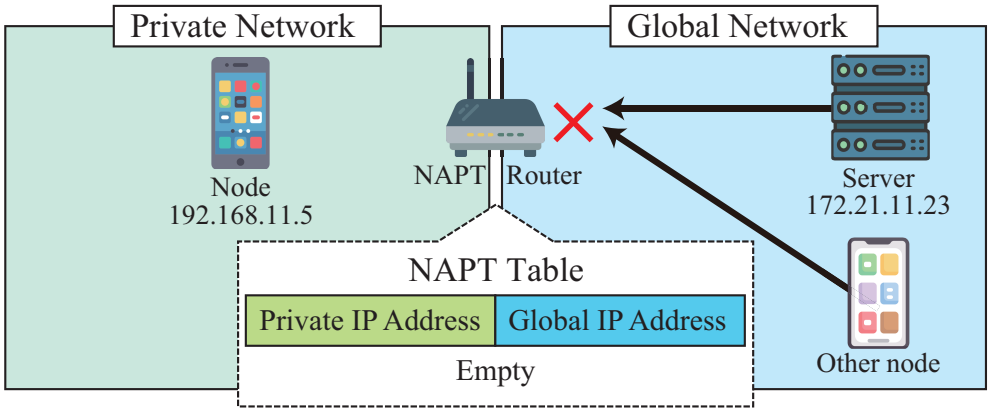
UDP Hole Punching とは，グローバルネットワークからプライベートネットワーク内の端末へ接続を行う NAPT 超えを実現する手法のひとつである．UDP Hole Punching の動作概要を図 2.4 に示す．UDP Hole Punching では，内部から通信を行なった後，一

定期間のみ外部ネットワークからの通信を許可する NAPT の性質を利用する．NAPT 配下の端末がグローバルネットワークへの通信を行う以前には，NAPT のマッピングテーブルにアドレス変換情報は記録されていない．そのため，グローバルネットワークからプライベートネットワークへの接続は不可能である．NAPT 配下の端末がグローバルネットワークへの通信を開始すると，NAPT は，パケットに含まれる IP アドレスとポート番号の変換情報を自身のマッピングテーブルに記録する．よって，グローバルネットワークからの通信が可能となる．しかし，NAPT はマッピングテーブルに記録されている変換情報を一定期間しか保持しないため，一定期間を過ぎると変換情報は削除されてしまう．そのため，UDP Hole Punching ではマッピングテーブルの変換情報が保持され続けるように，NAPT 配下の端末から一定間隔で UDP パケットを送信し続ける．この様にして，グローバルネットワークからの通信接続性を実現する．

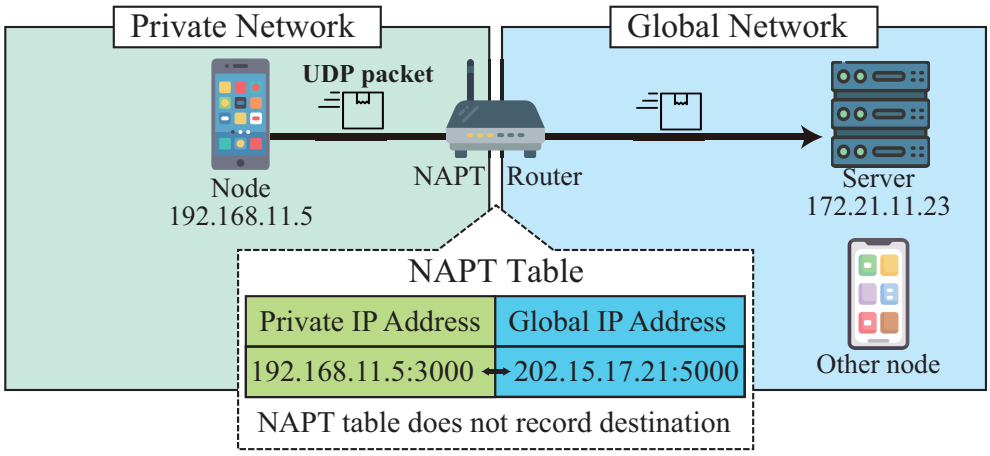
しかし，両端末が NAPT 配下に存在する場合，UDP Hole Punching を行っても，通信接続性の担保は困難である．これは，NAPT 配下の端末は，NAPT が生成した変換後のポート番号を互いに知ることが極めて困難なためである．

2.3.2　Session Traversal Utilities for NAPT

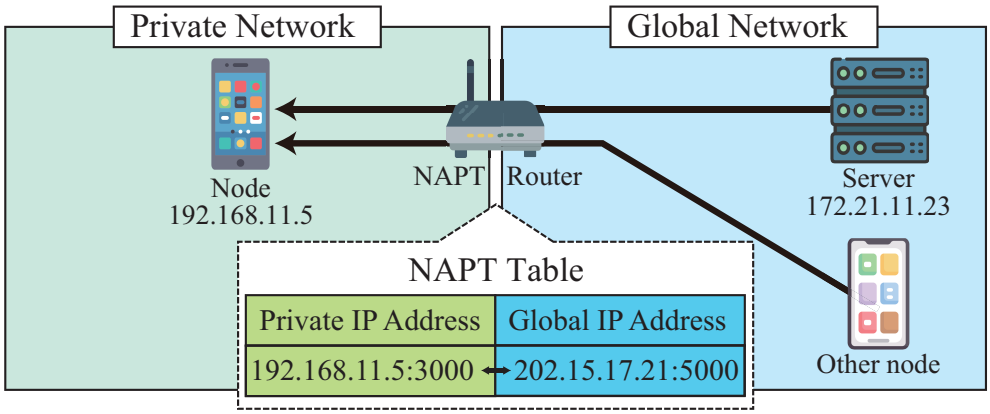
UDP Hole Punching では両端末が NAPT 配下に存在する場合，通信接続性を実現することが極めて困難である．これは，NAPT によって生成された変換情報を互いの端末は知る術がないためである．Session Traversal Utilities for NAPT (STUN) は，STUN サーバをグローバルネットワークに設置して，双方の変換情報を記録し，相手端末に通知することで通信接続性を実現するクライアントサーバ型プロトコルである．STUN の動作概要を図 2.5 に示す．STUN は，グローバルネットワークに設置された STUN サーバと NAPT 配下に設置された STUN クライアントから構成される．STUN クライアントは，相手端末への通信を開始する前に，STUN サーバに STUN Binding Request を UDP で送信する．STUN サーバは，STUN Binding Request に対して STUN Binding Response を返す．STUN Binding Response には，STUN サーバから見た STUN クライアントのグローバル IP アドレスとポート番号が含まれている．STUN サーバは双方のリクエストに対して STUN クライアントの変換後のアドレス情報をレスポンスとして返す．端末が自身の変換規則を取得した後，端末同士が通信接続を行う際に，先に



① **Unable to start communication from Global Network to Node**



② **Node sends UDP packets to generate NAPT table and update expiration time**



③ **Able to communicate with Node (202.15.17.21:5000) from Global Network**

NAPT Router: Network Address Port Translation Router

図 2.4 Overview of UDP Hole Punching

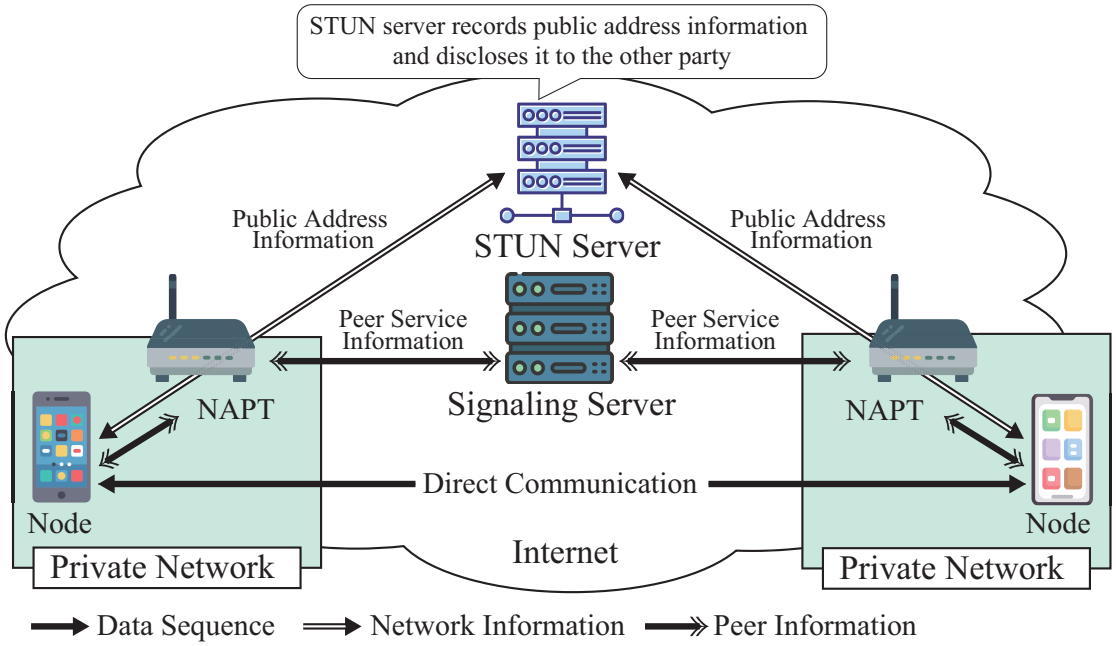


図 2.5 Overview of Session Traversal Utilities for NAPT

取得した変換規則をシグナリングサーバを通じて交換する．これにより，双方の端末は通信相手の変換情報を相互に得るため，両端末が NAPT 配下に存在する場合でも通信接続性の実現が可能となる．

しかし，STUN は，Symmetric NAPT を使用している環境において，通信接続性を確保することが困難である．Symmetric NAPT は，初めに通信をしたグローバル IP アドレスとポート番号からのみ通信を受け付けるため，最初に通信を行った STUN サーバからのアクセスのみを許可する．そのため，何れかの端末が Symmetric 型 NAPT 配下に存在する場合，STUN では通信接続性の実現が困難である．

2.3.3　Traversal Using Relays around NAPT

STUN では，両端末のどちらかが Symmetric 型 NAPT 配下に存在する場合，通信接続性を実現することが極めて困難である．この問題を解決し，Symmetric 型 NAPT が存在する場合でも NAPT 超えを実現する技術として，Traversal Using Relays around NAPT

(TURN) が存在する．Symmetric 型 NAPT 配下の端末は，グローバルネットワーク上に公開された外部サーバに対しては問題なく接続可能である．このため，Symmetric 型 NAPT 配下の端末同士で接続する際には，外部サーバにて通信を中継することにより NAPT 越えが実現可能である．TURN サーバは，上記の外部サーバの役割を担う．TURN の動作概要を図 2.6 に示す．TURN も STUN 同様，グローバルネットワークに設置された TURN サーバと Symmetric 型 NAPT 配下に存在する TURN クライアントから構成されたクライアントサーバ型プロトコルである．

TURN の動作は，認証処理と通信接続処理とリレー通信処理に分けられる．認証処理では，TURN クライアントが TURN Allocate Request により，TURN サーバに対して認証を行う．認証に成功すると，TURN Allocate Success Response により，TURN サーバは TURN クライアントのグローバル IP アドレスとポート番号を XOR-MAPPED-ADDRESS として付与し，TURN サーバが割り当てたリレーに用いるための IP アドレスとポート番号を XOR-RELAYED-ADDRESS として付与する．通信接続処理では，通信を開始する端末から，TURN サーバに対し，相手端末に対する接続を要求する．TURN クライアントは，Create Permission Request により，相手端末との通信開始を TURN サーバに依頼する．相手との通信が許可されると，Create Permission Response にて，相手のグローバル IP アドレスとポート番号が通知される．TURN サーバから Create Permission Response を受け取ると，XOR-PEER-ADDRESS に相手のグローバル IP アドレスとポート番号を含めて TURN サーバに Send Indication を送信する．リレー通信処理では，TURN サーバがクライアントから Send Indication を受信すると，相手先とのリレーに用いる XOR-RELAYED-ADDRESS を送信元として相互に UDP パケットを転送する．両端末が前述の手順を相互に行うことで，TURN サーバを介したリレー通信が実現される．よって，Symmetric 型 NAPT が存在する場合でも通信接続性を実現可能である．

ただし，Symmetric 型 NAPT が存在しない環境では TURN サーバによる通信の中継は不要である．TURN は通信の際に，常に TURN サーバを経由した通信となるため，このような場合には経路が冗長となってしまう課題を抱える．

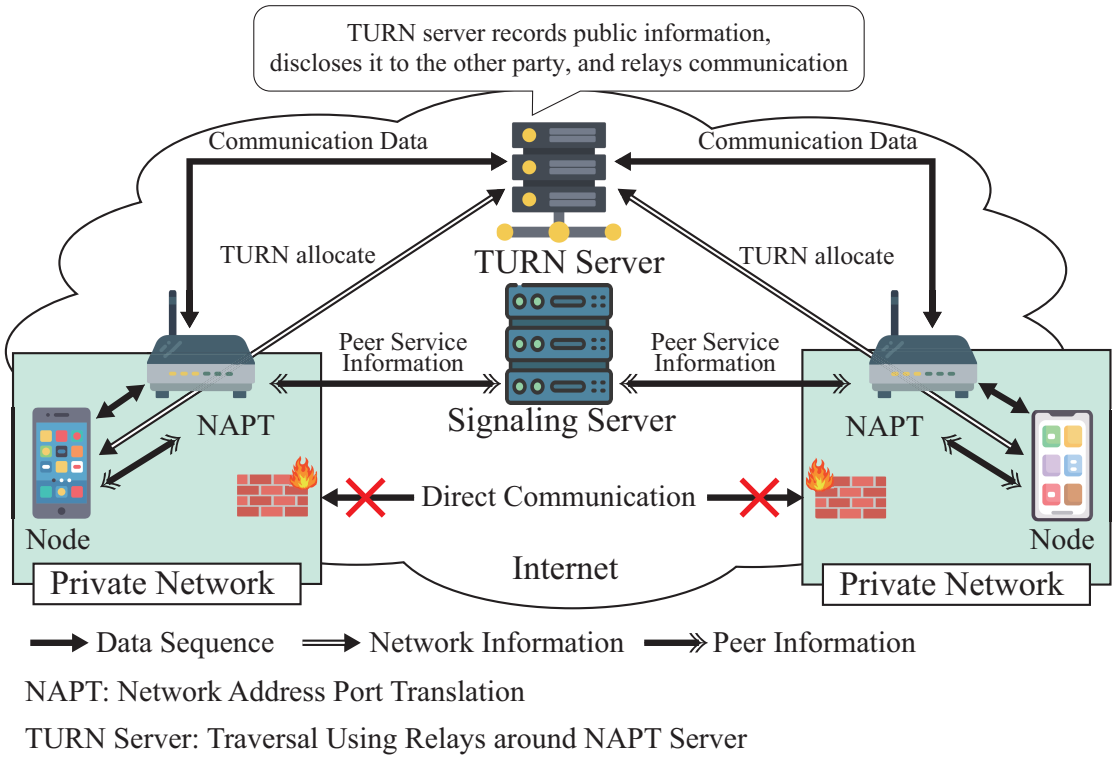


図 2.6 Overview of Traversal Using Relays around NAPT

2.3.4　Interactive Connectivity Establishment

TURN では，TURN サーバが両端末の間で中継処理を行うことで通信接続性を実現可能である．一方で，常に TURN サーバを経由した通信となってしまうため，Symmmetric 型 NAPT が存在しない環境では，経路が冗長となってしまう課題が存在した．Interactive Connectivity Establishment (ICE) では，STUN と TURN を統合し，端末が存在するネットワーク環境に応じて，TURN サーバを介した経路と相手端末への直接的な経路を状況に応じて使い分けることで，経路冗長問題を解消する．ICE の動作概要を図 2.7 に示す．ICE では，NAPT 配下に存在する ICE クライアントが，TURN サーバに TURN Allocate Request を送信し，Candidate と呼ばれる通信相手の候補と自身のネットワーク情報を収集する．Candidate は，IP アドレスとポート番号の組であり，NAPT ルータや TURN サーバ等の情報も収集する．ICE クライアントは収集した Candidate から Session Description Protocol (SDP) を生成し，通信相手と交換を行う．ICE クライアントが通信相手から SDP を受信すると，取得した全ての Candidate に対して STUN Binding

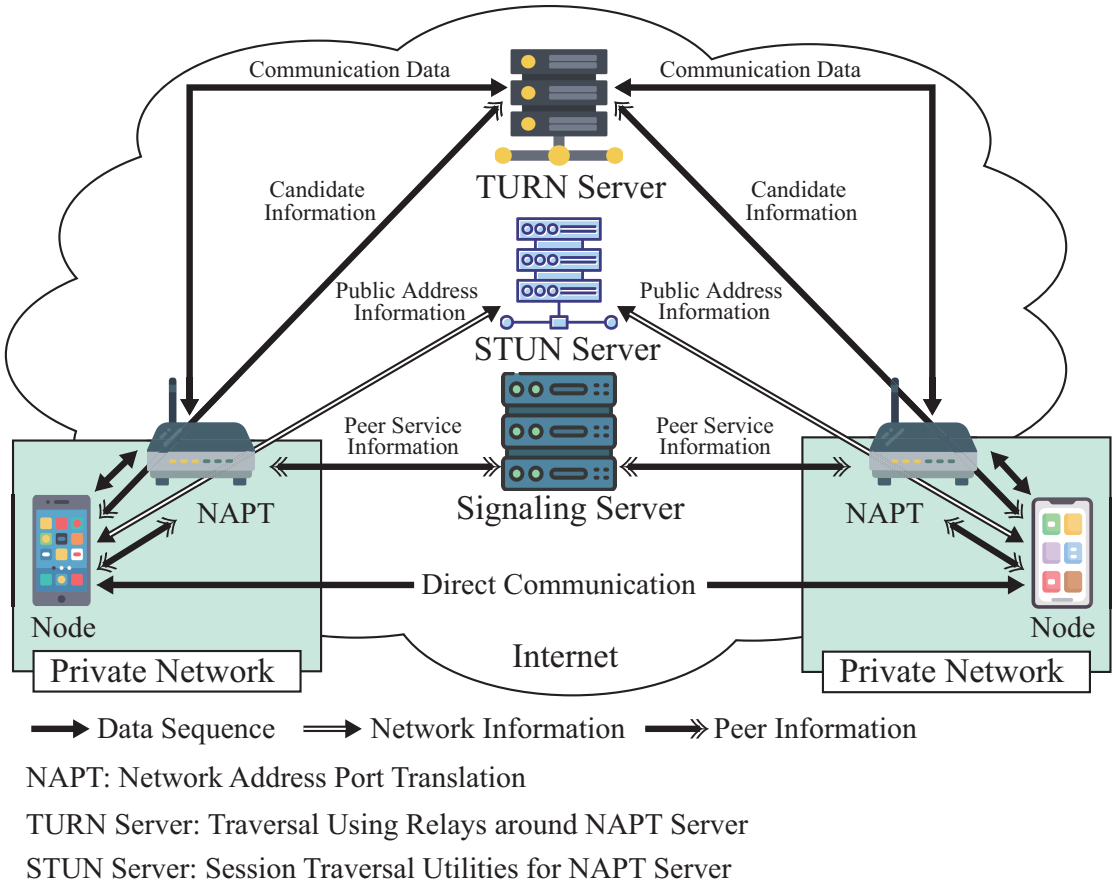


図 2.7 Overview of Interactive Connectivity Establishment

Update を送信して接続確認を行う．これにより，双方の端末は通信相手と通信を行う際に，NAPT の種類に応じて TURN サーバを経由するか，直接通信が可能かを判断可能となる，したがって，端末がどのような NAPT 配下に存在しようとも最適な NAPT 超えの実現が可能となる．

しかし，ICE は複数の NAT Traversal 技術を利用している点や，各技術の実行に伴う通信時間等のオーバーヘッドが増大するという課題が残ってしまう．加えて，ICE は後述する移動透過への検討をしていないため，モバイル端末での実用には適していない．

2.3.5　DualStack

現在のインターネットは，IPv4 アドレス枯渇問題に伴い，IPv6 アドレスの導入が進められている．しかし，IPv6 環境への完全な移行には時間を要するため，現在のインターネット環境は IPv4 と IPv6 が混在するデュアルスタック環境となっている．デュアルスタック環境での通信接続性を確保するためには，IPv4 と IPv6 の相互通信を実現する必要がある．DualStack は，単一の機器にて，IPv4/IPv6 の両プロトコルを使用可能とする技術である．DualStack の概要を図 2.8 に示す．DualStack を搭載した端末は，通信相手が使用している IP のバージョンに応じた使い分けが可能である．したがって，何れの IP バージョンを使用している機器と相互通信を実行可能となり，通信接続性が確保される．

一方で，DualStack を搭載する端末には，IPv4 と IPv6 の両方のプロトコルに対応させるため端末への特定の改良が必要となる．しかし，端末によっては IP バージョン対応のための特定の改良が困難な場合が存在するという問題が残されている．

2.3.6　Translator

DualStack は，一部の端末で DualStack の実装が困難であるという課題が存在する．端末に特定の改良を必要としない IPv4-IPv6 相互接続技術として，Translator が存在する．Translator は，中継サーバが IP ヘッダを変換することで，異なる IP バージョンでの相互通信を実現する技術である．Translator の概要を図 2.9 に示す．

具体的な例として，IPv4 を使用する端末と IPv6 を使用する端末が通信する場合を考えてみる．中継サーバは事前に端末の IPv4 ネットワークと IPv6 ネットワークの経路情報を保持している．まず，IPv4 端末は，中継サーバに向けて IP パケットを送信する．パケットを受信した中継サーバは，IPv4 ヘッダを IPv6 ヘッダへ変換し，IP パケットを IPv6 端末に転送する．逆の場合も同様に，IPv6 端末が中継サーバに向けてパケットを送信すると，中継サーバは IPv6 ヘッダを IPv4 ヘッダに変換し IPv4 端末へと転送する．この様にして，IPv4 端末と IPv6 端末は中継サーバを経由して相互に通信が可能となる．

中継サーバは中継の方式によって，Proxy 方式，Network Address Port Translation-Protocol Translation (NAPT-PT) 方式，Transport Relay Translator (TRT) 方式に分類され

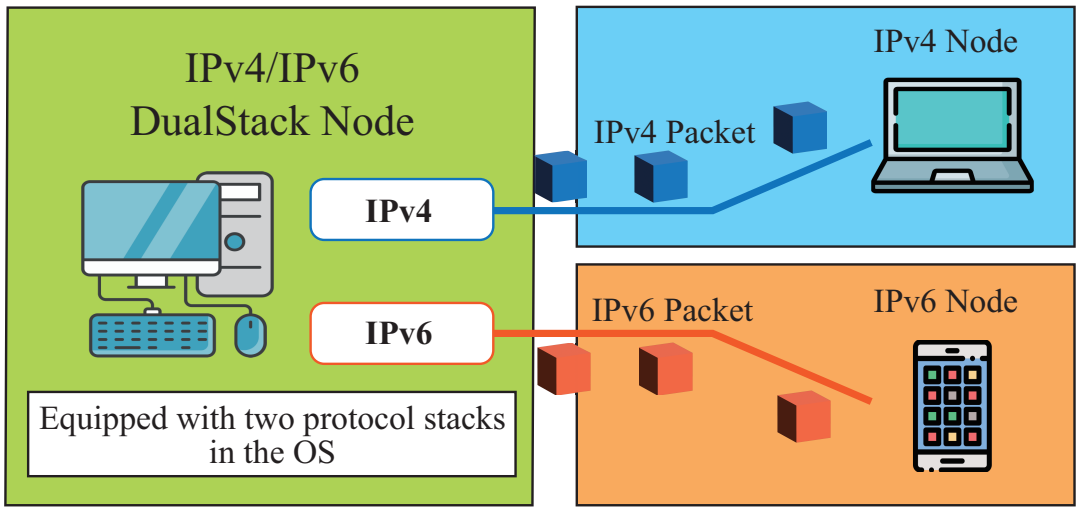


図 2.8 Overview of DualStack

る．Proxy 方式は，アプリケーション毎に送信元の代理となって通信を行う方式である．NAPT-PT 方式は，NAPT による IP アドレスの変換と同時に，IPv4-IPv6 のプロトコル変換を行う方式である．TRT 方式は，トランスポート層での TCP や UDP の通信を代理する方式である．Translator は，DualStack の実装が困難である端末に対して，IPv4-IPv6 の相互通信を実現する手段である．

一方で，Security Architecture for Internet Protocol (IPsec) や Session Initiation Protocol (SIP) 等の一部のプロトコルを使用する場合には，Translator の使用が困難である問題が存在する．この問題は IPsec や SIP 等の一部のプロトコルの動作が IP ヘッダに依存しているため，Translator による IP ヘッダ変換の影響を受けることにある．

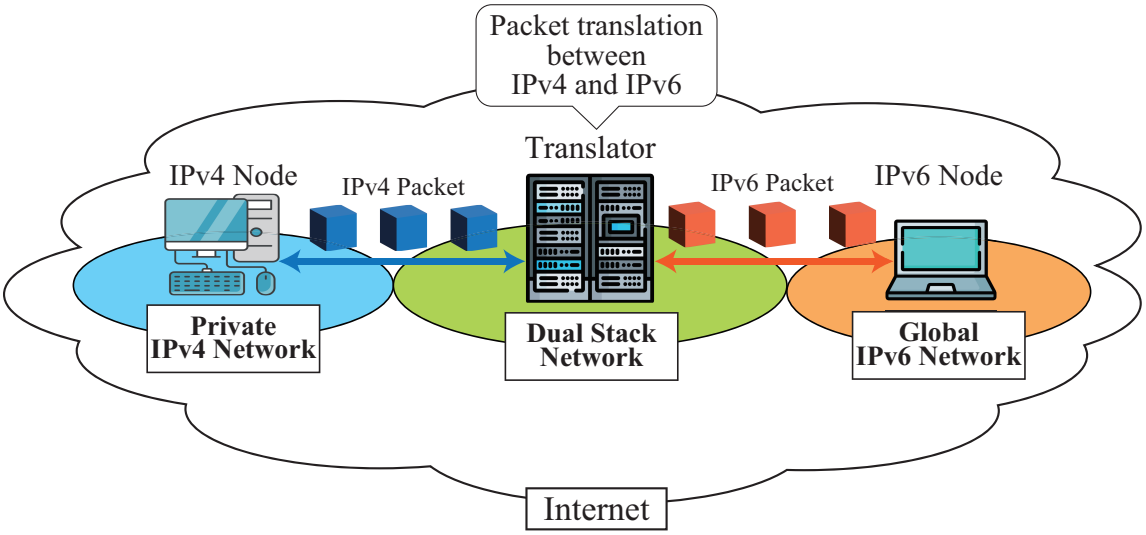


図 2.9 Overview of Translator

2.3.7　Tunneling

Tunneling は，カプセル化を利用して IPv4-IPv6 端末間の通信接続性を実現する技術である．カプセル化とは，通信パケット全体を別のプロトコルに埋め込む技術である．Tunneling は，両端末が同一の IP バージョンを使用しているものの，通信経路上に存在するルータなどのネットワーク中継機が，他方の IP バージョンのみ使用可能である場合において有効である．Tunneling の概要を図 2.10 に示す．

具体的な例として，両端末は IPv4 を用いて通信を行うが，通信経路上で IPv6 ネットワークを経由するという場合を考えてみる．送信元の IPv4 端末から送出された IPv4 パケットを，IPv6 を使用しているルータ等が受信すると，そのルータは IPv4 パケットに IPv6 ヘッダを付加しカプセル化を行う．カプセル化によって IPv6 パケットとなったパケットは，IPv6 ネットワークを経由し，宛先の IPv4 端末へ到達する際にデカプセル化が行われ IPv4 パケットとなる．こうして，IPv4 パケットが IPv6 ネットワークを経由する通信が可能となる．この様な，IPv6 でカプセル化された IPv4 パケットを IPv6 ネットワーク上に流通させる通信を IPv4 over IPv6 と呼ぶ．同様に IPv6 パケットを IPv4 パケットにカプセル化し，IPv4 ネットワーク上に流通させる通信を IPv6 over IPv4 と呼ぶ．また，両端末が異なる IP バージョンを使用している場合でも，受信側端末が使用する IP バージョンのパケットとなるようにカプセル化/デカプセル化処理

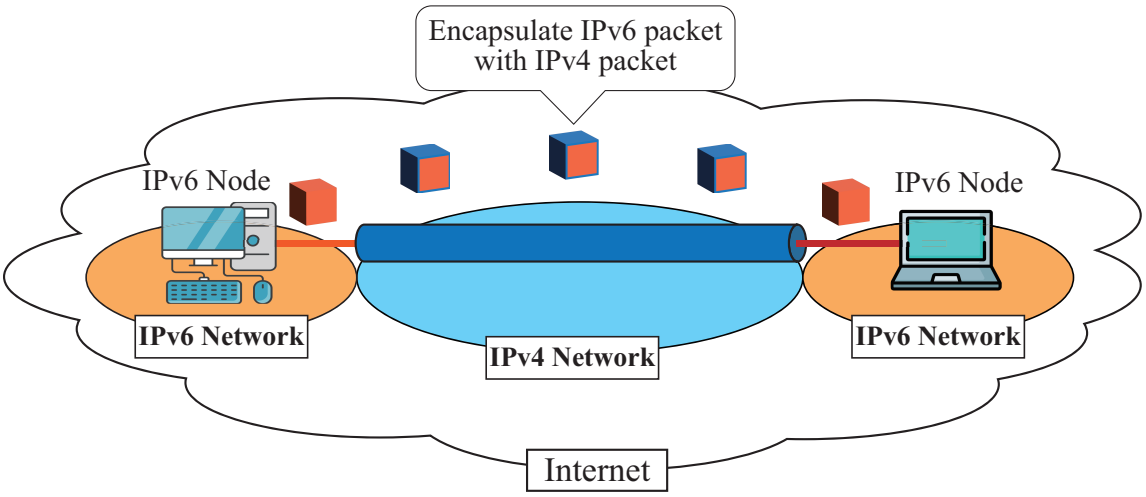


図 2.10 Overview of Tunneling

を実施することで通信が可能となる．

しかし，Tunneling は，NAT/NAPT 超え問題を考慮していない．それ故，IPv6 ネットワークに存在する端末から IPv4 ネットワークに存在する端末への接続は極めて困難である．

2.4　移動透過技術

スマートフォンに代表される無線通信端末には，複数のネットワークインターフェイスが搭載されており，通信を行う際にはこれらのネットワークインターフェイスを切り替えて通信が可能である．スマートフォンなどのモバイル端末では，第 5 世代移動通信システム (5G) の利用に代表されるセルラ通信や Wi-Fi 等の複数種類のネットワークを切り替えて通信を行っていることも多い．これはセルラ回線のトラフィック分散を実現するための手法である．セルラ通信は，Wi-Fi に比べて通信可能範囲が広く，建物や山等の障害物による通信妨害に強い．そのため，セルラ通信は屋外で通信を行う場合に適している．しかし，通信量が増加すると帯域制限が適用されてしまうというデメリットが存在する．一方で，Wi-Fi によるネットワーク通信は障害物に弱く，通信範囲が狭いデメリットを持ちつつも，通信量に応じた帯域制限が存在しないというメリットがある．この通信方法は屋内で通信を行う場合に適している．それ故，

屋外ではセルラ通信，屋内では Wi-Fi を使用した通信を行うことでセルラ回線の逼迫を抑制可能となる．このように適宜，適切な回線を切り替えて通信をすることによって，回線のトラフィックを分散させることを，トラフィックオフローディングと呼ぶ．トラフィックオフローディングの実現には，無線通信端末による接続ネットワークの切り替えが想定される．IP を用いた通信中にネットワークの切り替えが発生した場合，端末が使用する IP アドレスが変更される．一方で, IP アドレスはネットワーク上の位置情報と端末の通信識別子を兼ねているため，ネットワークの切り替わりによって IP アドレスが変更されると．端末の識別が不可能となり，通信が切断されてしまう．また，トラフィックオフローディングによるネットワークインターフェイスの切り替えの他にも，有線インターフェイスから無線インターフェイスへの切り替えやプライベートネットワークへの移動など，IP アドレスが変化する状況においては同様に通信切断が発生する．このような，端末のネットワーク移動に伴う IP アドレスの変更が原因で通信が切断される問題を，移動透過問題と呼ぶ．ネットワーク移動による通信切断を発生させない性質を移動透過性と呼び，移動透過を実現する技術を移動透過技術と呼ぶ．代表的な移動透過技術として，Mobile IP (MIP) が存在する．MIP は，IP のバージョンによって，Mobile IPv4 (MIPv4) と Mobile IPv6 (MIPv6) が存在する．しかし，MIPv4 と MIPv6 はそれぞれ独立した規格であるため互換性がない．さらに，MIP を応用した移動透過技術として，Dual Stack Mobile IPv6 (DSMIPv6) や Proxy Mobile IPv6 (PMIPv6) も検討されている．また，SIP を応用した SIP Mobility や Quick UDP Internet Connections (QUIC) と呼ばれる移動透過技術も存在する．以下のセクションにて，これら移動透過技術の詳細を説明する．

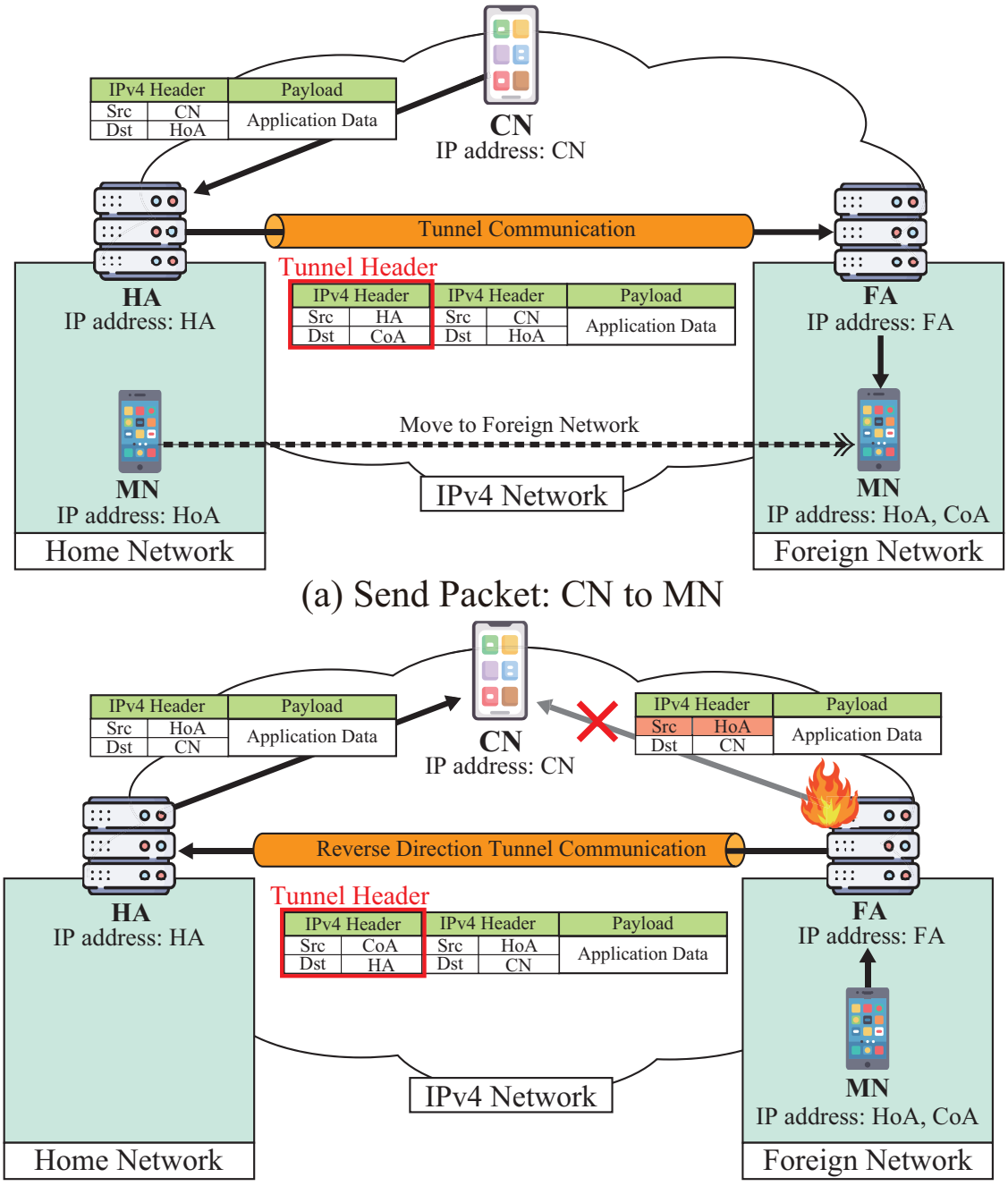
2.4.1　Mobile IPv4

Mobile IPv4(MIPv4) は，端末に対して不変な IP アドレスを割り当て，位置情報と通信識別子を分離した上で，ルータにてネットワーク移動に対応した転送処理を実施する移動透過技術である．MIPv4 の概要を図 2.11 に示す.

