# セキュアオーバーレイネットワークシステムにおける サービス拡張性実現手法に関する研究

# Study on Methods for Achieving Service Extensibility in Secure Overlay Network Systems

研究者 後藤廉　　指導教員 内藤克浩

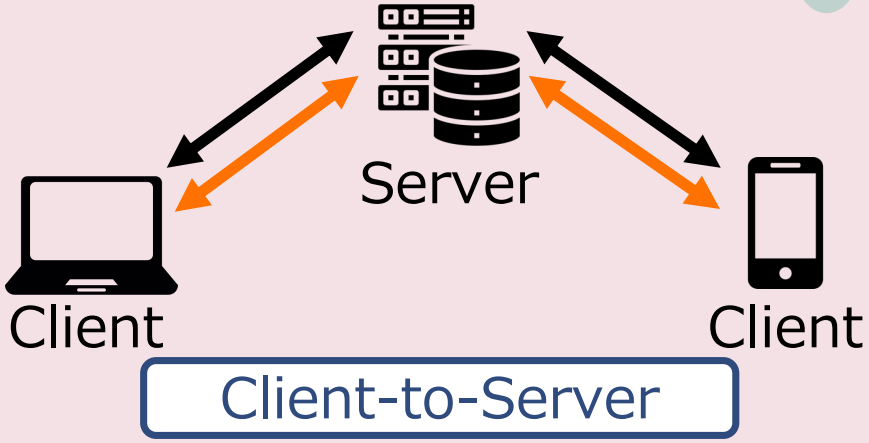
愛知工業大学大学院 経営情報科学研究科
修士論文発表審査会
2024年 2月 9日 (金)

# 目次

# サービスモデルとネットワークモデルの関係

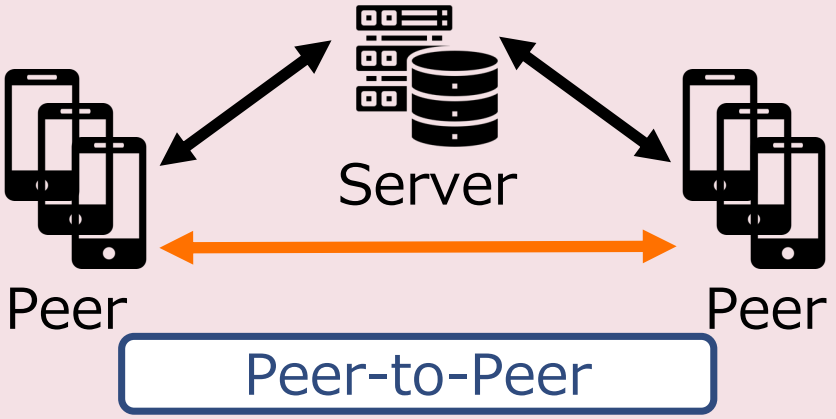
**Client-to-Server (C2S) 型**

- クライアント情報の管理が容易
- サーバ経由により通信経路が冗長化
- サーバにおける単一の脆弱性が存在




**Peer-to-Peer (P2P) 型**

- ピア情報の管理が複雑
- 直接通信による遅延の改善
- セキュリティリスクの分散が可能



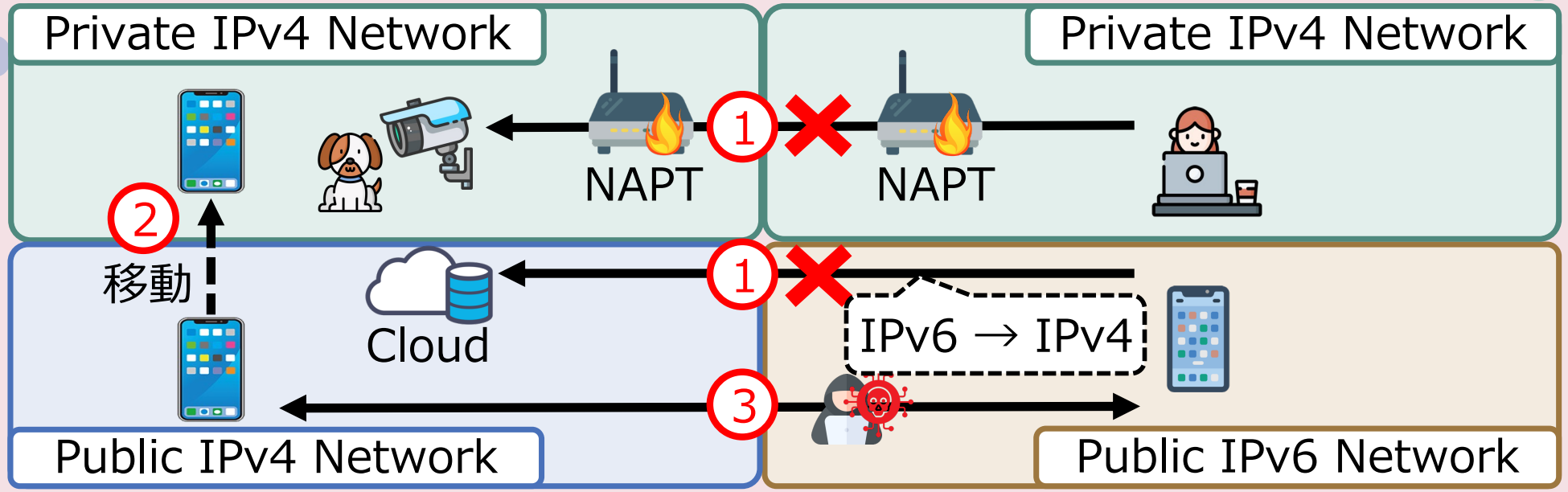

: Network model : Service model

C2S 型サービス
➡ C2S 型ネットワーク

P2P 型サービス
➡ C2S 型ネットワーク

P2P 型サービスには P2P 型ネットワークモデルが必要

# Peer-to-Peer 型ネットワークモデルの課題




NAPT : Network Address Port Translation

課題 1：NAPT 越え問題 および IPv4-IPv6 間の非互換問題（通信接続性）
➡ 端末の所属ネットワーク環境によっては直接通信が困難

課題 2：ネットワーク切り替えに伴う通信切断問題（移動透過性）
➡ IP が保持する情報の二重性により端末移動のセッション維持が困難

課題 3：相互接続・直接通信に伴うネットワーク脅威の考慮（機密性・完全性）
➡ ゼロトラストセキュリティに基づく暗号化やアクセス制御が必要

# 既存技術とCYPHONIC

| Requirement / Technology | NAPT Traversal | IPv4 - IPv6 Dual stack | Mobility Transparency | Zero-Trust Security |
|---|---|---|---|---|
| ICE | ○ | × | × | × |
| QUIC | × | ○ | ○ | × |
| OpenVPN (SoftEther) | ○ | ○ | × | × |
| Wireguard (Tailscale) | ○ | ○ | × | ○ |
| **CYPHONIC** | ○ | ○ | ○ | ○ |

既存技術では Peer-to-Peer 型ネットワークの構築に
必要とされる要素の包括的な実現が困難

# CYPHONIC 概要

CYber PHysical Overlay Network over Internet Communication
Peer-to-Peer 型ネットワークを構築可能な
セキュアオーバーレイネットワークシステム