MIPv4 では，ネットワーク上に Home Agent (HA)，Foreign Agent (FA) と呼ばれるルータを設置し，それらを介して移動ノードである Mobile Node(MN) とその相手ノードである Correspondent Node (CN) が通信を行う．IP アドレスは位置情報と通信識別子の

両役割を担うため，ネットワークの移動に伴う IP アドレスの変更によって，通信切断を引き起こす．MIP では，IP が持つ位置情報と通信識別子をそれぞれ，移動に伴って変化する Care-of Address (CoA) と不変な Home Address (HoA) に分離する．これにより，移動に伴う IP アドレスの変化を隠蔽し，移動透過性を実現する．MIPv4 において，MN はホームネットワーク内に設置される HA から HoA を取得し，CN に通知する．MN はホームネットワークに存在するとき，HoA を送信元 IP アドレスとして通信を行い，CN は HoA を宛先 IP アドレスとして通信を行う．MN が訪問先ネットワークへと移動すると，FA から CoA を取得する．次に，移動前のホームネットワークにある HA に対して，HoA と訪問先ネットワークで取得した，CoA との対応関係を通知するため，Binding Update Message を送信する．HA は，Binding Update Message を受信すると，Binding Cache テーブルに MN の訪問先ネットワーク情報を登録する．これにより，HA は，MN の訪問先ネットワーク情報を把握することが可能となる．その後，CN から MN へのパケットは，HA へ送信された後，送信元 IP アドレスを HA，宛先 IP アドレスを CoA とするパケットでカプセル化され，トンネル通信によって，訪問先ネットワークに存在する MN へ転送される．MN はパケットを受信すると，トンネルヘッダをデカプセル化して，CN が MN の HoA 宛に送信したパケットを取得可能である．これにより，CN は移動前と同様に MN のアドレスを HoA として認識可能なため，通信フローを識別して通信を継続することが可能となる．

しかし，MN が保持している HoA は本来，訪問先ネットワークで有効なアドレスではない．そのため，MN が訪問先ネットワークから CN にパケットを直接送信する場合，FA のイングレスフィルタリングによって破棄される可能性がある．イングレスフィルタリングとは，内部ネットワークからルータに入ってくるパケットに対するフィルタリングであり，不正なパケットをブロックする目的で設定される．また，MN が CoA を送信元 IP アドレスとしてパケットを CN に送信しても，CN が認識している MN のアドレスは HoA であるため，通信を行うことは不可能である．これらの問題を解決するため，逆方向トンネルを用いて，一度 HA へパケットを送信し，CN へ中継転送する方法が考えられる．しかしその方法では，MIPv4 利用時に常に HA を介した通信を行うこととなるため，経路冗長化問題となる．



MN: Mobile Node　CN: Correspondent Node　HA: Home Agent　FA: Foreign Agent
HoA: Home Address　CoA: Care-of Address

図 2.11 Overview of Mobile IPv4

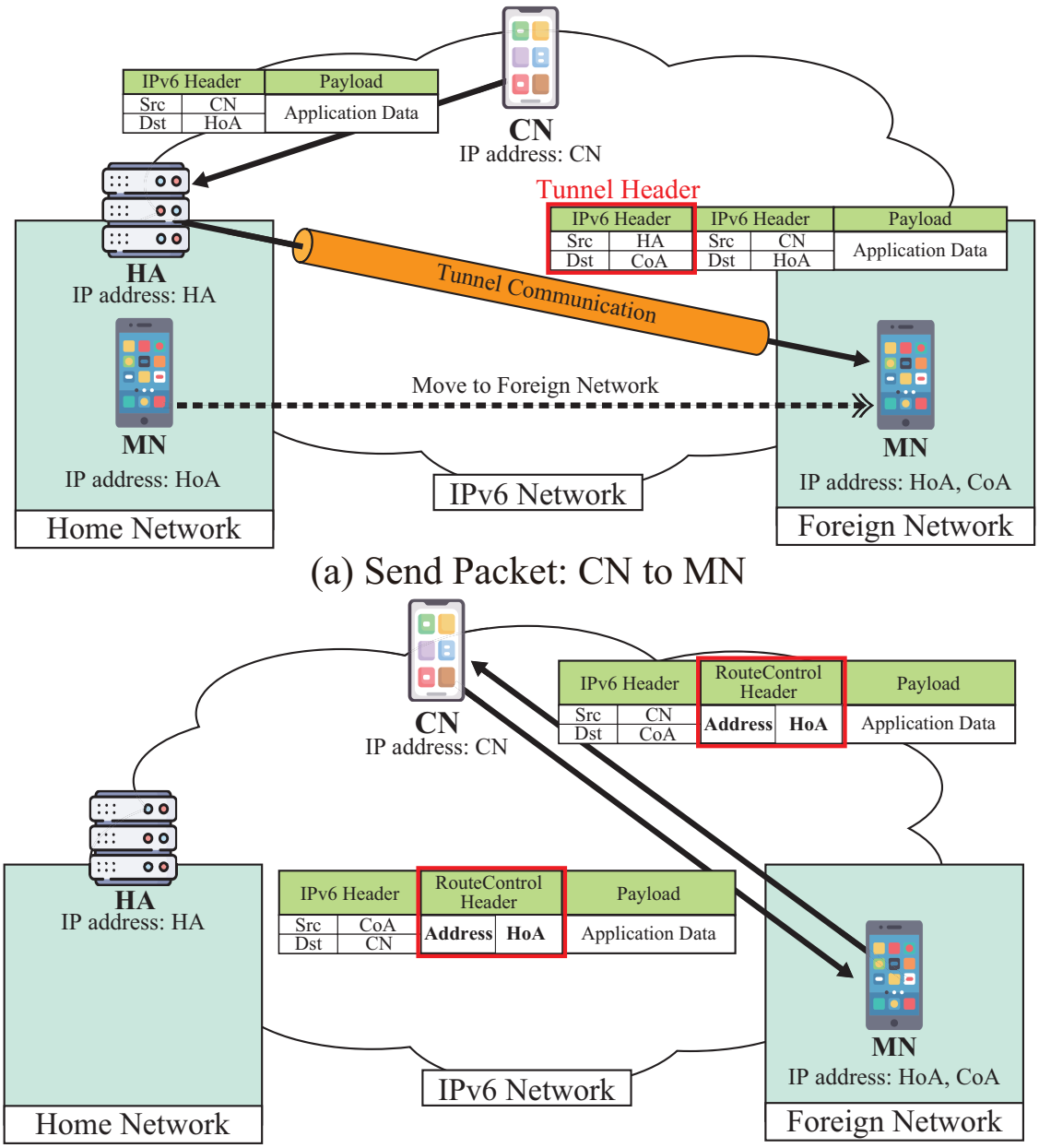
2.4.2　Mobile IPv6

MIPv4 では，MN と CN が通信を行う際に，常に HA を介する必要があるため，経路冗長問題が存在した．MIPv6 は，MIPv4 を IPv6 ネットワークに対応させた移動透過技術であり，経路最適化による経路冗長化問題を排除することが可能である．MIPv6 の概要を図 2.12 に示す.

MIPv6 は，MIPv4 をベースとしているため，MN が保持する HoA を用いた通信方法や，HA を経由したトンネル通信はそのまま引き継がれている．一方で，IPv6 の場合，すべての IP アドレスがグローバル IP アドレスとなるため，NAPT の概念は存在しない．故に，MIPv4 で CoA を割り当てていた FA は存在しない．その代わり，CoA は Stateless Autoconfiguration (SLLAC) 等の IPv6 アドレス自動生成機能，または Dynamic Host Configuration Protocol for IPv6 (DHCPv6) によって生成される．MN はネットワークを移動後に，HA に対して Binding Update Message を送信すると，HA との間に双方向トンネルが構築され，HA を経由した送受信を行う．また，イングレスフィルタリングによってパケットが破棄されてしまうことはないため，MN から CN へのパケットは直接送信することが可能である．加えて，Binding Update Message を HA の他，CN に対しても送信することで，CN 自身が MN の HoA と CoA の対応関係を把握可能となる．これにより，経路最適化が可能なため，MN と CN 間の通信は IPv6 の拡張ヘッダとして定義されている経路制御ヘッダを用いて，直接通信が可能となる．

一方で，CN は宛先 IP アドレスとして MN の CoA を指定するため，Binding Cache テーブルに不正な CoA を登録された場合に，セッションハイジャックが発生する恐れがある．MIPv6 ではセッションハイジャックを防ぐために，経路最適化を行う際，Return Routability と呼ばれる手続きを行う．Return Routability のシーケンス図を図 2.13 に示す.

Return Routability を行う際，MN は HoA を送信元とした Home Test Init パケットを HA 経由で CN へ送信する．次に，CoA を送信元とした Care-of Test Init パケットを直接 CN へ送信する．CN は Home Test Init パケット及び Care-of Test Init パケットを受け取ると Home Test パケットを HoA 宛に HA 経由で送信し，Care-of Test パケットを CoA 宛に直接送信する．MN は，Home Test パケット，Care-of Test パケットに含まれる Home Keygen Token と Care-of Keygen Token を元に認証鍵を生成し，Binding Update をハッ



(a) Send Packet: CN to MN

(b) Send Packet: MN to CN (Route optimization)

MN: Mobile Node　CN: Correspondent Node　HA: Home Agent
HoA: Home Address　CoA: Care-of Address

図 2.12 Overview of Mobile IPv6

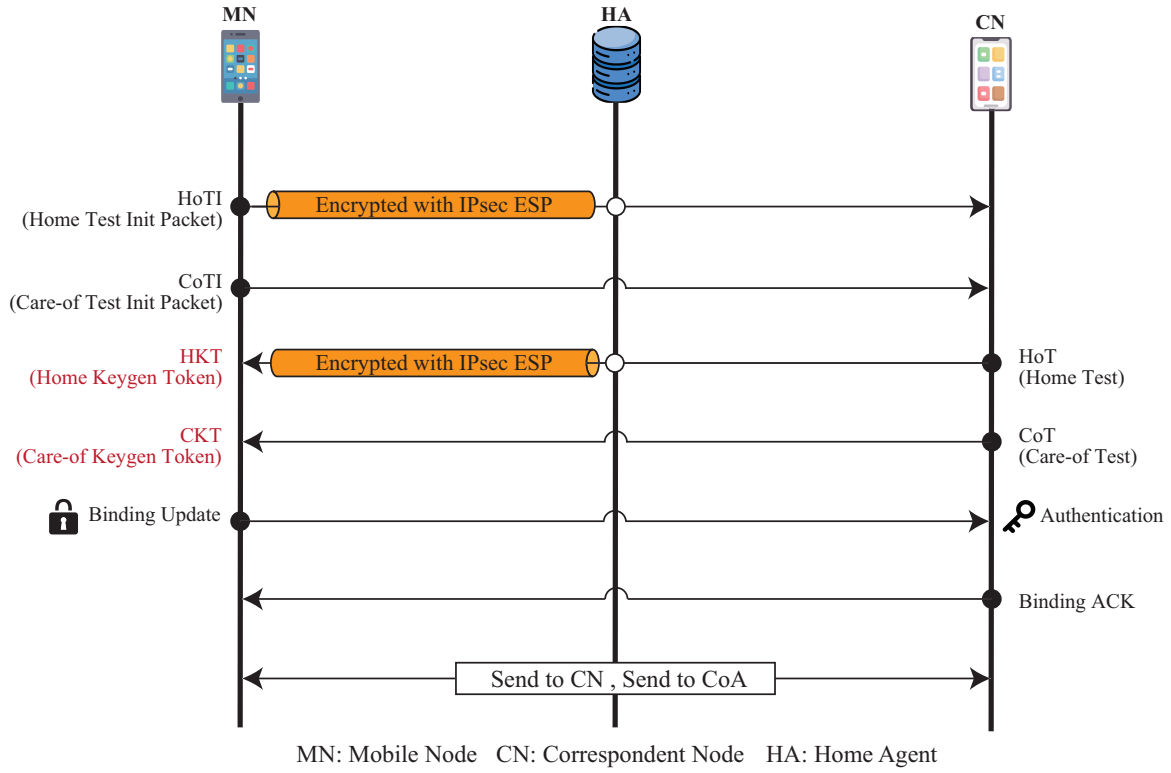


図 2.13 Overview of Return Routability

シュ化して CN へ送信する．CN は受信した認証データを検証し，完全性が確保されている場合に ACK を MN へ応答する．以後，MN と CN 間の通信は宛先 IP アドレス CN，CoA をそれぞれ指定してセキュアな送受信が可能となる．

MIPv6 は冗長な経路を排除して経路最適化が可能であるが，端末が IPv6 に対応していることを前提としている．そのため，元来使用されている IPv4 には対応しておらず，MIPv4 と MIPv6 では双方独立した技術となっている．

2.4.3　Dual Stack Mobile IPv6

MIPv4 は，移動透過性を実現可能であるが，経路が冗長になるという問題が存在した．また，MIPv6 では，MIPv4 での冗長な経路を排除可能であるが，端末が IPv6 に対

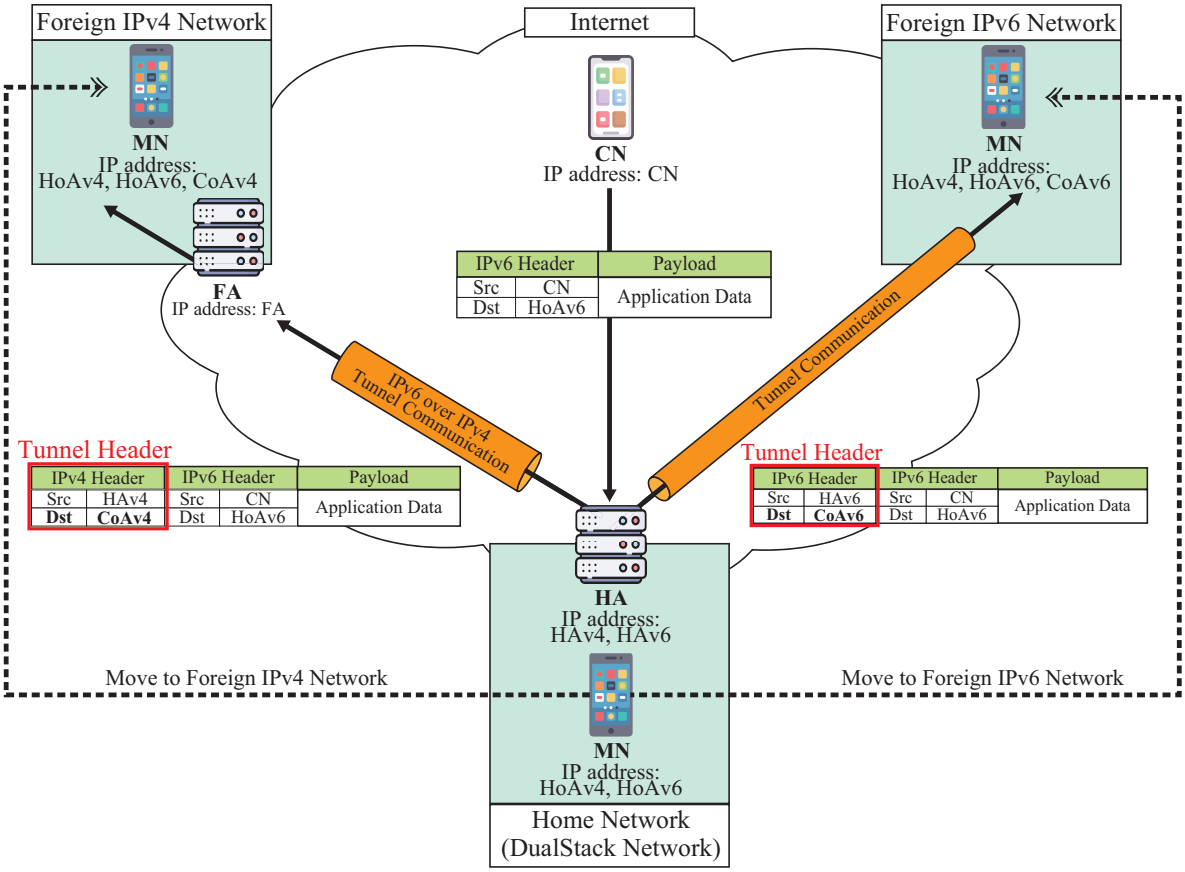
応していることが前提であった．そこで，IPv4 と IPv6 が混在するデュアルスタック環境向けに Dual Stack Mobile IPv6 (DSMIPv6) が提案されている．この移動透過技術は，MIPv4 と同様の手法で IPv6 に対応するように拡張した技術である．DSMIPv6 の概要を図 2.14 に示す.

前提条件として，MN と CN はデュアルスタックネットワークに存在しなければならない．DSMIPv6 は，MIPv4 を拡張しているため，MIPv4 同様に HA を介した通信となる．MN が CN と異なる IP バージョンを使用しているネットワークに移動した場合，Binding Update Message に移動前に保持していた HoA，及び訪問先ネットワークで割り当てられた CoA を含めて HA に送信する．MN は，ホームネットワークに存在するとき，HA によって HoAv4，HoAv6 の 2 種類の HoA を取得する．MN が IPv4 ネットワークに移動した場合，CN からのパケットは IPv4 ヘッダでカプセル化され，HA と MN の間に構築された IPv6 over IPv4 トンネルを通して転送する．また，IPv6 ネットワークに移動した場合，CN からのパケットは IPv6 ヘッダでカプセル化され，HA と MN の間に構築された IPv6 over IPv4 トンネルを通して転送する．この時，CN が MN の HoAv4 に対して通信を開始した後に，MN が IPv6 ネットワークに移動した際は，IPv4 パケットを IPv6 パケットでカプセル化し，IPv4 over IPv6 トンネルを通して転送する．また，MN が IPv4 ネットワークに移動した際は，HA と MN の間に構築された IPv4 トンネルを通して転送する．したがって，CN は HA と MN の間に構築されたトンネルを用いて通信を行うことにより，異なる IP バージョンを使用している環境においても，移動透過性を実現することが可能となる．

しかし，DSMIPv6 では IP バージョンが異なる場合，パケットのカプセル化処理が必要となる．そのため，通信の際は HA を経由する必要があり，経路冗長化問題を引き継ぐこととなる．

2.4.4　Proxy Mobile IPv6

前述してきた MIP は，モバイル端末等の MN が訪問先ネットワークで取得した CoA を HA に通知することで，移動透過性を実現する技術であった．したがって，MN が移動支援機能を持つためには，端末自体に MIP の機能を実装する必要がある．具体的には，特定のソフトウェアをインストールしたり，Operating System (OS) のカーネル

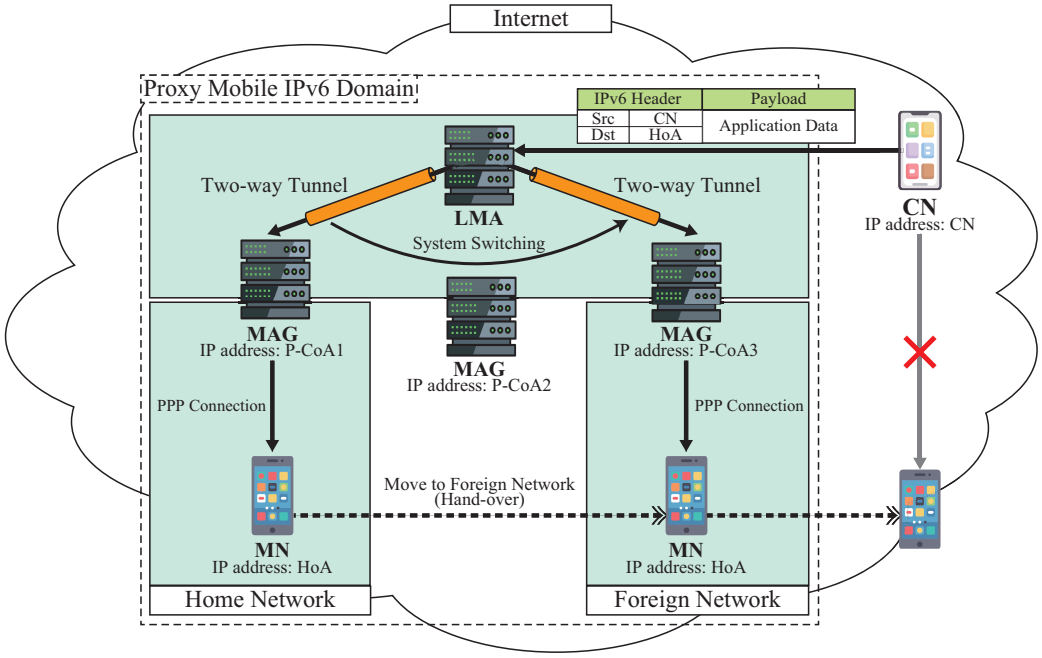


MN: Mobile Node　CN: Correspondent Node　HA: Home Agent　FA: Foreign Agent
HoAv4: Home IPv4 Address　CoAv4: Care-of IPv4Address
HoAv6: Home IPv6 Address　CoAv6: Care-of IPv6Address

図 2.14 Overview of Dual Stack Mobile IPv6

にプロトコルを実装したりする必要がある．そこで，MN に特定の改造を加えることを排除するため，ネットワーク側に移動支援機能を実装する方式が検討された．端末に手を加えることなく移動透過性を実現する代表的な技術として Proxy Mobile IPv6 (PMIPv6) が存在する．PMIPv6 の概要を図 2.15 に示す．

PMIPv6 では，ネットワークに一つの Local Nobility Anchor (LMA) と，複数の Mobilr Access Gateway (MAG) と呼ばれる機器を設置する．LMA，MAG は，いずれも Dual Stack ノードとして動作する．LMA は，MIPv6 において HA の役割を担う装置であり，MN の移動を管理したり，MN 宛のパケットを代理で受信して転送したりする機能を持つ．MAG は，MIPv6 における MN 側に実装していた移動支援機能を持ち，MN に代



MN: Mobile Node　CN: Correspondent Node　HoA: Home Address　P-CoA: Proxy Care-of Address
LMA: Local Mobility Anchor　MAG: Mobility Access Gateway

図 2.15 Overview of Proxy Mobile IPv6

わって LMA へ訪問先ネットワーク情報を通知する．PMIPv6 が適用されるネットワークを PMIPv6 ドメインと呼び，ドメイン内の移動であれば移動透過性を実現することが可能である．なお，MAG と MN 間は，Point-to-Point Protocol (PPP) で接続される．

MAG は，MN の接続を検知すると，LMA に Proxy bindingsUpdate Message と呼ばれる MN を識別するための情報を送信する．LMA は，MAG から取得した Proxy-CoA と呼ばれる MAG の IP アドレスと対応付けて，Binding Cache テーブルに登録する．その後，LMA は，HoA の生成に必要な IPv6 ネットワークプレフィックスを記載した Proxy Binding ACK を応答し，MAG との間に双方向トンネルを構築する．MAG は，LAM によって付与された IPv6 ネットワークプレフィックスを含めて Router Advertise (RA) パケットを生成し，MN へ通知する．MN は，MAG から受信したプレフィックスを元に，IPv6 アドレス自動生成機能を使用して，HoA を生成する．CN が，MN の HoA 宛にパケットを送信すると，LMA が代理で受信する．次に，CN からのパケットは，LMA と MAG との間に構築された双方向トンネルを用いて MAG へ転送され，MAG から

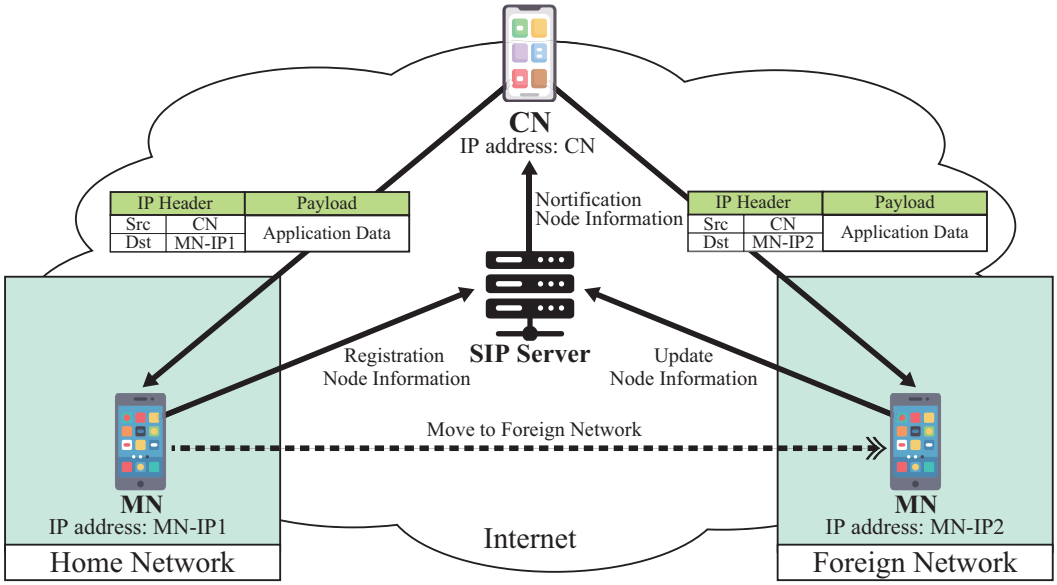


図 2.16 Overview of Session Initiation Protocol Mobility

MN へ送信される．また，MN が PMIPv6 ドメイン内を移動して MAG に接続するたびに，MAG は LMA への通知，及び双方向トンネルの構築を行う．したがって，MN は PMIPv6 ドメイン内であれば，MAG からプレフィックスを受け取ることが可能なため，同じ HoA を使用し続けることが可能である．LMA と MAG 間はトンネル通信を行うため，IPv4，IPv6 のいずれでも許容される．また，MN に割り当てる HoA も IPv4，IPv6 または，両方でも良いため，柔軟な運用が可能である．

しかし，端末は PMIPv6 ドメインを超えて移動透過支援を受けることは仕様上困難であるため，移動透過の行動範囲が限定されるという課題が存在する．

2.4.5　Session Initiation Protocol Mobility

Session Initiation Protocol Mobility (SIP Mobility) は，SIP と呼ばれるプロトコルを応用した移動透過技術である．SIP は，IP ネットワークを通じて，音声やビデオ通話等のリアルタイムコミュニケーションを可能にする通信プロトコルである．主に，Voice over INternet Protocol (VoIP) などの IP 電話で使用される．SIP Mobility の概要を図 2.16 に示す．

SIP を用いる端末は，Uniform Resource Identifer (URI) と呼ばれる一意の識別子を持つ．端末は，相手端末への通信を開始する前に SIP サーバに接続し，URI や IP アドレス，及びポート番号の登録処理を行う．端末は，通信する相手端末の URI を SIP サーバに問い合わせるか，または通知を受け取る．端末が通信をする際には，相手端末を URI で指定し，SIP サーバへ通信接続要求を送信する．SIP サーバは相手端末へ通信接続要求を転送し，相手端末と通信を行う．したがって，MIP のように HA を介した通信を行う必要がないため，経路冗長化問題は発生しない．

また，端末がネットワークを移動した場合，端末は IP アドレスの変化を SIP サーバに対して通知し，更新処理を行う．その後，SIP サーバから通信相手端末に向けて通信を行い，端末は移動した端末の IP アドレスを得る．そのため，直接，相手端末とのセッションを再構築することが可能である．さらに，SIP サーバはセッション情報を管理しているため，IP アドレスの変更に伴いセッション情報を更新する．したがって，UDP 等のコネクションレスなセッションであれば，端末が通信中に異なるネットワーク間を移動した場合にも，既存のセッションを維持し続けることが可能である．その結果，コネクションレス通信では，通信接続性と移動透過性を同時に実現することが可能である．

しかし，TCP 等のコネクションを確立した通信を行う際は，端末が移動した場合，IP アドレスが変化するため，セッションを継続することが困難である．また，SIP Mobility は IPv6 ネットワークでの利用が十分に検討されていない．

2.4.6　Quick UDP Internet Connections

Quick UDP Internet Connections (QUIC) とは，UDP を利用して高速化を図りつつ，TCP のような通信の信頼性を提供するトランスポート層のプロトコルである．従来のトランスポートプロトコルは，高信頼性でありながらオーバーヘッドが大きい TCP と，通信遅延が小さいものの信頼性について保証しない UDP が用途によって使い分けられていた．このような背景に基づき，TCP の高信頼性と UDP の高速通信の両立が望まれたことから QUIC が提案された．QUIC の仕様は，RFC9000, RFC9001, RFC9002 の 3 つに分かれている．順に，QUIC のトランスポートプロトコルとしての機能をを定めたもの，Transport Layer Security (TLS) を使用した暗号化の仕組みを定めたもの，パ

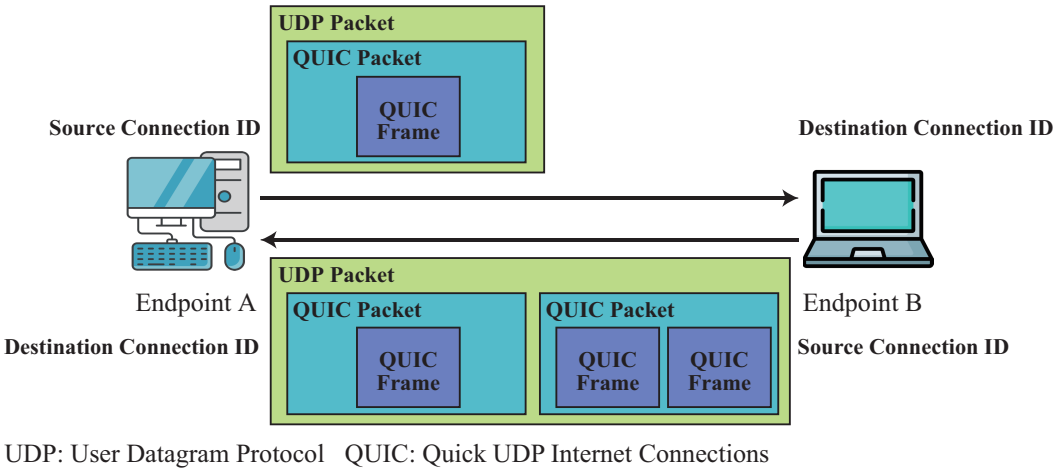


図 2.17 Overview of Quick UDP Internet Connections

ケットロスの検出や，帯域の逼迫を防ぐための通信を行う輻輳制御について定めたものである．QUIC には，トランスポート層における Head-of-Line-Blocking(HoL Blocking) の回避や再送制御の効率化，制御情報の暗号化といった特徴が存在する．QUIC は，UDP を基盤としており，TCP よりも高速なデータ通信を提供する．加えて，内部でトランスポート層における暗号化プロトコルである TLS を使用した信頼性の高いデータ通信を行う．

TCP や UDP のコネクションは，5 タプルと呼ばれる情報で識別される．5 タプルとは，送信元 IP アドレス，宛先 IP アドレス，IP ヘッダのプロトコル，送信元ポート番号，宛先ポート番号の 5 つの情報のことである．そのため，5 タプルの内 1 つでも値が変更されると，TCP や UDP のコネクションを継続することは不可能である．このことに起因して，端末がネットワークを移動すると，IP アドレスの変更が発生し，コネクションが断絶される．一方，QUIC はコネクション ID と呼ばれる識別子によって通信が識別される．故に，QUIC における通信の識別は IP アドレスに依存しないため，ネットワーク移動に伴う IP アドレスの変更が発生した場合であっても，通信を継続することが可能である．QUIC を利用した通信の概要を図 2.17 に示す．

QUIC を使用する双方のエンドポイントは，それぞれコネクション ID を生成する．自身が生成するものを送信元コネクション ID (Source Connection ID)，相手端末側が生成するものを宛先コネクション ID (Destination Connection ID) と呼ぶ．また，図 2.17 の

ように QUIC を使用する場合，一つの UDP パケットに一つ以上の QUIC パケットが内包され，一つの QUIC パケットには一つ以上の QUIC フレームが含まれる．QUIC パケットは，コネクションの確立に使用されるロングヘッダパケットと，コネクション確立後のデータ送受信に使用されるショートヘッダパケットに分けられる．ロングヘッダパケットのヘッダには，送信元コネクション ID と宛先コネクション ID の両方が格納されている．一方，ショートヘッダパケットのヘッダには宛先コネクション ID のみが格納されている．QUIC コネクションの確立後は，送信元コネクション ID と宛先コネクション ID の対応関係が既知となり，通信に必要な情報が一意に定まることからこのように規定されている．これにより，ヘッダを軽量化することで，通信のオーバーヘッドを削減することが可能となる．QUIC パケットには，一つ以上の QUIC フレームが格納される．QUIC フレームは目的に応じた様々なタイプに分類され，例えば，アプリケーションデータを送信するためには，STREAM を使用する．また，QUIC は TCP と TLS を使う場合より、コネクションの確立が迅速である．TCP と TLS を併用する場合は，TCP のハンドシェイクを行った後に TLS ハンドシェイクを行う，一方で，QUIC は，QUIC のコネクション確立と TLS のハンドシェイクを同時に行うため，アプリケーションデータを送信するまでの準備時間が短くなる．さらに，QUIC は，QUIC コネクションを切断することなく，端末から自発的に通信経路を切り替える，コネクションマイグレーションという機能を持っている．この機能はスマートフォンにおいて，セルラ回線から Wi-Fi 回線に切り替える際などに利用可能である．よって，モバイル端末において，通信中のネットワーク切り替えが発生した場合においてもコネクションを継続可能であるため，移動透過性を実現する．

しかし，QUIC では，NAPT 配下の端末に対してグローバルネットワークから通信を開始することが極めて困難となる NAPT 超え問題を抱える．そのため，両エンドポイントが NAPT 配下である場合，NAT Traversal 技術の併用が必要である．

2.5　暗号化技術

インターネットを活用した EC サイトでの電子商取引の普及や，デジタル技術の活用を通して生活やビジネスを変革する Digital Transformation (DX) 化に伴い，個人情報や重要なデータの送受信が増加し，情報漏洩や不正アクセスのリスクが増大して

いる．このような状況で，暗号化技術はデータの保護とプライバシの確保において重要な役割を担っている．暗号化とは，データをある規則に則って意味のある形から不可解な形式へと変換する処理である．これによって，データを不正なアクセスから保護し，通信内容を盗み見たり，改ざんしたりすることを防止する．また，暗号化技術を利用してデータを暗号化することで，改ざんや破損の検知が容易となるため，信頼性の高い通信を実現可能である．以下セクションで，暗号化技術に関連した技術の詳細を説明する．

2.5.1　Transport Layer Security

TLS とは，インターネット上でやり取りされるデータの盗聴，改ざん，なりすましを防止し，セキュアな通信を実現するための暗号化プロトコルである．TLS は SSL の次世代規格であり，SSL が使用される中で重大な脆弱性が発見されたため，設計を見直した TLS が登場した．現在，SSL というと TLS のことである場合が多いが，はじめに普及した SSL という名称が広く認知されていることから，SSL や SSL/TLS と表現されることも珍しくない．TLS は，公開鍵認証方式を用いたサーバの認証や共通鍵暗号方式を用いた暗号化通信，ハッシュを用いた改ざん検知を規定している．公開鍵認証とは，公開鍵と秘密鍵からなるキーペアを用いて，ノードの真正性を確認する方式である．TLS を使用する場合，サーバは秘密鍵を用いて電子署名を作成し，クライアントへ送信する．この際使用される電子署名は，サーバ証明書とも呼ばれる．電子署名を受信したクライアントは，サーバの公開鍵からその署名の妥当性を確認する．署名が正しいと確認された際には，その署名を作成したサーバは正しい秘密鍵を所有していることから，正規のサーバであると判断される．TLS はオプションとして，サーバとクライアントの役割を入れ替え，クライアントの真正性を確認するクライアント認証と呼ばれる機能を有している．また，TLS は認証プロセスにおける通信を経て共通鍵を交換し，以降の通信を暗号化する．さらに，暗号化通信に加え，一方向ハッシュ関数によって生成されたハッシュ値を通信データに付加することで，改ざん検知を行う．

2.5.2　OpenSSL

OpenSSL は，SSL/TLS の実装を提供するオープンソースの暗号化ライブラリである．SSL/TLS では，クライアントとサーバ間でデータを暗号化して送受信する際に，暗号化と復号に同じ鍵を使用する共通鍵暗号方式を使用する．また，作成した共通鍵をクライアントとサーバ間で共有する際には，暗号化と復号に異なる鍵を使用する公開鍵暗号方式を使用する．公開鍵暗号方式では，サーバが暗号化用の鍵と復号用の鍵の組を作成し，復号用の鍵は秘密鍵としてサーバが保持し，暗号化用の鍵は公開鍵としてクライアントに付与する．クライアントは公開鍵によって暗号化したデータを送信することで，通信内容の復号は秘密鍵を所持する正規のサーバのみが可能となる．このように，共通鍵暗号方式と公開鍵暗号方式を併用する方式をハイブリッド暗号方式と呼ぶ．

OpenSSL の代表的な使用例として，Hyper Text Transfer Protocol Secure (HTTPS) が挙げられる．HTTPS は，Web ブラウザの閲覧時等に利用される HTTP を OpenSSL の機構を用いて暗号化したネットワークプロトコルである．また，OpenSSL は暗号化を実装するための crypto ライブラリが含まれる．OpenSSL は，ライブラリに含まれる関数をコマンドラインから実行可能なコマンドラインツールが含まれる．コマンドラインツールには，Rivest Shamir Adleman (RSA) 等の秘密鍵の作成，SSL サーバ証明書の Certificate Signing Request (CSR) の作成，SSL/TLS クライアントとサーバのテスト等，様々な機能が実装されている．

2.5.3　Advanced Encryption Standard

Advanced Encryption Standard (AES) は，米国が 2001 年に標準暗号として定めた共通鍵暗号アルゴリズムである．共通鍵暗号アルゴリズムでは，データの暗号化と復号に同一の鍵を使用する．AES は，現在最も一般的に使用されているブロック暗号の一つである．ブロック暗号とは，共通鍵暗号の一種で，固定サイズのブロックにデータを分割し，鍵を使用してそれぞれのブロックを暗号化または復号する暗号化方式である．AES は，1997 年に米国立標準技術研究所によって暗号化標準の選定プロセスが開始され，2001 年に AES として採用された．AES では，主に，128 ビット，192 ビット，256 ビットの 3 つの鍵長をサポートしており，現在では，256 ビットの鍵長が最も一般

的に使用される．また，Substitution Permutation Network (SPN) 構造を採用しており，セキュリティマージンが低く抑えられている．前述の OpenSSL においても AES が利用されている．

2.5.4　Hash-based Message Authentication Code

Hash-based Message Authentication Code (HMAC) は，ハッシュ関数を利用したメッセージ認証コードである．メッセージ認証とは，ネットワークを通じて伝送されたメッセージが途中で改ざんされていないことを確認する技術である．HMAC では，送信者がハッシュ関数を用いてメッセージをハッシュ化した値をメッセージに付加して送信する．受信者は，メッセージから送信者と同じハッシュ関数を使用してハッシュ値を算出し，受信したメッセージに付加されたハッシュ値と比較する．自身で算出したハッシュ値と受信メッセージに付加されたハッシュ値が一致すれば，メッセージが改ざんされていないことが確認される．

HMAC は，2 回ハッシュ関数を用いた認証子の生成や，鍵付きハッシュ関数として利用されることがある．HMAC は，Secure Shell (SSH) プロトコルや SSL プロトコルにおける鍵生成の際に利用されている．また，インターネット層でのセキュリティを担保するプロトコルである IPsec は複数のプロトコルの集合であるが，そのうちの一つに Internet Key Exchange (IKE) が存在する．IKE では，key derivation と呼ばれる手続きに HMAC を利用している．

2.6　スレッドを用いた並行処理技術

Linux に代表される UNIX 系列の OS (Operating System)，Windows，Mac OS，iOS、Android など，マルチタスクが可能な OS は，マルチタスク OS と呼ばれ，複数の処理を同時実行可能である．マルチタスクとは，複数のタスクを頻繁に切り替えて処理する処理方式である．マルチタスクを利用するケースは様々であり，例として，複数クライアントからのリクエストを同時に処理するサーバが挙げられる．マルチタスクを使用しない場合，特定のクライアントとの処理が完了するまで，後続のクライアントから受信した処理が待機状態となる．各クライアントあたりの処理要求が小さ

く，サーバの作業量が限られるアプリケーションでは，深刻な問題には至らない．しかし，サーバが一つのクライアントの処理に大きな負荷がかかる場合，他の全てのクライアントに対する返信時間が許容限度を超える恐れがある．そのため，マルチタスク OS を利用することにより，複数クライアントからの要求を同時に処理することでこの問題を回避する．

マルチタスク OS は，プロセスやスレッドを利用することで，並行処理を実現する．プロセスとは，CPU (Central Processing Unit) 上で実行状態になっているタスクのことである．一方，スレッドとは，CPU 上で動作しているプロセスを最小に分割した処理単位である．プロセスやスレッドを利用すると処理を並行して行うことが可能となり，処理効率が向上する．スレッドは，プログラムを実行する際のコンテキスト情報が最小であるため，コンテキストを切り替える際のオーバーヘッドが比較的小さい．スレッドは，プログラムのソースコードによる管理が可能なユーザスレッドと，OS カーネルが管理するカーネルスレッドに分類され，いずれも CPU コアに対してマッピングされる．

また，並行処理とは，スレッドを利用することで単一のプロセスが複数の処理を並行して作業する方式である，一般的に，高スループットが求められるミドルウェアや複数クライアントからの接続および処理を受け入れるサーバは，並行処理を用いて，タスクを効率的に実行する．以下のセクションにて，アプリケーションのタスクに対して，プロセスが管理するスレッドを複数に分けることで並行処理を実現するマルチスレッドと，Go 言語のランタイムによって管理される軽量スレッドを複数生成して並行処理を実現する Goroutine について説明する．

2.6.1　マルチスレッド

マルチスレッドとは，一つのプロセスを複数のスレッドで処理することにより，処理速度の向上を図る並行処理方式である．CPU が複数の処理を同時実行する場合，処理時間を分割しタスクに均等に割り当てる，タイムスライスが実施される．タイムスライスでは，その時点で処理しているプロセスやスレッドを停止し，処理対象を他のプロセスやスレッドに切り替える，コンテキストスイッチが実行される．プロセスの切り替えによるコンテキストスイッチが発生した場合，メモリ空間およびアドレス空

間の切り替えが必要となる．また，一つのプロセスは一つのアドレス空間でもあるため，プロセスを生成する際には新たなアドレス空間を確保する必要があり，メモリを浪費する．故に，プロセスの並列実行による並行処理は，オーバーヘッドに起因する処理時間の増加が比較的大きい．一方で，スレッドの切り替えによるコンテキストスイッチは，同一プロセス内であればアドレス空間の切り替えが発生しない．加えて，スレッドはプロセス内で生成されるため，スレッドの生成に際して生成元プロセスのアドレス空間を使用する．故に，スレッドの生成は低負荷で実行することが可能である．したがって，スレッドは，プロセスに比べ小さいオーバーヘッドで切り替えることが可能である．

図 2.18 に，シングルコア CPU とマルチコア CPU における，マルチスレッドの処理時間の比較を示す．シングルコアにおけるマルチスレッドによる並行処理は，シングルスレッドに比べて，一つの処理を完了するまでの時間が増加する場合がある．これは，シングルコア CPU が同時に一つの処理しか実行可能でないことに起因する．一方，マルチコア CPU においては，マルチスレッドを用いることで，処理速度と精度を飛躍的に向上させることが可能である．これは，スレッドを用いることにより，アプリケーション内で必要に応じて多数の処理を並列実行可能となるためである．現代の CPU のほとんどはマルチコアを実装しているため，複数のコアで一度に並行処理を実現することが可能であり，マルチスレッドは有用性が高い並行処理技術である．

2.6.2　Goroutine

Goroutine とは，Go 言語で使用される軽量なスレッドであり，Go プログラムの最も基本的な構成単位である．Gorutine は，軽量スレッドを用いて処理をすることで処理速度の向上を図る並行処理方式である．Go 言語で記述されたプログラムが動作すると，Main Goroutine と呼ばれる Goroutine が最低一つ起動する．Main Goroutine は，プロセスが開始する際に自動的に生成され，起動される．生成された他の Goroutine 間の処理に優先度はなく，排他制御を実装しない限り，全ての Goroutine は非同期で実行される．スレッドと Goroutine の位置付けを図 2.19 に示す．スレッドは，CPU コアに対してマッピングされ，OS によって管理されるが，Goroutine は，OS のスレッドに対してマッピングされ，Go ランタイムによって管理される．Goroutine は通常の OS スレッ

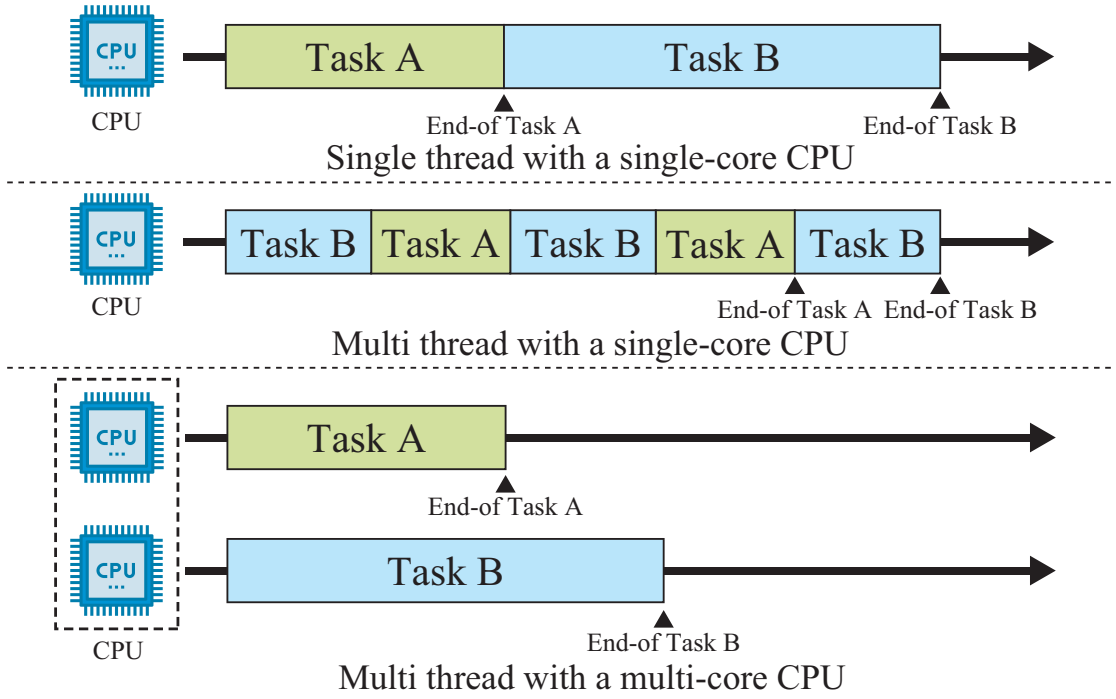


図 2.18 Comparison of processing times with Single-Thread and Multi-Thread

ドの ID と異なり，指定しない限りどのスレッドにマッピングされるかは決定されていない．Go ランタイムは生成された Goroutine に対して以下の処理を実行する．

- カーネルから割り当てられたメモリを分割し，必要に応じて割り当てる．
- ガベージコレクタを動かす．
- Goroutine のスケジューリングを行う．

Go ランタイムの概要図を図 2.20 に示す．Go ランタイムの内部は，主にマシンスレッド，Goroutine，Go ランタイムスケジューラから構成される．マシンスレッドは Linux カーネルにおける物理 CPU コアに対応する．Goroutine はプロセスに相当し，Go ランタイムスケジューラは，OS が提供するスレッドスケジューラと同等の機能を提供する．現代のカーネルは CPU の M 個のコア数に対し，同時に N 個の複数の処理が可能な M:N スケジューラを実装しており，ハイブリッドスレッディングを提供する．Go ランタイムスケジューラも同様に，M 個のカーネルスレッドに，N 個のユーザスレッド

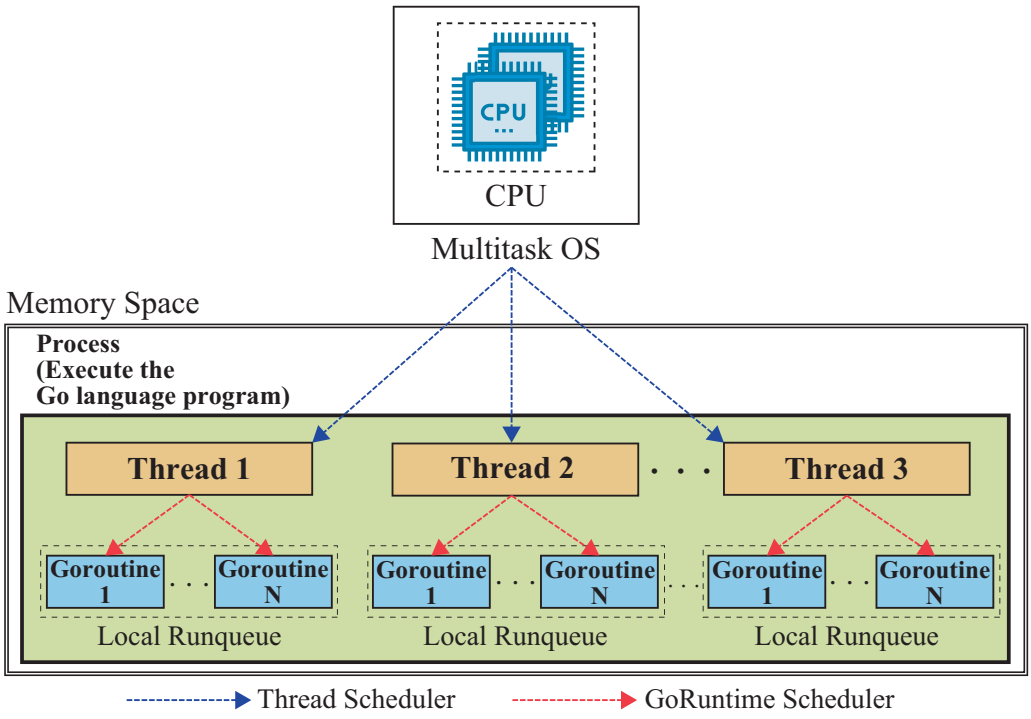


図 2.19 Relationship between thread and goroutine

を対応させたものにスケジューリングされるため，複数の CPU コアを扱うことが可能であり M:N スケジューラを実現する．Go ランタイムは sysmon と呼ばれる特殊なマシンスレッドが，プログラムの実行においてボトルネックとなる処理を常に監視する．その際，sysmon はスケジューラによって実行が止められることがないよう，sysmon が稼働しているマシンスレッドは，特定の OS スケジューラから分離される．Go ランタイムスケジューラはランキュー等のスレッドが行う作業を束ね，マシンスレッド毎にタスクとして Goroutine のリストを保持する．各スケジューラはマシンスレッドを保持しており，実行可能な Goroutine をグローバルキューから取り出して，ローカルキューに追加する．タスクは Goroutine が実行される際，ローカルキューから Go ランタイムスケジューラによって取り出され，マシンスレッド上で実行される．また，G0 はマシンスレッドに割り当てて実行する Goroutine とは別に割り当てた，特別な Goroutine であり，実行中の Goroutine が待ち状態，もしくはブロック状態になると起動する．さらに，Go ランタイムスケジューラを動かすことの他に，Goroutine によって割り当てられたスタックの増減処理や，ガベージコレクションを独自に実行する．ガベージコレ

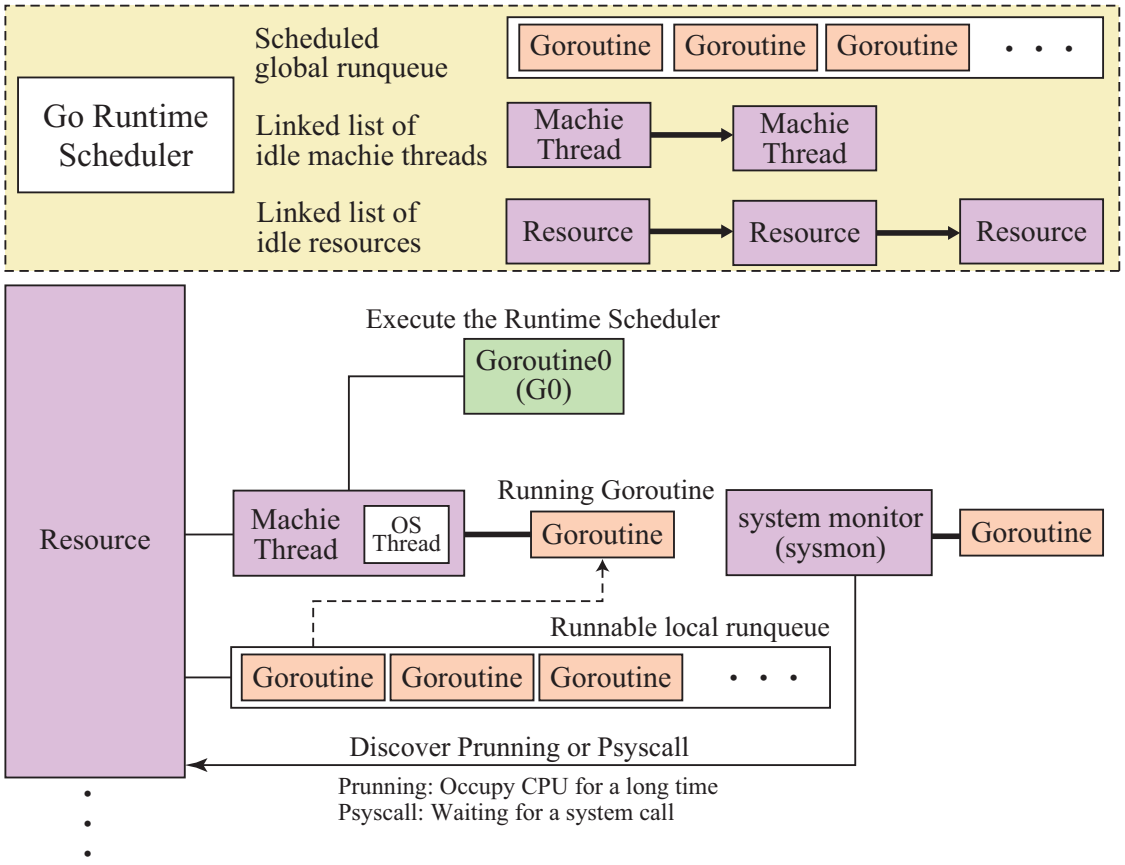


図 2.20 Overview of Go Runtime

クションとは，プログラムが動的に確保したメモリ領域のうち，断片化したメモリ領域及び不要になったメモリ領域を自動的に解放する機能である．Go ランタイムは常に一つの Goroutine を実行させることはなく，適度に実行する Goroutine を取り替えることにより，プログラム実行の効率化を図る．実行中の Goroutine を一旦中断する処理は，プリエンプションと呼ばれ，sysmon がボトルネックとなっている Goroutine を見つけると実行される．実行のボトルネックになっている Goroutine とは，CPU を長時間占有している Goroutine や，システムコール待ち状態の Goroutine を指す．以上より，Go は内部で Go ランタイムスケジューラを動作させることにより，OS カーネルが提供するスレッドのスケジューラとは分離して管理され，異なる Goroutine を実行する際も OS スレッドの切り替えを伴わない仕組みを持つ．そのため，マシンスレッド上で実行されている Goroutine が Go ランタイムスケジューラによって切り替えられた場合

にも，OS がコンテキストスイッチを認識することはない．つまり，Goroutine の切り替えは，カーネルスレッドの切り替えのように，プログラムカウンタやメモリ参照場所の切り替え処理が発生しない．加えて，Goroutine は生成と破棄に関する操作が非常に低コストであるため，生成，破棄のたびに OS に要求を行うカーネルスレッドと比較して，実行時間が極めて短縮され，軽量に動作することが可能である．

2.7　ソフトウェアアーキテクチャ

現在のクラウドサービスは，大きく分けて二種類のアーキテクチャで設計されている．一つ目はスレッド駆動型アーキテクチャであり，主な使用事例として Apache が挙げられる．スレッド駆動型アーキテクチャは，端末から受信した通信リクエスト一つに対して専用のスレッドを生成し，リクエストの処理を行う．二つ目はイベント駆動型アーキテクチャであり，主な使用事例として Nginx が挙げられる．イベント駆動型アーキテクチャは，端末から受信した通信リクエストをイベントキューに格納し，事前に生成された複数のワーカースレッドに渡すことによって，ワーカースレッドが処理を行う．スレッド駆動型アーキテクチャは，一つのリクエストに対する計算量が多い場合の処理を得意としており，イベント駆動型アーキテクチャは，同時発生する多数のリクエストの処理を得意としている．以下セクションにて，スレッド駆動型アーキテチャおよびイベント駆動型アーキテクチャについて詳細に説明する．

2.7.1　スレッド駆動型アーキテクチャ

スレッド駆動型アーキテクチャの概要図を図 2.21 に示す．スレッド駆動型アーキテクチャは，主に Apache で利用されている．スレッド駆動型アーキテクチャは，リクエストから生成されるジョブ毎に専用のスレッドを生成するため，単一のジョブの計算量が多い処理を得意とする．

まず，スレッド駆動型アーキテクチャでは，クライアントからのリクエストジョブを受け取ると，ジョブを処理する専用のスレッドを生成する．その後，生成されたスレッドが処理を行うことで，レスポンスを生成し，送信する．故に，スレッド駆動型アーキテクチャはマルチスレッド方式を活用して実現される．スレッド駆動型アーキテク

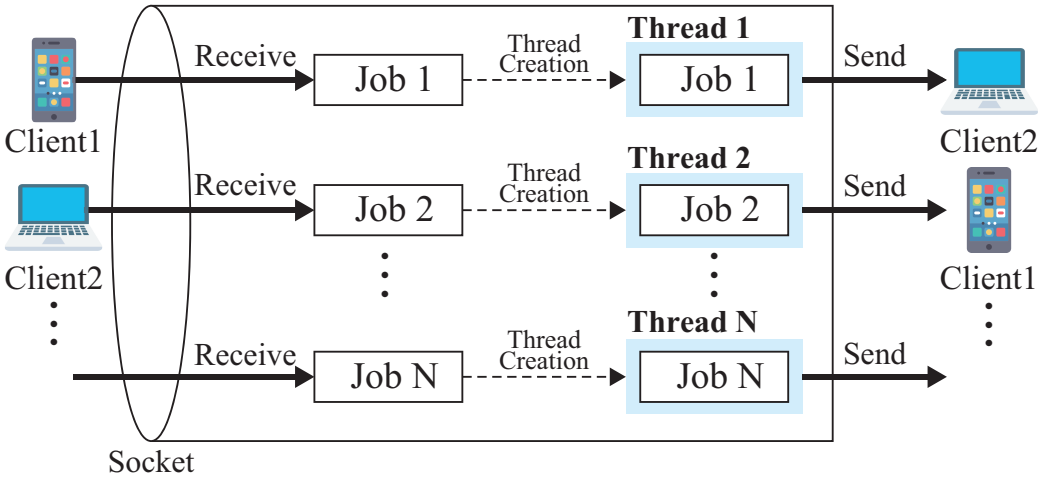


図 2.21 System Model of Thread driven architecture

チャは，専用のスレッドを生成するため，各リクエストの計算量が多く負荷が高い場合でも，他のリクエストに対する処理に対して影響が発生しづらい．一方で，同時接続数が増加した場合，スレッドの生成が都度発生することに起因する処理遅延が発生することが懸念される．

2.7.2　イベント駆動型アーキテクチャ

イベント駆動型アーキテクチャの概要図を図 2.22 に示す．イベント駆動型アーキテクチャは，主に Nginx で利用されている．イベント駆動型アーキテクチャは事前に生成されたワーカースレッドで処理を行うため，同時発生する多数のリクエストの処理を得意とする．

まず，イベント駆動型アーキテクチャでは，クライアントからのリクエストジョブを受け取ると，ジョブを親スレッドに格納する．次に，親スレッドはジョブ処理用のワーカースレッドの状態を確認し，ワーカースレッドが処理可能な状態であれば，ジョブをワーカースレッドに引き渡す．その後，ジョブを引き渡されたワーカースレッドが処理を行うことで，レスポンスを生成して，送信を行う．また，イベント駆動型アーキテクチャは，コンテキストスイッチの回避を目的として，シングルスレッド方式を活用して実現される．さらに，イベント駆動型アーキテクチャは事前に生成されたワー

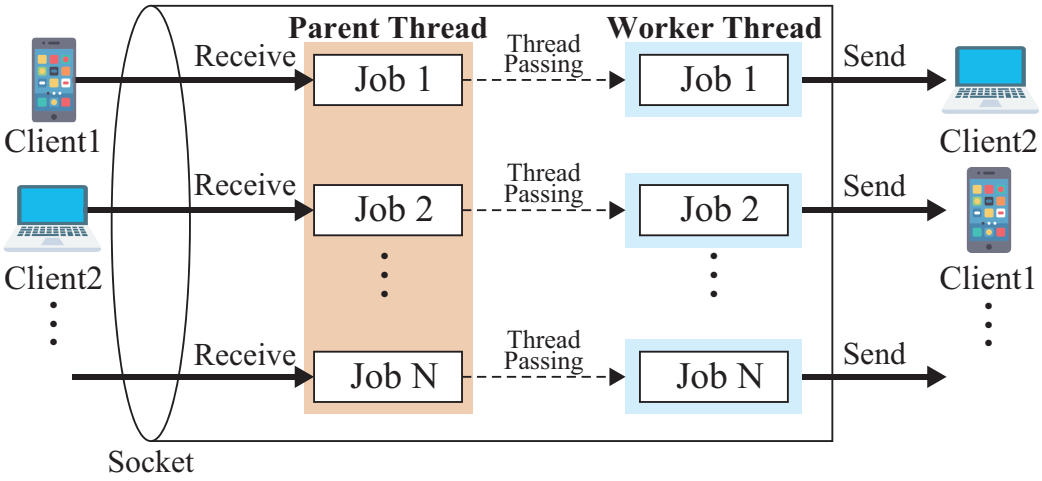


図 2.22 System Model of Event driven architecture

カースレッドで処理を行うため，スレッド生成や待機による大きな処理遅延は発生しない．

2.8　セキュリティモデル

近年では，ビジネスシーンにおけるシステムの在り方が大きく変わり，それに伴って新たなセキュリティの在り方が登場した．企業はクラウドサービスを利用した業務システムを構築するようになった．また，テレワークの普及によって，BYOD を採用する企業も増え、企業のシステムやビジネスはリアル拠点からオンラインへその舞台を移しつつある．故に，社内外に多くの境界が生まれ、高度化するサイバー攻撃から境界を完全に守ることが困難となり，従来の境界型セキュリティモデルでは通用しなくなっている．以下のセクションでは従来のセキュリティモデルである境界防御モデルと新たなセキュリティモデルであるゼロトラストモデルの詳細を説明する．

2.8.1　境界防御モデル

境界防御モデルは，社内等の信用する内部ネットワークとインターネット等の信頼しない外部ネットワークに境界を設け，境界内部を保護するセキュリティモデルであ

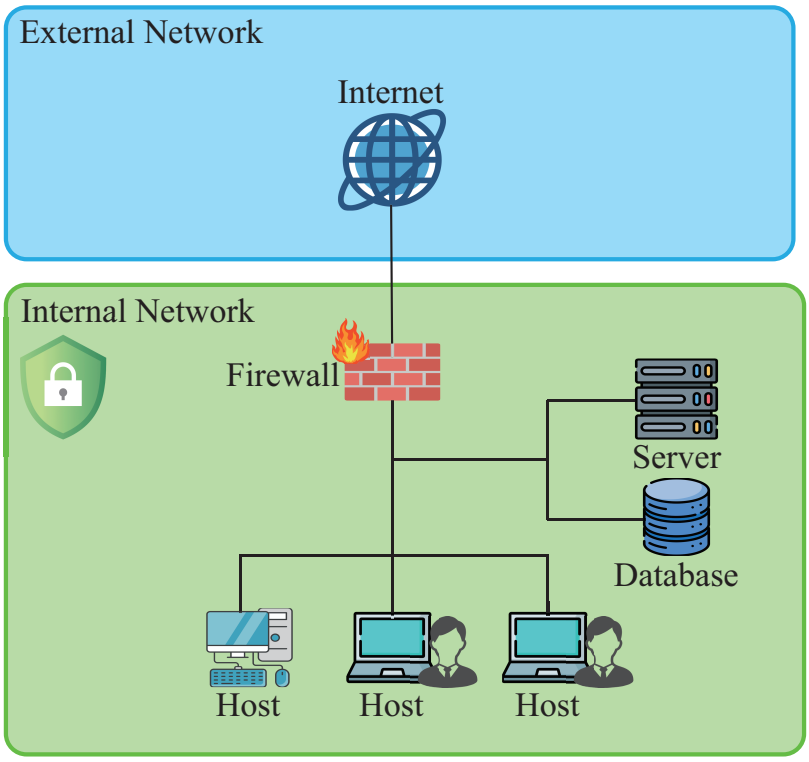


図 2.23 Overview of Perimeter Defence Model

る．境界防御モデルの概要を図 2.23 に示す．組織が保護すべき情報資産とそれを取り扱うユーザや端末は境界内部に存在することが前提であり，境界内部のものは全て信頼可能なものとして，境界外部からの脅威を境界で検証することにより，境界内部への侵害を防ぐ．

境界防御モデルの代表的なソリューションとして，ファイアウォール，Virtual Private Network (VPN)，プロキシサーバなどが存在する．

2.8.2　ゼロトラストモデル

ゼロトラストモデルは，企業におけるクラウドサービスの普及やモバイル端末の活用によるサイバー攻撃の増加や，組織内部の不正によって生じるセキュリティ事故によって，従来の境界防御モデルのようなセキュリティの境界が曖昧になったことで登場した新たなセキュリティモデルである．ゼロトラストモデルの概要を図 2.24 に示す．

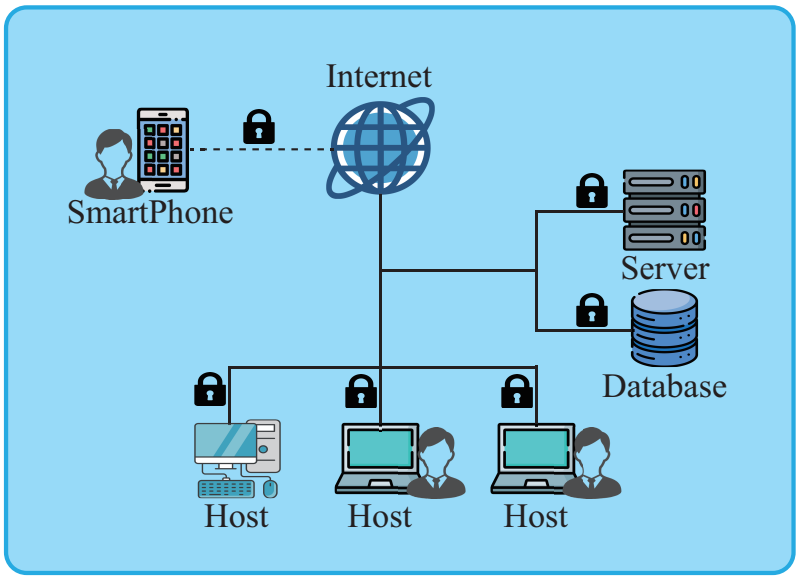


図 2.24 Overview of Zero Trust Model

ゼロトラストモデルは，2010 年に米国の調査会社である ForresterResearch 社によって提唱された．ゼロトラストモデルは境界防御モデルと違い，全ての通信を信頼しないという考え方に基づく．社内外のネットワーク環境における従来の境界の概念を捨て去り，保護すべき情報資産にアクセスするものは全て信用せず，その安全性を検証することで情報資産への脅威を防ぐ．これにより，外部からのサイバー攻撃だけでなく，内部からの情報漏洩やマルウェア感染等の脅威に対しても対策が可能となる．

2.9　サイバー攻撃手法

世の中には非常に様々なサイバー攻撃が存在している．サイバー攻撃とは，サーバ，パソコン，スマートフォンなどの情報端末に対して，ネットワークを通じシステムの破壊やデータの窃盗，改ざんなどをする行為のことである．サイバー攻撃の被害件数は世界規模で増加しており、その種類や手口は多様化，巧妙化している．また，攻撃の目的は，どの不正行為にも共通する金銭窃盗や国家や企業の機密情報を窃取し組織体の戦略変更やイメージダウンを狙う組織犯罪，ハクティビストによる政治的・社会的な主張など、攻撃者のタイプにより異なる．近年は、従来の脆弱性につけ込む攻

撃方法にくわえて、特定の企業、団体、個人をターゲットにする標的型メール攻撃、アカウントへの不正アクセス、ランサムウェア、DDoS 攻撃、IoT 機器やスマートフォン、タブレットを狙ったサイバー攻撃が増加している．以下のセクションではサイバー攻撃の中でもサービス提供を停止させて妨害行為を行う攻撃について詳述する．

2.9.1　Denial of Service 攻撃

Denial of Service (DoS) 攻撃とは，アクセスが集中することでサーバに負荷が掛かることを利用し，悪意を持ってサーバに大量のデータを送信するサイバー攻撃のことである．DoS 攻撃の概要を図 2.25 に示す.

同じ時間帯にサーバに大量のアクセスが送信されると，アクセスに関するデータ送受信でサーバは想定より多くの処理をしなければならない．サーバに処理負荷が掛かるとリクエスト/レスポンスの遅延やサーバがダウンが発生し，サービスの提供を行うことが不可能になってしまう恐れがある．サービスが提供不可能になってしまうと，経済的損失だけでなく，社会的信頼が失墜することも考えられる．

DoS 攻撃にはフラッド型と脆弱性型の 2 種類が存在する．フラッド型の攻撃には UDP フラッド攻撃やコネクションフラッド攻撃などがある．UDP フラッド攻撃とは，サーバに対しサイズの大きなパケットを送信し続けることによって負荷をかける攻撃である．また，コネクションフラッド攻撃は，サーバに大量の接続を要求し，機能停止を狙う攻撃である．一方で，脆弱性型の攻撃は，サーバやアプリの脆弱性を利用して不正処理を実行し、サービス機能を停止させる手法である．この手法では，サーバ内部に侵入し、不正処理を大量に行わせて機能停止や異常終了に追い込む．

2.9.2　Distributed Denial of Service 攻撃

Distributed Denial of Service (DDoS) 攻撃とは，攻撃対象となるサーバに複数のコンピュータから大量のデータを送信することで正常なサービス提供を妨害するサイバー攻撃のことである．DDoS 攻撃の概要を図 2.26 に示す.

DoS 攻撃との違いは，DoS 攻撃は一台のコンピュータから攻撃を仕掛けられるのに対し，DDoS 攻撃は複数のコンピュータを踏み台にし，数万台から数千万台規模で DoS

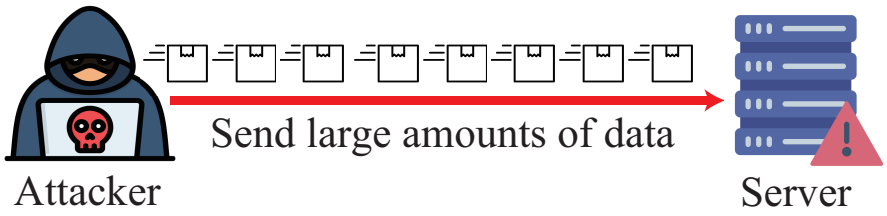


図 2.25 Overview of Denaial of Service attack

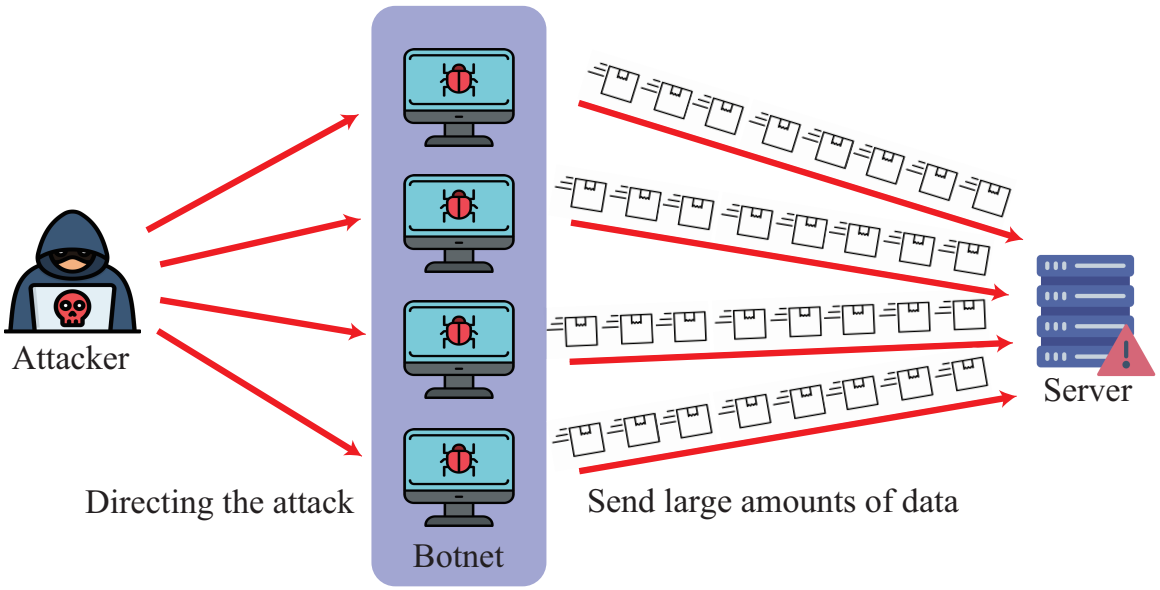


図 2.26 Overview of Distributed Denaial of Service attacks

攻撃を行うことである．DDoS 攻撃は，ボットと呼ばれるマルウェアに感染したコンピュータが使用されることが多い．攻撃者はマルウェアに感染したコンピュータを支配下に置き，ボットネットと呼ばれるネットワークを形成することによって DDoS 攻撃を行う．DDoS 攻撃では，他者のコンピュータを踏み台にしているため，加害者を特定するのが困難である．

DoS/DDoS 攻撃の対策方法としては，ファイアウォールや Intrusion Detection System (IDS), Intrusion Prevention System (IPS) などが挙げられる．ファイアウォールとは，ネットワークの境界に設置され、不正なアクセスや攻撃から情報を守る防御システムである．このシステムを活用すると，特定の IP アドレスやポートへの通信の許可をコントロールすることが可能となる．また，通信の監視と管理者への警告を行う IDS は，不正侵入検知システムと呼ばれる．IDS の機能に加えて，通信の遮断までを行う IPS は不正侵入防止システムと呼ばれる．IDS や IPS は，通信内容を検査し，異常な量のトラフィックや不正な処理要求の有無を分析する．IDS が通信の異常を検出した際には、迅速に管理者へ通知し、管理者は異常な通信のブロック、ファイアウォールのフィルタリングを強化することで攻撃対する対処が可能である．IPS は通信の異常を通知することに加え、ワームや DoS などのパケットが持つ特徴を検知した直後に、人手を介することなく自動的に通信を遮断するところまでの対応を行う．IPS は IDS と比べ、迅速なインシデント対応が可能なメリットが存在する反面、業務に大きな影響が出やすいというデメリットも存在する。万一、設定が不正な場合など誤検知を行ってしまうと、即時にシステムが停止してしまうケースが想定される．

2.9.3　リプレイ攻撃

リプレイ攻撃とは，認証情報などのパケットをネットワーク上の通信を盗聴することによって取得し，そのパケットを再送信することによってそのユーザになりすます攻撃のことである．リプレイ攻撃の概要を図 2.27 に示す.