## 通信接続性（課題 1 への対応）

- エンド端末の所属ネットワーク環境に応じた通信経路の確立

## 移動透過性（課題 2 への対応）

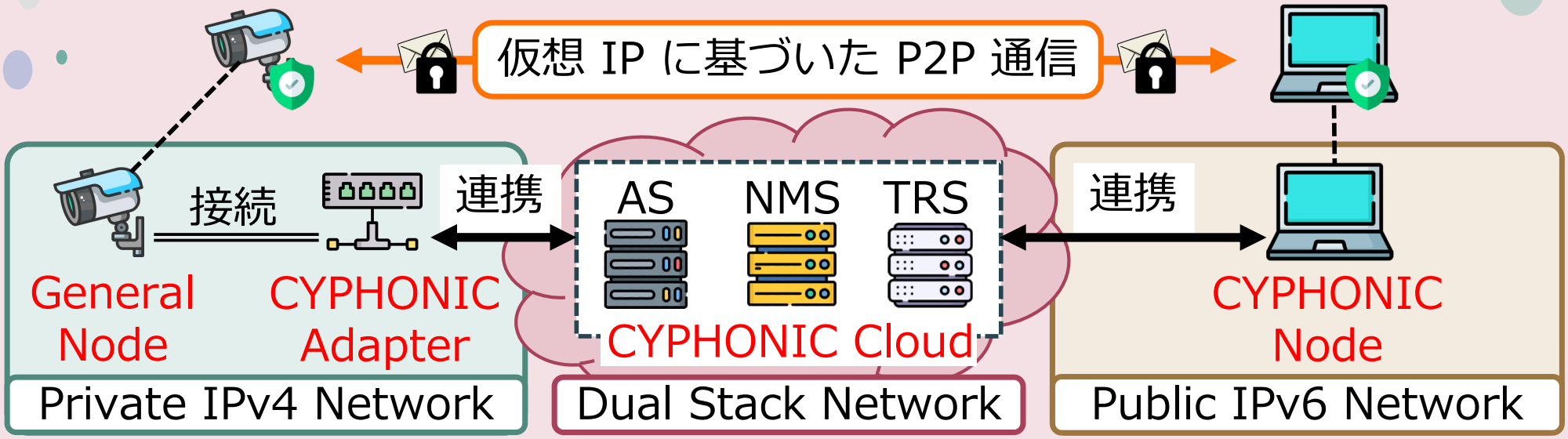
- 仮想 IP 通信により実 IP の変化をアプリケーションから隠蔽

## 機密性・完全性（課題 3 への対応）

- ゼロトラストセキュリティに基づき全端末の認証を実施
- 通信を行う双方の端末のみが知り得る共通鍵でデータを暗号化

端末間のセキュアな Peer-to-Peer 接続を
容易に実現する通信フレームワークとして機能

# CYPHONIC 構成要素




## CYPHONIC クラウド

Authentication Service (AS)
➡ 端末認証及び端末情報の管理

Node Management Service (NMS)
➡ ネットワーク情報管理と経路指示

Tunnel Relay Service (TRS)
➡ 直接通信が困難な場合に通信を中継

## CYPHONIC ノード

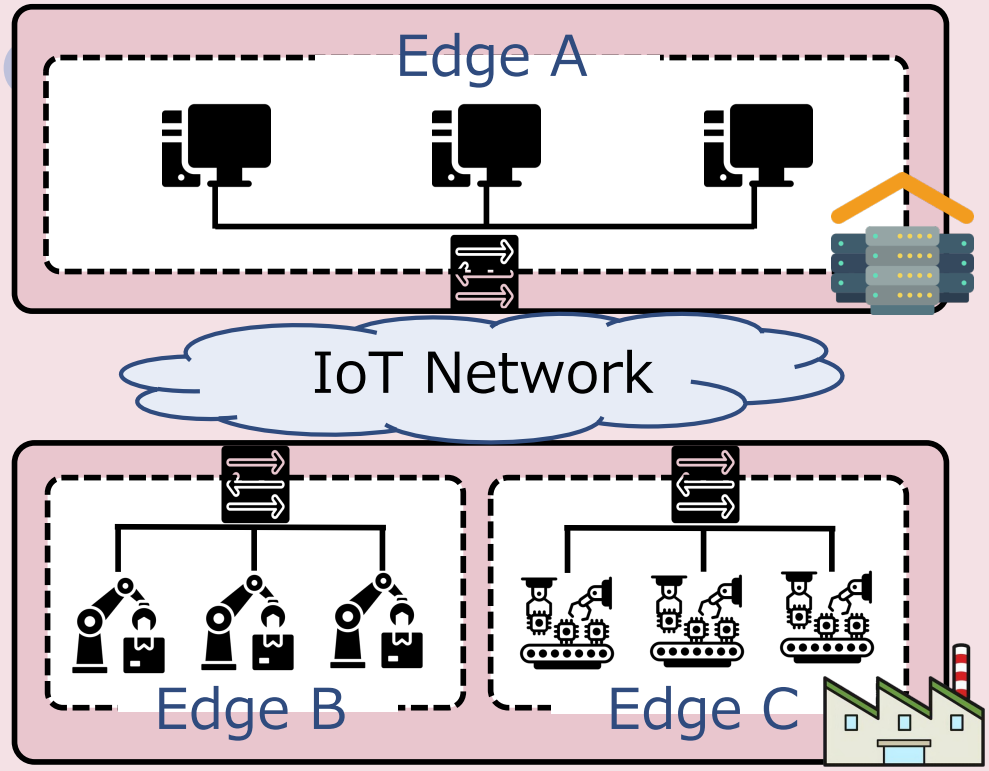
クライアントプログラムを
搭載した端末

## CYPHONIC アダプタ

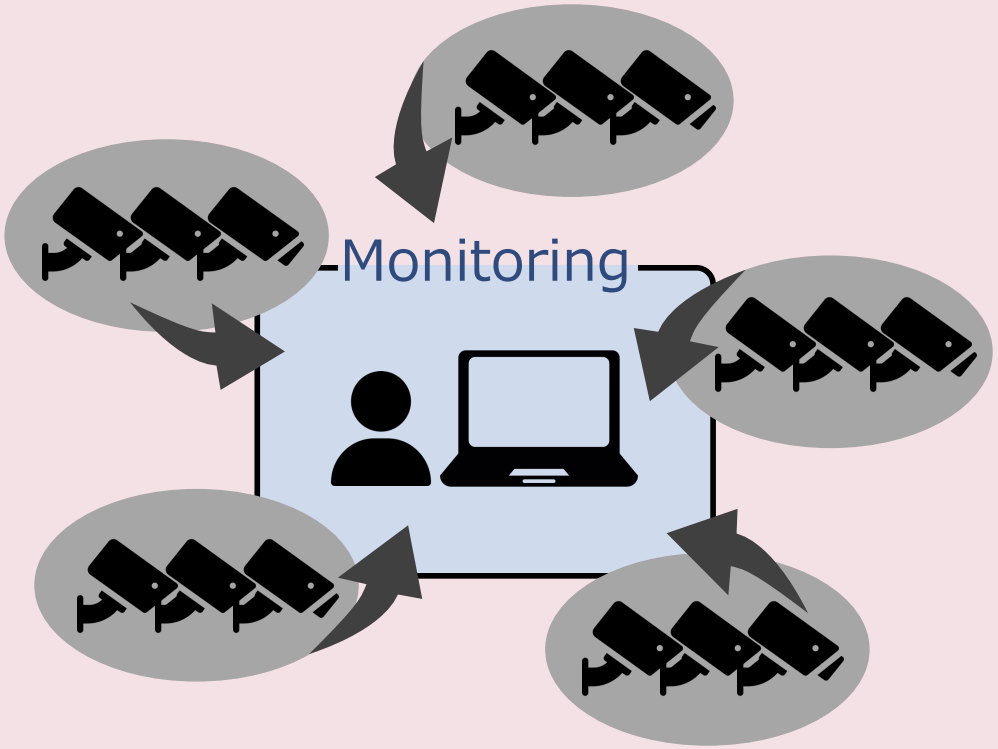
一般ノード に CYPHONIC の
通信機能を提供するアダプタ端末

※ 一般ノード：既存プログラムの改良が困難な端末（例：IoT 機器）

# 近年の IoT 利活用形態




ファクトリオートメーション



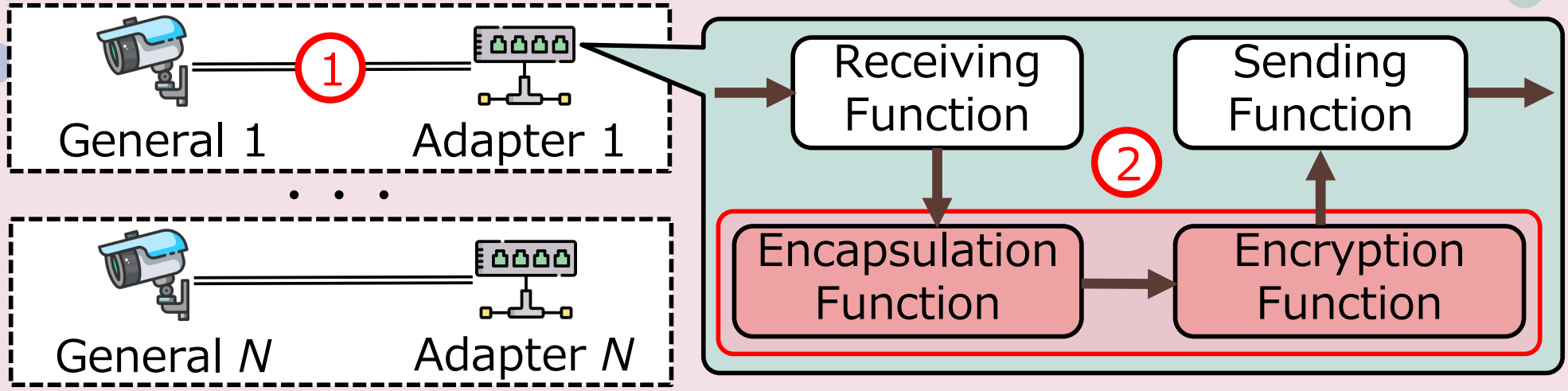

カメラーソリューション

- ファクトリオートメーション：多数の IoT 機器を連携した自動化システム
- カメラソリューション：IoT 機器を用いた複数区画からのデータ収集

膨大な IoT 機器間をセキュアに接続する技術が必要

➡ CYPHONICアダプタによりユースケースに対応可能

# CYPHONICアダプタの必要機能

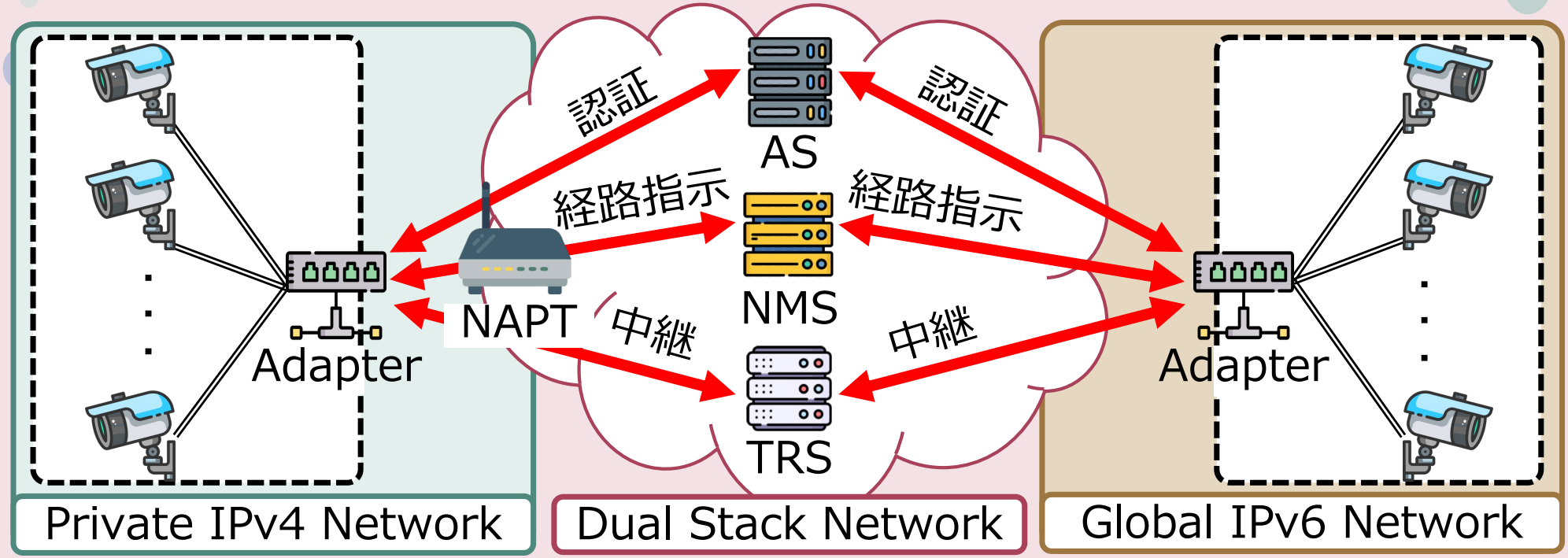



1. 一般ノードとCYPHONICアダプタが1対1対応
   - CYPHONIC の利用ケースが限定的
   - IoT 機器を利用した近年のサービス需要への対応が困難

2. アダプタは送受信データをシングルスレッドで処理
   - 一般ノードにおける通信品質の低下が懸念
   - 複数一般ノードの同時接続対応が極めて困難

複数一般ノードの同時接続対応に伴い
処理機能のマルチスレッド化が必要

# CYPHONICクラウドの必要機能




AS: Authentication Service NMS: Node Management Service
TRS: Tunnel Relay Service NAPT: Network Address Port Translation

P2P ネットワークの構築に伴い CYPHONIC クラウドと連携
➡ 利用者やエンドノードの増加により負荷増大が懸念

CYPHONIC を継続して利用可能な可用性の確保が必要

- 障害許容性：サービス機能の多重化と障害からの自動的な復旧
- 高負荷耐性：リクエストや負荷に応じたスケール機能

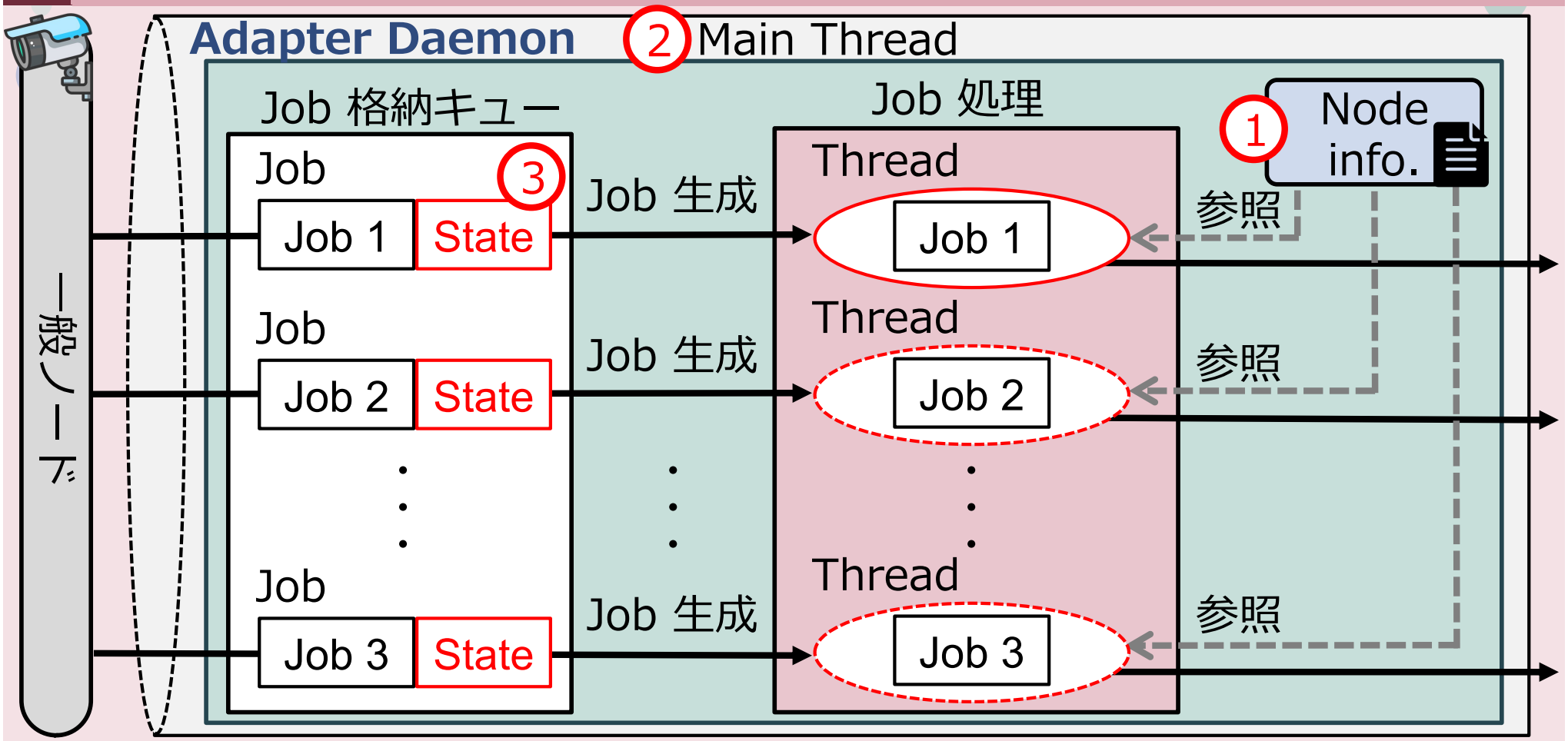
# 本研究の目的

CYPHONIC における
サービス拡張性実現手法の提案

1. CYPHONICアダプタにおける複数一般ノードのサポート
- 送受信パケット処理機能のマルチスレッド化
- フラグメントパケットの順序維持機構の追加

2. CYPHONICクラウドの障害許容性および高負荷耐性の実現
- コンテナによるクラスタリングとサービス機能の多重化
- オートスケーリングと負荷分散機能の追加

# CYPHONICアダプタの既存実装




問題 1：予め決められた一般ノード情報のみを管理
問題 2：パケット処理機能がシングルスレッドで動作
問題 3：ステート情報をスレッド内に保持

マルチスレッド環境に対応した処理機能の設計が必要

# マルチスレッド化に伴う要求事項

## 1. イベント駆動型処理モデルによる実装

スレッドの生成と破棄に要するオーバーヘッドを削減

➡ 事前に生成したワーカースレッドを用いた処理の効率化が必要

## 2. スレッド間での情報共有機能の追加

別々のスレッドに割り当てられた処理の生合成を担保

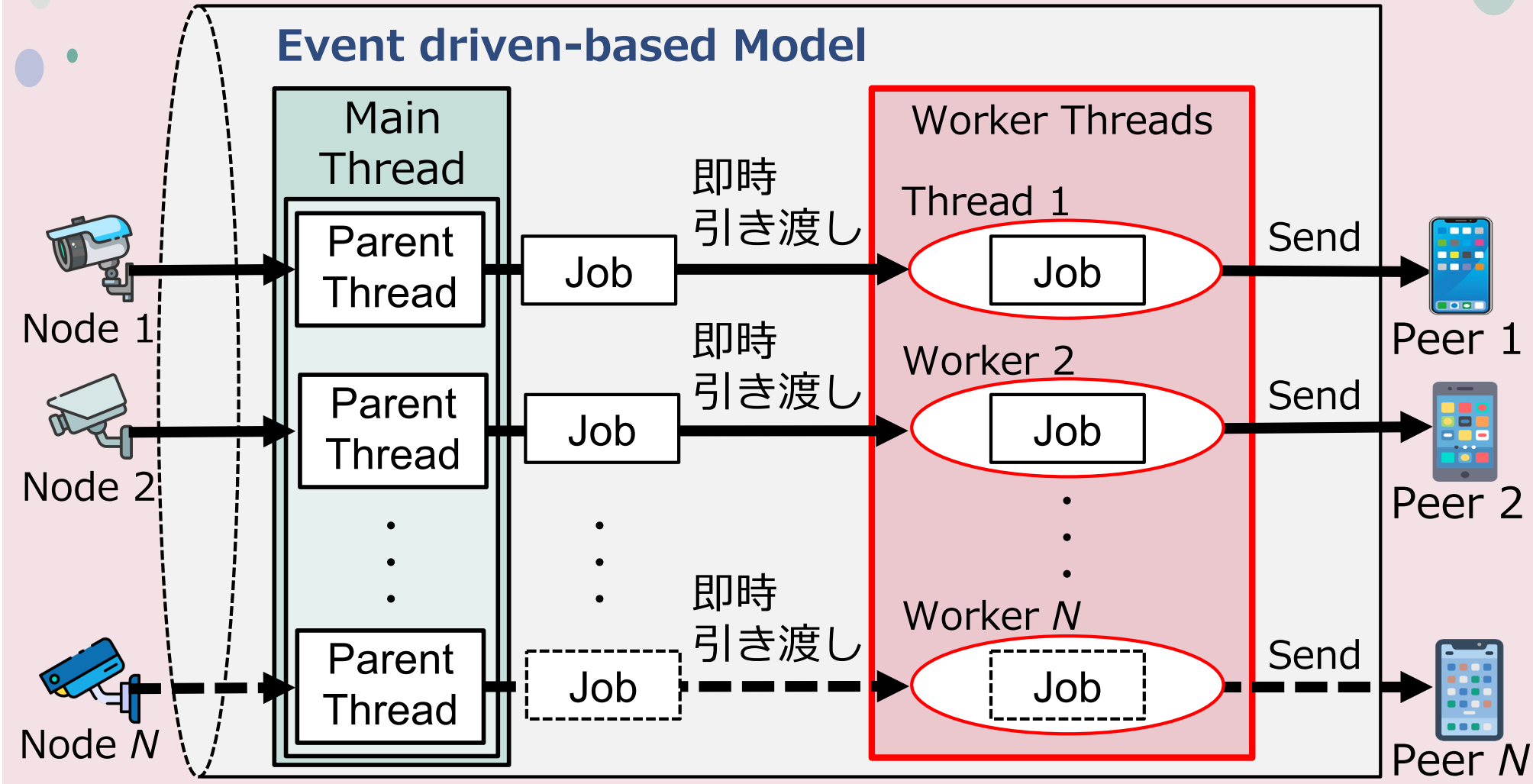
➡ ステート情報を共有するためのキャッシュストアが必要

## 3. フラグメントパケットの順序維持機構

各ワーカースレッドは非同期的にフラグメントデータを処理

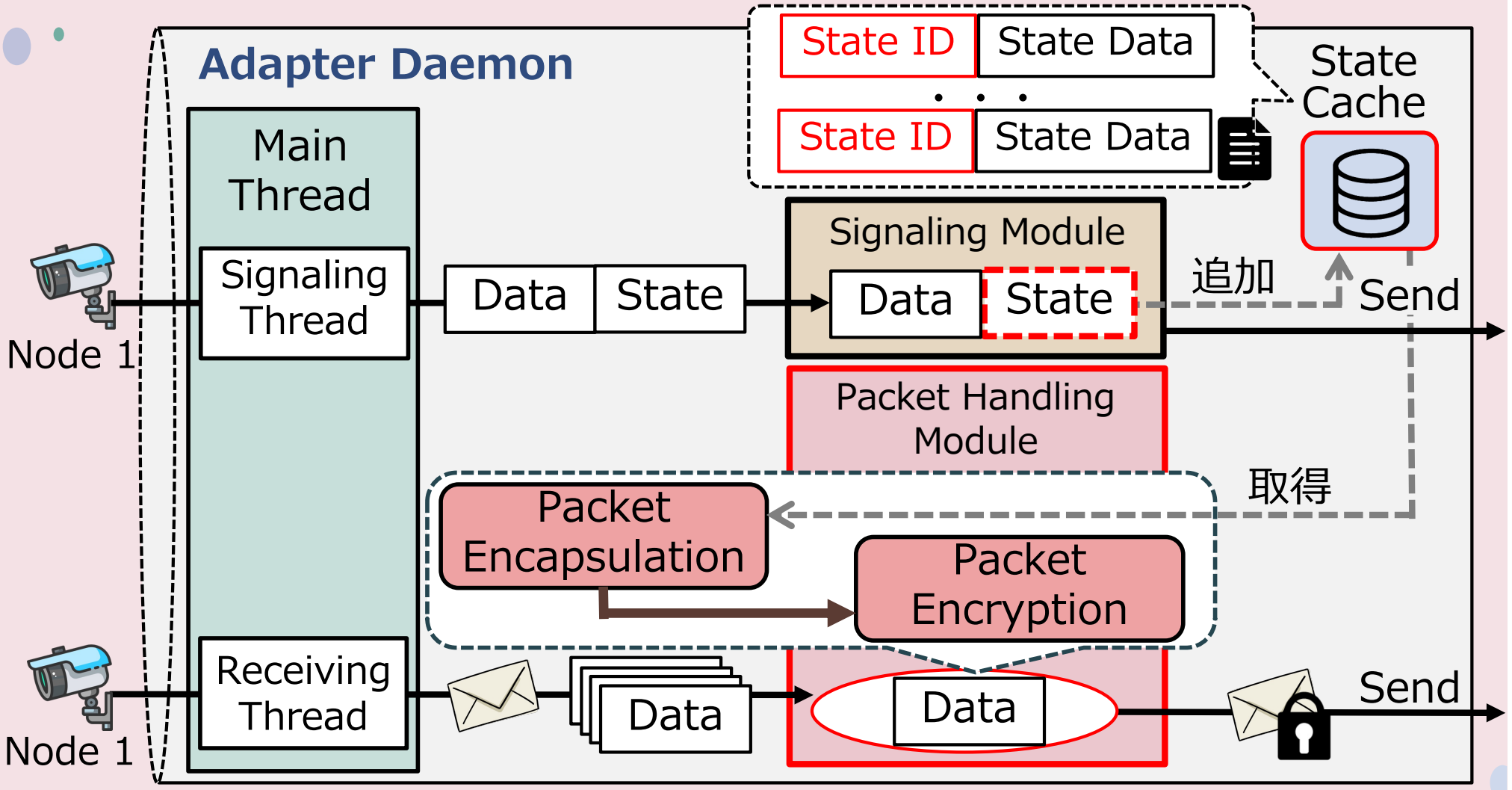
➡ 送受信時点におけるパケットの順序を維持する仕組みが必要

# イベント駆動型処理モデル



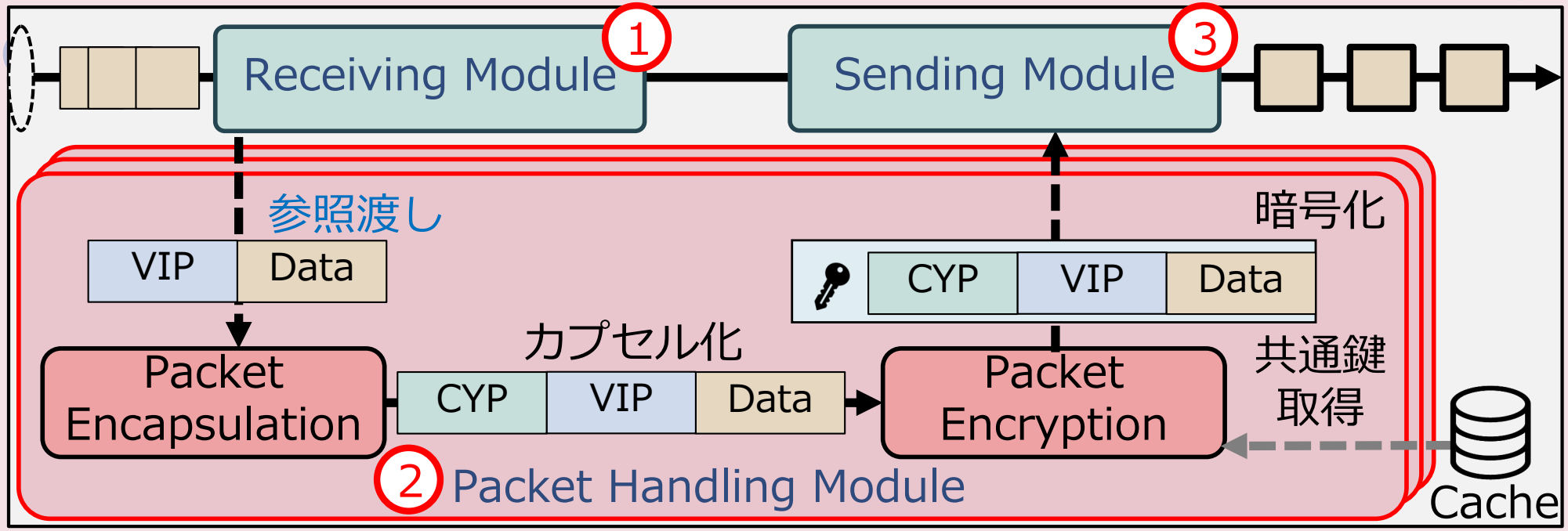

- リクエストジョブを受け取る親スレッドと実際にジョブを処理するワーカースレッドが分離
- 同時発生する通信リクエストの効率的な処理が可能

# ステート情報の共有




- シグナリング時にステート情報を取得してキャッシュに追加
- パケット処理用のワーカースレッドが情報を参照

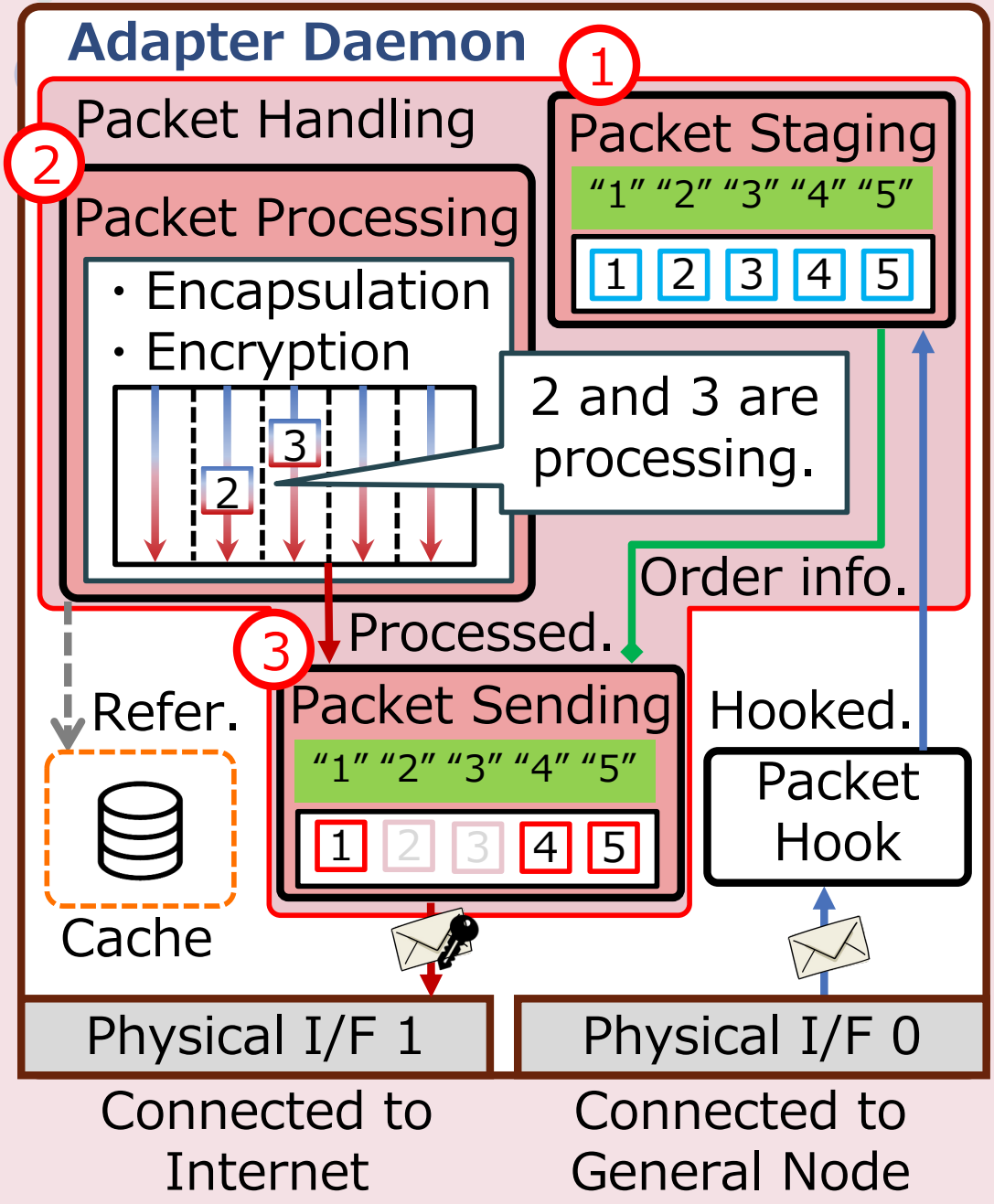
# CYPHONICアダプタのパケット処理




CYP: CYPHONIC Header　　VIP: Virtual IP Header

1. Receiving Module
   - ➡ ノードから取得したパケットを格納する受信キューを準備
2. Packet Handling Module
   - ➡ 受信キューを参照してパケットを処理
3. Sending Module
   - ➡ 送信可能となった処理済みパケットを逐次的に送信