リプレイ攻撃では，盗聴したパケットが暗号化されていたとしても，サーバはパケットを解釈可能であるため，攻撃者はユーザの意図しない行動を実行可能となってしまう．また，リプレイしたパケットを用いることで DoS 攻撃にも応用可能である．

リプレイ攻撃の対策手法には，ワンタイムパスワード，nonce を用いた手法などが

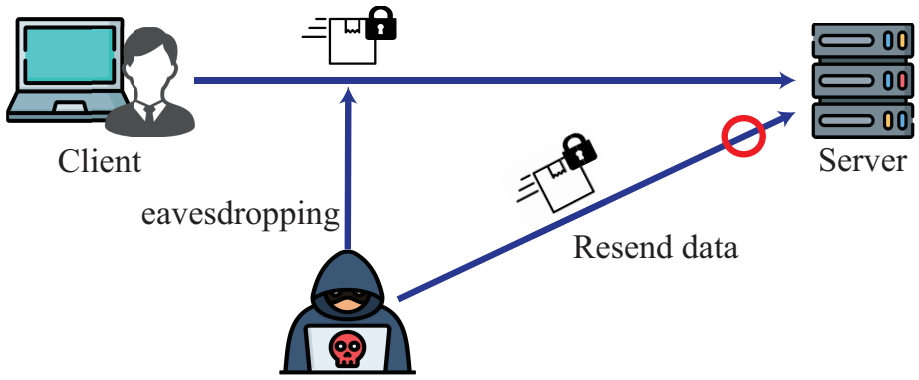


図 2.27 Overview of Replay attack

挙げられる．ワンタイムパスワードとは，文字通り一度のみ使用可能なパスワードのことである．また，nonce とは，number used once の略称であり，一度のみ使用されるランダムな値のことである．ユーザが認証情報等をサーバに送信する際には，認証情報をハッシュ化し，生成されたハッシュ値に nonce を付け加えて再度ハッシュ化した値を送信する．サーバはユーザと同じ方法でハッシュ値を生成し比較することによって認証情報の正当性を確認する．nonce は，認証の度に変更されるため，ユーザが送信するハッシュ値が毎回変わることとなる．そのため，攻撃者はハッシュ値を傍受したとしても，そのハッシュ値を使い回すことは不可能である．

第3章

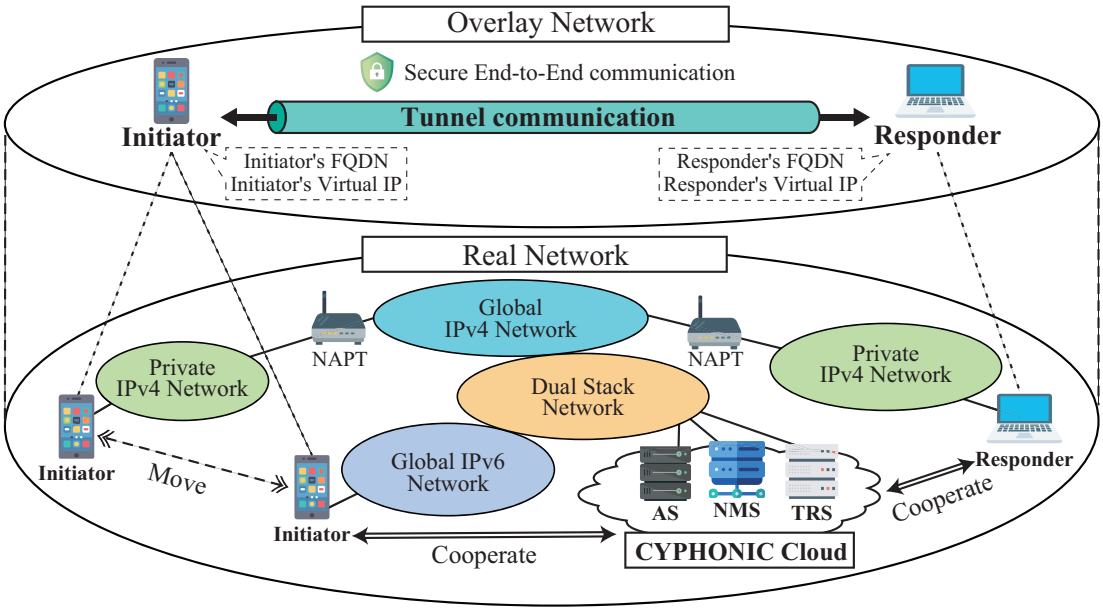
CYPHONIC

3.1 概要

端末間の安全で確実な直接通信を実現するには，通信に関する複数の問題に対処する必要がある．そのためには，NAPT や IP バージョンの差異による通信接続問題，ネットワーク切り替えにより通信が切断される移動透過問題の解決が必要である．また，直接通信での第三者による通信内容の盗聴や改ざん，不正な端末との接続など，セキュリティの問題に対しても対処が必要である．

CYPHONIC は，オーバーレイネットワークを構築することで，通信接続性や移動透過性を包括的に提供する通信プロトコルである．加えて，認証済み端末同士の暗号化通信によって，ゼロトラストセキュリティに基づくセキュアな直接通信を実現する．

CYPHONIC の概要図を図 3.1 に示す．CYPHONIC は，通信端末内に仮想 IP アドレスを割り当てた，仮想的なネットワークインターフェイスを生成する．CYPHONIC を利用するアプリケーションは，生成された仮想ネットワークインターフェイスに対してデータを送受信することで，仮想 IP アドレスを用いて通信をする．これにより，実ネットワーク上に実 IP アドレスではなく，仮想 IP アドレスを使用する仮想的なネットワークが構築される．このような，仮想 IP アドレスを使用して通信を行うネットワーク空間をオーバーレイネットワークという．オーバーレイネットワークは仮想的なネットワークであり，実際の通信パケットは実ネットワーク上を流通する必要がある．そのため，仮想 IP アドレスを用いた通信は実 IP アドレスによってカプセル化され，UDP トンネルによる通信を行う．



AS: Authentication Service　NMS: Node Management Service　TRS: Tunnel Relay Service

図 3.1 Overview of CYPHONIC

CYPHONIC が構築するオーバーレイネットワークによって，通信接続性や移動透過性が実現可能となる．CYPHONIC には，NAPT や IP バージョンの差異の概念が存在しないため，オーバーレイネットワーク上の通信は，実ネットワーク上の NAPT や IP バージョンの影響を受けない．また，仮想 IP アドレスは不変であるため，接続先ネットワークが変更された場合でも変更されることがない．そのため，移動などのネットワーク切り替えに伴って通信が切断されることなく通信の継続が可能である．

加えて，CYPHONIC は，端末認証と通信内容の暗号化によってセキュアな通信を実現可能である．CYPHONIC を利用する端末は，CYPHONIC のクラウドサービスに対して認証を行う．認証により，CYPHONIC が構築したオーバーレイネットワーク上には，認証済み端末のみが存在する．よって，通信端末が不正な端末へ接続されることがない．さらに，相手端末と通信をする際には，通信内容を暗号化する暗号鍵を直接交換する．そのため，CYPHONIC のクラウドサービスや第三者に暗号鍵を知られることなく，セキュアな直接通信が可能となる．

CYPHONIC を利用して通信を行う端末は，端末を一意に識別する Fully Qualified Domain Name (FQDN) と不変な仮想 IP アドレスを用いて通信を行う．FQDN と仮想 IP

アドレスは，端末認証後に付与される．アプリケーションは，FQDN によって相手端末を識別する．通信を開始する端末は，相手端末の FQDN を指定して CYPHONIC のクラウドサービスに通信開始要求を送信する．その後，クラウドサービスは，FQDN から IP アドレスを求める名前解決を行い，通信開始端末に相手端末の仮想 IP アドレスを通知する．これにより，送信元および送信先に仮想 IP アドレスを使用した通信が可能となる．

また，端末は CYPHONIC のクラウドサービスに対してネットワーク環境の変化を随時通知することによって，状況に応じた通信経路を使用する．端末は認証後とネットワーク移動時に，自身が接続しているネットワーク情報をクラウドサービスに対して登録する．その後，クラウドサービスは，端末に対して双方のネットワーク環境に応じて選択された通信経路を指示する．端末は，クラウドサービスから指示された通信経路を用いることで，NAPT が存在する場合でも直接通信が可能となる．NAPT の種類により，直接通信が困難な場合は，クラウドサービスが通信中継を行う．

そして，端末は暗号鍵を生成し，トンネルを確立することによって，端末間での安全な直接通信を実現する．通信経路を指示された端末は，端末間の送受信データを暗号化する暗号鍵を生成する．生成された暗号鍵は，通信経路上の誰にも知られることなく相手端末に共有される．暗号鍵が共有されると，端末間でトンネルを確立し，データの送受信を行う．データは暗号鍵によって暗号化されるため，盗聴されることなく安全な直接通信が可能となる．

3.2　CYPHONIC の構成要素

CYPHONIC は，CYPHONIC を利用した通信を管理するクラウドサービスである CYPHONIC クラウドと，CYPHONIC を利用するための機能を搭載した端末である CYPHONIC ノードから構成される．さらに，CYPHONIC クラウドは，Authentication Service (AS)，Node Management Service (NMS)，Tunnel Relay Service (TRS) の 3 種類のサービスから構成される．CYPHONIC クラウドは，IPv4 ネットワークと IPv6 ネットワークからの通信を処理可能とするため，デュアルスタックネットワークに設置される．AS は認証処理を行うサービスであり，事前に登録されたユーザの情報から CYPHONIC ノードが正規のユーザであることを確認する．NMS は，CYPHONIC ノードのネット

ワーク情報を管理し，通信経路の選択および指示を行うサービスである．TRS は，CYPHONIC ノード間の直接通信が困難な場合に，CYPHONIC ノードの送受信するデータを中継する通信中継サービスである．以下に，AS，NMS，TRS と CYPHONIC ノードの詳細を説明する．

- Authentication Service (AS)
  AS は，CYPHONIC ノードを認証するサービスである．AS は，CYPHONIC を利用する際に，CYPHONIC ノードが最初に通信を行うサービスである．CYPHONIC ノードの認証要求を受信した AS は，事前に登録されたユーザ情報と通信に含まれるアカウント情報や証明書を照合し，CYPHONIC ノードが正規のユーザであることを確認する．認証完了後に AS は，CYPHONIC ノードに対して FQDN と仮想 IP アドレスを付与する．加えて，次に通信を行う NMS の IP アドレスと NMS との通信を暗号化する共通鍵 (Common Key) を配布する．ノードの FQDN と仮想 IP アドレス，Common Key は NMS へも共有する．

- Node Management Service (NMS)
  NMS は，CYPHONIC ノードの現在のネットワーク情報を管理し，CYPHONIC ノード間の通信経路を選択および指示を行うサービスである．CYPHONIC ノードは，AS の認証処理後，NMS に現在接続しているネットワークの情報を登録する．接続ネットワークが変更した際には，その都度 NMS に対して，接続ネットワーク情報の変更を通知する．ネットワーク情報には，NAPT の有無や使用している IP アドレスが含まれる．この際 NMS は，データベースから取得した Common Key を用いて CYPHONIC ノードと暗号化通信を行う．CYPHONIC ノードが相手端末の FQDN を指定して通信接続要求を送信すると，NMS は，双方のネットワーク環境の情報から通信経路を選択し，双方の端末に指示する．通信経路は，基本的に CYPHONIC ノードを直接結ぶ経路となる．しかし，NAPT の種類により CYPHONIC ノード同士の直接通信が困難な場合は，TRS を介した通信経路を指示し，TRS に対して通信中継を依頼する．一方で，端末間の通信にはセキュアな通信を実現するため，暗号鍵 (End Key) を用いた通信を行う．CYPHONIC はゼロトラストの設計思想に則るため，End Key は CYPHONIC ノード間での直接交換が望ましい．そこで NMS は，End Key を安全に配布するための暗号鍵 (Tunnel

Key と Temporary Key) を生成する．Tunnel Key は CYPHONIC ノードと TRS に，Temporary Key は，CYPHONIC ノードのみに配布される．Tunnel Key は TRS を介さない End Key の交換において，鍵の盗聴を防止する．Temporary Key は，TRS を介した End Key の交換に使用される．これにより，TRS に鍵を盗聴されることなく，端末間の End Key の交換が可能となる．

- Tunnel Relay Service (TRS)
  TRS は，CYPHONIC ノード間の直接通信が困難な場合に，CYPHONIC ノードの送受信するデータを中継する通信中継サービスである．以下二つのうちいずれかを満たす場合は，TRS による通信中継が必要となる．
  - 双方の CYPHONIC ノードが NAPT 配下に存在する場合．しかし，NAPT の種類や組み合わせにより，後述する経路最適化処理が行われ，TRS を経由しない直接通信が可能となる．
  - 双方の CYPHONIC ノードが使用する実 IP アドレスのバージョンが異なる場合．

  TRS は，NMS の経路選択処理における中継依頼に基づいて，CYPHONIC ノード間の送受信データの中継を行う．CYPHONIC ノード間の直接通信が困難な場合は，End Key の交換も TRS を経由して行う．その際，CYPHONIC ノードと TRS 間の通信は Tunnel Key を用いて暗号化するため，End Key を Temporary Key を用いて暗号化する．TRS は，Temporary Key を所持していないため，TRS を中継して End Key を交換する場合でも，TRS に対して End Key は秘匿される．

- CYPHONIC ノード
  CYPHONIC ノードは，CYPHONIC を利用して通信を行う端末のことである．CYPHONIC ノードは，一連の通信プロセスを開始する Initiator と，一連の通信プロセスの中で返答を行う相手端末である Responder に分類される．CYPHONIC クラウドとの連携やオーバーレイネットワーク上での通信を実現するための処理実行は，デーモンプロセスである CYPHONIC Daemon が担う．CYPHONIC ノードは自身の端末に CYPHONIC Daemon をインストールする．CYPHONIC Daemon の起動後，CYPHONIC ノードは以下のいずれかの方法を用いて，AS に対して認

証を要求する．

- ID・パスワード
- 証明書
- Single Sign On (SSO)

認証が成功すると，AS から自身の FQDN と仮想 IP アドレスが付与される．加えて，次に通信を行う NMS の IP アドレスと NMS との暗号化通信に使用する共通鍵 (Common Key) を受け取る．仮想 IP アドレスは，実 IP アドレスとの競合を回避する必要があるため，実ネットワークでは使用されないアドレス帯を使用する．仮想 IP アドレスのバージョンとして IPv4 を使用する場合は，ネットワーク相互接続装置のベンチマークテスト用のアドレス帯として割り当てられている 198.18.0.0/15 を活用する．一方，仮想 IP アドレスのバージョンとして IPv6 を使用する場合は，文書記述用のアドレス帯として割り当てられている 2001:db8::/32 を活用する．認証処理後，CYPHONIC ノードは NMS に接続し，現在接続しているネットワーク情報を登録する．CYPHONIC ノード上で動作するアプリケーションが相手端末と通信を行う際には，相手端末の FQDN を指定して，NMS に通信開始要求を送信する．NMS は Domain Name System (DNS) サーバと同様に，FQDN から仮想 IP アドレスを求める名前解決を行い，CYPHONIC ノードに相手端末の仮想 IP アドレスを通知する．これにより仮想 IP アドレスを用いた通信が可能となる．さらに，CYPHONIC ノードは，NMS からその後の端末間通信で使用する通信経路を受信する．通信経路が指示されると，CYPHONIC ノードは端末間の暗号化通信に使用する暗号鍵 (End Key) を生成し，相手端末との間にトンネルを構築する．アプリケーションにより生成された仮想 IP パケットは，CYPHONIC Daemon によって実 IP パケットにカプセル化され，UDP トンネルにて送受信される．他の CYPHONIC ノードから受信した実 IP パケットは，CYPHONIC Daemon によってデカプセル化され，仮想 IP パケットがアプリケーションに届けられる．また，移動などによって接続先ネットワークが変更された場合には，ネットワーク情報の登録からの処理プロセスを再度実行する．

3.3　各プロセスの通信シーケンス

CYPHONIC では，オーバーレイネットワーク上での端末間通信を実現するため，認証処理，位置情報登録処理，経路選択処理，トンネル確立処理，データ通信処理の五つの通信プロセスを実施する．さらに，双方の CYPHONIC ノードが NAPT 配下に存在する場合，TRS を介した冗長な経路を排した，直接通信を試みる経路最適化処理を実施する．以下に，各通信処理の詳細を説明する．

3.3.1　認証処理

認証処理は，CYPHONIC ノードの正当性を確認し，認証する処理である．認証処理の存在によって，オーバーレイネットワーク上で通信する端末は，信頼した認証済み端末のみに限られる．これにより，端末が不正な端末に接続することを防ぐ．CYPHONIC ノードは，初回起動時に，AS に対して認証を要求する．その際，CYPHONIC ノードは，事前に取得したルート証明書を使用して，通信相手である AS の真正性を確認する．また，CYPHONIC ノードと AS 間の通信は，TLS によって暗号化される．

図 3.2 に認証処理の通信シーケンスを示す．認証処理では，まず，CYPHONIC ノードが AS に対して Login Request メッセージを送信する．Login Request メッセージには，CYPHONIC ノードを利用するユーザの ID・パスワードからなるアカウント情報，または，クライアント証明書の情報が含まれる．AS が Login Request メッセージを受信すると，事前に登録されたユーザ情報と照合を行い，CYPHONIC ノードの正当性を確認する．認証が完了すると，AS は CYPHONIC ノードの FQDN と仮想 IP アドレスを選定する．さらに，CYPHONIC ノードと NMS 間の暗号化通信で利用する共通鍵 (Common Key) を生成する．その後，AS が共通鍵をデータベースへ保存することによって NMS へ共通鍵を共有する．AS は，Login Request メッセージに応じて Login Response メッセージを CYPHONIC ノードに応答する．Login Response メッセージには，認証を要求した CYPHONIC ノードに割り当てる FQDN および仮想 IP アドレスを NMS との暗号化通信に使用する共通鍵が含まれる．

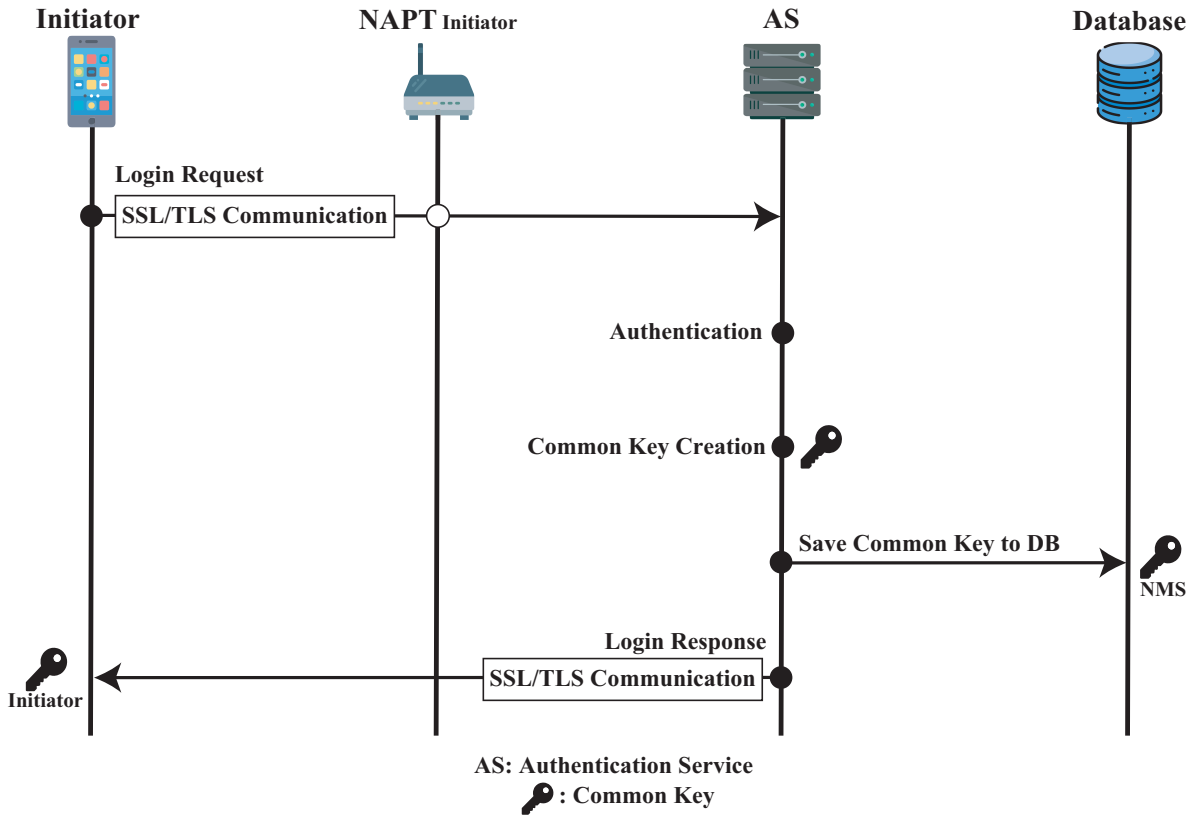


図 3.2 Authentication process

3.3.2　位置情報登録処理

位置情報登録処理は，CYPHONIC ノードが現在接続しているネットワーク情報を，NMS に登録する処理である．位置情報登録処理は，AS の認証後および CYPHONIC ノードの移動に伴う接続先ネットワークの切り替わりが発生する度に実施される．位置情報登録処理に際して，CYPHONIC ノードと NMS は，認証処理で AS から配布された共通鍵を用いて暗号化通信を行う．なお，共通鍵は．CYPHONIC ノードは Login Response メッセージ，NMS は，データベースからそれぞれ取得する．

図 3.3 に位置情報登録処理の通信シーケンスを示す．位置情報登録処理では，まず，CYPHONIC ノードが NMS に対して Registration Request メッセージを送信する．Registration Request メッセージには，CYPHONIC ノードが使用している実 IP アドレスやポート番号が含まれる．データ部分に CYPHONIC ノードの実 IP アドレスが含ま

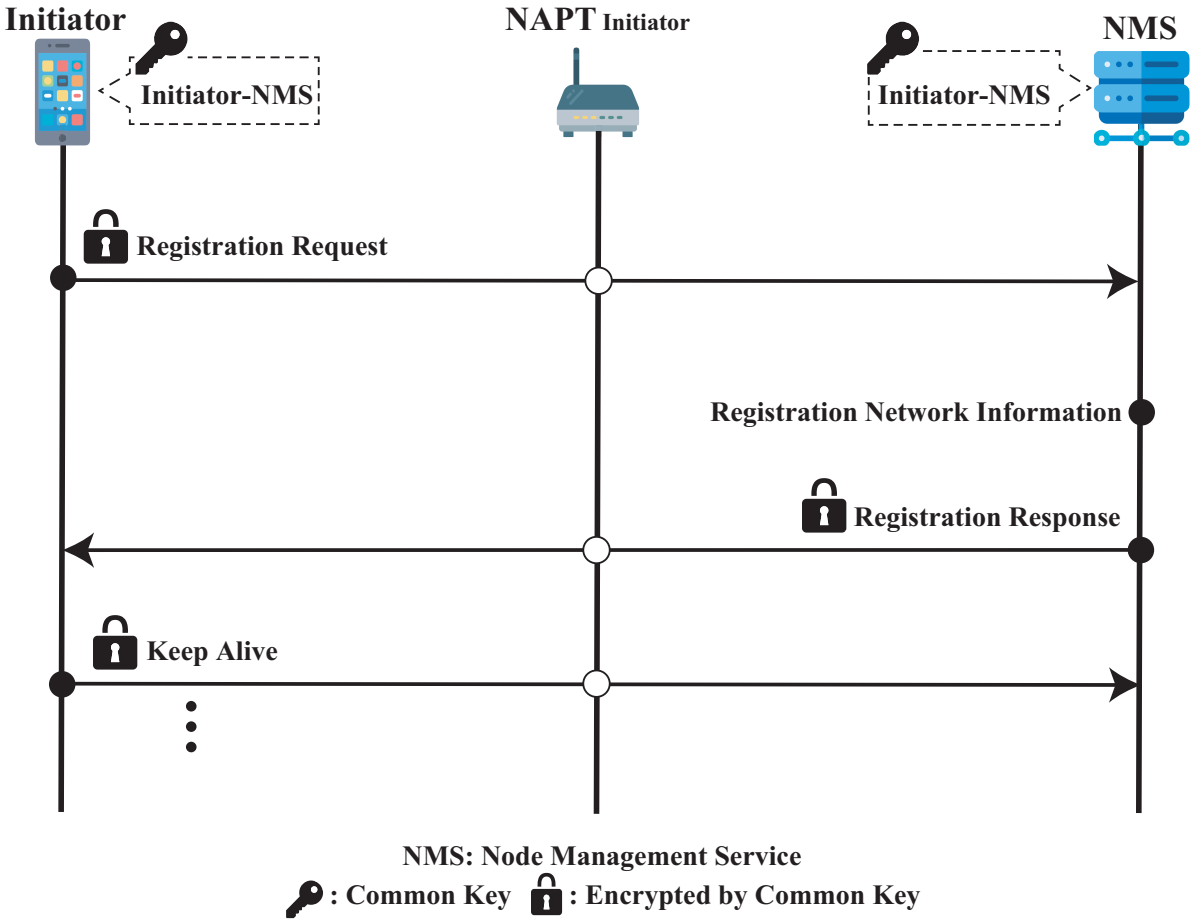


図 3.3 Registration process

れているため，Registration Request メッセージの IP ヘッダに格納された IP アドレスとの比較により，通信経路上で IP アドレスが変更されたか否かを検知可能である．これにより，NMS は NAPT の有無を判断し，CYPHONIC ノードのネットワーク情報を登録もしくは更新する．ネットワーク情報の登録が完了すると，NMS は Registration Response メッセージを CYPHONIC ノードに返信する．また，CYPHONIC ノードは，NMS とのコネクションを維持するために，定期的に Keep Alive メッセージを送信する．これにより，CYPHONIC ノードが NAPT 配下に存在する場合は，NAPT のマッピングテーブルを維持することが可能なため，NMS からの通信を受信可能となる．

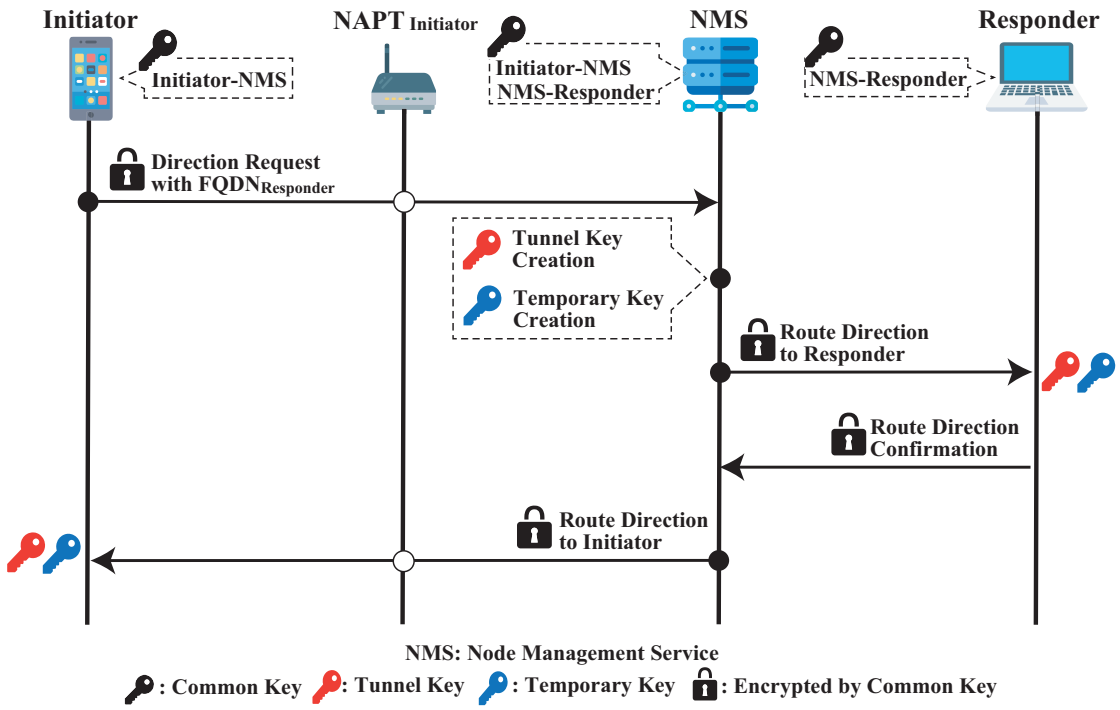


図 3.4 Route selection process

3.3.3　経路選択処理

経路選択処理は，CYPHONIC ノード間の通信に使用する，通信経路の選択を行う処理である．通信経路は，通信を行う CYPHONIC ノード双方のネットワーク環境情報によって選択され，直接通信が可能な場合と TRS を介した通信中継が必要な場合の 2 種類の処理手順が存在する．また，経路選択処理，後述するトンネル確立処理，データ通信処理で使用する 3 種類の暗号鍵を以下のように規定する．

- End Key

  End Key は，CYPHONIC ノード間の通信確立後，アプリケーションの送受信データを暗号化するために使用する共通鍵である．End Key は CYPHONIC ノードが生成し，Initiator および Responder のみが所持することで CYPHONIC クラウドを含めた第三者に対して通信内容を秘匿する．

- Tunnel Key

  Tunnel Key は，End Key の交換メッセージを隣接コンポーネント間において暗号

化する共通鍵である．アプリケーションの送受信データの暗号化に End Key を使用するには，End Key が第三者に盗聴されることなく，CYPHONIC ノード間で交換することが重要である．そのため，End Key の交換メッセージを Tunnel Key で暗号化することで，第三者から End Key を秘匿することが可能となる．Tunnel Key は，NMS によって生成され，Initiator，Responder および TRS に配布される．

- Temporary Key
  Temporary Key は，End Key の交換に TRS を経由する場合，TRS から End Key を秘匿するために使用する共通鍵である．TRS を経由して End Key の交換を行う際，TRS は Tunnel Key を所持しているため，End Key の覗き見が可能となってしまう．そのため，TRS が所持しない Temporary Key を用いて End Key を暗号化する．さらに，それを Tunnel Key で暗号化した交換メッセージの送信によって，TRS から End Key を秘匿しつつ，TRS を経由した End Key の交換が可能となる．Temporary Key は，NMS によって生成され，Initiator と Responder のみに配布される．

図 3.4 に CYPHONIC ノード間の直接通信が可能な場合の通信シーケンスを示す．また，図 3.5 に TRS を介した通信中継が必要な場合の通信シーケンスを示す．図 3.4 において，Initiator が Responder と通信を開始する際には，Responder の FQDN を指定して，Direction Request メッセージを NMS に送信する．NMS は，Initiator から Direction Request メッセージを受信すると，トンネル確立処理で使用する二種類の暗号鍵，Tunnel Key と Temporary Key を生成する．その後，生成した二種類の暗号鍵と経路指示を Route Direction to Responder メッセージに含め，Responder に送信する．Responder は，Route Direction to Responder メッセージを受信すると，Initiator を認識するとともに，Tunnel Key および Temporary Key を取得する．次に，Responder は，NMS に経路指示を承諾したことを示すため，Route Direction Confirmation メッセージを送信する．NMS は，Responder からの Route Direction Confirmation メッセージを受信すると，Initiator に対しても経路を指示するため，二種類の暗号鍵を含めて Route Direction to Initiator メッセージを送信する．また，Route Direction to Initiator メッセージには，Responder の仮想 IP アドレスが含まれているため，相手の FQDN を指定して，仮想 IP アドレスを求める一連の処理は名前解決に相当する．

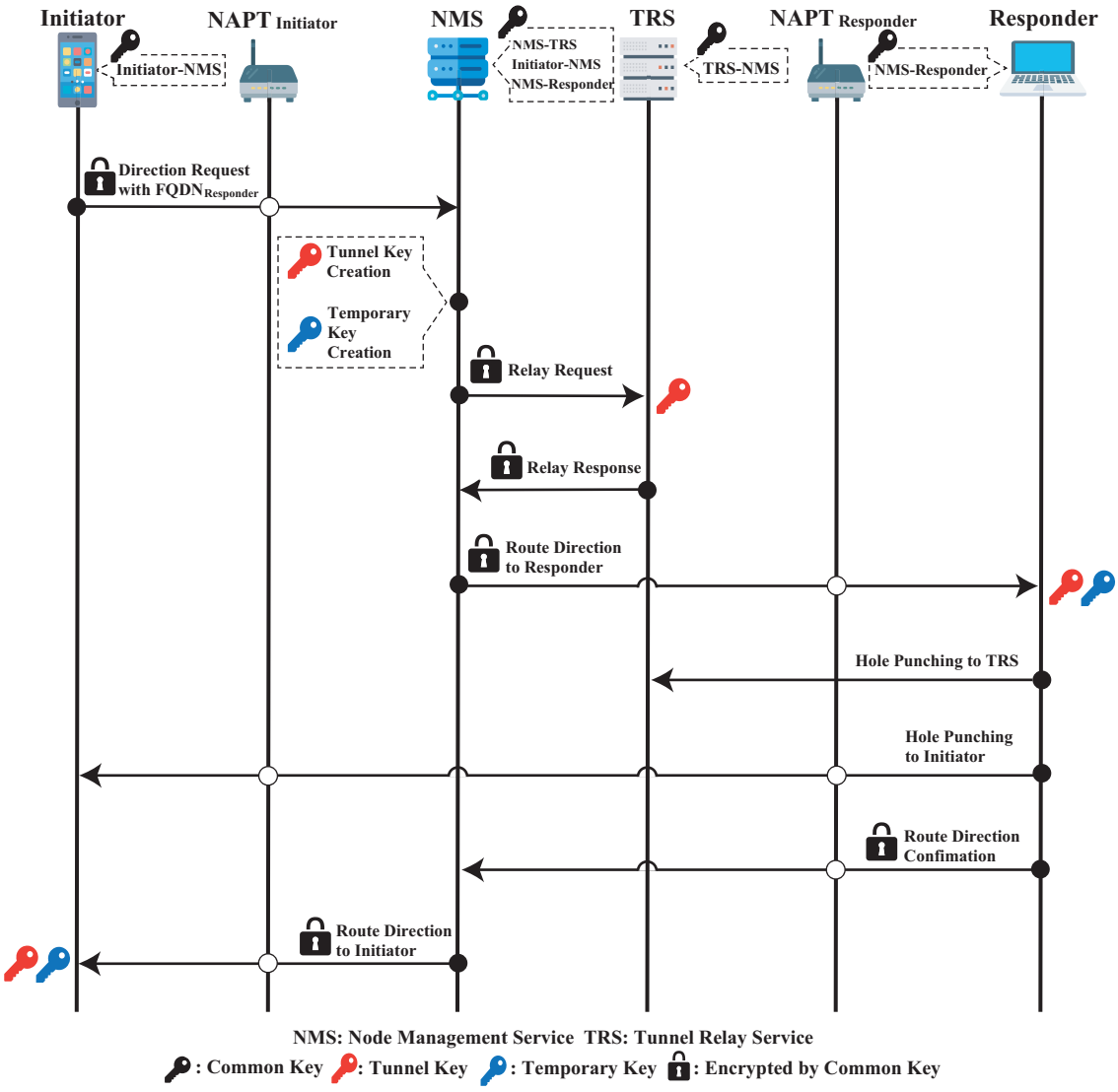


図 3.5 Route selection (via TRS) process

一方，Initiator と Responder の双方が NAPT 配下に存在する場合や，使用する IP バージョンが異なる場合は，CYPHONIC ノード間の直接通信が不可能である．よって，このような場合には TRS による通信中継が必要となる．ただし，双方の CYPHONIC ノードの NAPT の組み合わせが，表 3.1 の×印以外の場合，後述する経路最適化処理が実施され，TRS を介さない直接通信が可能となる．図 3.5 において，Initiator からの Direction Request メッセージを受信した NMS は，二種類の暗号鍵を生成する．経路選択によって TRS による中継が必要と判断した場合は，TRS に Relay Request メッセージを送信する．Relay Request メッセージには二種類の暗号鍵の内 Tunnel Key のみが含まれている．その後，TRS が，Realy Response メッセージを返して，中継依頼が終了する．TRS を用いた通信を行う場合，NMS は Route Direction to Responder メッセージにて，TRS による中継処理が必要であることを指示する．経路指示を受けた Responder は，NMS に Route Direction Confirmationa メッセージを送信する．そして NMS が Initiator に Route Direction to Initiator メッセージを送信する．この時，同時に Responder は，TRS に対して UDP Hole Punching を行う．これにより，TRS によって中継されたメッセージの受信が可能となる．また，後述する経路最適化処理を実現するため，Initiator に対しても UDP Hole Punching を行い，双方向通信を可能にする．