# フラグメントパケットの順序維持機構



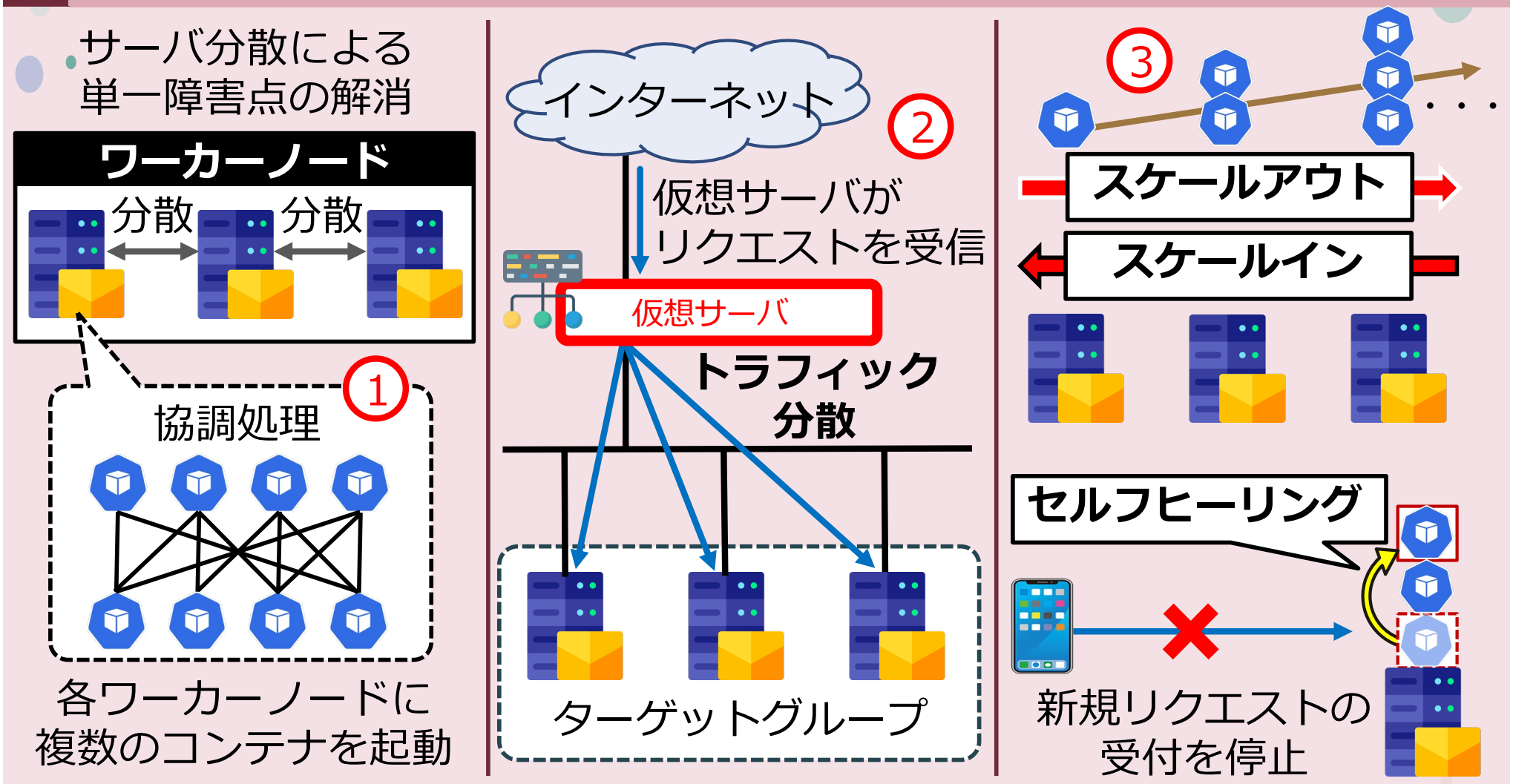

1. Packet Staging Module
受信パケットをバッファに格納し受信順序を保存

2. Packet Processing Module
ワーカースレッドに引き渡してカプセル化・暗号化を非同期実行

3. Packet Sending Module
処理済みパケットをキューに追加して受信順序に基づき送信

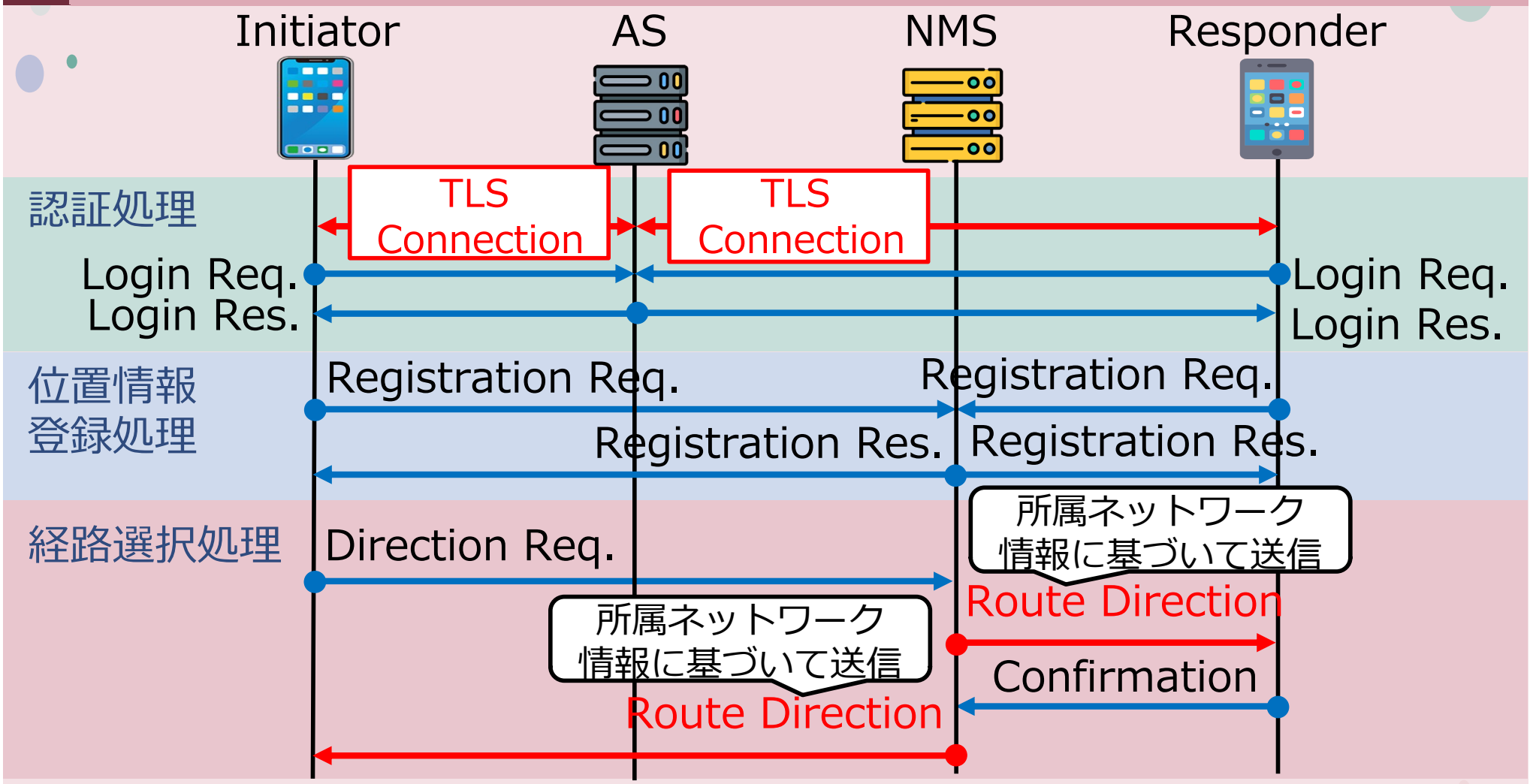
ワーカースレッドの処理状況に依らずフラグメントパケットの順序維持が可能

# 高可用性を実現する一般的な手法



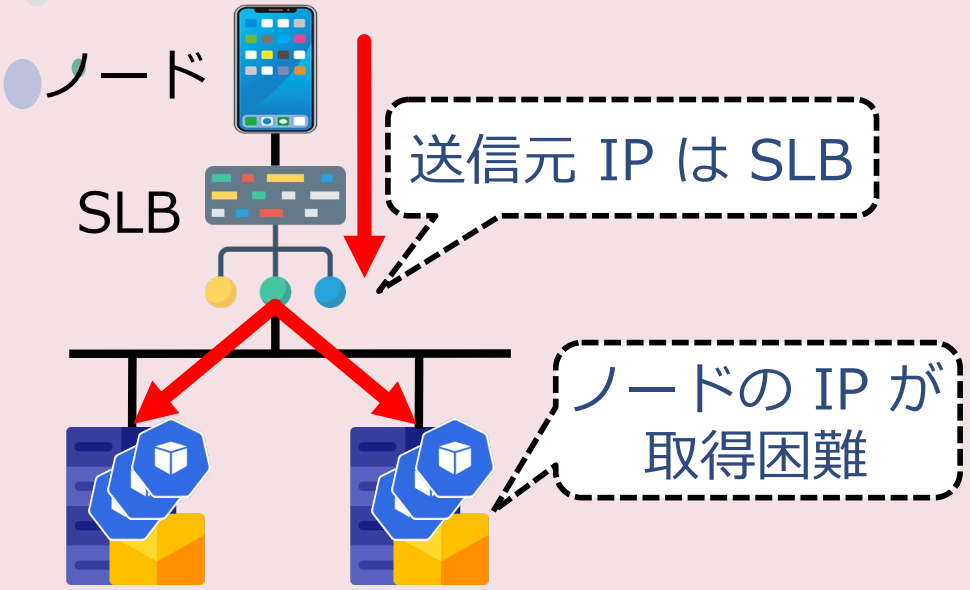

1. コンテナによるクラスタリングおよびサービス機能の多重化
2. リクエスト分散によるサーバ負荷の平準化
3. 状況に応じたスケーリングと自動的な障害復旧

# CYPHONICのシグナリングシーケンス

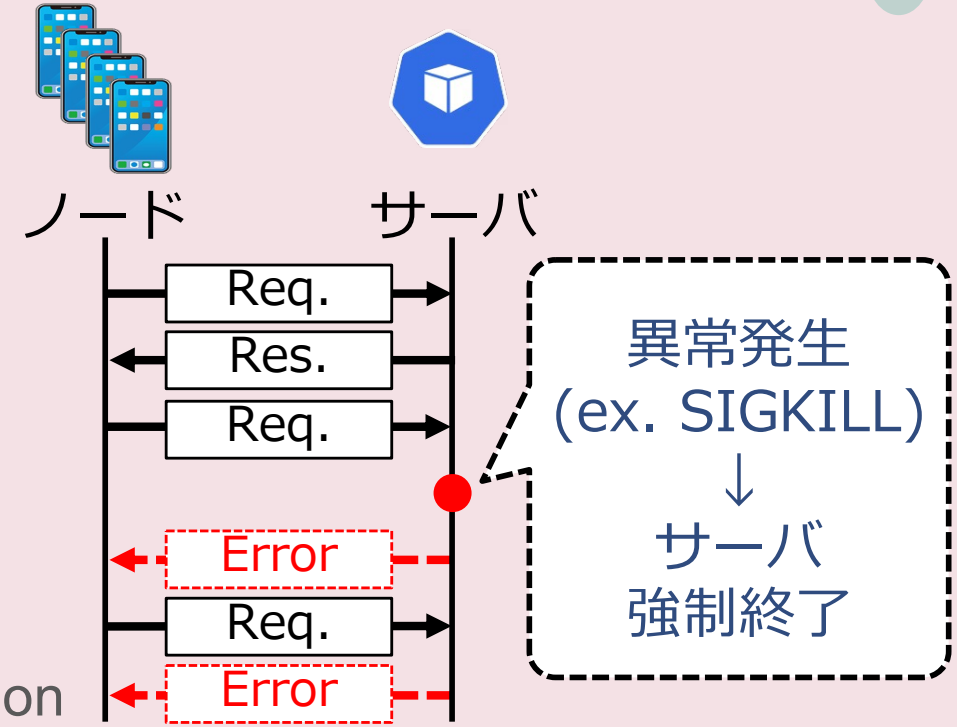



- 認証処理：TLSコネクションを確立して認証要求を送信
- 位置情報登録処理：端末の所属ネットワーク情報を取得
- 経路選択処理：ネットワーク情報に基づいて経路指示

# スケール設計に伴う考慮事項



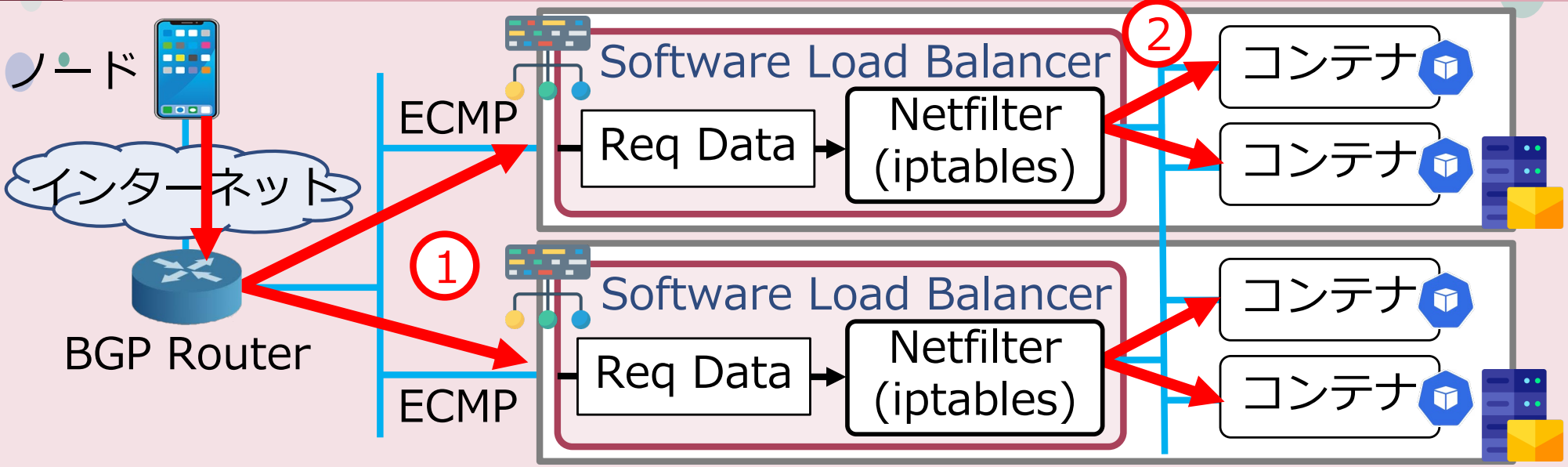

SLB: Software Load Balancer
SNAT: Source Network Address Translation




- 負荷分散時に送信元 IP を SNAT してリクエストを転送
- エンドノードの所属ネットワーク情報の正確な取得が必要
  ➡ 送信元 IP アドレスを維持した負荷分散を考慮

- スケーリングに伴いサーバの強制終了が発生する可能性
- 既存コネクションの処理を正常に終了した後の終了が必要
  ➡ クラウドサービスの Graceful Shutdown を考慮

# 送信元IPを維持した負荷分散機構




BGP: Border Gateway Protocol　ECMP: Equal Cost Multi Path

1. クラスタ前段：BGP における ECMP を活用して負荷分散

送信元 IP を維持してリクエストをワーカノードに分散

➡ ワーカノード自体の負荷を平準化

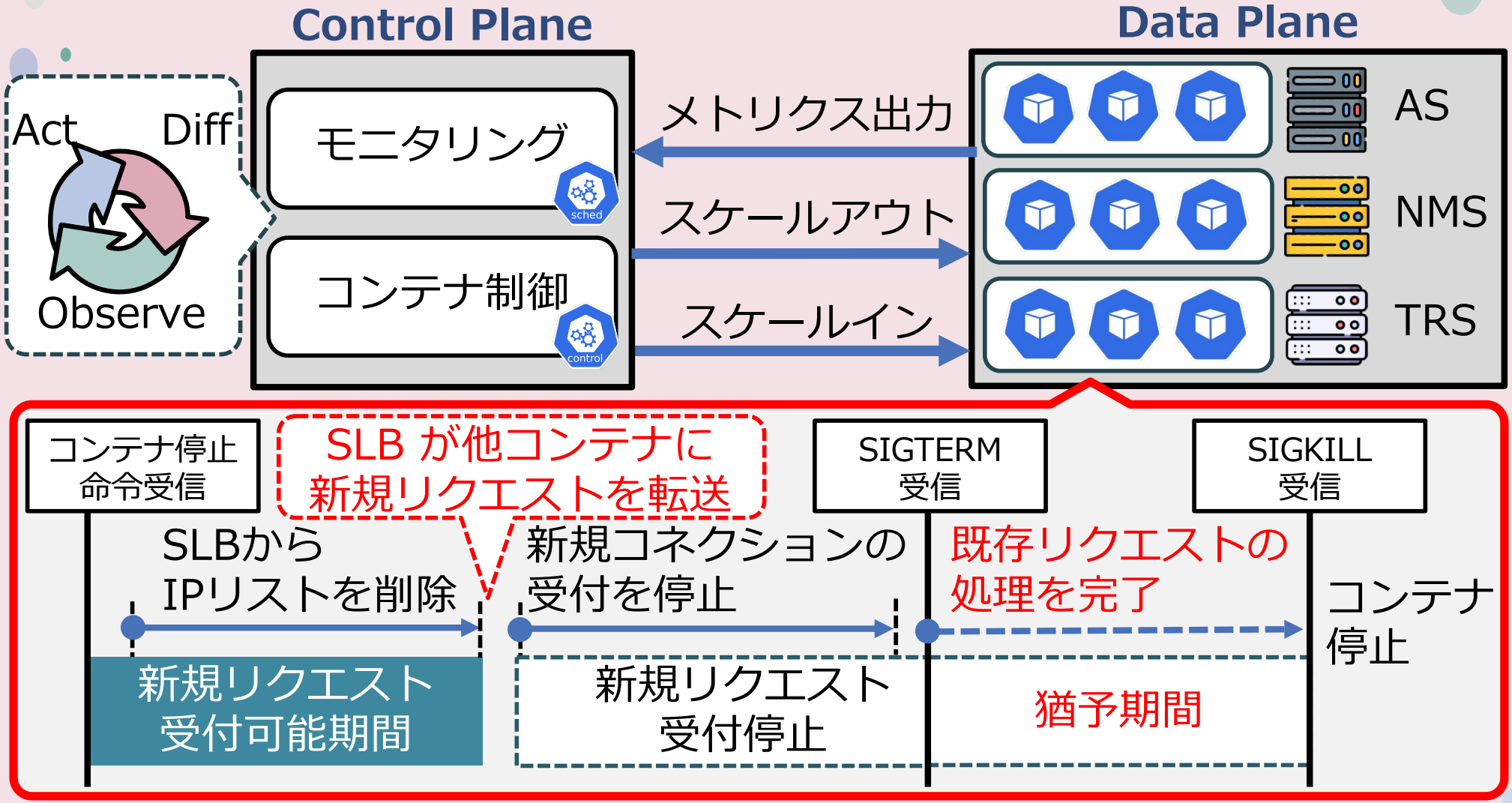
2. クラスタ後段：ワーカノードの Netfilter による負荷分散

ワーカノードに流入したリクエストを各コンテナに分散

➡ コンテナにおけるリクエスト処理の負荷を平準化

2段階の分散により送信元IPの維持と負荷平準化を実現

# サーバコンテナ群のオートスケーリング




SLB: Software Load Balancer

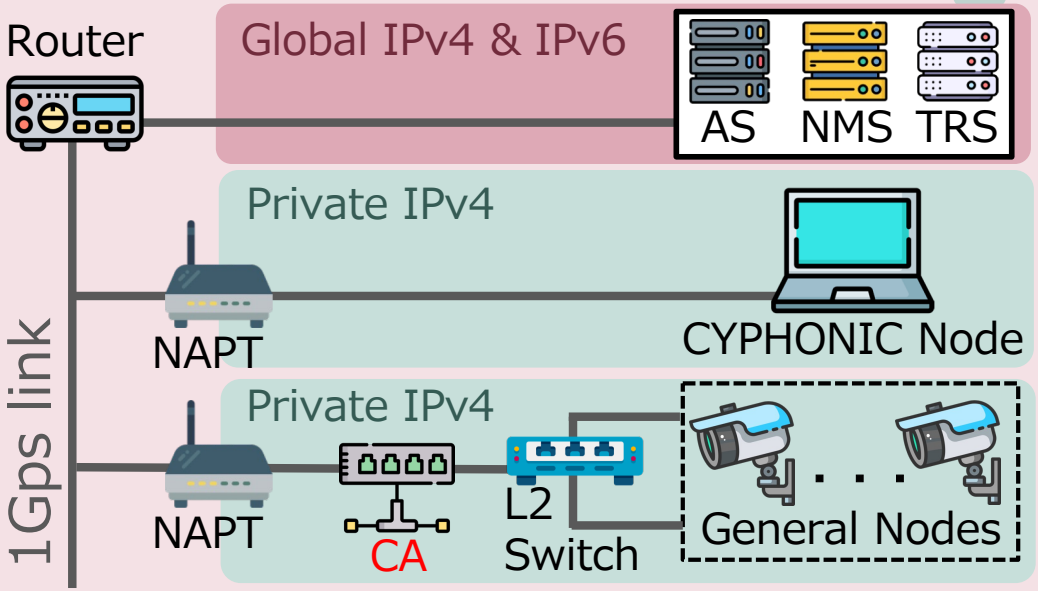
スケールアウト：メトリクス (ex. CPU使用率, RAM使用率) の閾値超過

スケールイン：新規リクエストを転送するとともに既存の処理を完了

# CYPHONICアダプタの検証

一般ノードの通信性能で評価

- 通信遅延時間の測定
  - ping を使用
- 通信スループットの測定
  - iperf3 を使用




**CYPHONIC Adapter (CA) - 1**

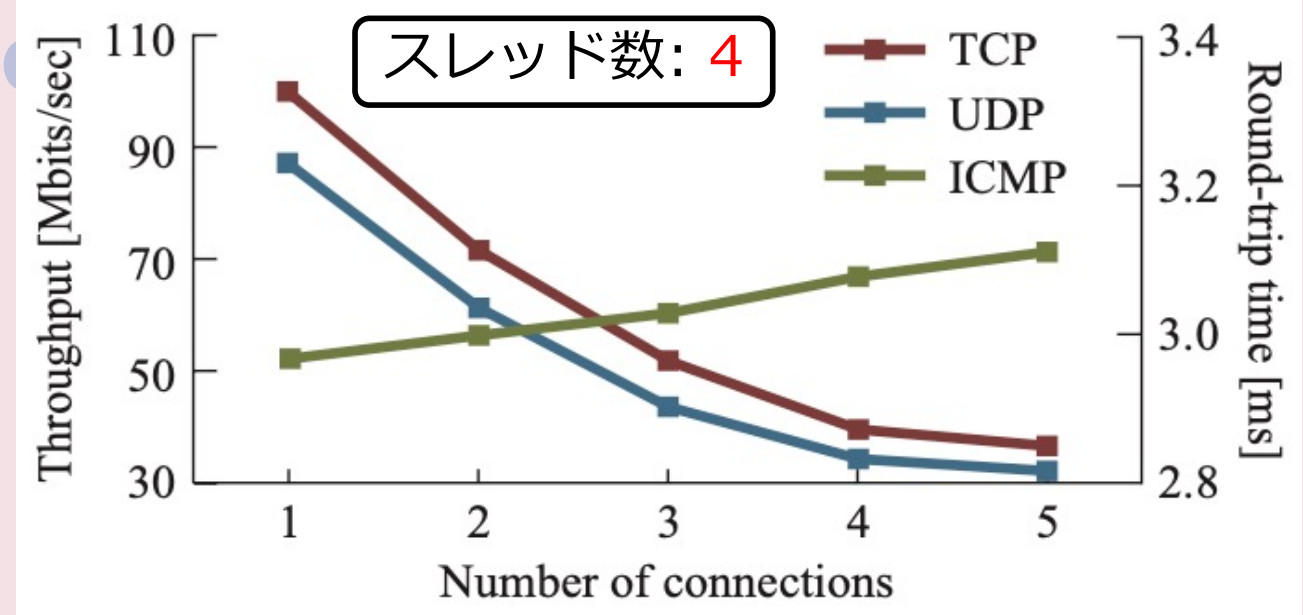
| | |
|---|---|
| Machine | Raspberry Pi Model B |
| OS | Debian GNU/Linux |
| CPU | Cortex-A72 BCM2711 @1.5GHz 4cores |
| RAM | 8 GiB LPDDR4-3200 |

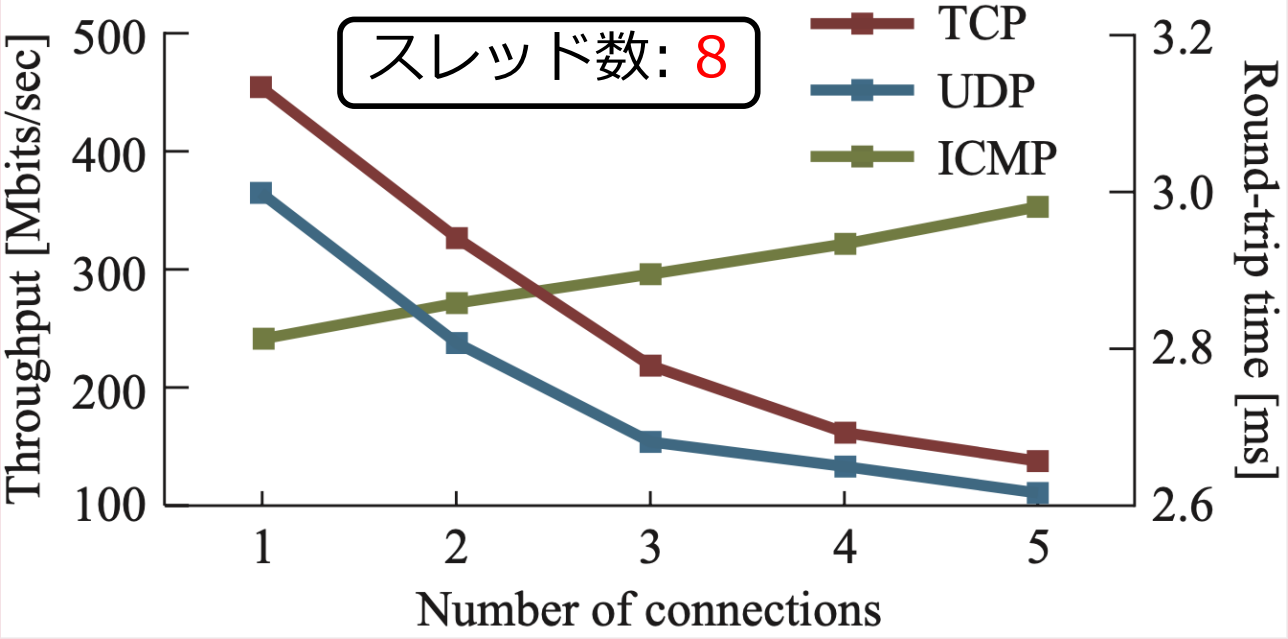
**CYPHONIC Adapter (CA) - 2**

| | |
|---|---|
| Machine | Yoga Slim 7-14ITL05 |
| OS | Ubuntu 22.04.3 |
| CPU | Intel(R) i5-1135G7 @2.40GHz 4cores |
| RAM | 8 GiB DDR4-3200 |

- 各NAPT下に配置した一般ノードとCYPHONICノードが通信
- アダプタに使用する機器のスペックを変化させて計測

# CYPHONICアダプタの検証結果




従来のアダプタシステム
(※一般ノード 1 台のみ)

| | |
|---|---|
| TCP | 37.3 Mbits/sec |
| UDP | 30.0 Mbits/sec |
| ICMP | 3.27 ms |

マルチスレッド化により
一般ノードの通信品質の
大幅な改善を確認

ワーカースレッド数の
増加によりさらなる
性能向上が可能

# 実運用を想定した考察

## CYPHONICアダプタの評価結果

- 提案システムを用いた一般ノードの通信品質を測定
  ➡ マルチスレッド化により通信品質の大幅な改善を確認
- 処理モジュールに割り当てるワーカースレッド数を変化
  ➡ ワーカースレッドの増加でさらなる性能の向上が可能