3.3.4　トンネル確立処理

トンネル確立処理は，Initiator と Responder のみが通信内容を把握可能な通信経路を構築する処理である．Initiator と Responder 間の通信経路は，実際には UDP トンネルである．Initiator と Responder は，経路選択処理で受信した Route Direction メッセージの内容に沿って通信経路を構築する．従って，トンネル確立処理の手順も CYPHONIC ノード間の直接通信が可能な場合と TRS を介した通信中継が必要な場合に分けられる．図 3.6 に CYPHONIC ノード間の直接通信が可能な場合の通信シーケンスを示す．また，図 3.7 に TRS を介した通信中継が必要な場合の通信シーケンスを示す．

図 3.6 において，Initiator はトンネル構築に先立ち，Initiator と Responder 間の通信内容を暗号化する暗号鍵である End Key を生成する．次に，End Key を Tunnel Request メッセージに含め，Tunnel Key で暗号化して Responder へ送信する．Responder が Tunnel Request メッセージを受信すると，Tunnel Key を用いてメッセージを復号し，End Key

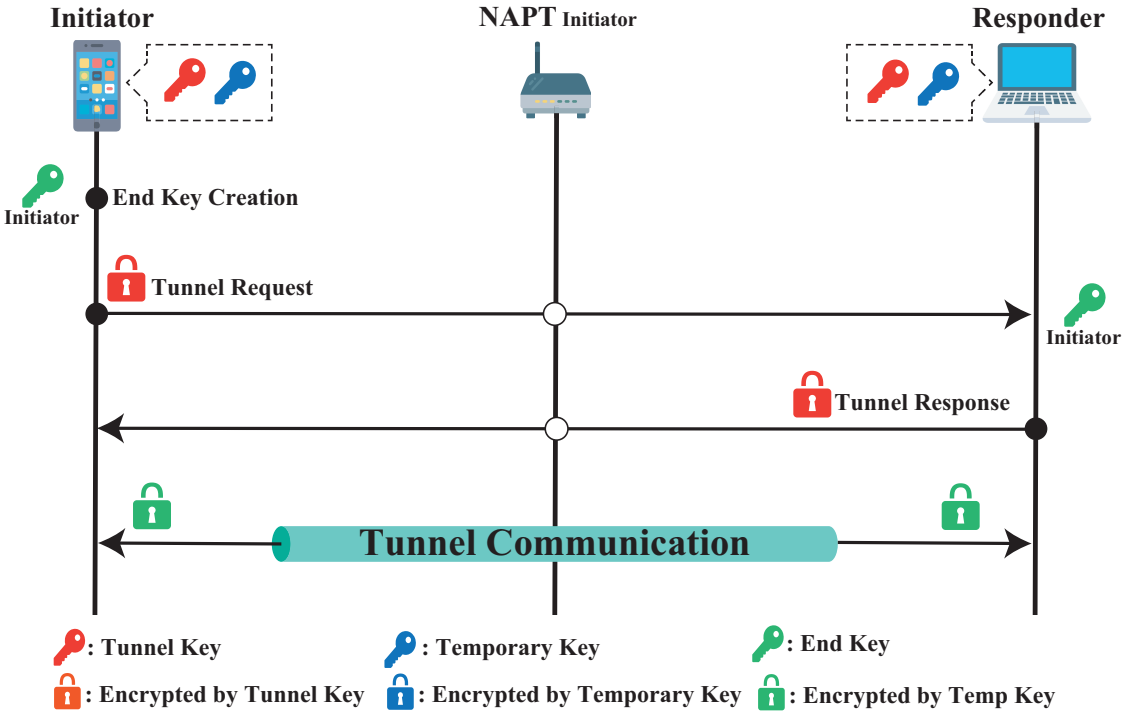


図 3.6 Tunnel establishment process

を取得する．その後，Responder は，Initiator に対し，Tunnel Key で暗号化した Tunnel Response メッセージを送信する．Initiator は，受信した Tunnel Response メッセージを Tunnel Key で復号し，End Key が正常に共有されたか否かを確認する．以上の処理を以て，End Key が CYPHONIC ノード間で交換され，トンネル確立処理が完了する．

一方，CYPHONIC ノード間の直接通信が困難な場合には，TRS を介してトンネル確立処理を実施する．図 3.7 において，Initiator は，End Key を生成した後，Temporary Key を用いて End Key を暗号化する．次に，Temporary Key で暗号化した End Key を，Tunnel Request メッセージに含め，Tunnel Key で暗号化して TRS へ送信する．メッセージを受信した TRS は，Responder に対して Tunnel Request メッセージを転送する．この際 TRS は，Tunnel Key を所持しているためメッセージを復号可能な一方で，Temporary Key は所持していないため，End Key の内容把握は不可能である．その後，Tunnel Request メッセージを受信した Responder は，Tunnel Key と Temporary Key を用いてメッセージを復号し，End Key を取得する．Responder は End Key を取得後，Tuunel Response メッセージを Tunnel Key で暗号化し，TRS を経由して Initiator へ送信する．

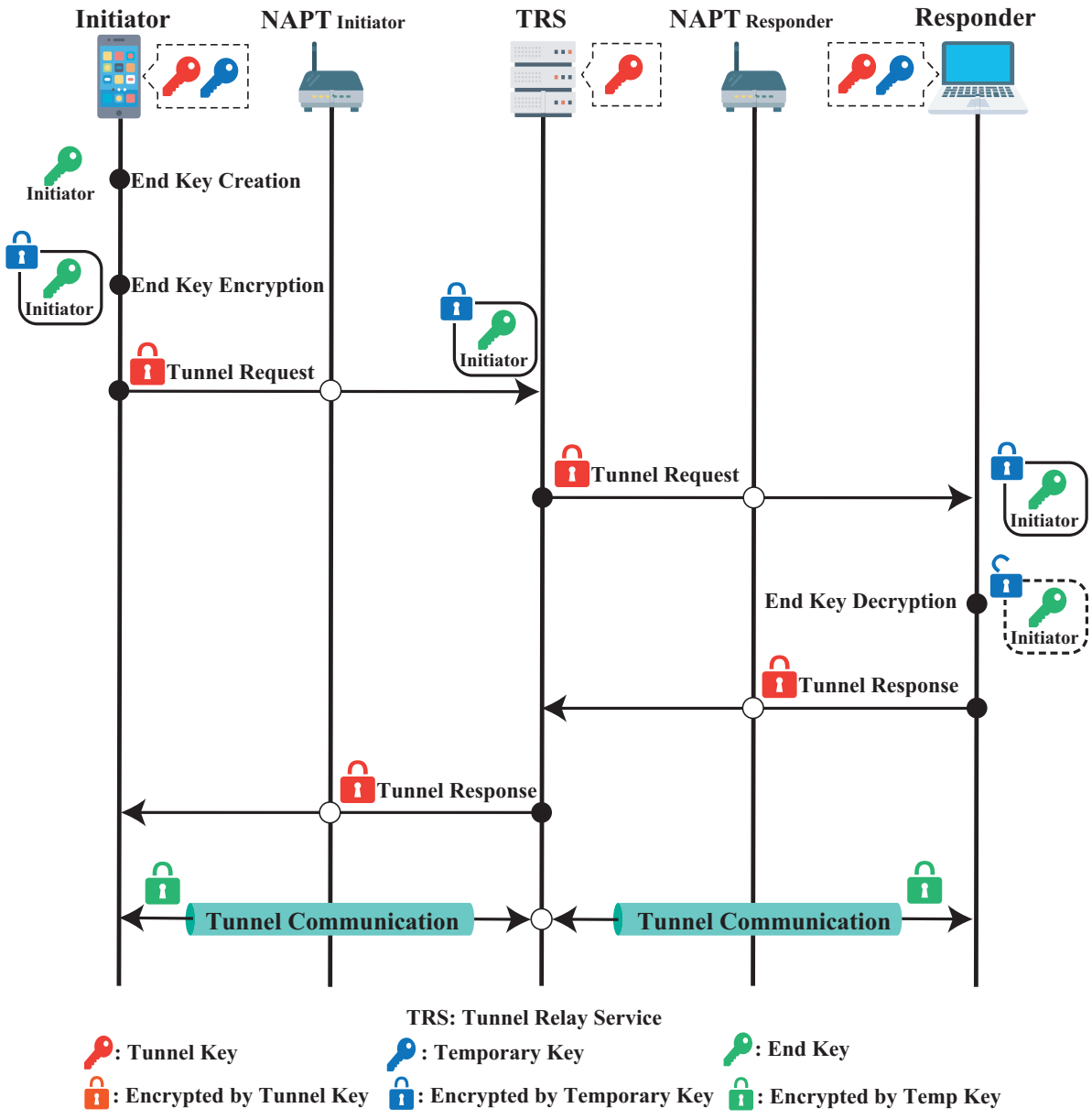


図 3.7 Tunnel establishment (via TRS) process

Initiator は，受信した Tunnel Response メッセージを Tunnel Key で復号し，End Key が正常に共有されたか否かを確認する．以上の処理を以て，直接通信が困難な場合においても，TRS を経由して End Key が CYPHONIC ノード間で交換され，トンネル確立処理が完了する．

3.3.5　データ通信処理

データ通信処理は，CYPHONIC ノード間でアプリケーションデータの送受信を行う処理である．CYPHONIC 間のデータ通信はオーバーレイネットワーク上で行われる．まず，アプリケーションは，送信先の仮想 IP アドレスを指定し，仮想ネットワークインターフェースから仮想 IP パケットを送信する．次に，CYPHONIC Daemon が仮想ネットワークインターフェースから仮想 IP パケットを取り込むと，End Key を用いて仮想 IP パケットを暗号化する．一方で，オーバーレイネットワークは仮想的なネットワークであるため，実際の通信においてパケットは実ネットワーク上を流通する必要がある．そのため，仮想 IP パケットを送信先の相手端末や TRS の実 IP アドレスを指定した IP ヘッダでカプセル化を行い，実 IP パケットを生成する．そして，通信はトンネル確立処理で構築した UDP トンネルを介して行う．実 IP パケットを受信した端末は，デカプセル化によって仮想 IP パケットを取得し，End Key による復号を行う．復号された仮想 IP パケットは受信端末の仮想ネットワークインターフェースを通じてアプリケーションに渡される．以上の処理の繰り返しによって，CYPHONIC ノード上で動作するアプリケーションは，仮想 IP アドレスを使用したデータ通信の実現が可能となる．

3.3.6　経路最適化処理

経路最適化処理とは，TRS による通信中継が必要な経路の選択後，CYPHONIC ノード同士が直接通信を試みることによって，経路冗長化の排除を目的とした処理である．Initiator と Responder の双方が NAPT 配下に存在する場合は，TRS による通信中継が必要である．そのため，データ通信では常に TRS を経由する必要があり，経路冗長化の問題が発生する．そこで，トンネル確立処理において，End Key の交換後，

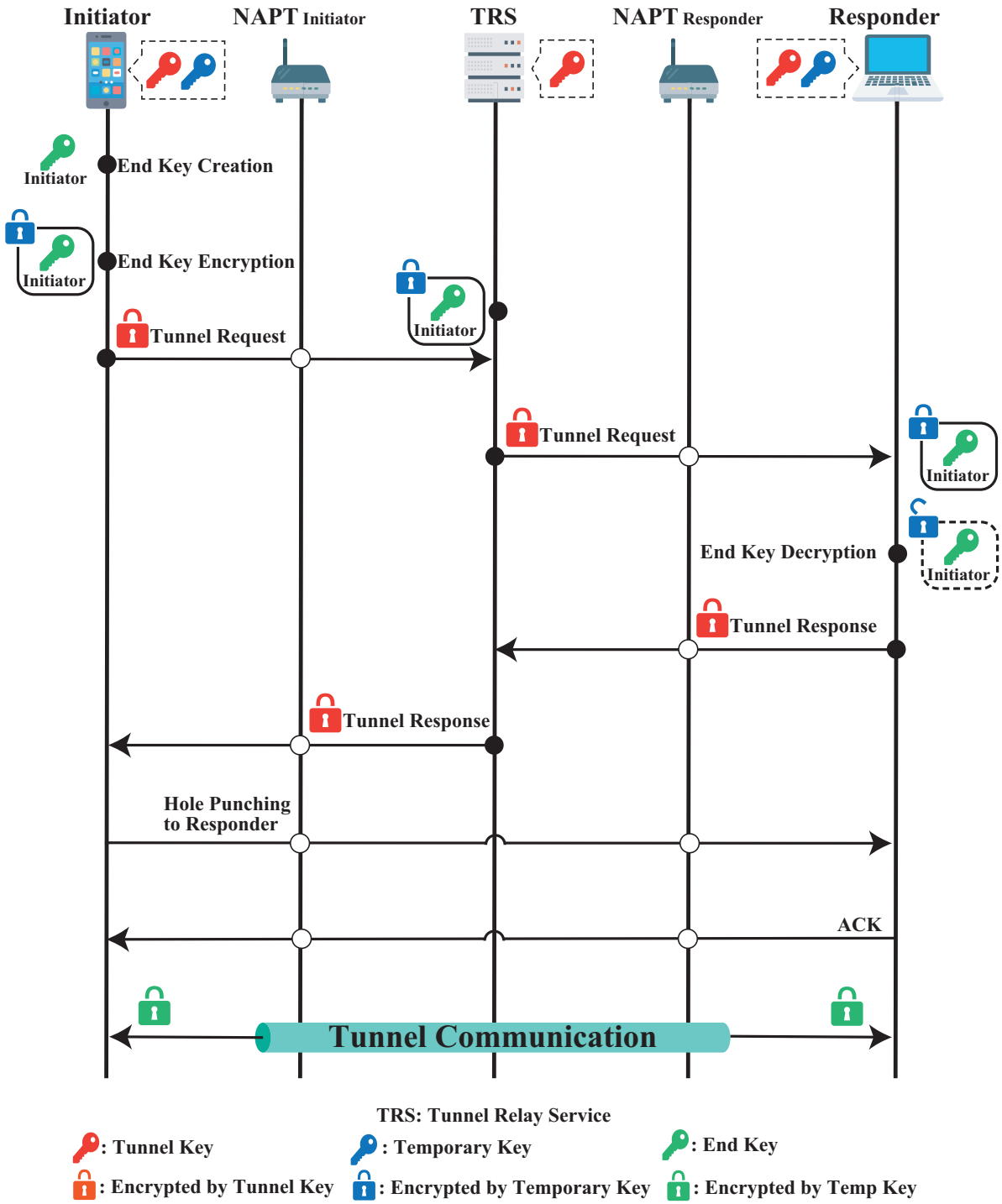


図 3.8 Route optimization process

CYPHONIC ノード同士の直接通信を試みる経路最適化処理が実施される．

図 3.8 に経路最適化処理の通信シーケンスを示す．TRS を介した End Key の交換後，Initiator は Responder に対して，UDP Hole Punching を実行する．これにより，Initiator 側の NAPT には，マッピングテーブルのエントリが作成される．また，前項の経路選択処理において，TRS を介した通信が選択された場合，Responder は Initiator に対して，UDP Hole Punching を実行する．そのため，Responder 側の NAPT にはマッピングテーブルが作成済みである．Initiator の UDP Hole Punching を Responder が受信すると，受信確認応答として ACK を Initiator に返信する．Initiator が ACK の受信に成功した場合，経路最適化が成功したとみなされる．よって，以降 Initiator が送信するパケットは，直接 Responder に宛てて送信される．

しかし，Initiator と Responder が接続されている NAPT の組み合わせにより，経路最適化が困難な場合が存在する．表 3.1 に NAPT の組み合わせにおける，経路最適化の可否を示す．表 3.1 中の○で示されている NAPT の組み合わせは，経路最適化が可能である．表 3.1 中の△で示されている NAPT の組み合わせは，短期間に複数回の経路最適化処理を実行することで，経路最適化が可能となる．表 3.1 中の×で示されている NAPT の組み合わせは，経路最適化の実現が困難である．これは，Port Restricted Cone NAPT や Symmetric NAPT が強い制約を持つことに起因している．Port Restricted Cone NAPT や Symmetric NAPT は，パケットの送信元 IP アドレスやポート番号の制約が強いため，UDP Hole Punching や ACK パケットをブロックする．従って，Initiator と Responder の双方が制約の強い NAPT に接続されている場合，経路最適化処理を実現が不可能であり，CYPHONIC ノード間の直接通信が極めて困難である．以上より，経路最適化による経路冗長化の排除が達成されない場合も存在する．

3.4　通信処理で用いられるメッセージフォーマット

CYPHONIC では，3.3 節で説明したデータ送受信の前準備に当たる通信制御シグナリング，および実際にデータの送受信を行うデータ通信処理が実行される．これらの処理では，CYPHONIC 独自のヘッダを付与した CYPHONIC メッセージによる通信を行う．CYPHONIC メッセージは，Base Header，Message Body，Hash-based Message Authentication Code (HMAC) から構成される．NMS，TRS，CYPHONIC ノードは，通

表 3.1 Combination of NAPT types that can apply route optimization process

Initiator \ Responder	Full Cone	Restricted Cone	Port Restricted Cone	Symmtric
Full Cone	○	○	○	○
Restricted Cone	○	○	○	△
Port Restricted Cone	○	○	○	×
Symmetric	○	○	×	×

信時に共通鍵を用いて Message Body を暗号化した後，HMAC を付与して相手端末へ送信する．HMAC は，共通鍵とハッシュ関数を用いて，メッセージの改ざん検知に使用されるメッセージ認証符号の一つである．以下に，CYPHONIC メッセージのフォーマットについて詳細を述べる．

- Base Header
  付録の図 A.1 で示される Base Header は，全ての CYPHONIC メッセージに付与される．CYPHONIC メッセージを受信した CYPHONIC コンポーネントは，Base Header の内容をもとに Message Body を解読する．以下に，Base Header の各フィールドについて詳細に説明する．
  - Transaction ID (32bit)
    シグナリングで用いられる識別子を格納する．一連の処理を識別可能な Universally Unique Identifier (UUID) やハッシュ値のような一意な値を挿入する．
  - Version (8bit)
    CYPHONIC のバージョンを示す.
  - Flag (8bit)
    処理結果の成否，送信者の位置情報登録処理が初回であるか否か，などの状態を示す.
  - Type (8bit)
    CYPHONIC メッセージの種類を示す.

- Count (8bit)
  メッセージの同時送信数を示す．一つの Base Header に対し，複数のメッセージを付与して送信する場合，追加分のメッセージ数を格納する．証明書など巨大なデータを送信する場合に利用する．
- Sequence Number (32bit)
  一連の処理によるシグナリング中に，メッセージカウンタとして用いる．Sequence Number は，メッセージの送信時に格納されている値をインクリメントする．Sequence Number と HMAC を用いることでリプレイ攻撃への対策を実現する．
- Message Length (16bit)
  Base Header，Message Body，HMAC を含めたシグナリングメッセージの全体の長さを格納する．
- Next Option (8bit)
  Base Header と Message Body 部の間に挿入している拡張ヘッダを使用する場合に用いる．Next Opt は，拡張ヘッダの種類を示す．Next Opt は，IPv6 で規定されている拡張ヘッダと同様の形式であり，複数個の拡張ヘッダを用いる場合は，先頭にチェーンする．
- Reserved (8bit)
  パディングおよび将来的に利用するための空き領域であり，予約フィールドである．
- ID (128bit)
  経路選択処理以前のシグナリングにおいては，送信元を示す Node ID が格納される．トンネル確立処理以降のシグナリングおよびデータ通信においては，通信経路を示す Path ID が格納される．

- Hash-based Message Authentication Code (HMAC)
  付録の図 A.2 で示される HMAC は，CYPHONIC メッセージの末尾に付与され，メッセージの改竄検知に使用される．

- ACK
  付録の図 A.3 で示される ACK は，シグナリングメッセージの受信に成功したこ

とを示す確認応答用のメッセージである．特別な情報を返信する必要がない場合に利用される．Base Header の Flag フィールドには，処理結果の成否を格納する．

- Login Request
  付録の図 A.4 で示される Login Request は，CYPHONIC ノードが AS に認証処理を依頼するメッセージである．Login Request には，アプリケーションの識別子と認証情報を格納する．認証に成功した場合，オーバーレイネットワークへの参加が許可される．Login Request は，TLS によって暗号化されるため，Message Body の暗号化および HMAC の付与は行われない．

- Login Response
  付録の図 A.5 で示される Login Response は，AS が認証した CYPHONIC ノードに対して，認証結果を返信するためのメッセージである．Login Response には，認証結果，認証された CYPHONIC ノードを管理する NMS の IP アドレス，CYPHONIC ノードと NMS の通信で使用する共通鍵，CYPHONIC ノードに割り当てる FQDN と FQDN に紐づく仮想 IP アドレスが含まれる．Login Response は，TLS によって暗号化されるため，Message Body の暗号化および HMAC の付与は行われない．

- Registration Request
  付録の図 A.6 で示される Registration Request は，CYPHONIC ノードが，現在接続しているネットワーク情報を NMS に登録するためのメッセージである．Registration Request には，NAPT による変換前の IP アドレス及びポート番号，NMS からのシグナリングを受信する際の通知方法が格納される．今後，CYPHONIC のモバイル対応に伴い，Keep Alive に代わるエネルギー消費を抑えた新たな通知方法が登場する可能性がある．新しい通知方法を選択可能とするため，通知方法を指定するフィールドが予約されている．

- Registration Response
  付録の図 A.5 で示される Registration Response は，NMS が CYPHONIC ノードに対して Registration Request の結果を応答するためのメッセージである．Registration Response には，Registration Request の処理結果と，CYPHONIC ノードが NAPT 配下に存在するか否かを示すフラグが含まれている．

- Keep Alive
  付録の図 A.8 で示される Keep Alive は，CYPHONIC クラウドの各サービスと CYPHONIC ノード間で通信経路を維持するために用いるメッセージである．CYPHONIC では，CYPHONIC ノードが Keep Alive を定期的に送信することで，NAPT のマッピングテーブルのエントリを維持する．これにより，UDP Hole Punching と同様の効果を得る．一方で，モバイル端末ではエネルギー消費が激しくなる．

- Direction Request
  付録の図 A.9 で示される Direction Request は，CYPHONIC ノードが NMS に対して経路指示を依頼するためのメッセージである．Direction Request には，アプリケーションの識別子と Responder の FQDN が格納される．

- Route Direction
  付録の図 A.10 で示される Route Direction は，NMS が Initiator と Responder それぞれに対して，通信経路を指示するためのメッセージである．Route Direction には，通信経路を一意に示す PathID，相手端末のネットワーク環境情報と FQDN，通信経路情報が含まれる．また，End Key 交換用の暗号鍵である，Tunnel Key と Temporary Key が格納される．

- Relay Request
  付録の図 A.11 で示される Relay Request は，NMS が TRS に対して通信中継を依頼するためのメッセージである．Relay Request には，通信経路を一意に示す Path ID と Tunnel Request，Tunnel Response の暗号化通信に使用する Tunnel Key が含まれる．

- Relay Response
  付録の図 A.12 で示される Relay Response は，TRS が NMS に対して Relay Request の受理に成功したことを通知するためのメッセージである．NMS は，TRS から Relay Response を受信すると，TRS を経由する通信経路を CYPHONIC ノードへ指示する．

- Hole Punching

付録の図 A.13 で示される Hole Punching は，CYPHONIC ノードが TRS や相手端末に対して，自身宛メッセージの宛先情報を通知するメッセージである．Hole Punching は，通信において送信元となる CYPHONIC ノードが NAPT 配下に存在する場合に使用する．Hole Punching の受信者は，宛先情報としてメッセージの送信元アドレスを記録する．記録した送信元アドレスは，NAPT 変換後の情報であるため，Hole Punching メッセージを送信した際に作成されたマッピングテーブルと合わせて，NAPT 超えが可能となる．

- Tunnel Request
  付録の図 A.14 で示される Tunnel Request は，NMS からの経路指示に従い，Responder もしくは TRS に対してトンネル構築を依頼するメッセージである．Tunnel Request には，送信元を一意に示す Node ID と，後述の Capsule Message を暗号化するための End Key を格納する．

- Tunnel Response
  付録の図 A.15 で示される Tunnel Response は，Tunnel Request の受理に成功したことを通知するメッセージである．CYPHONIC ノードは，Responder からの Tunnel Response の受信を以て，End Key の交換を完了する．

- Capsule Message
  付録の図 A.16 で示される Capsule Message は，CYPHONIC 上で動作するアプリケーションのデータを送受信するためのメッセージである．Capsule Message の Payload フィールドには，仮想 IP パケットの IP ヘッダ，トランスポートヘッダ，通信データが格納される．オーバーレイネットワークは，実態を伴わない仮想的なネットワークであるため，Capsule Message を実 IP アドレスを使用した実 IP パケットにカプセル化することで，実際に物理ネットワークを流通することが可能となる．

3.5　CYPHONIC クラウドのシステムモデル

図 3.9 に CYPHONIC クラウドのシステムモデル図を示す．CYPHONIC クラウドを構成する各サービスは，CYPHONIC ノードや他のクラウドサービスからリクエスト

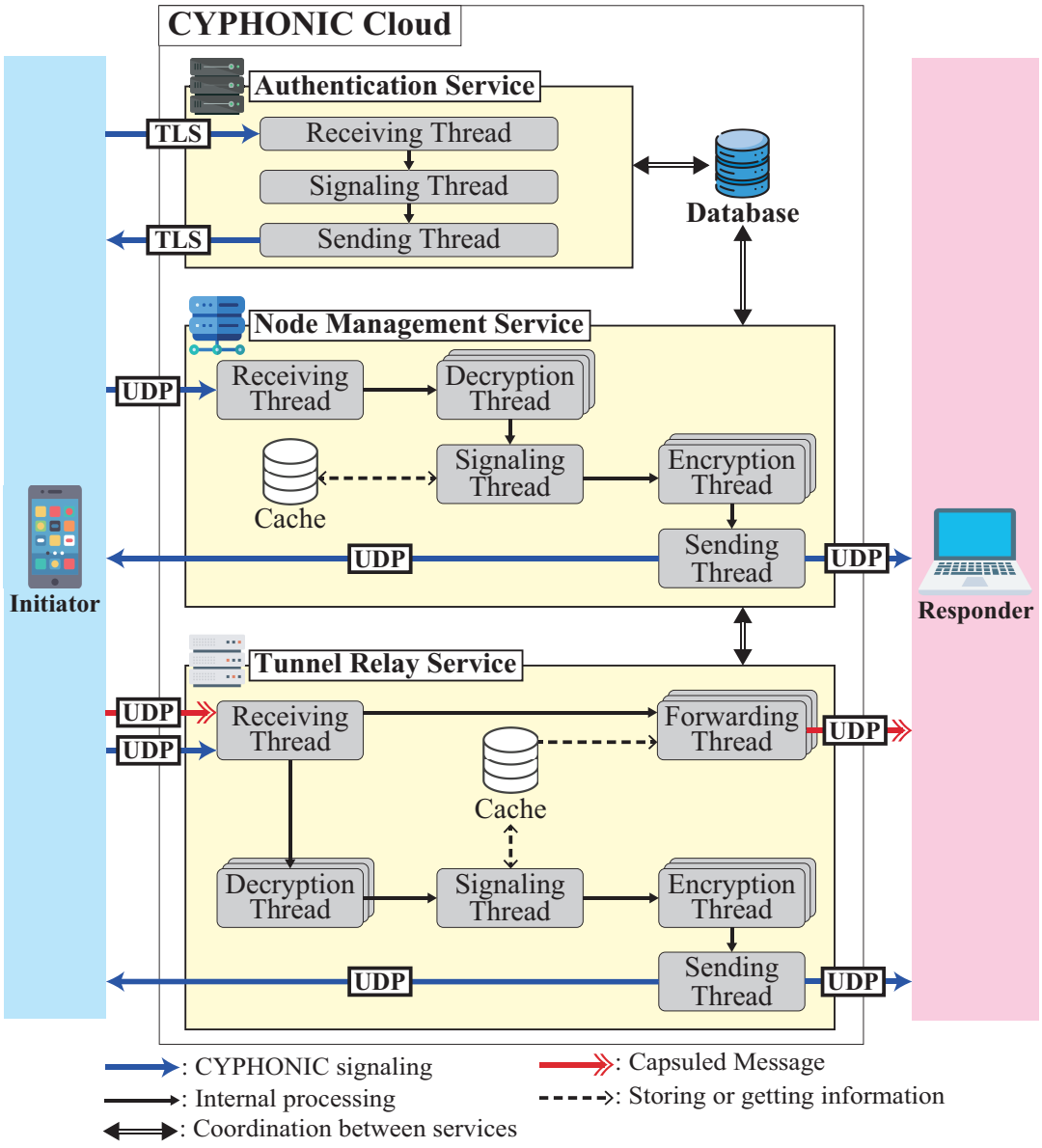


図 3.9 System model of CYPHONIC Cloud

を受け取り，処理を行う．CYPHONIC クラウドは，内部処理の種類ごとに専用のスレッドが生成される．イベント駆動型アーキテクチャを基に実装されており，Receving Thread でリクエストを受け取ると，他の様々なスレッドに処理を受け渡す．そのため，リクエストの受け取りと内部処理を分離可能である．これにより，CYPHONIC クラウドに対して大量のリクエストが想定される場合に，内部処理の遅延によって後続の通信がブロックされる事を防止する．

また，暗号化・復号処理を行う Encryption Thread/Decryption Thread，TRS において通信中継を行う Forwading Thread は，リクエスト数が増加すると処理のボトルネックとなる恐れがある．そのため，これらのスレッドは CPU のコア数に応じた個数のスレッドを多重に生成し，並列化による処理速度の向上を図る．

さらに，CYPHONIC クラウドではインメモリキャッシュを採用し，ステート情報を一時的に保存するデータ領域を利用する．スレッドがステート情報を保持していると，マルチスレッド環境では応答パケットを別のスレッドが処理する可能性があるため，ステート情報を損失する恐れがある．各スレッドに通信状態を保持させず，インメモリキャッシュに保持させることによって，マルチスレッド処理が維持される．

AS は，TLS を利用して通信するため，Encryption Thread/Decryption Thread は不要である．一方で，NMS と TRS は UDP による通信を行うため，Encryption Thread/Decryption Thread が必要である．以下に，CYPHONIC クラウドを構成するスレッドについて詳細に説明する．

- Receiving Thread

  Receiving Thread は，CYPHONIC ノードまたは他の CYPHONIC クラウドサービスからリクエストを受け取るスレッドである．Receiving Thread は，受信したリクエストを受信キューに渡す．Receiving Thread は，単一スレッドでリクエストを待機し，ノンブロッキング Input/Output (I/O) で受信したリクエストを処理する．そのため，リクエストの種類によって後続のリクエストをブロックするということがない．よって，待ち状態であるリクエストの発生で CYPHONIC クラウドサービスの処理パフォーマンスが劣化することはない．

- Sending Thread

  Sending Thread は，CYPHONIC ノードまたは他の CYPHONIC クラウドサービス

へレスポンスを送信するスレッドである．Sending Thread は，送信キューにレスポンスを格納し，送信キューから取り出した順序に従って送信処理を行う．Sending Thread も Receiving Thread と同様，ノンブロッキング I/O で動作し，単一スレッドでレスポンスを送信する．従って，TCP や UDP パケットの順序の整合性が保たれる．

- Encryption Thread
  Encryption Thread は，セキュア通信のための暗号化処理を行うスレッドである，起動する Encryption Thread の個数は，CPU のコア数に応じて変更され，並列処理が図られる．暗号化処理を並列化するために，送信キューに対して紐付いている暗号鍵で暗号化する．

- Decryption Thread
  Decryption Thread は，セキュア通信のための復号処理を行うスレッドである．起動する Decryption Thread の個数は，Encryption Thread と同様，CPU のコア数に応じて変更され，並列処理が図られる．復号処理を並列化するために，受信キューに対して紐付いている暗号鍵で復号する．Decryption Thread で復号されるパケットは Capsule Message 以外の CYPHONIC 制御用メッセージであるため，Signaling Thread へと渡される．

- Signaling Thread
  Signaling Thread は，CYPHONIC のシグナリング処理に基づいて，シグナリングパケットを生成するスレッドである．受信キューから Decryption Thread を介して復号されたパケット情報をもとに，Signaling Thread が返信パケットを生成し，送信キューに格納する．送信キューによって格納されたパケットは，Encryption Thread で暗号化され，Sending Thread によって送信される．

- Forwarding Thread
  Forwarding Thread は，Capsule Message を中継するスレッドである．Fording Thread は，通信を中継する際に，IP ヘッダを変更する．よって，通信相手に応じて IP バージョンを使い分けることにより，使用する IP バージョンが異なる CYPHONIC ノード間でも通信が実現される．そのため，Forwarding Thread は，デュアルスタック

ネットワークでの動作を前提としている．また，CYPHONIC ノードが Symmetric NAPT 配下に存在する場合でも，TRS からの通信は許容される特性から，Forwarding Thread による通信中継によって NAPT 超えが実現可能となる．

3.6　CYPHONIC ノードのシステムモデル

CYPHONIC ノードは，端末に CYPHONIC Daemon をインストールすることにより，CYPHONIC を利用した通信が可能となる．CYPHONIC Daemon とは，CYPHONIC の機能を提供する端末プログラムであり，CYPHONIC クラウドとの連携 (シグナリング) を通じて，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，オーバーレイネットワーク上での通信を行うために，仮想ネットワークインタフェースを生成する．仮想ネットワークインタフェースには，CYPHONIC クラウドから割り当てられた仮想 IP アドレスが付与される．CYPHONIC ノード上で動作するアプリケーションは，仮想ネットワークインタフェースを介して通信を行うことで，仮想 IP アドレスを使用したオーバーレイネットワーク上での通信が実施される．アプリケーションが生成する仮想 IP パケットは，そのままでは実ネットワーク上を流通することは不可能であるため，CYPHONIC Daemon によって制御情報の付加，実 IP パケットへのカプセル化を行うことによって，実ネットワークを流通可能となる．以下に，CYPHONIC Daemon が提供する機能の概要と，CYPHONIC ノードのオーバーレイネットワークを使用した通信について詳細に説明する．

3.6.1　CYPHONIC Daemon の概要

図 3.10 に CYPHONIC ノードのシステムモデル図を示す．CYPHONIC ノードでは，CYPHONIC Daemon が動作している必要がある．CYPHONIC Daemon は，CYPHONIC を利用した通信を実現する機能を提供する端末プログラムであり，CYPHONIC ノードのユーザ空間にバックグラウンドプロセスとして常駐する．CYPHONIC Daemon には，オーバーレイネットワークでの通信を実現するために，以下の機能が必要となる．