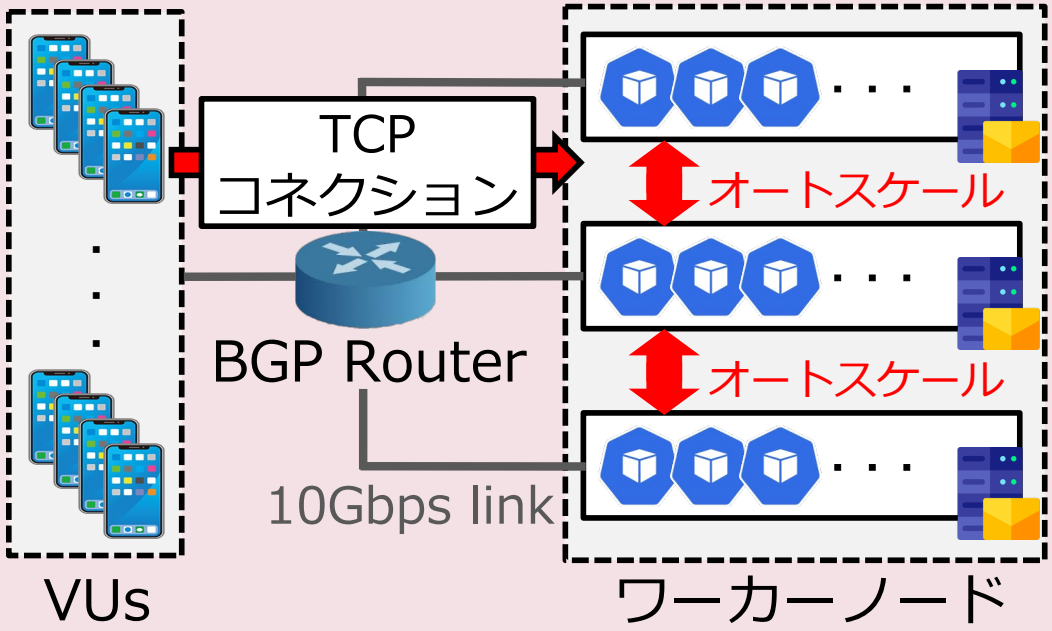
## 防犯カメラ（定常的なストリーム配信）を複数台設置する例

- 結果より 5 台の場合, 1 台あたり 4K, 30 fps 程度を処理可能
- 一般的に設置される防犯カメラは 3 ~ 5 fps 程度で十分
- 動画データの処理方式やCYPHONICアダプタのスペックを上げることでさらに多くの一般ノードを収容可能

# CYPHONICクラウドの検証

クラウドを Kubernetes 上に構築

負荷分散・オートスケールの検証




SLB: Software Load Balancer
VUs: Virtual Users

| Worker Node 3 台 | |
|---|---|
| OS | Ubuntu 22.04.3 |
| CPU | Intel(R) i9-13900 @5.60GHz 24cores |
| RAM | 128 GiB DDR4-3200 |

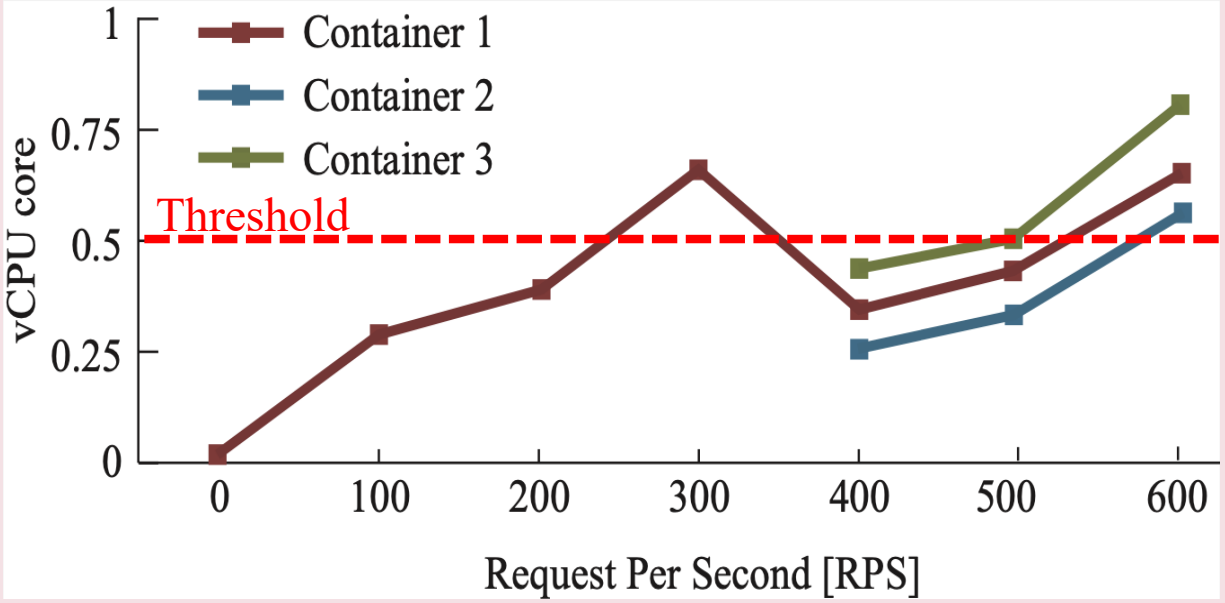
| BGP Router (Kernel-based Virtual Machine) | |
|---|---|
| OS | VyOS 1.5 |
| CPU | Intel(R) i9-11900K @3.50GHz 8cores |
| RAM | 32 GiB DDR4-3200 |

- 検証 1：送信元 IP アドレスを維持した負荷分散を確認
- 検証 2：リクエスト数の増加に伴うスケールアウトを確認
- 検証 3：スケールインに際するデータ損失の確認

# CYPHONICクラウドの検証結果

- 毎分 100 RPS ずつリクエストを増加

CPU使用率 50% におけるスケールアウト傾向




終了までの待機時間：0s

| | |
|---|---|
| 総リクエスト数 | 54230 |
| 500番台応答 | 2371 |
| エラー率 (%) | 4.37 |

終了までの待機時間：5s

| | |
|---|---|
| 総リクエスト数 | 54230 |
| 500番台応答 | 0 |
| エラー率 (%) | 0 |

検証 1：等コストパスルーティングを活用した負荷分散を確認
➡ ノードの所属ネットワーク情報を正確に取得可能

検証 2：単一コンテナの平均 CPU 使用率に基づいたスケールアウトを確認
➡ リクエストの増加に伴うサーバ負荷の平準化が可能

検証 3：コンテナ終了までの待機時間を延ばすことでエラー率の改善を確認
➡ スケールインの影響をエンドノードに対して隠蔽可能

# まとめ

CYPHONIC における
サービス拡張性実現手法の提案

1. CYPHONICアダプタにおける複数一般ノードのサポート
- 送受信パケット処理機能のマルチスレッド化
- フラグメントパケットの順序維持機構の追加

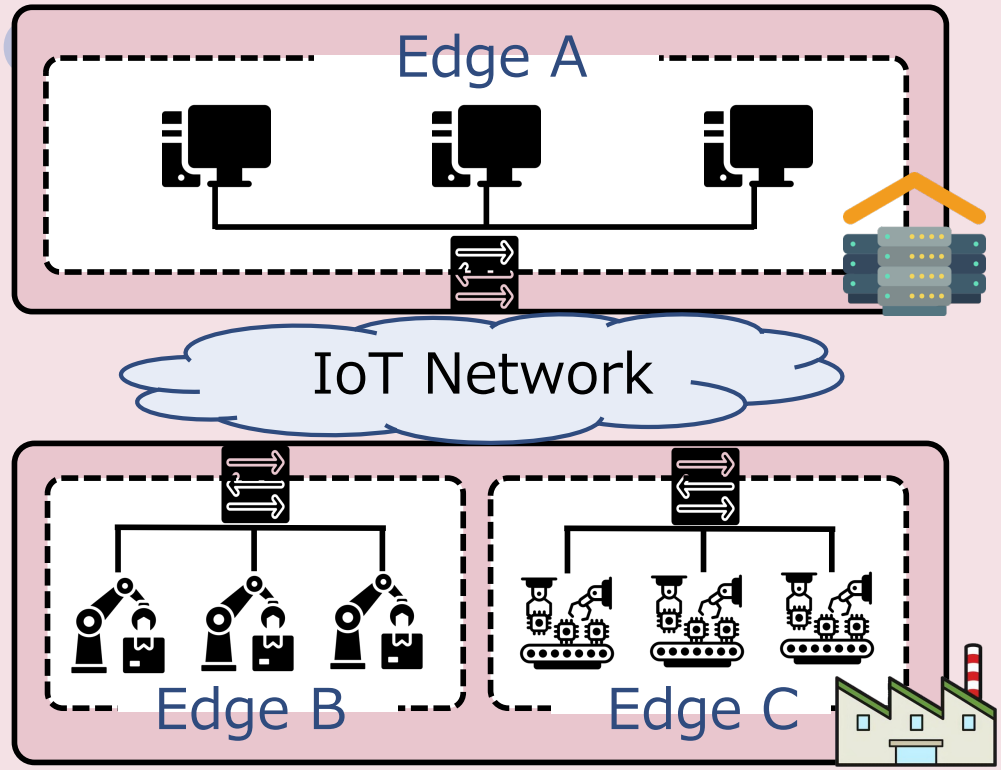
2. CYPHONICクラウドの障害許容性および高負荷耐性の実現
- コンテナによるクラスタリングとサービス機能の多重化
- オートスケーリングと負荷分散機能の追加