- CYPHONIC クラウドとシグナリングを実行する機能

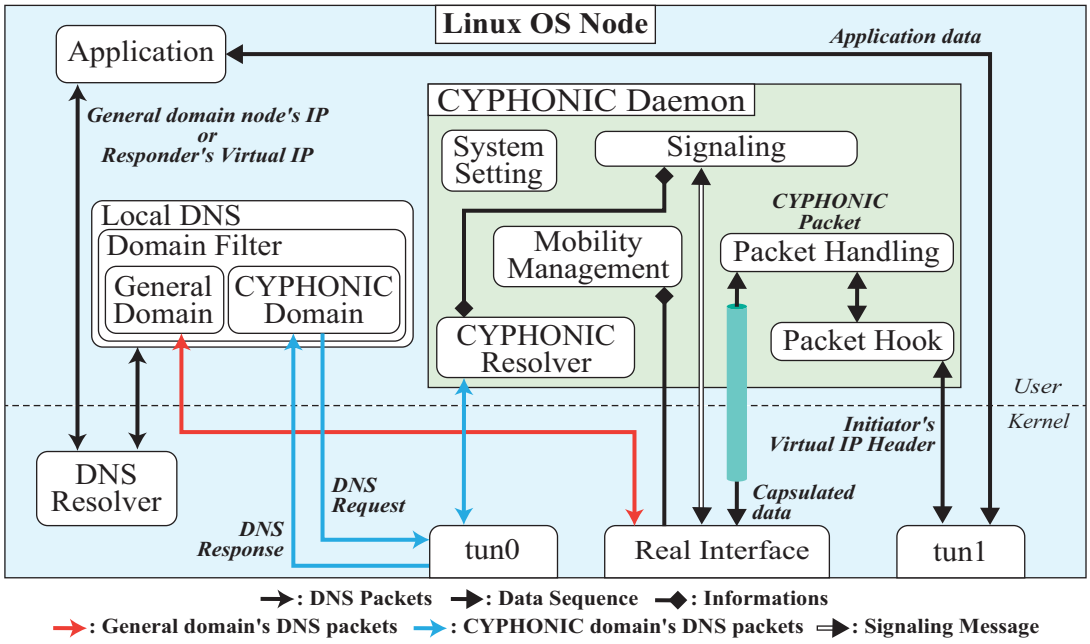


図 3.10 System model of CYPHONIC Daemon

- FQDN から仮想 IP アドレスの名前解決を行う DNS リゾルバの機能
- アプリケーションから仮想 IP アドレスを取得 (フック) する機能
- 仮想 IP パケットをカプセル/デカプセル化する機能
- 実ネットワークを介して相手ノードと Capsule Message を送受信する機能

また，CYPHONIC ノードには，オーバーレイネットワーク上で通信を行うために，仮想 IP アドレスが割り当てられたネットワークインタフェースが必要である．そのため，CYPHONIC Daemon は，TUN/TAP によって仮想ネットワークインタフェースを生成する．仮想ネットワークインタフェースとは，実際のネットワークインタフェースと同様に取扱い可能な，ソフトウェアで再現されたネットワークインタフェースである．CYPHONIC Daemon が生成した仮想ネットワークインタフェースには，仮想 IP アドレスが割り当てられ，アプリケーションと CYPHONIC Daemon の間で仮想 IP パケットを受け渡すために使用される．CYPHONIC Daemon は，仮想ネットワークインタフェースを介して，仮想 IP パケットの送受信を行うことで CYPHONIC クラウドや相手ノー

ドと通信を行う．以下に，CYPHONIC Daemon を構成するモジュールについて詳細に説明する．

- Signaling Module
  Signaling Module は，CYPHONIC クラウドとのシグナリング処理を実行するモジュールである．Signaling Module は，CYPHONIC Daemon を起動した際に最初に呼び出され，シグナリング用パケットの生成，仮想ネットワークインタフェースの生成，仮想ネットワークインタフェースへの仮想 IP アドレスの割り当てを行う．また，仮想 IP アドレスを使用した通信が仮想ネットワークインタフェースへルーティングされるように，ルーティングテーブルの設定を行う．

- Packet Hook Module
  Packet Hook Module は，仮想ネットワークインタフェースへ到着した仮想 IP パケットを取得するモジュールである．アプリケーションから取得した仮想 IP パケットは，後述の Packet Handling Module に受け渡され，カプセル化される．一方で，Packet Handling Module からデカプセル化された仮想 IP パケットを受け取った場合は，仮想ネットワークインタフェースを介してアプリケーションへ転送する．

- Packet Handling Module
  Packet Handling Module は，パケットの生成，カプセル化/デカプセル化，暗号化/復号を行うモジュールである．Packet Hook Module から取得した仮想 IP パケットをカプセル化，暗号化して実ネットワークへ転送する．また，実 IP パケットを受け取った際には，復号，デカプセル化して得られた仮想 IP パケットを Packet Hook Module に受け渡す．また，送信する CYPHONIC パケットには改ざん検知用の HMAC が付与される．受信側は，受信パケットに付与された HMAC と受信パケットから計算した値を照合し，パケットの改ざん検知を行う．これにより，データの完全性を保証した通信が実現される．

- CYPHONIC Resolver Module
  CYPHONIC Resolver Module は，DNS パケットを処理するモジュールである．CYPHONIC ノードは，CYPHONIC クラウドとのシグナリングによって FQDN が割り当てられ，仮想 IP アドレスと紐付けられる．アプリケーションは，相手ノー

ドを FQDN を用いて指定するため，FQDN から仮想 IP アドレスを名前解決する仕組みが必要である．CYPHONIC Resolver Module は，アプリケーションからの FQDN の名前解決を求める DNS Request を生成し，Signaling Module へ受け渡す．Signaling Module からシグナリングを介して仮想 IP アドレスを取得すると，仮想 IP アドレスを名前解決結果とする DNS Response を生成し，仮想ネットワークインタフェースを介してアプリケーションへと伝達する．

- System Setting Module
  System Setting Module は，設定ファイルから CYPHONIC の通信に必要な諸情報を CYPHONIC Daemon 全体に適用するモジュールである．設定ファイルには，仮想ネットワークインタフェースの設定や，クラウドサービスとの通信に用いるポート番号，AS での端末認証方法が記述されている．

- Mobility Management Module
  Mobility Management Module は，移動透過性を実現するためのモジュールである．Mobility Management Module は，実 IP アドレスの変更を監視しており，実 IP アドレスが変更された際には Signaling Module へ通知する．Signaling Module から NMS に対して位置情報登録以降の処理を再度行うことにより，NMS が保持する端末のネットワーク情報の更新やトンネルの再構築を行う．

3.6.2　オーバーレイネットワークでの通信

図 3.11 に，オーバーレイネットワーク通信の DNS パケットフローを示す．CYPHONIC Daemon は，仮想ネットワークインタフェースとして，アプリケーションと仮想 IP パケットをやり取りするための tun1 と，アプリケーションと DNS パケットのやり取りを行う tun0 を生成する．アプリケーションは，通信相手を CYPHONIC で使用する FQDN で指定する．CYPHONIC ノードが名前解決を行う際には，通信相手の FQDN から仮想 IP アドレスを取得したい場合と，一般的なドメイン名から実 IP アドレスを取得したい場合の二種類が存在する．そのため，CYPHONIC ノード上で CYPHONIC の FQDN と一般的なドメイン名を振り分ける DNS リゾルバが必要である．Local DNS は，CYPHONIC の FQDN と一般的なドメイン名を振り分け，CYPHONIC の FQDN の

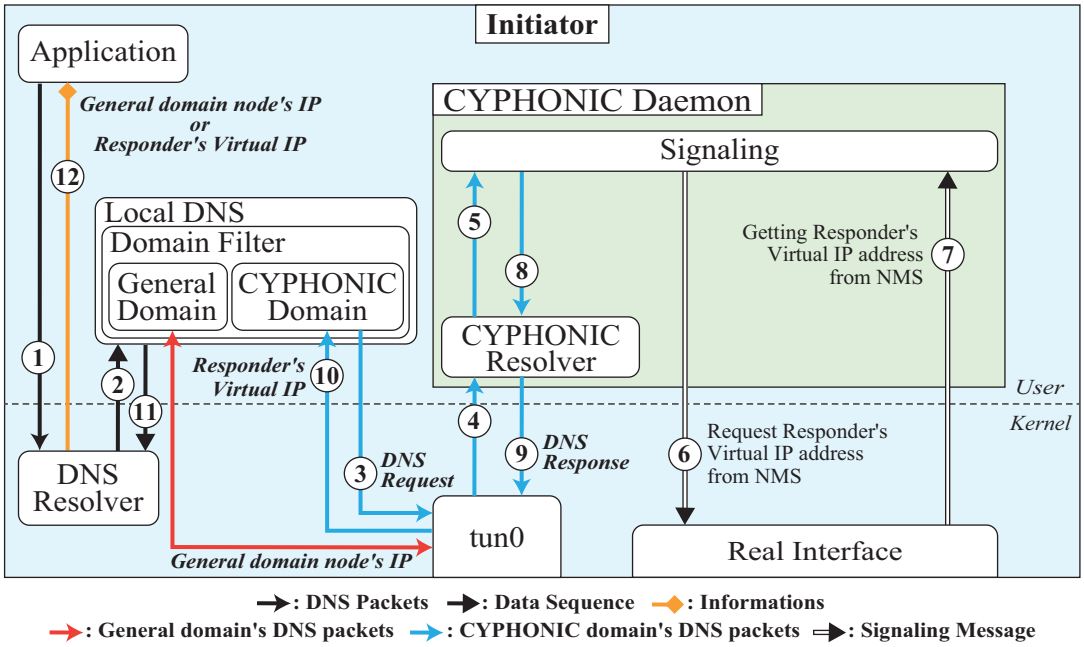


図 3.11 DNS Packet Flow when using CYPHONIC

DNS Request のみを仮想ネットワークインタフェースの tun0 へと転送する．CYPHONIC Resolver Module は，DNS 処理専用の仮想ネットワークインタフェースである tun0 から DNS パケットを取得し，Signaling Module に対して名前解決を依頼する．Signaling Module は，NMS に対して通信相手の FQDN と共に Direction Request メッセージを送信する．NMS から Route Direction to Initiator メッセージが返されると，メッセージから通信相手の仮想 IP アドレスを取得可能となる．Signaling Module は，Route Direction to Initiator から仮想 IP アドレスを取得し，CYPHONIC Resolver Module へと引き渡す．CYPHONIC Resolver Module は，受け取った仮想 IP アドレスを応答する DNS Response を生成し，tun0 を介してアプリケーションへ名前解決結果として転送する．

図 3.12 にオーバーレイネットワーク通信のパケットフローを示す．CYPHONIC ノード上で動作しているアプリケーションは，FQDN の名前解決を経て仮想 IP アドレスを取得すると，その後のアプリケーションデータの宛先に相手ノードの仮想 IP アドレスを指定して，ソケット通信を行う．また，CYPHONIC Daemon は，デフォルトルートを仮想ネットワークインタフェースである tun1 に設定する．これにより，アプリケーションは送信元と宛先の双方が仮想 IP アドレスを指定した，仮想 IP パケットでの通

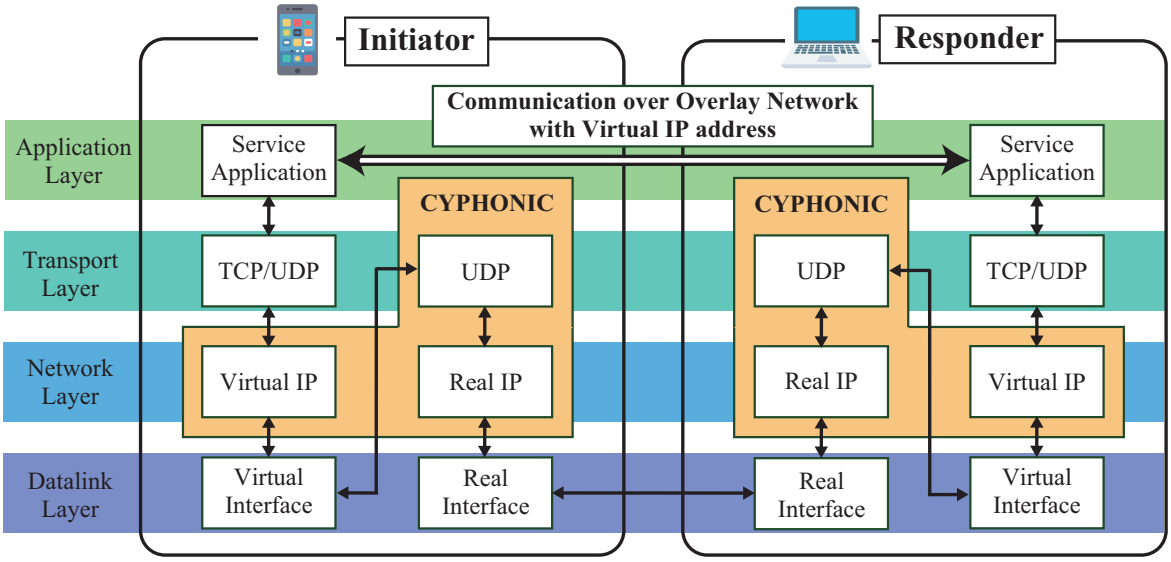


図 3.12 Protocol layer model when using CYPHONIC

信となる．仮想 IP パケットが，仮想ネットワークインタフェースである tun1 に送信されると，Packet Hook Module によって CYPHONIC Daemon に送られる．Packet Hokk Module によって取得された仮想 IP パケットは，Packet Handling Module によって暗号化等の処理が実行される．CYPHONIC は，実ネットワークにおいて送受信するパケットは UDP パケットであるため，生成した仮想 IP パケットに対して UDP ヘッダを付加し，CYPHONIC ノードの実 IP アドレスでカプセル化を行う．その後，生成された UDP パケットは CYPHONIC ノードの実ネットワークインタフェースを介して実ネットワークへと送信される．実ネットワークインタフェースから送信された通信パケットは，事前にシグナリングによって相手ノードとの間に構築した UDP トンネルを用いて，相手ノードへ届けられる．相手ノードが通信パケットを受信した際には，実 IP ヘッダと UDP ヘッダがデカプセル化され，CYPHONIC パケットとなる．CYPHONIC パケットは Packet Handling Module によって復号，デカプセル化処理等がなされ，仮想 IP パケットが取り出される．仮想 IP パケットは Packet Hook Module によって取得され，仮想ネットワークインタフェース tun0 を介してアプリケーションへと届けられる．このような処理の繰り返しによって，オーバーレイネットワークでの通信が実現する．

第4章

簡易パケット認証手法の提案

4.1 概要

本章では，CYPHONIC クラウドをリプレイ攻撃や DoS 攻撃から保護するための簡易パケット認証手法を提案する．CYPHONIC の通信では，端末間の直接通信を開始する事前準備として，CYPHONIC クラウドとのシグナリングを実行する．しかし，サーバの働きをする CYPHONIC クラウドはサイバー攻撃の標的となる恐れがある．リプレイ攻撃によるサーバに対する不正な操作や，DoS 攻撃によるサーバダウンが発生すると，サービス提供不能になる危険を伴う．CYPHONIC クラウドが，リプレイ攻撃や DoS 攻撃の被害を受け，サービス提供不能な状態に陥ると，その後の端末間直接通信の実現が不可能となる．

CYPHONIC クラウドが攻撃の被害を回避するためには，CYPHONIC クラウドが受信するパケットを認証する仕組みが必要である．CYPHONIC において，リプレイ攻撃や DoS 攻撃は，CYPHONIC 独自のパケットを再利用した場合に生じると考えられるため，正規のパケットに有効期限を設定することで正規のパケットと不正なパケットの判断を行う．有効期限を超過したパケットは不正なパケットと判断しパケットを破棄することで，CYPHONIC クラウドの継続稼働を可能にする．

したがって，本研究では，CYPHONIC クラウドをリプレイ攻撃や DoS 攻撃から保護するための簡易パケット認証手法を提案する．

4.2　簡易パケット認証が必要な背景

CYPHONIC は通信する際に，CYPHONIC ノードと CYPHONIC クラウドの通信が必要不可欠である．CYPHONIC の通信では，端末間の直接通信を開始する事前準備として CYPHONIC クラウドとのシグナリングを実行する．しかし，サーバを用いたシステムは，サイバー攻撃の標的となる傾向があり，サーバの働きをする CYPHONIC クラウドは攻撃される危険を伴う．

シグナリングに使用される CYPHONIC パケットは，CYPHONIC が独自に定義したパケット構造のパケットである．そのため，攻撃者にネットワーク上を流れる CYPHONIC パケットをキャプチャされ，DoS 攻撃に利用することやキャプチャしたパケットを書き換えて再送信するリプレイ攻撃に利用して，CYPHONIC クラウドを攻撃される懸念が存在する．CYPHONIC クラウドがリプレイ攻撃や DoS 攻撃の被害を被ると，CYPHONIC クラウド内部の不当な情報変更や大量の不正パケットの処理によって負荷が集中するリスクがある．そのような事態になると，CYPHONIC クラウドの処理遅延やサーバダウンが発生し，サービス提供不能な状態によって端末間直接通信の実現が不可能となる．

攻撃者によって送信された不正なパケットを CYPHONIC クラウドが処理回避するためには，CYPHONIC クラウドが受信するパケットを認証する仕組みが必要である．加えて，不正なパケットには正規のパケットを再送信した遅延パケットが想定されるため，時間情報を利用したパケット認証手法が求められる．

4.3　簡易パケット認証手法の提案

図 4.1 に提案する簡易パケット認証のシステム概要図を示す．簡易パケット認証では，送信される CYPHONIC パケットに印 (Token) を付与し，CYPHONIC クラウドが受信したパケットに付与された Token を検証することによってパケットの正当性を判断する．不正なパケットには，攻撃者がネットワーク上を流れる CYPHONIC 独自のパケットをキャプチャし，再送信した遅延パケットが想定される．そのため，時間情報を利用して，正規のパケットが送信されてから CYPHONIC クラウドが受信するまでの一定時間のみパケットを受理する仕組みが必要である．

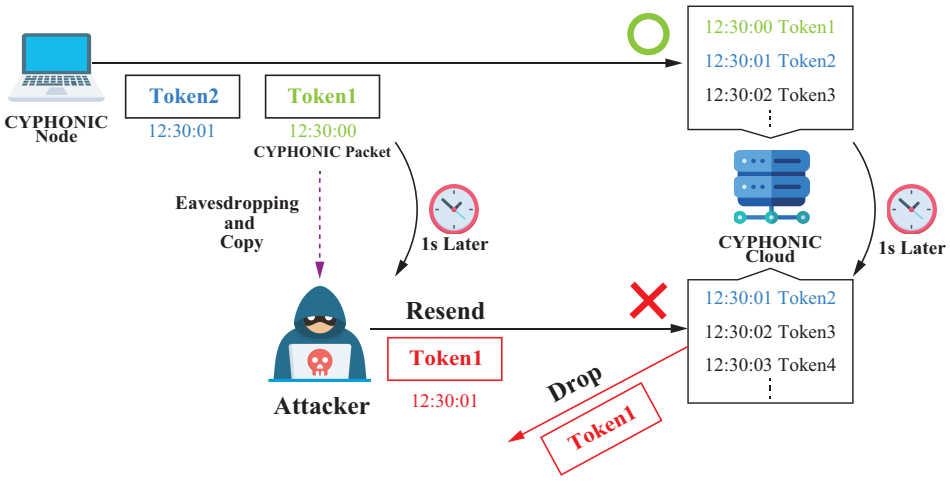


図 4.1 System overview of Simple Packet Authentication

そのため，パケットを送受信する双方で時間を同期し，一定時間毎に特定のルールで送信パケットに印 (Token) を付与することによって，受信側で Token が付与されたパケットの正当性を判断する．まず，CYPHONIC ノードと CYPHONIC クラウドの双方で時刻を同期する．この際，世界の標準時刻の基準である協定世界時 (UTC) を用いて時刻を同期するため，パケットを送受信する双方で同一な時間情報の扱いが可能となる．次に，パケットを送信する CYPHONIC ノードは，一定時間毎に特定のルールで送信パケットに Token を付与し，CYPHONIC クラウドに対して送信する．パケットを受信した CYPHONIC クラウドは，自身が承認する Token と比較し，値が一致すればパケットを受理，一致しなければパケットを破棄する．この際，CYPHONIC クラウドは，CYPHONIC ノードと同一の方法で検証用の Token を毎秒計算する．パケットの送信には，通常，良好なネットワーク環境であれば，1 秒未満であるため，秒単位で Token を更新することにより，正規のパケットを CYPHONIC クラウドで受理可能となる．

また，提案手法は運用面を考慮し，正当なパケットの判断にデータベースへの問い合わせやパケットの復号等の処理を必要としないため，CYPHONIC クラウドへの負荷を低減した迅速なパケット認証が実現可能となる．厳密なパケットの認証には，正当なパケットの判断にデータベースへの問い合わせやパケットの復号等の処理が必要である．しかし，DoS 攻撃等によって，不正なパケットを大量に受信した場合に，全

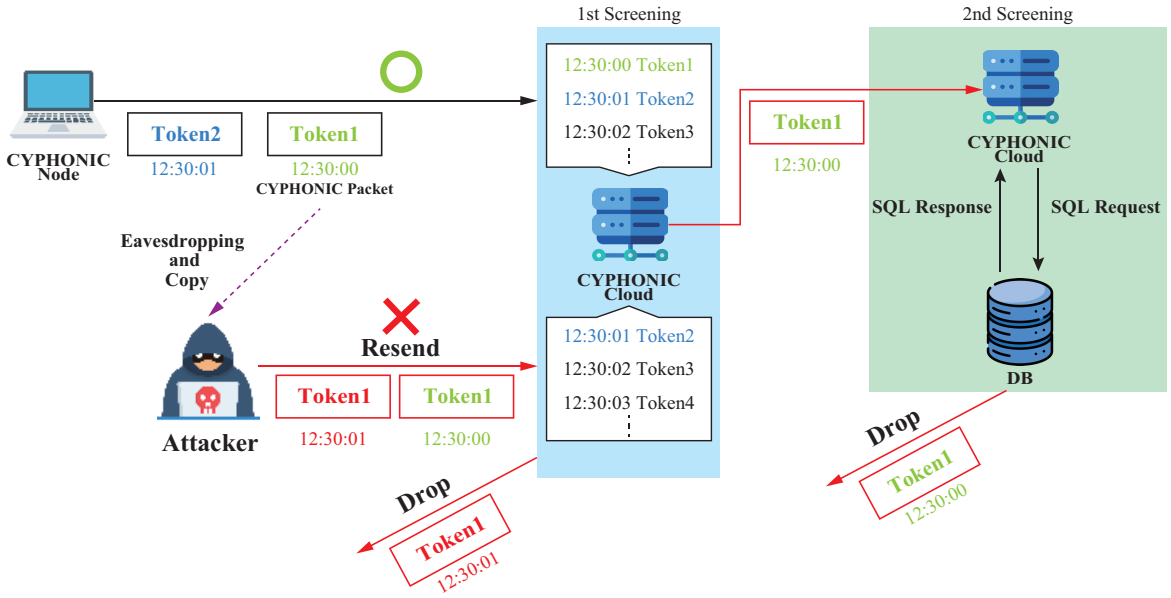


図 4.2 Positioning of Simple Packet Authentication

てのパケットに対しデータベースへの問い合わせやパケットの復号等の処理を実施すると，CYPHONIC クラウドの処理負荷や処理時間が増大し，サービスを継続して提供不可能になる．そのため，簡易パケット認証では，データベースアクセスや復号処理等の負荷の高い処理を極力避けることにより，CYPHONIC クラウドへの負荷を低減した迅速なパケット認証を実現する．

一方で，不正パケットの遅延時間が極めて短い場合は，不正パケットが簡易パケット認証を通過してしまう恐れがある．その場合は，簡易パケット認証後にデータベース等を用いた厳密なパケット認証を実施する．しかし，本研究では，CYPHONIC クラウドがサービスの提供に支障をきたさない範囲でパケットを破棄可能となることが目的であるため，不正パケットの一次選別を行う．本研究において提案する簡易パケット認証の位置付けを図 4.2 に示す．

第5章

簡易パケット認証手法の実装

5.1 概要

本章では，CYPHONIC クラウドをリプレイ攻撃や DoS 攻撃から保護するための簡易パケット認証手法の実装について述べる．簡易パケット認証手法の実装において，CYPHONIC パケットを生成する CYPHONIC ノードと，パケットを検証する CYPHONIC クラウドに対する実装が必要である．CYPHONIC ノードには，生成した CYPHONIC パケットにパケット認証用の値 (Token) を計算し，付与する機能を追加する．また，CYPHONIC クラウドの AS には Token の生成に必要な CYPHONIC 内で共有する値 (SecretCommonValue) の生成と CYPHONIC ノードおよび他の CYPHONIC クラウドへ共有する方法の追加を行う．さらに，NMS，TRS には受信した CYPHONIC パケットに付与された Token を検証して正規のパケットか否かを判断するパケット認証の機能を追加する．AS との通信は TLS 通信であり，セッションの確立がなければ通信が成立しないため，UDP 通信を行う NMS と TRS に簡易パケット認証機能を追加する．

5.2 簡易パケット認証手法の実装

提案する簡易パケット認証手法の実装では，大きく以下の五つの事柄の実装が必要である．

- 時間の同期
- SecretCommonValue の生成と共有

- CYPHONIC パケットに付与する Token の生成
- パケットを認証するための検証用 Token の生成
- パケットの認証機能

5.2.1　時間の同期

簡易パケット認証では，時間情報を用いてパケットに有効期限を設定するため，パケットを送受信する CYPHONIC ノードと CYPHONIC クラウドで時刻を同期する必要がある．時刻の同期には，世界の標準時刻の基準である協定世界時 (UTC) を使用する．パケットの送信から受信までに要する時間は，一般的に良好なネットワーク環境であれば 1 秒未満であるため，秒単位で時刻を同期することによって，CYPHONIC ノードと CYPHONIC クラウドは同一の時間情報を扱うことが可能となる．

5.2.2　SecretCommonValue の生成と共有

送信する CYPHONIC パケットには，同期した時間情報を用いて Token を付与する．しかし，時刻情報のみでは Token を安易に推測される懸念があるため，CYPHONIC ノードと CYPHONIC クラウドで共通な値 (SecretCommonValue) を共有する．そして，共有された SecretCommonValue と時間情報を合わせて Token を計算する．

SecretCommonValue は，AS がランダムに生成する文字列である．AS は，起動した直後に SecretCommonValue を生成し，CYPHONIC ノードと CYPHONIC クラウドに値を共有する．AS は，SecretCommonValue の生成後，CYPHONIC クラウドに値を共有するため，データベースに値を保存する．CYPHONIC クラウドの AS, NMS, TRS は，データベースに保存された SecretCommonValue を取得することによって，SecretCommonValue を共有する．

一方で，CYPHONIC は，クラウドサービスの処理負荷に対応するため水平スケーリングが想定されている．よって，複数の AS インスタンスが異なる SecretCommonValue を生成し，同時にデータベースへ格納することが起こり得る．異なる SecretCommonValue が同時にデータベースに格納された場合，初めに格納された値は後に格納された値に上書きされるため，初めに格納を行った AS インスタンスが持つ値は無効となる．

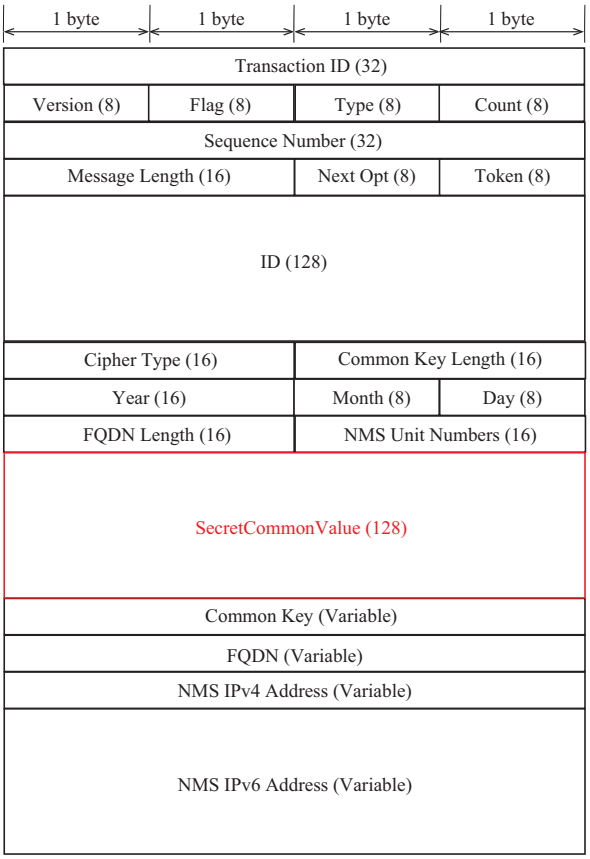


図 5.1 Extended-LoginResponse

このような状況を回避し，確実に CYPHONIC クラウドの全てのインスタンスが同一の SecretCommonValue を保持するように，AS は Secret CommonValue をデータベースに格納した後，データベースに格納された SecretCommonValue を取得する．

また，CYPHONIC ノードへの SecretCommonValue の共有は，端末認証時に行う．AS での端末認証が完了すると，認証応答パケットの中に SecretCommonValue を含めて CYPHONIC ノードに SecretCommonValue を配布する．認証応答パケットには新しい情報を載せる必要があるため，パケットを拡張した．拡張した認証応答パケットを図 5.1 に示す．SecretCommonValue は攻撃者に値を推測されることを防止するため，データ長は 128bit とした．以降，CYPHONIC ノードが CYPHONIC パケットを生成する際には，認証応答パケットから取得した SecretCommonValue を用いて Token を計算し，パケットに付与する．

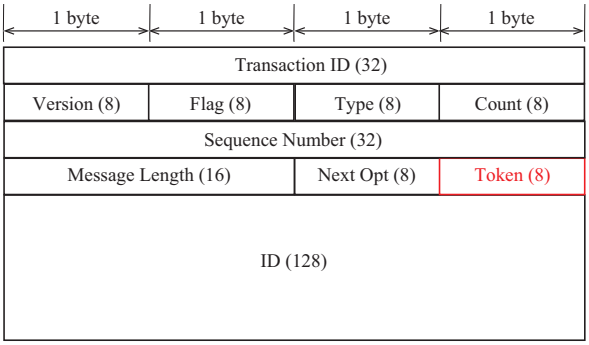


図 5.2 Extended-BaseHeader

5.2.3　CYPHONIC パケットに付与する Token の生成

NMS，TRS では受信したパケットが正規のパケットか否かを判断するパケット認証が行われるため，送信する CYPHONIC パケットに認証用の値 (Token) を付与する必要がある．Token は，CYPHONIC パケット生成時のタイムスタンプと取得した SecretCommonValue を連結し，ハッシュ化させて生成する．ハッシュ関数には，軽量ハッシュアルゴリズムである FNV ハッシュ関数を使用した．FNV ハッシュ関数は，一般的にはセキュリティ用途に不向きとされているが，今回の提案は CYPHONIC クラウドがサービス提供不能に陥らない程度のパケットを破棄可能となれば良いため，極力 CYPHONIC クラウドに負荷がかからない軽量ハッシュ関数を採用した．

また，生成された Token は CYPHONIC パケットの BaseHeader に付与する．格納箇所は，従来の CYPHONIC の BaseHeader では Reserved フィールドとして予約されていたフィールドを使用する．変更した BaseHeader を図 5.2 に示す．8bit のフィールドを利用する理由は，ヘッダ長が長くなるとスループットが低下するためである．一方で，8bit で表現可能なデータ数は 256 通りであり，衝突可能性が非常に高い懸念がある．しかし，CYPHONIC クラウドがパケット認証用に保持する Token の有効時間は 1 秒程度であるため，多大な影響はないと考えられる．また，ヘッダは暗号化されないため，CYPHONIC クラウドがパケットを受信した際，復号することなくパケット認証が実施可能となる．これにより，高速で CYPHONIC クラウドに負荷をかけないパケット認証処理が実現される．

5.2.4　パケットを認証するための検証用 Token の生成

受信した CYPHONIC パケットを認証するために，NMS と TRS では検証用の Token を生成する．検証用 Token は CYPHONIC パケットに付与された Token と一致する必要があるため，パケットに付与する Token と同一の方法で生成する．

また，パケット認証において，CYPHONIC クラウドはリプレイされた不正な遅延パケットを破棄し，その時点で有効な正規のパケットを処理する必要がある．そのため，Token は時間変化することが望ましい．さらに，パケットの送信から受信までの時間は基本的に 1 秒未満であり，CYPHONIC クラウドに負荷がかかり過ぎない計算量を考えると，Token は毎秒単位で変化させることが妥当である．よって，CYPHONIC クラウドの起動後に検証用の Token を毎秒計算する処理を行う．検証用 Token の生成は，毎秒の計算処理のためにその他のパケットに対する処理が滞ることを防ぐため，gorutine を用いた並行処理を行う．これにより，検証用 Token を毎秒計算しつつ，他のパケットに対する処理を実行することが可能となる．

5.2.5　パケットの認証機能

受信したパケットの認証を行うには，受信パケットに付与された Token と CYPHONIC クラウドが保持している，パケット認証時に有効な Token の比較を行う．CYPHONIC クラウドが保持しているその時点で有効な Token はキューに格納する．有効な Token は毎秒更新されるため，新たな Token をキューに追加するとキューに存在する最も古い Token を削除する．

CYPHONIC クラウドがパケットを受信すると，一番初めにパケットが正規のものであるかを判断するパケットの認証が行われる．パケットを受信すると，CYPHONIC クラウドは検証用 Token を計算しているスレッドから現在有効な Token を保持しているキューを参照する．そして，キューに格納されている Token が受信パケットに付与された Token と一致するかを確認する．一致していれば，正規のパケットであると判断し，その後のパケットの処理を行う関数を呼び出す．一致しなければ，リプレイされた不正な遅延パケットであると見做し，パケットを破棄する．パケットの破棄は，その後のパケットを処理する関数を呼び出さないことで実現される．最後に簡易パケット認証手法の実装の全体像を図 5.3 に示す．

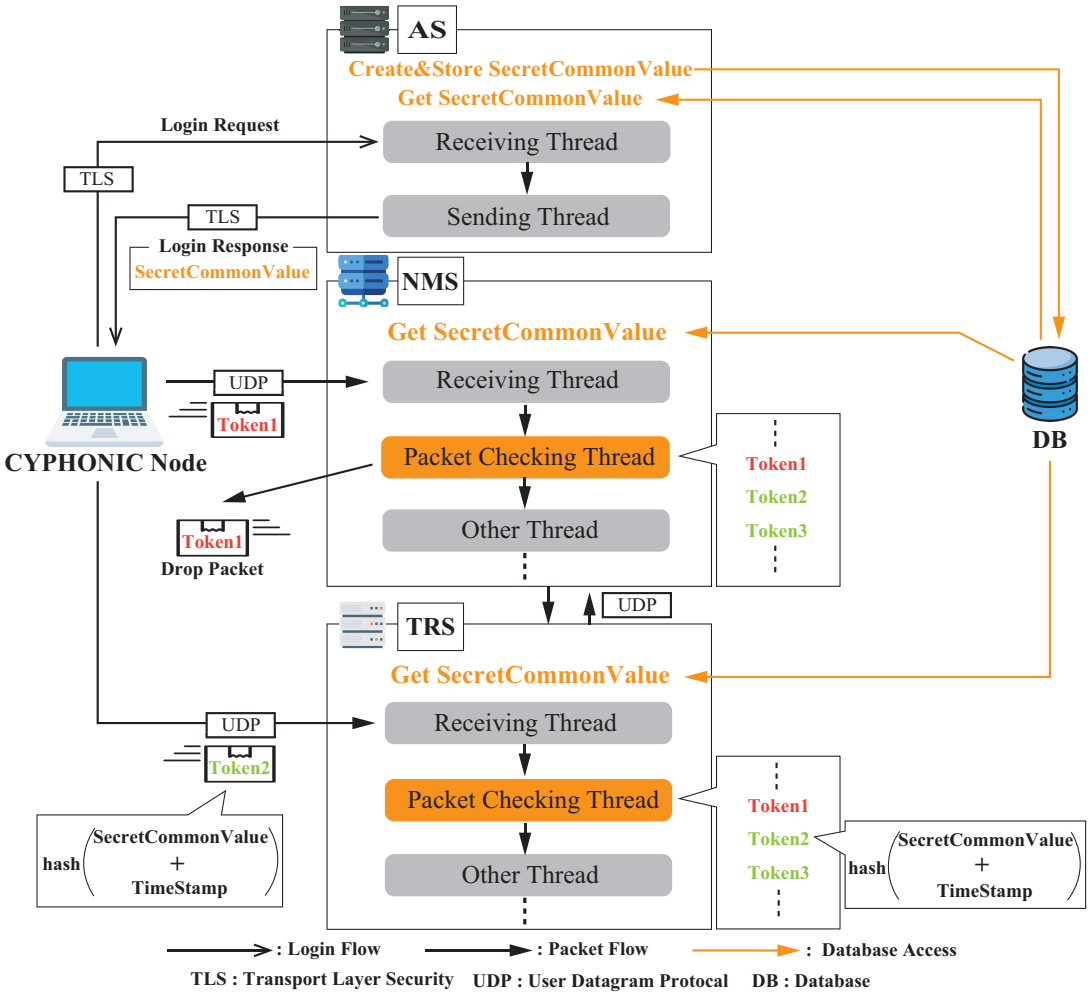


図 5.3 Overview of Packet Authentication Implementation

第6章

検証および評価

6.1 概要

本章では，実装した簡易パケット認証機能に関する動作検証および性能評価について述べる．本研究では，攻撃者が送信した不正なCYPHONICパケットを，受信側であるCYPHONICクラウドがサービス提供不能にならない程度に破棄することが重要である．CYPHONICクラウドを攻撃されサービス提供が不能になると，その後の通信サービスが提供不可能となるためである．動作検証では，CYPHONICの一連の通信を通して，NMSやTRSが受信するパケットが認証された後に処理されていることを確認する．