1. 処理機能のマルチスレッド化による通信性能の改善を確認
2. オートスケールと負荷分散機能が有用であることを確認

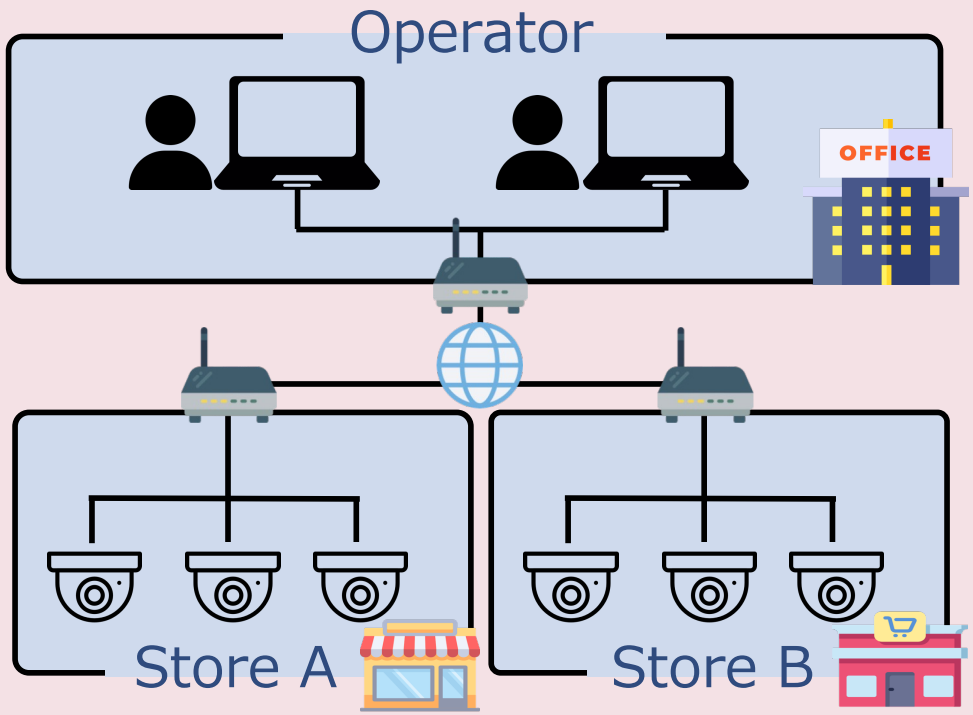
# 以下、予備スライド

# CYPHONICアダプタに関する予備資料

# 近年の IoT 利活用形態




ファクトリオートメーション



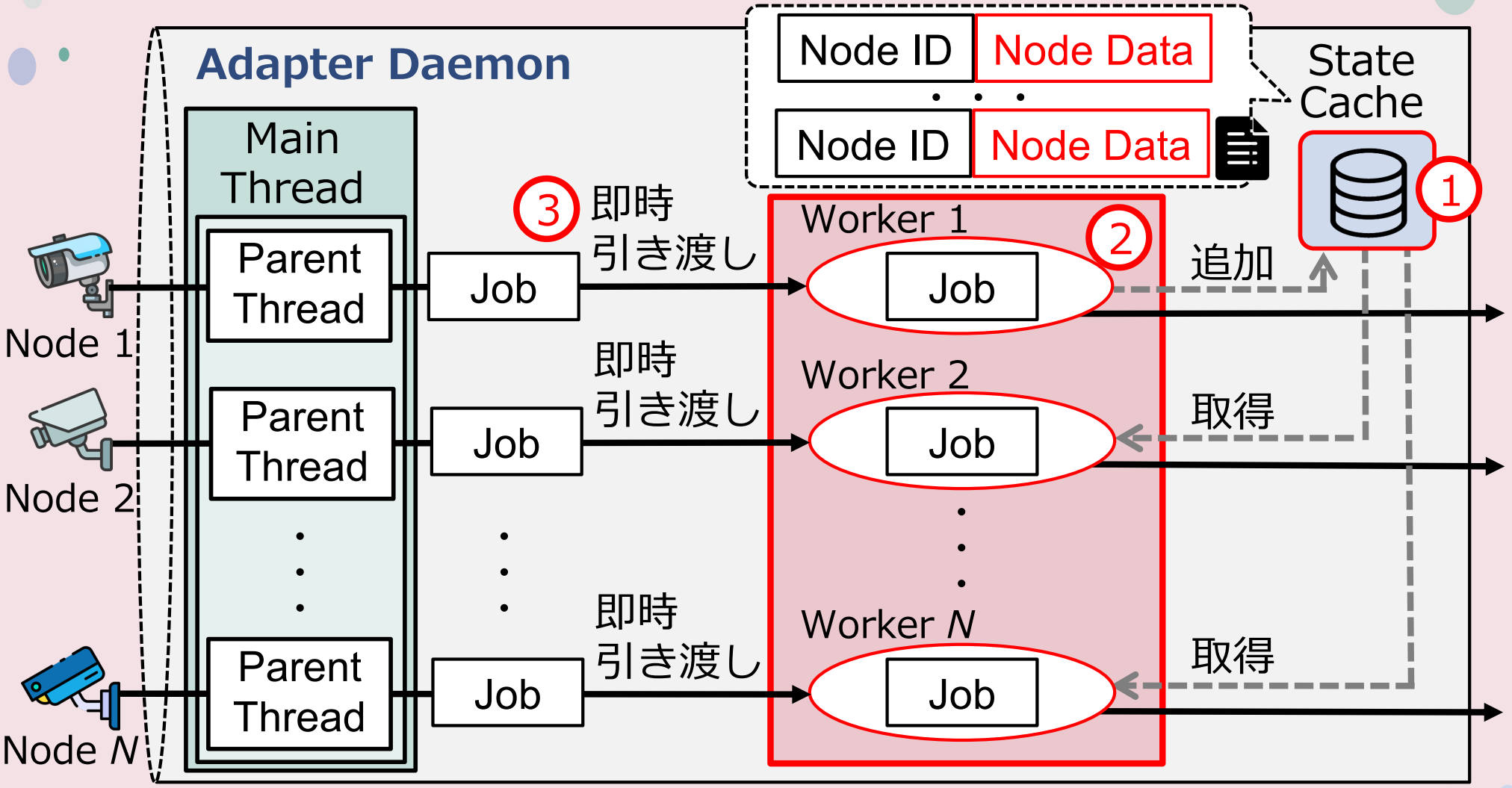

監視カメラーソリューション

- ファクトリオートメーション：多数の IoT 機器を連携した自動化システム
- 監視カメラソリューション　：IoT 機器を用いた複数区画からのデータ収集

膨大な IoT 機器をセキュアに接続する技術が必要

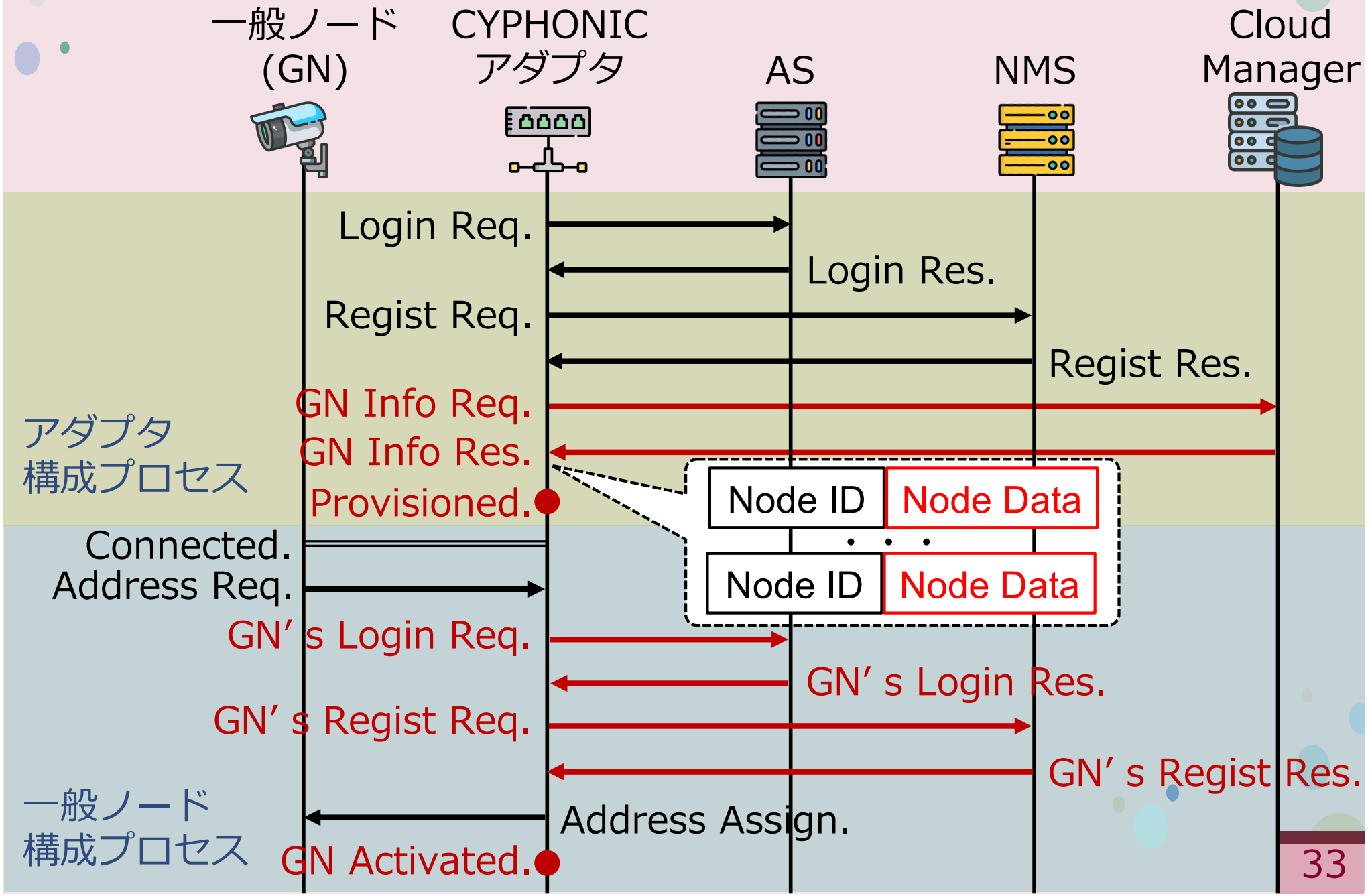
➡ CYPHONICアダプタによりユースケースに対応可能

# マルチスレッド処理モデル



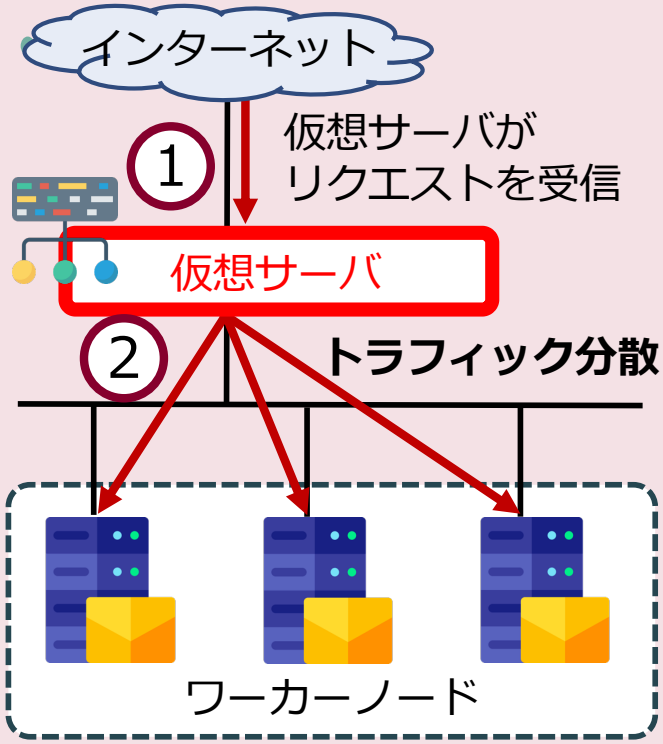

1. ステート情報を管理するキャッシュストアを準備
2. 親スレッドとは別にワーカースレッドを事前に生成
3. ジョブをワーカースレッドに引き渡して並行処理

# シグナリングプロセスの拡張



# CYPHONICクラウドに関する予備資料

# ロードバランサの導入と負荷分散の課題




**ロードバランサの一般的な考え方**

1. 単一エンドポイント（ロードバランサ）が
   リクエストを一括受

2. バックエンド（ワーカーノードグループ）に
   トラフィックを分散

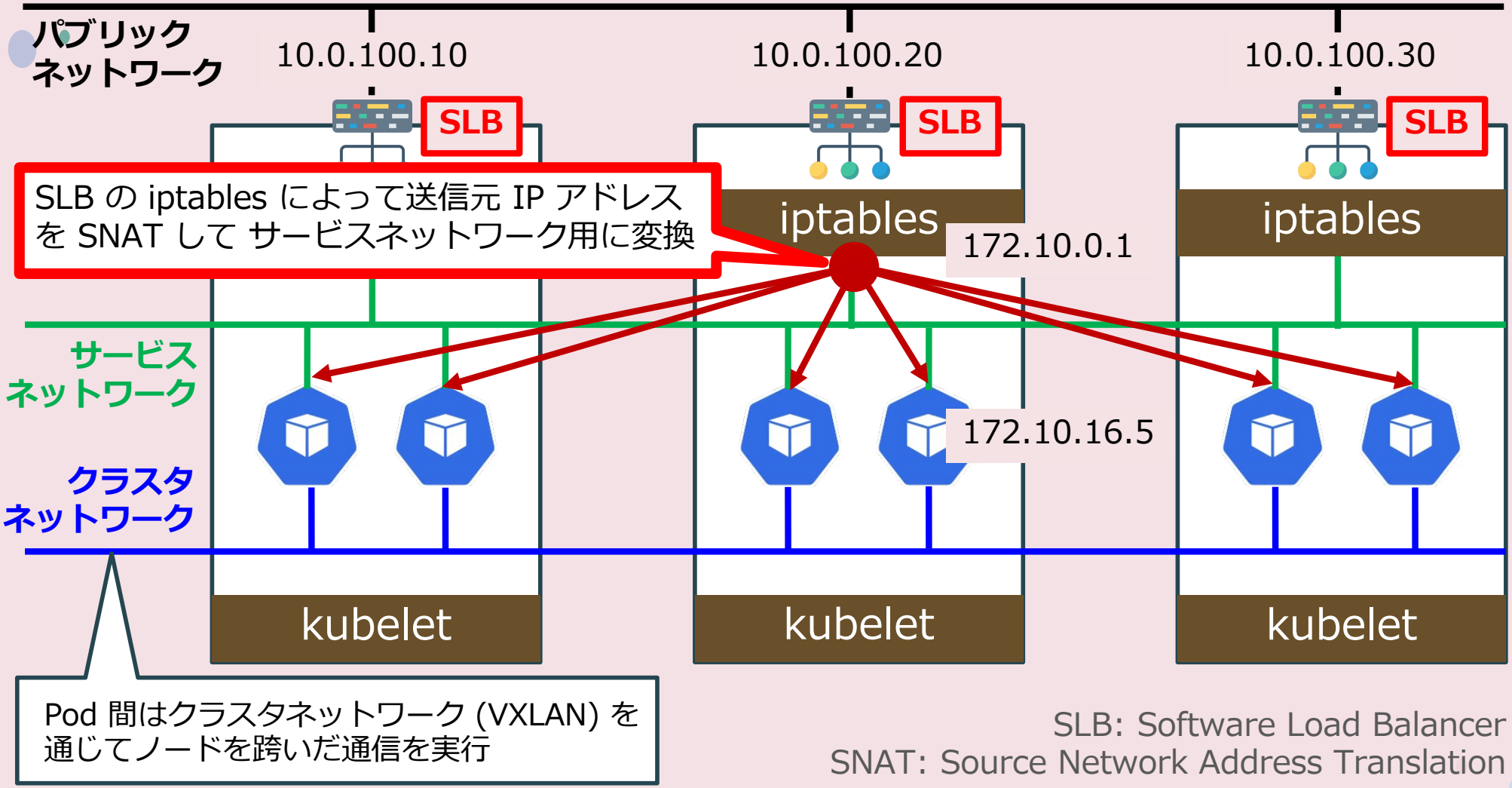
**単一ノード辺りの負荷を下げる**

リクエストを分散する際にクラスタネットワークに有効なアドレスに変換

ロードバランサは SNAT によってワーカーノードにトラフィックを分散

送信元 IP アドレスが不明となるため位置情報登録処理や経路確立処理が困難

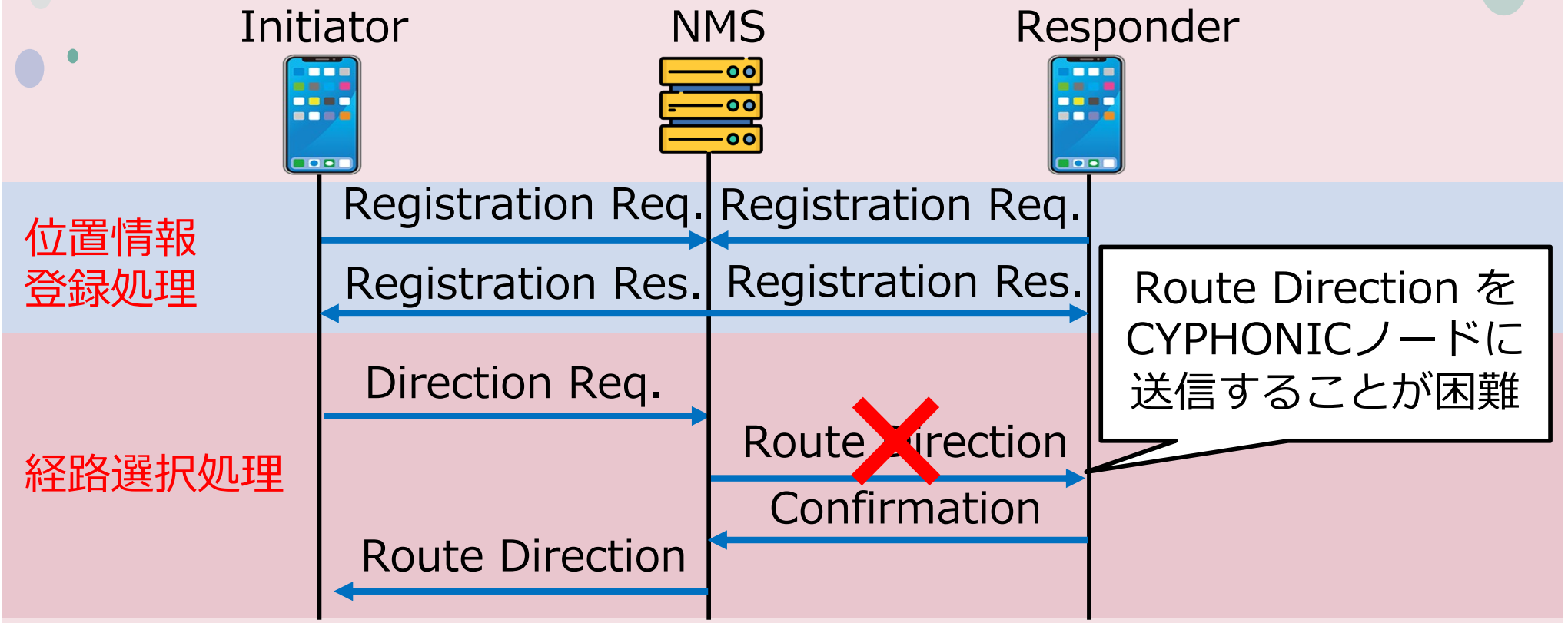
# SLB-SNAT による送信元 IP の特定困難性




SLB: Software Load Balancer
SNAT: Source Network Address Translation

- エンドノードは SLB (仮想サーバ) に向けて通信リクエストを送信
- SLB (iptables) はリクエストをサービスネットワークに転送する際に送信元 IP アドレスをクラスタ内で有効な IP アドレスに SNAT