一方，CYPHONICクラウドは，リプレイ攻撃やDoS攻撃の標的となる恐れがあり，CYPHONIC独自のパケットを不正に利用することで，CYPHONICクラウドは危害を加えられる可能性がある．不正なCYPHONICパケットとは，正規のCYPHONICノードが送信したCYPHONICパケットをキャプチャし，手を加えて再送信されたパケットである．不正なCYPHONICパケットを利用すると，CYPHONICクラウドに対する不正な操作を試みるリプレイ攻撃やリプレイパケットを利用したDoS攻撃が実行され，CYPHONICクラウドがサービス提供不能になるリスクが存在する．そのため，CYPHONICクラウドがリプレイ攻撃やDoS攻撃の被害を受け，サービス提供不能にならない程度に不正なCYPHONICパケットを破棄することが求められる．再送信された不正なCYPHONICパケットは正規のCYPHONICパケットと比較して遅延する確率が高いと推測されるため，CYPHONICクラウドではどれほど遅延したCYPHONICパケットであれば破棄

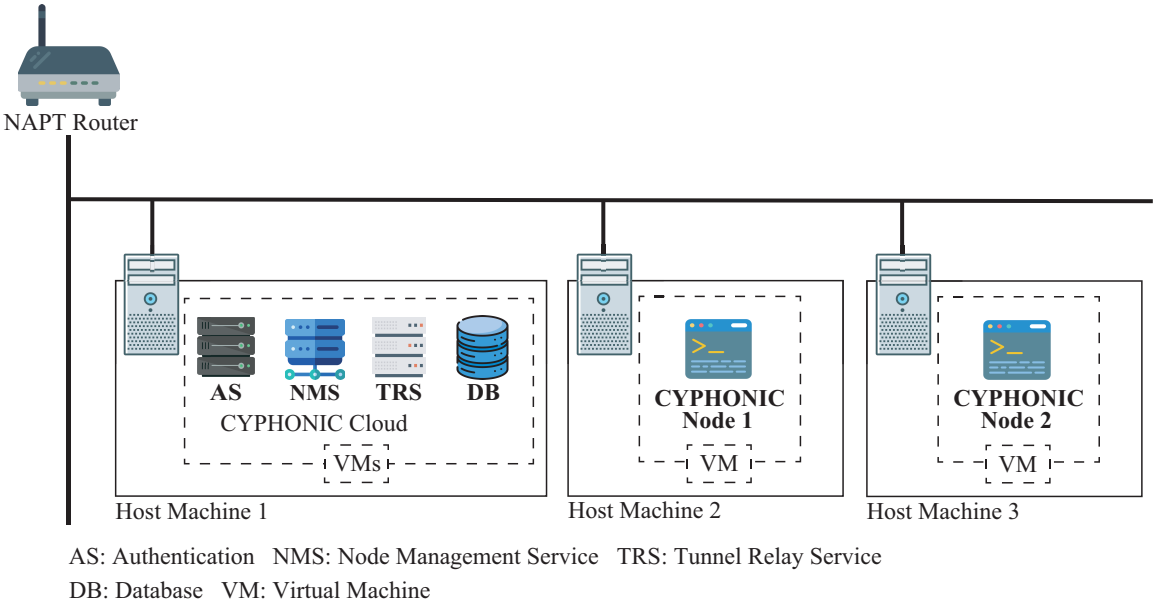


図 6.1 Operation verification environment

可能であるかという性能評価が必要である．したがって，CYPHONIC クラウドが受信する CYPHONIC パケットの遅延時間に対する破棄率を検証および評価する．また，性能評価によって，攻撃者の攻撃時間を考察する．

6.2　動作検証環境

簡易パケット認証機能の動作検証を実施する環境について，構成を図 6.1 に諸元を表 6.1 に示す．検証において，CYPHONIC クラウド，CYPHONIC ノード共に仮想マシン上にデプロイした．仮想マシンはホストマシンを介した通信により NAPT が適用されるため，CYPHONIC ノードが NAPT 配下に存在する状況と同等になる．そのため，本検証環境では，CYPHONIC ノード間の通信は TRS による中継が実行されるため，CYPHONIC の一連の通信を通して，NMS，TRS の両方に対する簡易パケット認証の動作検証を実施することが可能である．

表 6.1 Computer specifications for verification

Host Machine	
OS	Ubuntu 22.04
CPU	Core(TM) i9-13900 @ 5.60GHz 24cores, 48threads
Memory	128GB RAM
Virtual Machine (KVM)	
OS	Ubuntu 22.04
CPU	Core(TM) i9-13900 @ 5.60GHz 2cores, 2threads
Memory	2GB RAM

6.3　動作検証

動作検証では，NMS と TRS に実装した簡易パケット認証機能が正常に機能していることを確認した．CYPHONIC クラウドを起動すると簡易パケット認証に使用される検証用 Token の計算が開始される．そして，CYPHONIC ノードが送信する CYPHONIC パケットには認証に必要な Token が付与されている．そのため，CYPHONIC の一連の通信を実施することで簡易パケット認証が機能する．簡易パケット認証機能を導入して CYPHONIC の一連の通信を行ったところ，正常に CYPHONIC ノード間の通信が実行された．これは，CYPHONIC ノード間の通信が行われる以前の CYPHONIC クラウドとのシグナリングにおいて，パケットがパケット認証を正常に通過したことを意味する．また，CYPHONIC パケットに不適切な Token を付与した場合，そのパケットが簡易パケット認証によって破棄されることも確認した．

6.4　性能評価環境

簡易パケット認証機能の性能評価を実施する環境は，6.2 節の動作検証環境の一部を使用する．性能評価環境を図 6.2 に示す．遅延した CYPHONIC パケットに対する CYPHONIC クラウドの破棄率の測定には，NMS と TRS で同じため，本検証では，NMS と CYPHONIC ノード一台を用いて性能を検証および評価する．

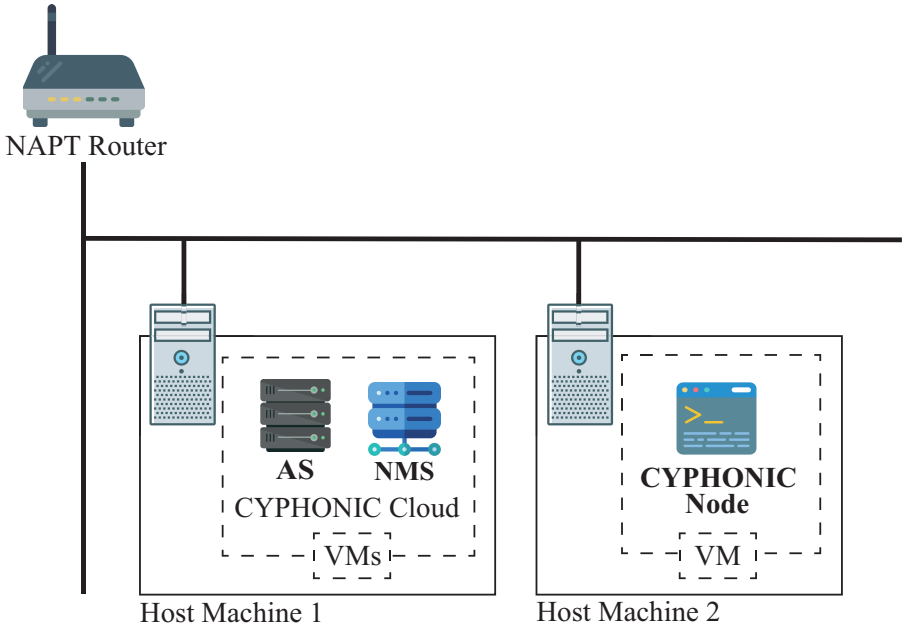


図 6.2 Performance evaluation environment

6.5　性能評価

性能評価では，CYPHONIC クラウドが受信する CYPHONIC パケットの遅延時間に対する破棄率を検証および評価する．性能評価は，有線環境と無線環境を想定し，二つの性能評価を実施した．有線環境を想定した場合，パケットには一般的なネットワーク遅延である Ping 値 50ms，Jitter 値 20ms の遅延をかけて検証を実施した．一方で，無線環境を想定した場合は，パケットに Ping 値 150ms，Jitter 値 100ms の遅延をかけて検証を実施した．また，検証は，信頼区間 95%，許容誤差 5%の場合の破棄率を算出するため 400 回試行した．400 回という試行回数は，以下の式を考慮した回数である．

確率 p を推定するために必要な試行回数 N を求めるためには、以下の式を使用する．

$$N = \frac{Z^2 \cdot p \cdot (1-p)}{E^2}$$

Z は信頼区間に対応する値である．例えば，95%の信頼区間なら $Z = 1.96$ である．また，p は予想される確率で，パケットの破棄率が p である．E は許容誤差であり，±5%の誤差を許容する場合は，$E = 0.05$ となる．

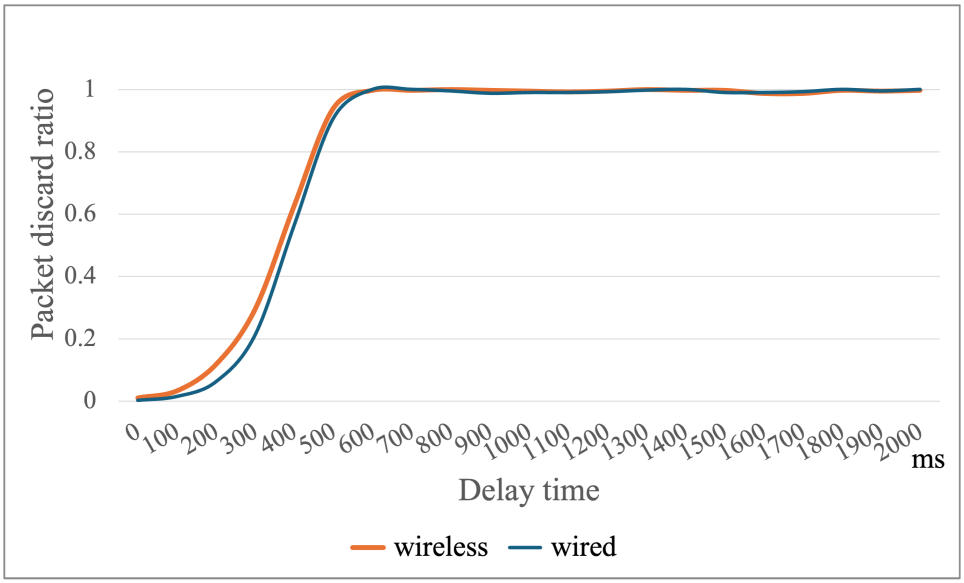


図 6.3 Packet discard ratio by Delay time

今回の性能評価の場合，遅延時間において，遅延した不正な CYPHONIC パケットが CYPHONIC クラウドで破棄されるか否かの確率は不明なため，最大の不確実性を考慮して，50% ($p = 0.5$) とする．これにより，最も保守的な試行回数を算出することが可能であり，算出した回数以上の試行回数であれば信頼区間 95%，許容誤差 5%の場合の破棄率に近づくこととなる．

条件を代入すると以下のようになる．

$$N = \frac{1.96^2 \cdot 0.5 \cdot 0.5}{0.05^2}$$

$$N = \frac{3.8416 \cdot 0.25}{0.0025} = 384.16$$

よって，パケットの破棄率 p が不明な場合，最低でも 384 回以上の試行回数であれば，95%の信頼区間，5%の許容誤差のパケット破棄率 p が算出される．

性能評価の結果を図 6.3 に示す．図は，パケットの遅延時間を 100ms 間隔で変化させた時の CYPHONIC クラウドのパケット破棄率を表している．結果のグラフによると，有線無線共に概ねパケットに 200ms の遅延をかけた時に破棄率は増加し始め，600ms 以上の遅延でほぼ全てのパケットが破棄されることが確認された．また，無線の場合

は，Ping 値や Jitter 値が高いため，短い遅延時間のパケット破棄率が増加した．加えて，この結果から，CYPHONIC クラウドに対する攻撃者の攻撃可能時間は 1 秒未満であると結論付けられる．攻撃可能時間が 1 秒程度であれば，CYPHONIC クラウドはサービス提供を継続することが可能であると推測される．

第7章

総括

7.1 本研究のまとめ

本研究では，CYPHONIC クラウドをリプレイ攻撃や DoS 攻撃から保護し，セキュリティを強化する簡易パケット認証手法を提案した．簡易パケット認証機能の実装では，CYPHONIC パケットに認証用の値である Token を付与し，受信側の CYPHONIC クラウドで Token を検証する機能を実装した．また，検証および評価は，動作検証と性能評価を行った．動作検証では，実装した簡易パケット認証機能が正常に動作することを確認した．性能評価では，CYPHONIC パケットに遅延をかけ，CYPHONIC クラウドが受信する CYPHONIC パケットの遅延時間に対するパケット破棄率を算出した．

7.2 今後の課題

本研究では，簡易パケット認証の概念実証を行ったに過ぎず，実装面の課題が残されている．現実装では AS が再起動しない限り，一度データベースに格納された SecretCommonValue を使用し続けることになり，セキュリティ上の懸念がある．そのため，SecretCommonValue には有効期限を設定し，有効期限を超過すると値を更新する仕組みが必要である．また，提案したパケット認証は，CYPHONIC クラウドが攻撃を受けてもサービス提供を継続するための簡易的な対策であるため，リプレイパケットの遅延時間が短いと認証を通過してしまう場合がある．そのため，簡易パケット認証後には，パケットの妥当性を確認するための追加認証手順を経ることが重要である．

付録

A　CYPHONIC のパケットフォーマット定義

B　拡張した CYPHONIC のパケットフォーマット定義

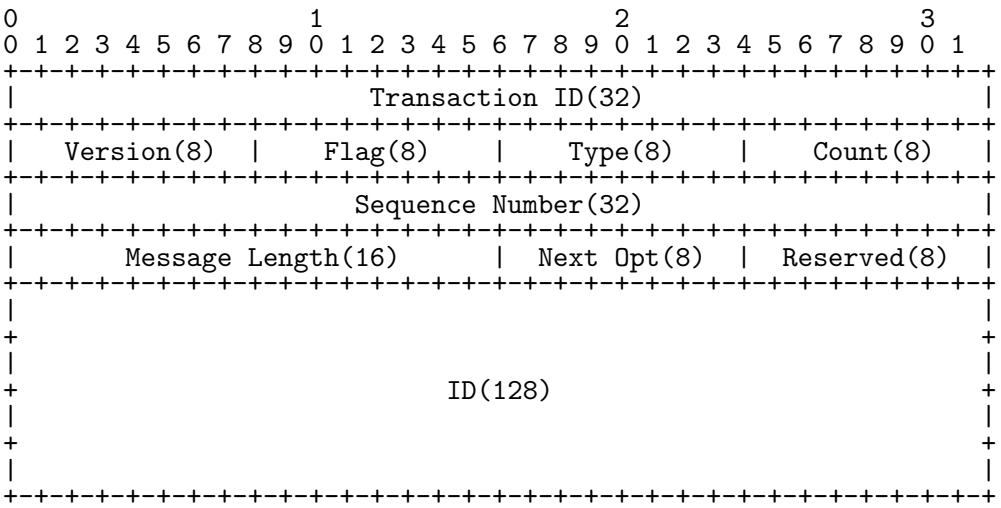


図 A.1 Structure of Base Header

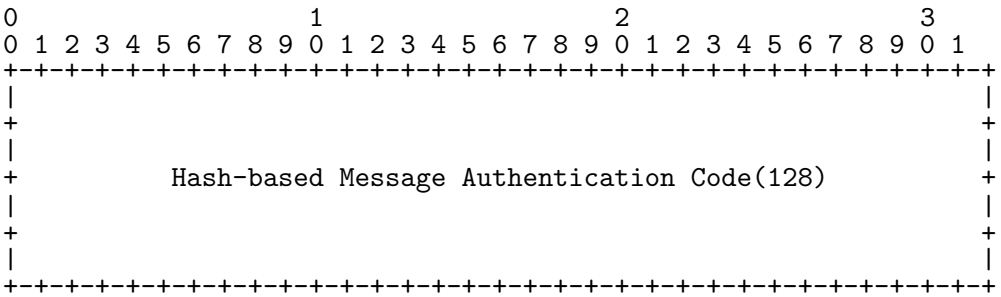


図 A.2 Structure of Hash-based Message Authentication Code

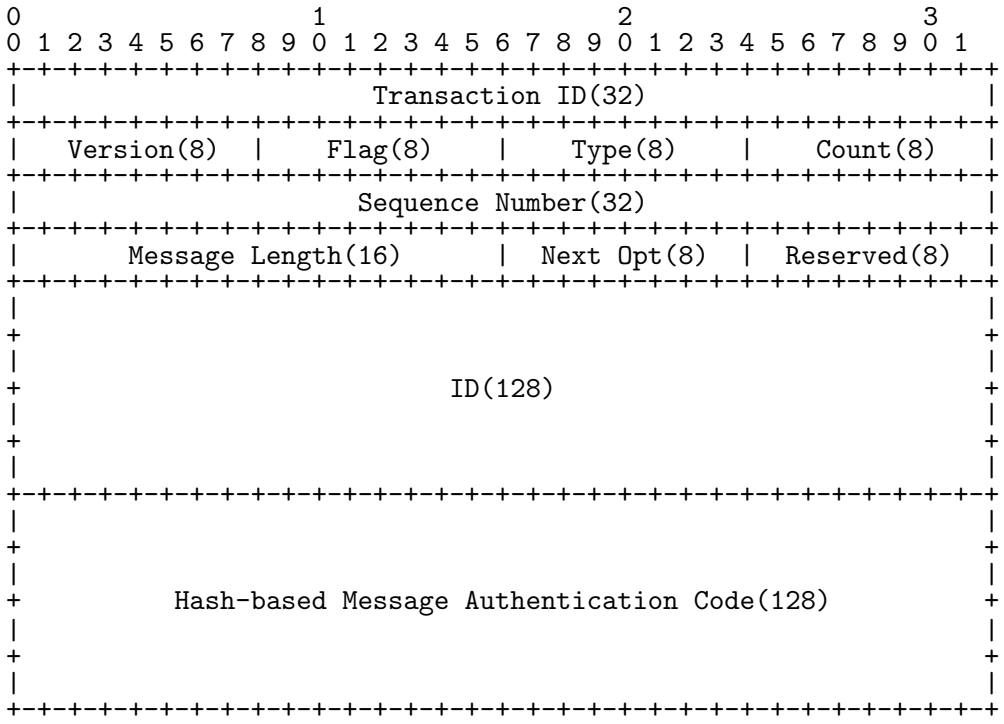


図 A.3 Structure of ACK

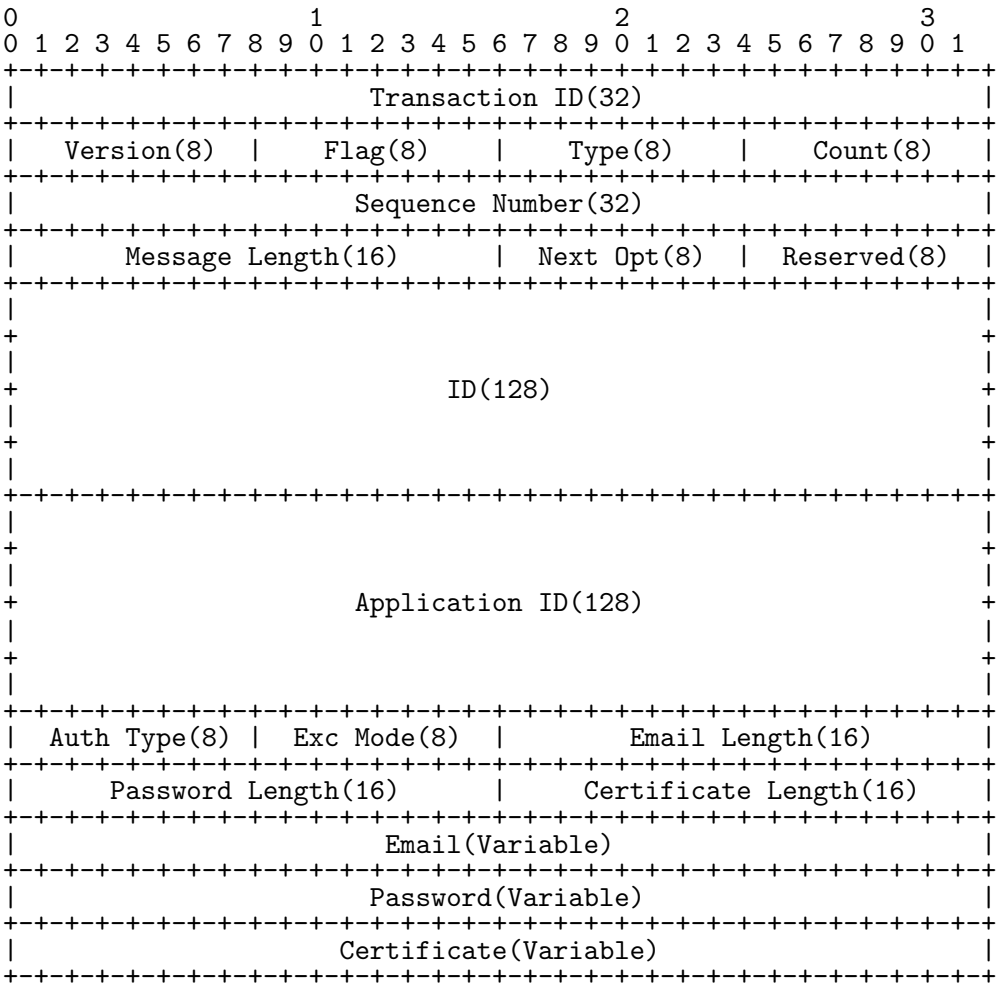


図 A.4 Structure of Login Request

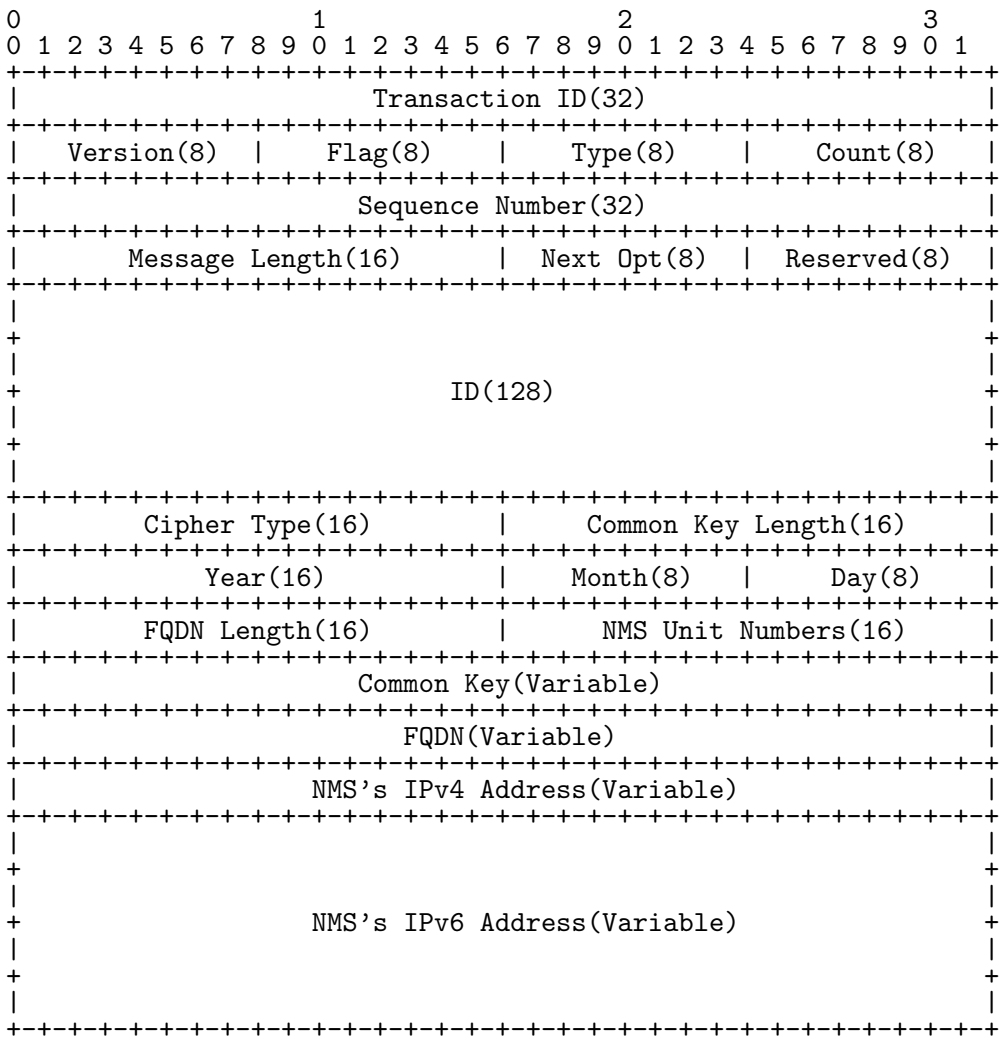


図 A.5 Structure of Login Response

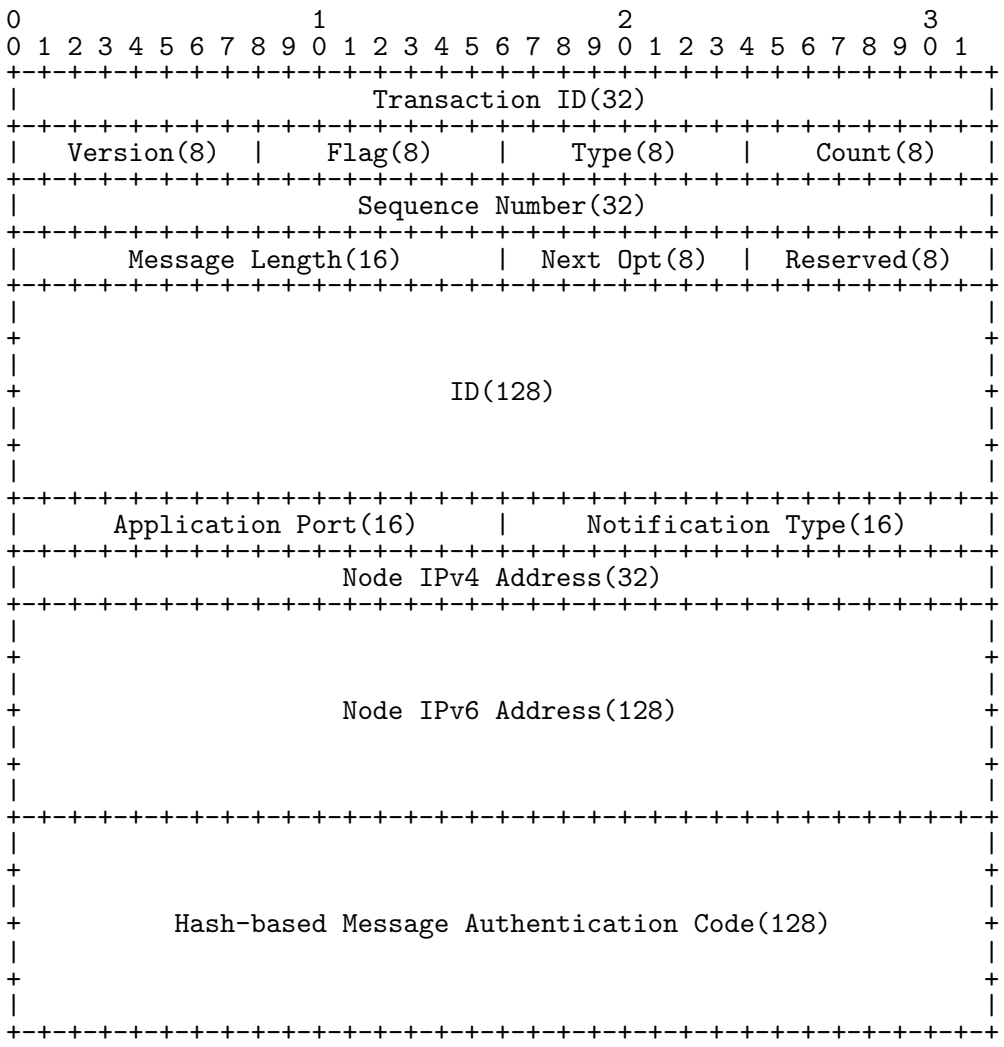


図 A.6 Structure of Registration Request

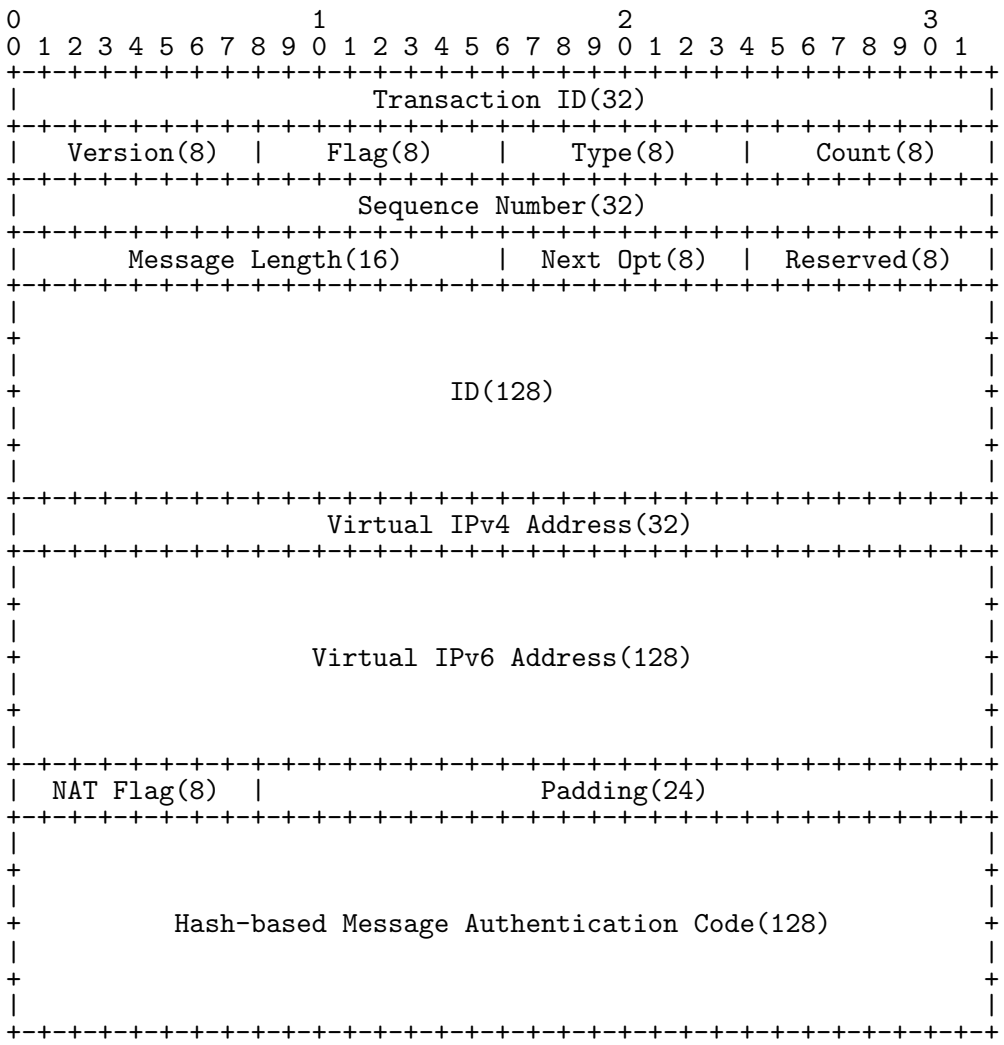


図 A.7 Structure of Registration Response

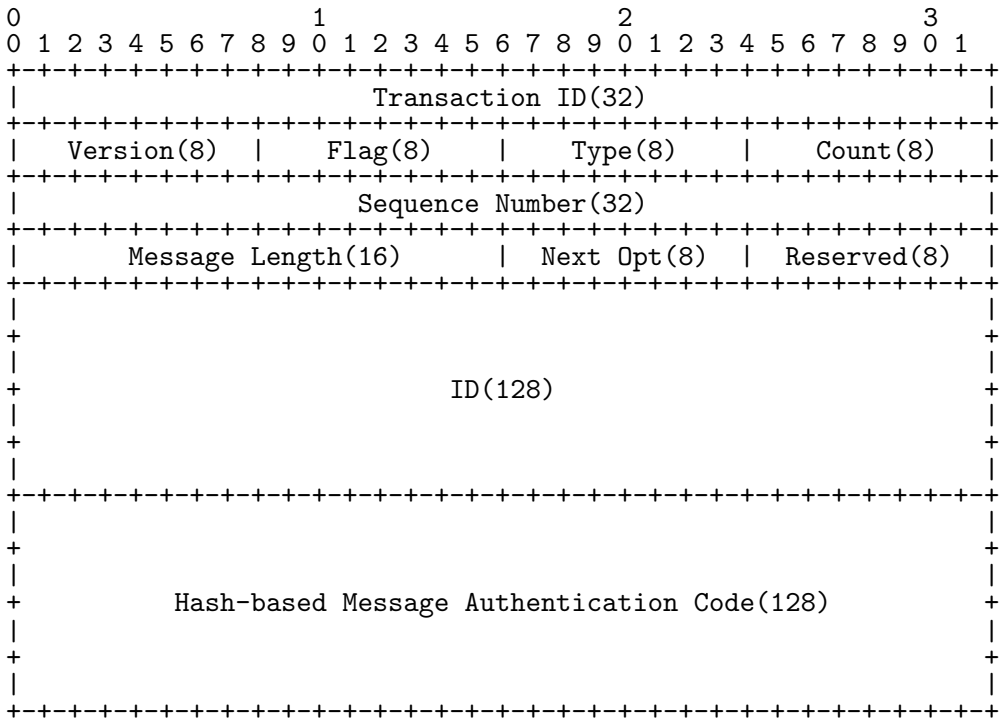


図 A.8 Structure of Keep Alive

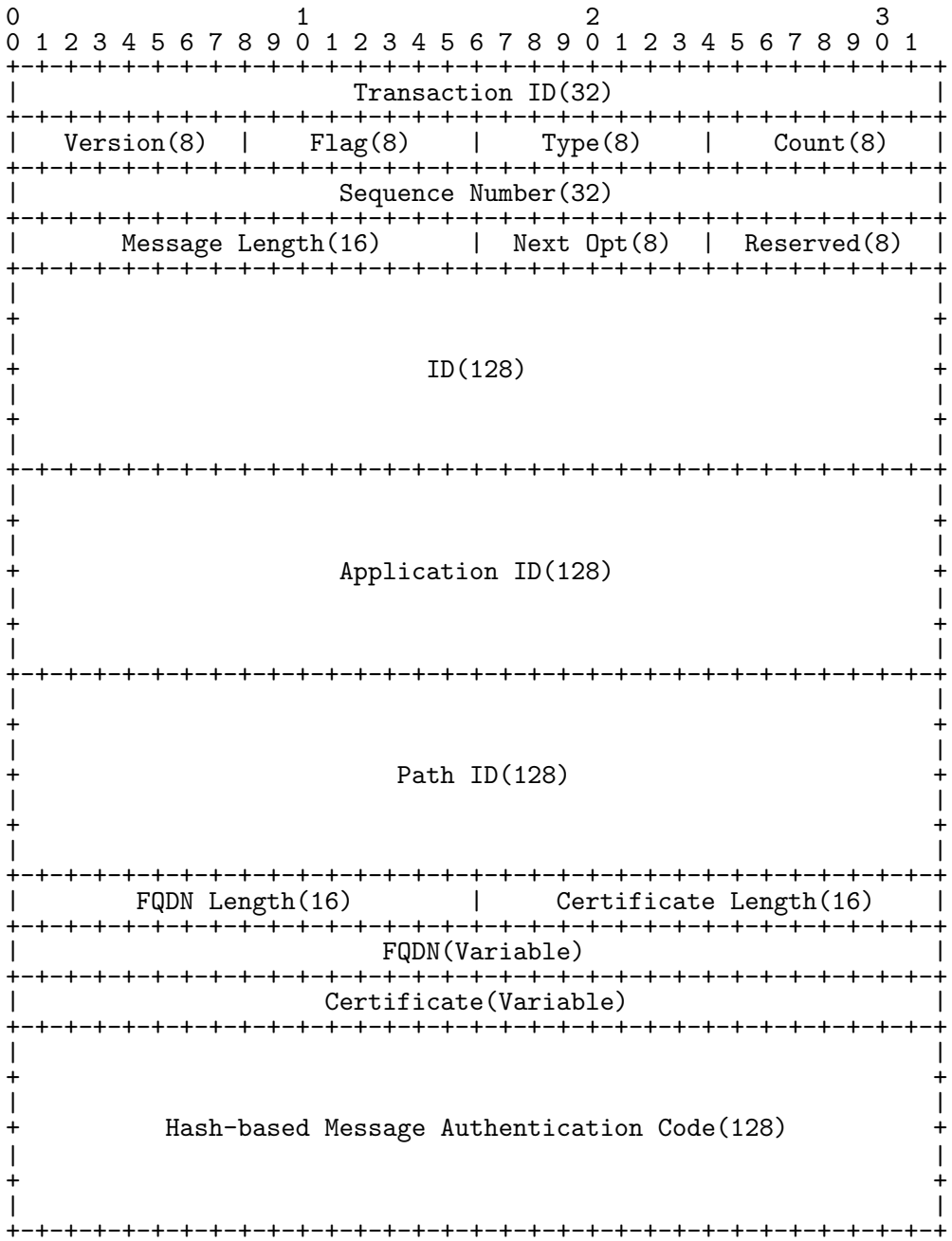


図 A.9 Structure of Direction Request

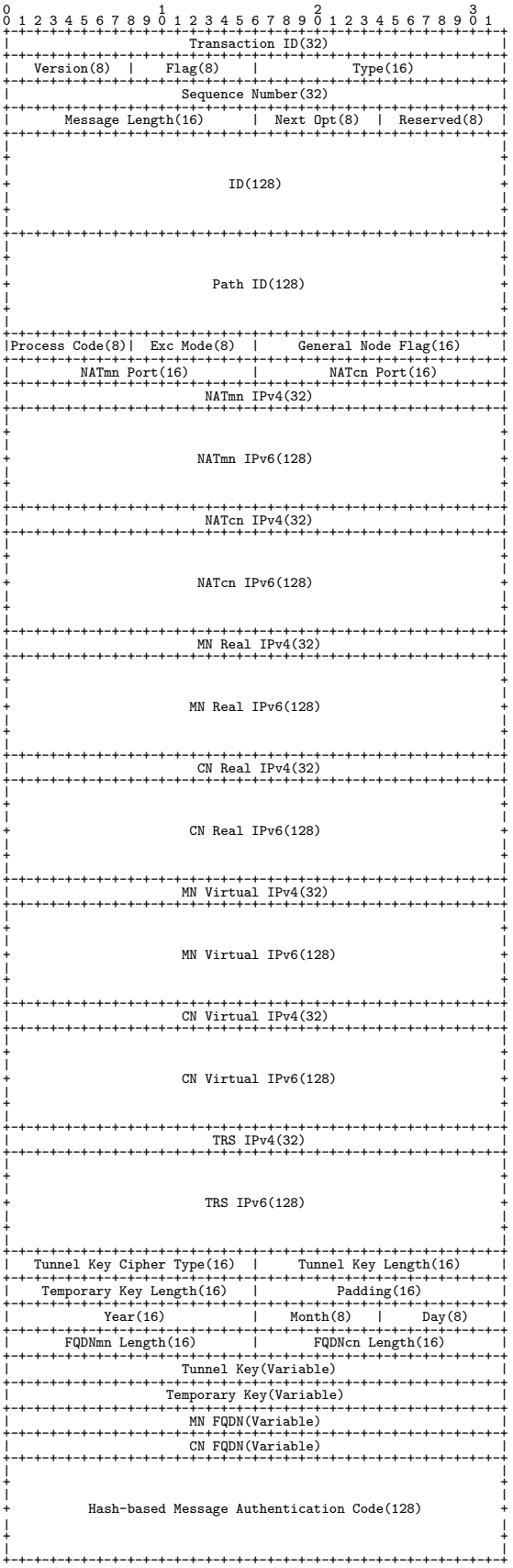


図 A.10 Structure of Route Direction

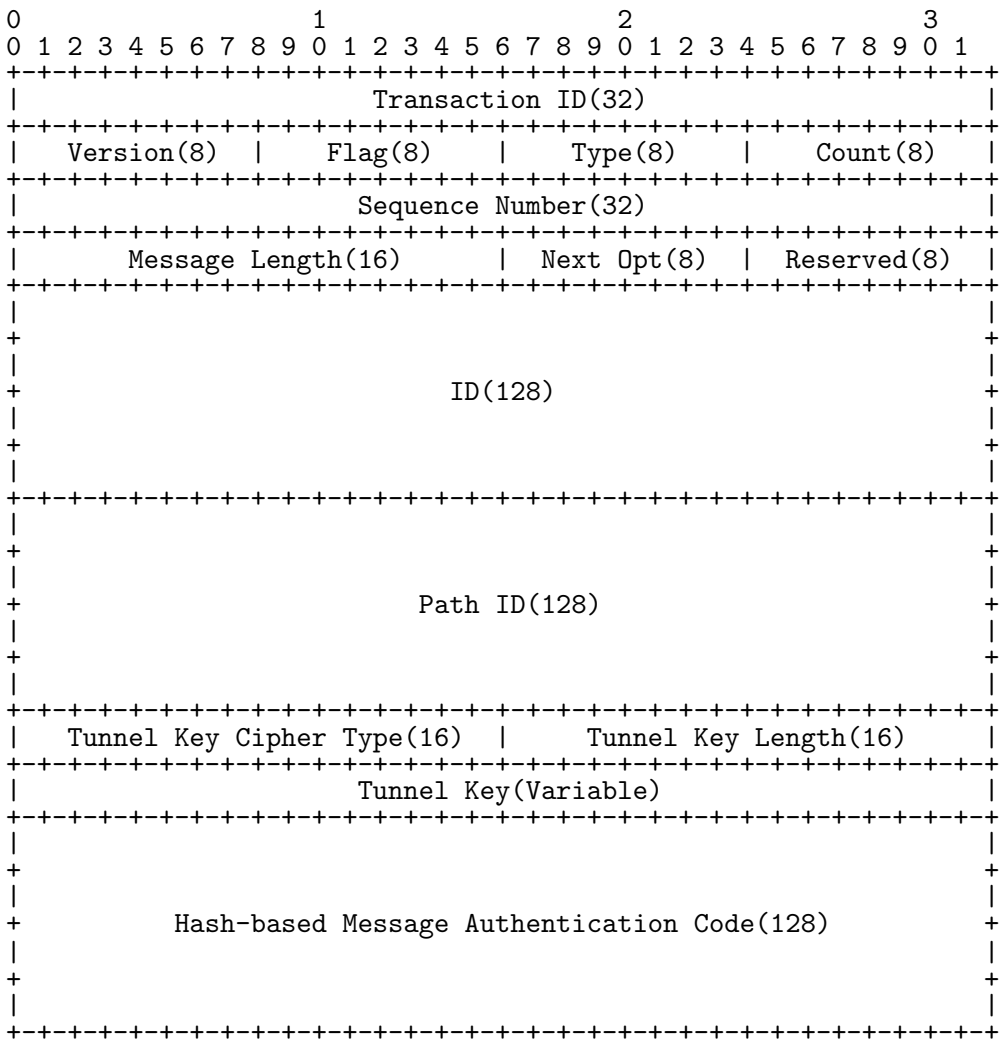


図 A.11 Structure of Relay Request

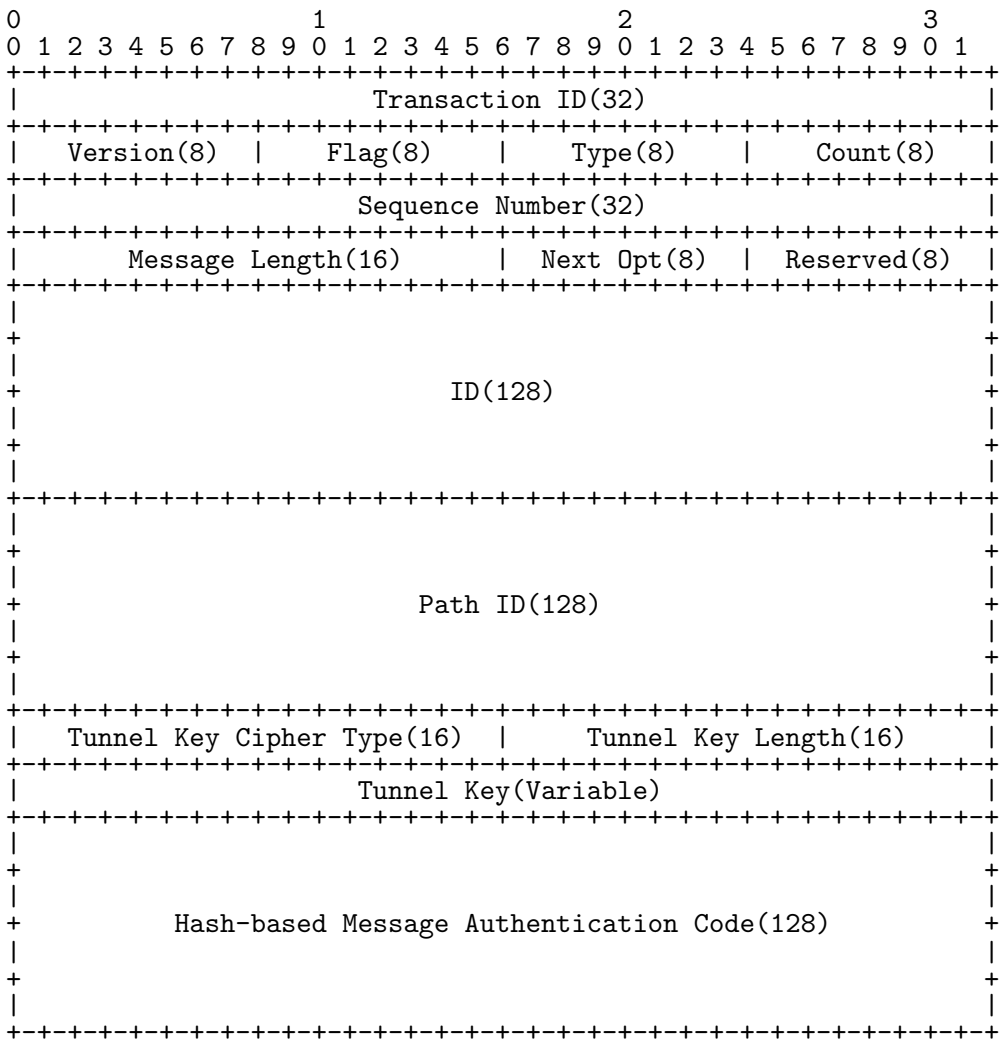


図 A.12 Structure of Relay Response

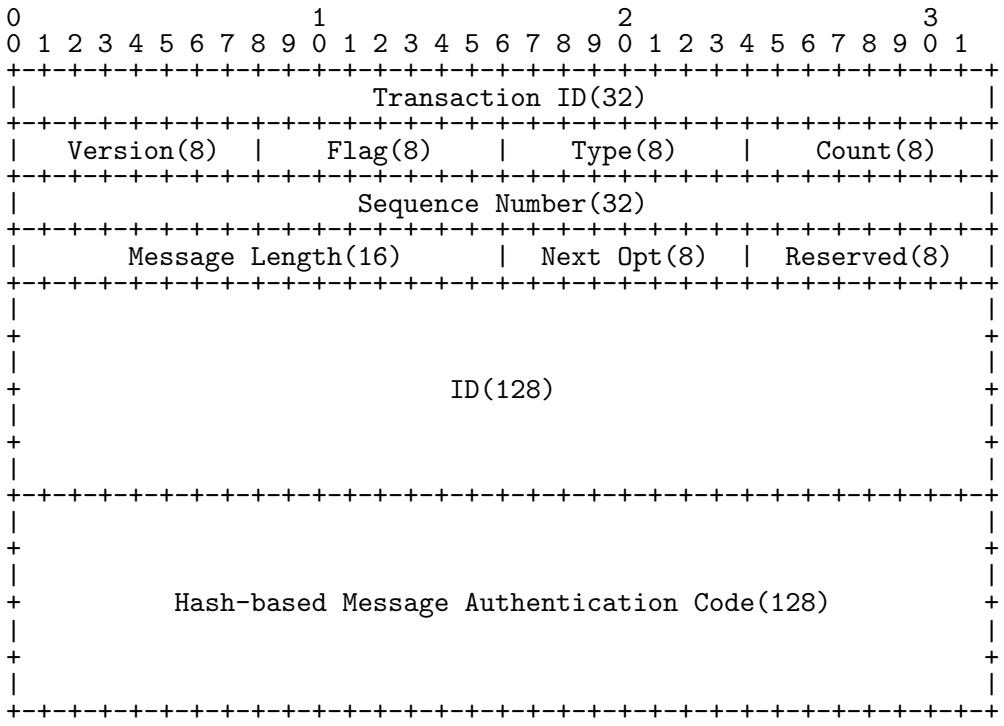


図 A.13 Structure of Hole Punching

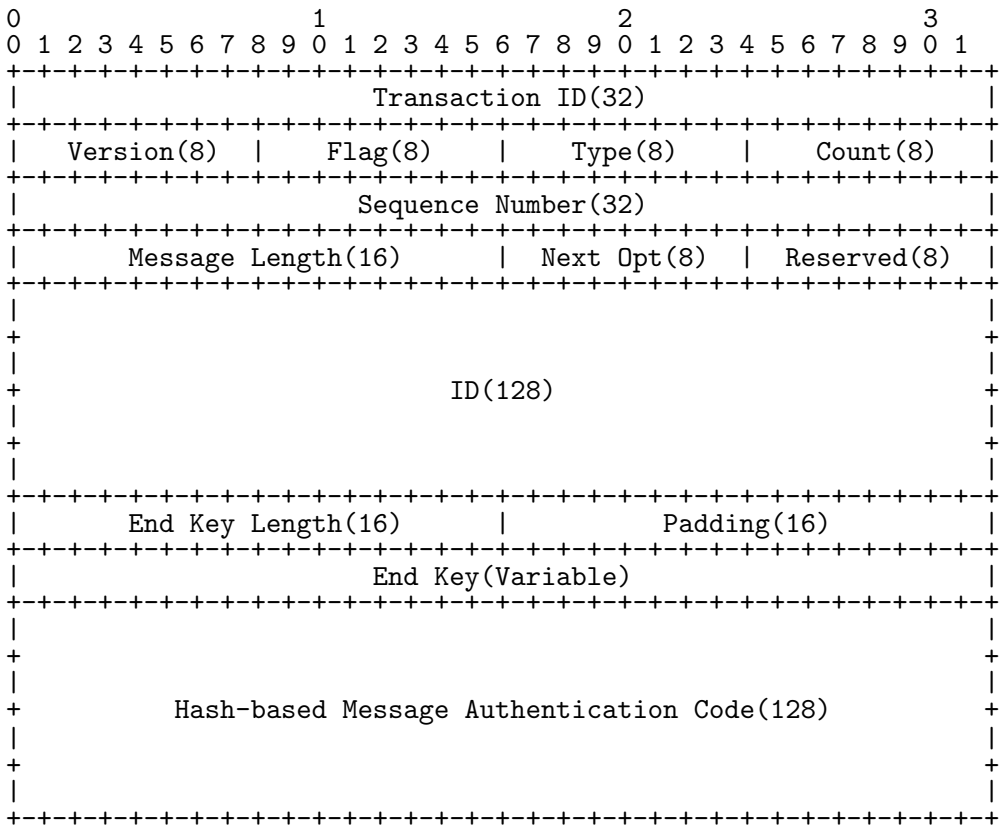


図 A.14 Structure of Tunnel Request

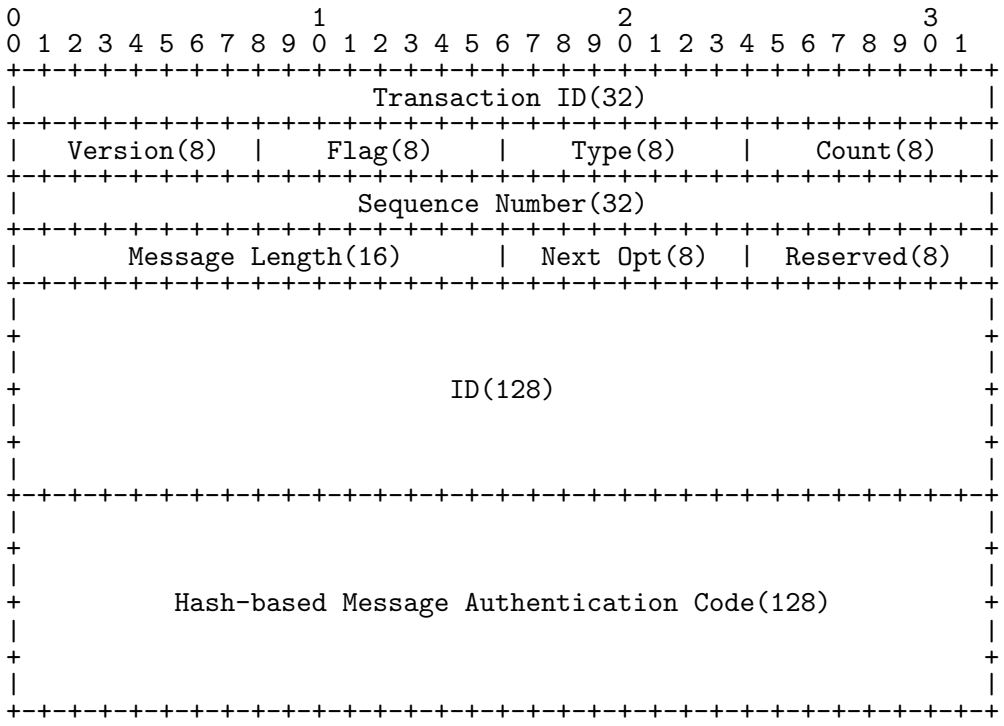


図 A.15 Structure of Tunnel Response

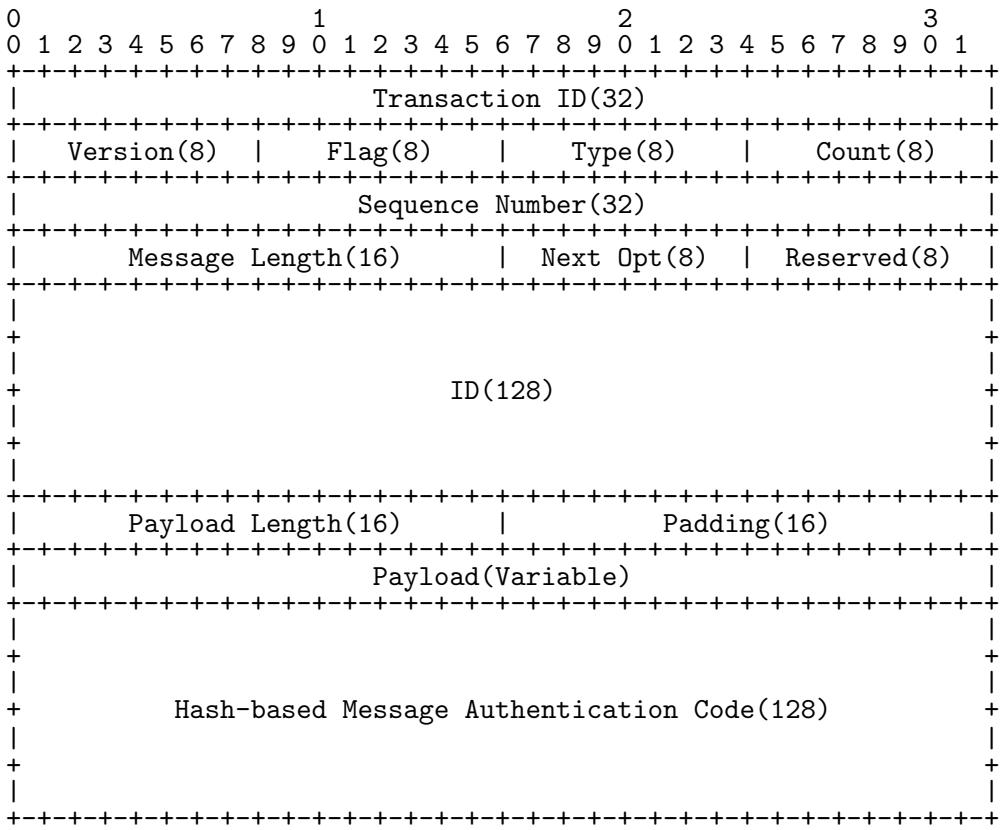


図 A.16 Capsule Message

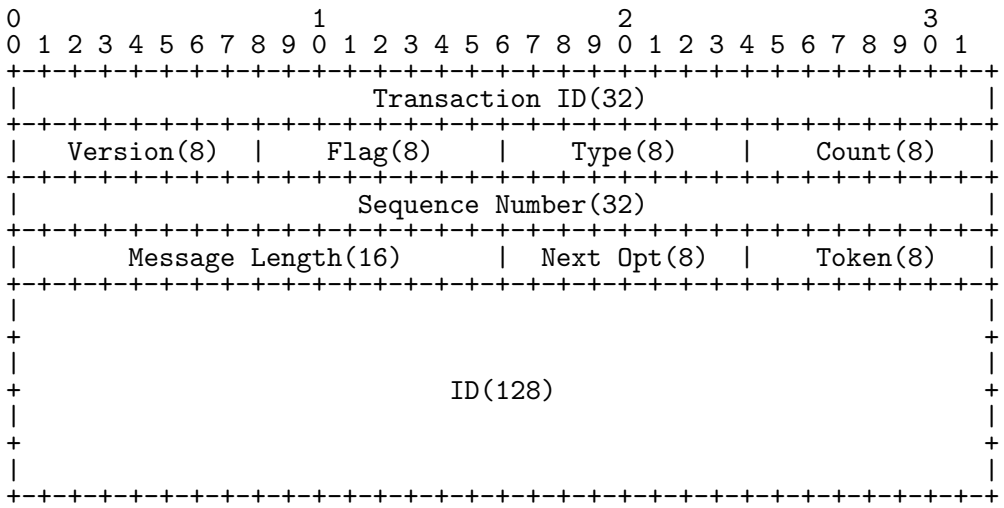


図 B.1 Structure of Extended Base Header

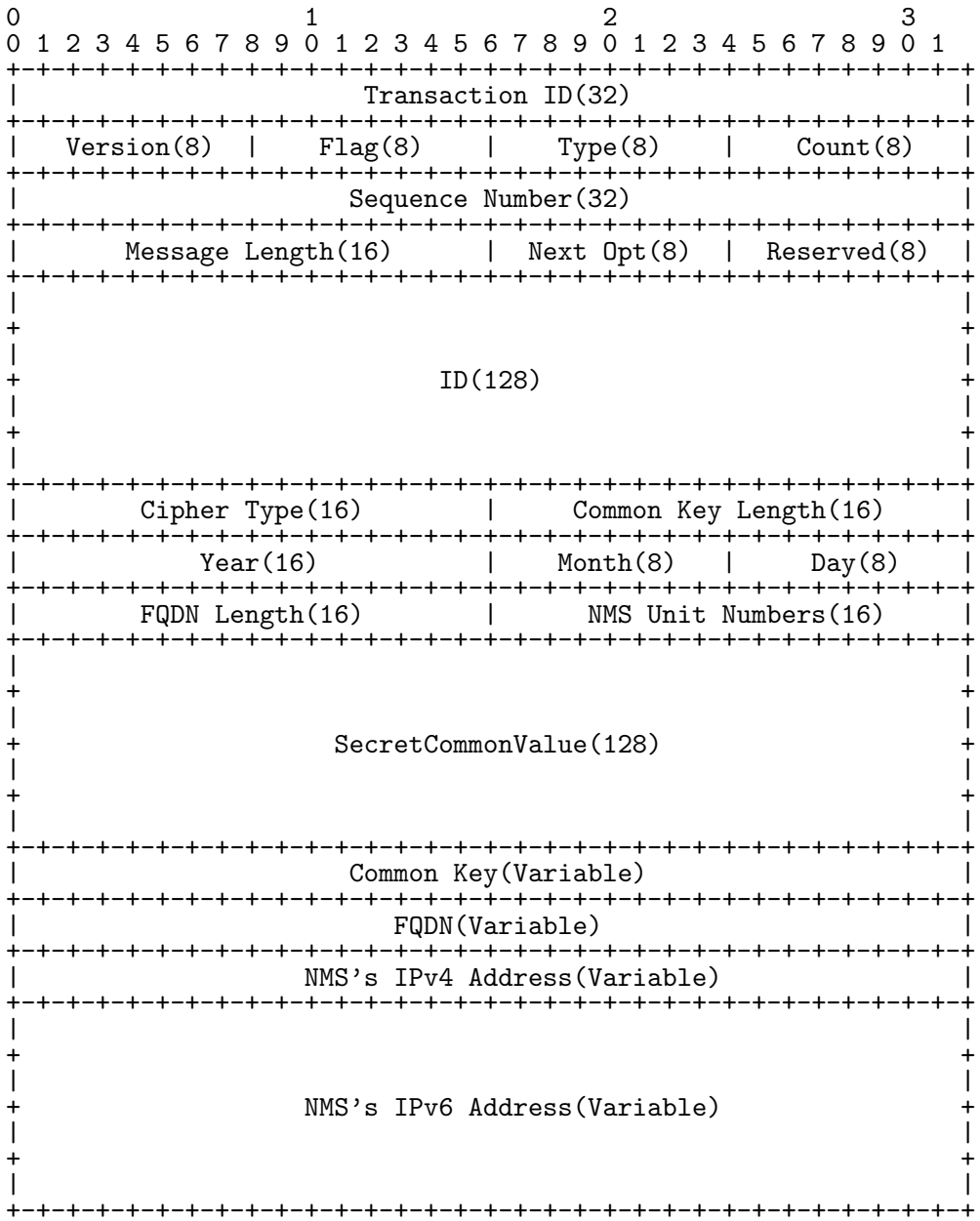


図 B.2 Structure of Extended Login Response

謝辞

本研究の遂行にあたり，多忙な中ご支援いただいた多くの方々に深く感謝の意を表する．愛知工業大学情報科学部情報科学科 内藤克浩 教授には，多忙でありながらも時間を割き，本研究の進行において，ご指導を賜りましたことを深く感謝申し上げる．同大学大学院経営情報科学研究科情報科学専攻内藤研究室の大学院生の皆様には，日頃の研究活動から要旨，論文添削に至るまで様々なご指導とご教授を賜り賜りましたことを深く感謝申し上げる．また，同研究室の同期の皆様には，互いに切磋琢磨しながらご助力いただき、同大学大学院経営情報科学研究科情報科学専攻梶研究室の OB の方には，何気ない相談から耳の痛い話までご教授いただいたことに感謝の意を表する．

参 考 文 献

[1] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys & Tutorials*, vol. 7, no. 2, pp. 72–93, 2005.

[2] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Peer-to-peer Networking and Applications*, vol. 1, pp. 18–28, 2008.

[3] M. A. Uddin, A. Stranieri, I. Gondal, and V. Balasubramanian, "A survey on the adoption of blockchain in iot: Challenges and solutions," *Blockchain: Research and Applications*, vol. 2, no. 2, p. 100006, 2021.

[4] R. Ranjan and L. Zhao, "Peer-to-peer service provisioning in cloud computing environments," *The Journal of Supercomputing*, vol. 65, pp. 154–184, 2013.

[5] "Internet Protocol," RFC 791, Sep. 1981. [Online]. Available: https://www.rfc-editor.org/info/rfc791

[6] L. Prehn, F. Lichtblau, and A. Feldmann, "When wells run dry: the 2020 ipv4 address market," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, pp. 46–54, 2020.

[7] M. Holdrege and P. Srisuresh, "IP Network Address Translator (NAT) Terminology and Considerations," RFC 2663, Aug. 1999. [Online]. Available: https://www.rfc-editor.org/info/rfc2663

[8] K. B. Egevang and P. Srisuresh, "Traditional IP Network Address Translator (Traditional NAT)," RFC 3022, Jan. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3022

[9] D. S. E. Deering and B. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 8200, Jul. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8200

[10] C.-L. Huang and S.-H. Hwang, "The asymmetric nat and its traversal method," in *2009 IFIP International Conference on Wireless and Optical Communications Networks*, IEEE, pp. 1–4, 2009.

[11] P. Wu, Y. Cui, J. Wu, J. Liu, and C. Metz, "Transition from ipv4 to ipv6: A state-of-the-art survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1407–1424, 2012.

[12] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing, "Rfc 5389: Session traversal utilities for nat (stun)," 2008.

[13] R. Mahy, P. Matthews, and J. Rosenberg, "Rfc 5766: Traversal using relays around nat (turn): relay extensions to session traversal utilities for nat (stun)," 2010.

[14] J. Rosenberg, "Interactive connectivity establishment (ice): A protocol for network address translator (nat) traversal for offer/answer protocols," Tech. Rep., 2010.

[15] S. Aravind and G. Padmavathi, "Migration to ipv6 from ipv4 by dual stack and tunneling techniques," in *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, IEEE, pp. 107–111, 2015.

[16] P. Wu, Y. Cui, J. Wu, J. Liu, and C. Metz, "Transition from ipv4 to ipv6: A state-of-the-art survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1407–1424, 2013.

[17] Q. Zhang, C. Guo, Z. Guo, and W. Zhu, "Efficient mobility management for vertical handoff between wwan and wlan," *IEEE Communications Magazine*, vol. 41, no. 11, pp. 102–108, 2003.

[18] P. Singh, "Enhancement of handover decision in heterogeneous networks," in *2017 International Conference on Signal Processing and Communication (ICSPC)*, IEEE, pp. 436–441, 2017.

[19] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "Impact of ieee 802.11 n/ac phy/mac high throughput enhancements on transport and application protocols—a survey," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2050–2091, 2017.

[20] A. Taghizadeh, T.-C. Wan, and R. Budiarto, "A fast inter-domain network-based ip mobility scheme for urban areas," *Journal of Communications and Networks*, vol. 16, no. 6, pp. 645–655, 2014.

[21] Z. Xiang and Z. Ma, "Mobility management based on mip table in mixed ipv4/v6 networks," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, IEEE, pp. 1132–1136, 2012.

[22] S. Tomic, T. Hoeher, R. Menedetter, R. Maslenka, M. Banfield, and R. Lauster, "Study of sip-based voip application interworking with ipv4-ipv6 transitioning mechanisms," in *2006 IEEE Sarnoff Symposium*, IEEE, pp. 1–4, 2006.

[23] A. Wylde, "Zero trust: Never trust, always verify," in *2021 international conference on cyber situational awareness, data analytics and assessment (cybersa)*, IEEE, pp. 1–4, 2021.

[24] K. Naito, K. Tanaka, and K. Tanaka, "Cyphonic: Overlay network technology for cyber physical communication," in *The 10th International Multi-Conference on Complexity, Informatics and Cybernetics: IMCIC 2019*, pp. 1–6, 2019.

[25] S. Isomura, T. Yoshikawa, H. Komura, S. Kubota, C. Nishiwaki, and K. Naito, "Prototyping evaluation of cyphonic: Overlay network technology for cyber-physical communication," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, pp. 1–2, 2021.

[26] T. Yoshikawa, H. Komura, C. Nishiwaki, R. Goto, K. Matama, and K. Naito, "Evaluation of new cyphonic: Overlay network protocol based on go language," in *2022 IEEE International Conference on Consumer Electronics (ICCE)*, IEEE, pp. 1–6, 2022.

[27] S. Horisaki, K. Matama, K. Naito, and H. Suzuki, "A proposal of quic-based cyphonic for encrypted end-to-end communications," in *2022 Tenth International Symposium on Computing and Networking (CANDAR)*, IEEE, pp. 27–35, 2022.

[28] R. Goto, K. Matama, R. Aihata, S. Horisaki, H. Suzuki, and K. Naito, "Implementation and evaluation of cyphonic client focusing on sequencing mechanisms and concurrency for packet processing," in *2023 IEEE 12th Global Conference on Consumer Electronics (GCCE)*, IEEE, pp. 997–1001, 2023.

[29] Z. Peng, Z. Duan, J.-J. Qi, Y. Cao, and E. Lv, "Hp2p: A hybrid hierarchical p2p network," in *First International Conference on the Digital Society (ICDS'07)*, IEEE, pp. 18–18, 2007.

[30] M. K. Kagita, N. Thilakarathne, T. R. Gadekallu, P. K. R. Maddikunta, and S. Singh, "A review on cyber crimes on the internet of things," *Deep Learning for Security and Privacy Preservation in IoT*, pp. 83–98, 2022.

[31] H. Ahmetoglu and R. Das, "A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions," *Internet of Things*, vol. 20, p. 100615, 2022.

[32] H. S. Lallie, L. A. Shepherd, J. R. Nurse, A. Erola, G. Epiphaniou, C. Maple, and X. Bellekens, "Cyber security in the age of covid-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic," *Computers & security*, vol. 105, p. 102248, 2021.

[33] R. Verma, N. Dhanda, and V. Nagar, "Enhancing security with in-depth analysis of brute-force attack on secure hashing algorithms," in *Proceedings of Trends in Electronics and Health Informatics: TEHI 2021*. Springer, 2022, pp. 513–522.

[34] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE access*, vol. 8, pp. 6249–6271, 2020.

[35] M. A. Al-Shareeda, S. Manickam, S. A. Laghari, and A. Jaisan, "Replay-attack detection and prevention mechanism in industry 4.0 landscape for secure secs/gem communications," *Sustainability*, vol. 14, no. 23, p. 15900, 2022.

[36] A. A. Farea, C. Wang, E. Farea, and A. B. Alawi, "Cross-site scripting (xss) and sql injection attacks multi-classification using bidirectional lstm recurrent neural network,"

in *2021 IEEE International Conference on Progress in Informatics and Computing (PIC)*, IEEE, pp. 358–363, 2021.

[37] R. Khader and D. Eleyan, "Survey of dos/ddos attacks in iot," *Sustainable Engineering and Innovation*, vol. 3, no. 1, pp. 23–28, 2021.

[38] K. Coulibaly, "An overview of intrusion detection and prevention systems," *arXiv preprint arXiv:2004.08967*, 2020.

[39] M. D. Junior and N. F. Ebecken, "A new waf architecture with machine learning for resource-efficient use," *Computers & Security*, vol. 106, p. 102290, 2021.

[40] W. S. Admass, Y. Y. Munaye, and A. A. Diro, "Cyber security: State of the art, challenges and future directions," *Cyber Security and Applications*, vol. 2, p. 100031, 2024.

[41] U. Inayat, M. F. Zia, S. Mahmood, H. M. Khalid, and M. Benbouzid, "Learning-based methods for cyber attacks detection in iot systems: A survey on methods, analysis, and future prospects," *Electronics*, vol. 11, no. 9, p. 1502, 2022.

[42] A. A. Elsaeidy, N. Jagannath, A. G. Sanchis, A. Jamalipour, and K. S. Munasinghe, "Replay attack detection in smart cities using deep learning," *IEEE Access*, vol. 8, pp. 137 825–137 837, 2020.

[43] M. Gniewkowski, "An overview of dos and ddos attack detection techniques," in *International Conference on Dependability and Complex Systems*, Springer, pp. 233–241, 2020.

[44] T. Li, Z. Wang, L. Zou, B. Chen, and L. Yu, "A dynamic encryption–decryption scheme for replay attack detection in cyber–physical systems," *Automatica*, vol. 151, p. 110926, 2023.

研究業績

- R.Goto, K.Matama, R.Aihata, M.Suda, H.Suzuki, and K.Naito, “CYPHOIC Adapter: Enabling Secure P2P Connectivity for Diverse IoT Devices Active,” *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, January 2024.
- 鈴木 誉写, 須田 倫代, ウジイエ ギレルメ セイジ, 宮村 一希, 相畑 亮太, 鈴木 秀和, 内藤 克浩 “CYPHONIC クラウドにおける効率的な処理追跡に向けたロギング機構の設計と基礎評価, ” 情報処理学会 第 *40* 回 コンシューマ・デバイス*&*システム研究会 *(CDS)*, May 2024.
- Yoshiya Suzuki, Michiyo Suda, Seidy Guilherme Ujiie, Kazuki Miyamura, Hidekazu Suzuki and Katsuhiro Naito “Effective Process Tracking in Horizontally Scalable CYPHONIC Cloud Services,” *2024 IEEE 13th Global Conference on Consumer Electronics (GCCE)*, October 2024.
- 須田 倫代, 鈴木 誉写, 杉原 充稀, 鈴木 秀和, 内藤 克浩 “CYPHONIC クラウドのセキュリティ強化に向けた簡易パケット認証の設計と評価, ” 情報処理学会 第 *114* 回モバイルコンピューティングと新社会システム研究会 *(MBL)*, February 2025.