# 位置情報登録処理における SNAT の課題




NMS は位置情報登録処理によって取得した CYPHONIC ノードの
ネットワーク環境情報を元に通信経路を指示

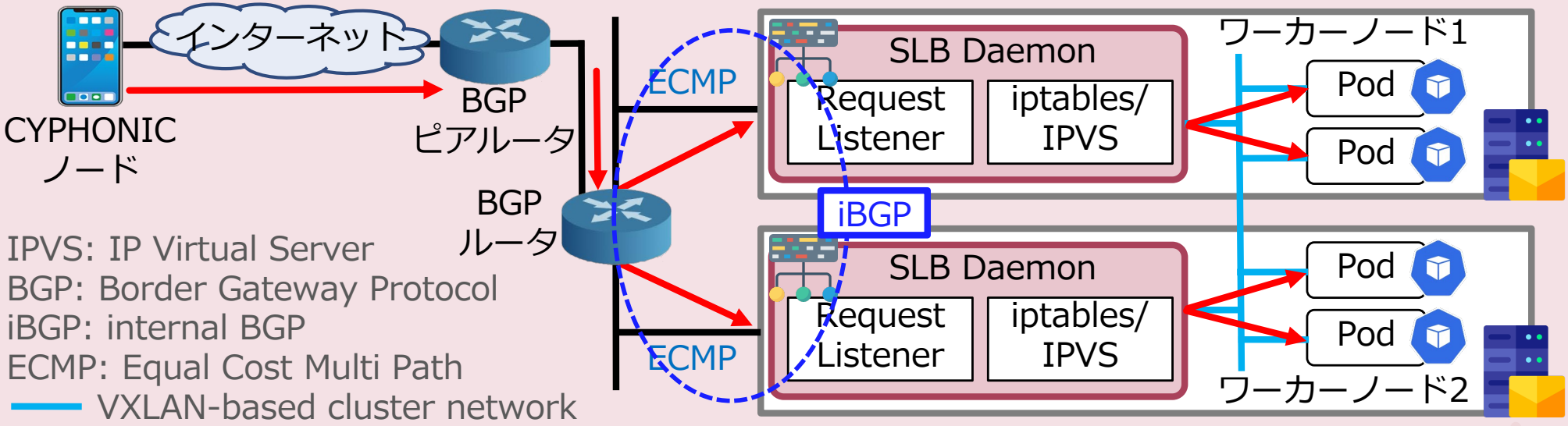
SLB によるトラフィック転送における SNAT によって
全ての CYPHONIC ノードが同一のネットワークにいると判断

経路選択処理における Route Direction の実行が不可能

# BGP によるクラスタワイドな負荷分散

SLB Daemon と BGP ルータを連携した
ECMP による等コストパスベースの負荷分散

- Kubernetes クラスタの前段に BGP ルータを配置
- SLB Daemon はワーカーノード間を iBGP で接続して経路アドバタイズ
- ワーカーノードに Pod がスケジュールされない場合は経路アドバタイズを停止して BGP ルータからネクストホップエントリを削除

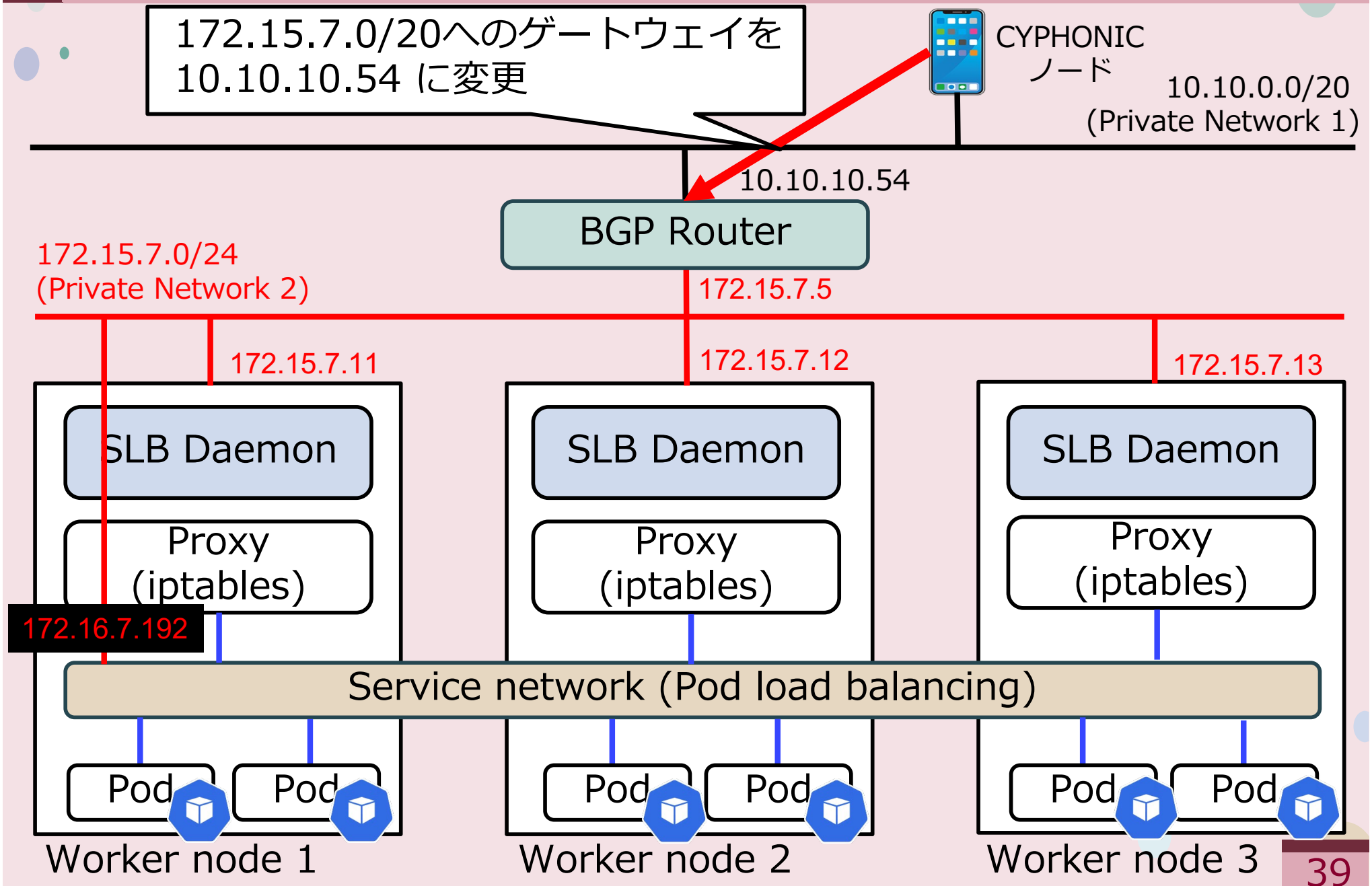



CYPHONIC ノードからのリクエスト

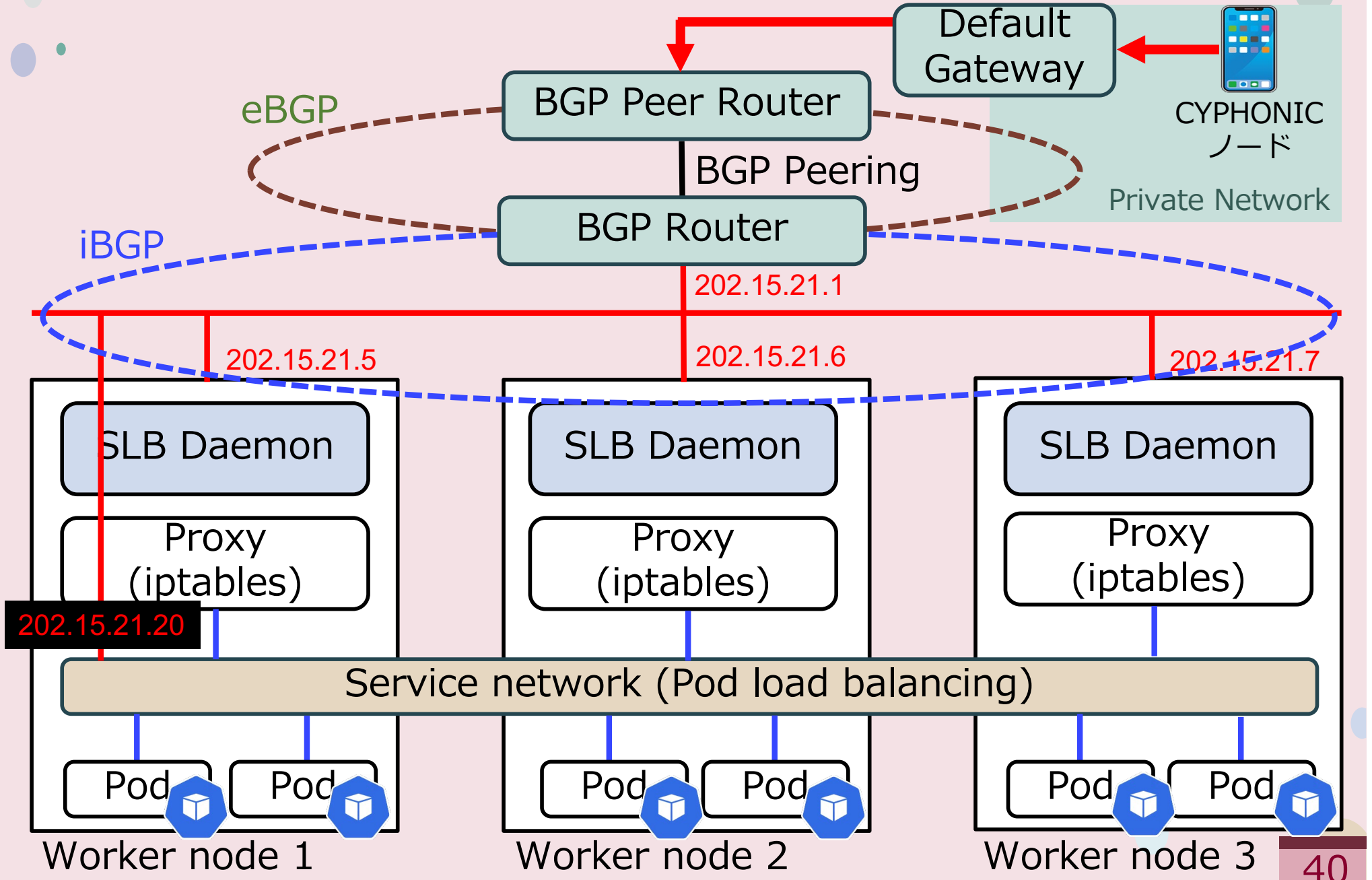
- 各シグナリングは BGP ルータを経由して SLB に送信
- パケットを受信したワーカノード内の Pod にリクエストを分散

➡ **ワーカーノードを跨ぐ負荷分散を防止することで SNAT を回避**

# 検証環境におけるネットワーク構成



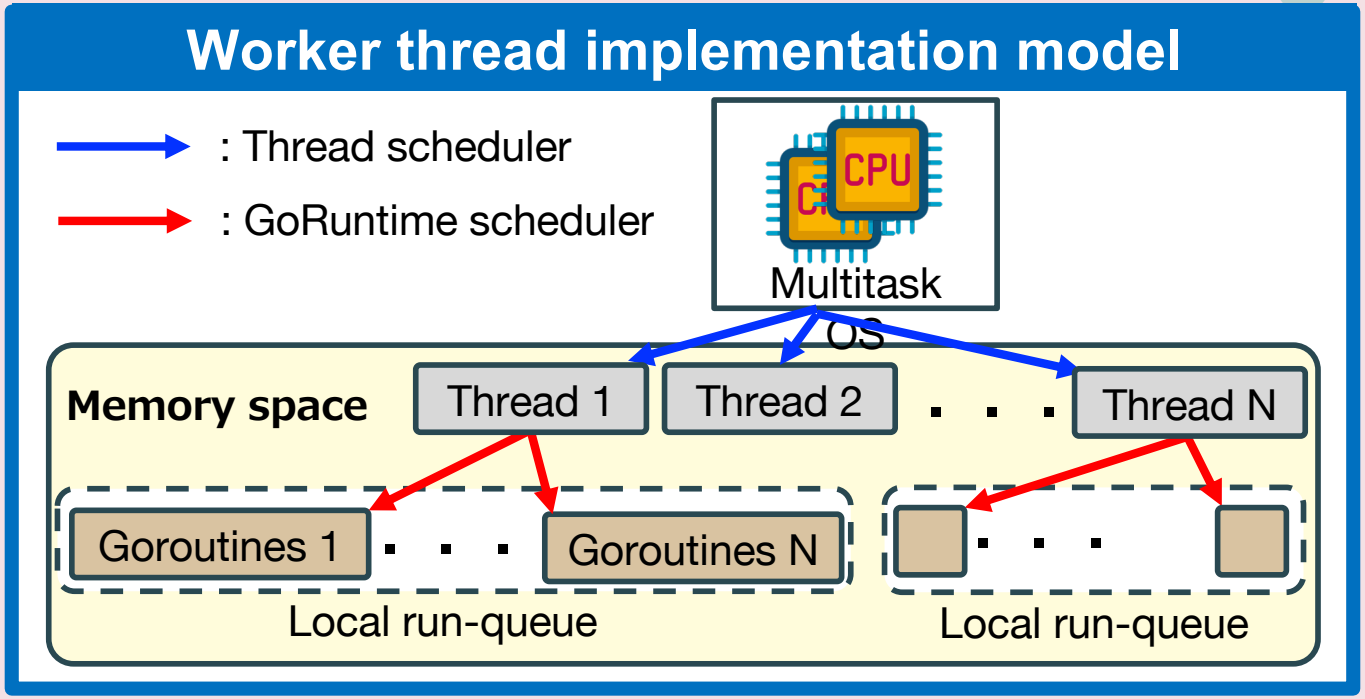
# パブリックネットワークでのクラスタ設計



# CYPHONICクラウド評価に関する予備資料

# イベント駆動アーキテクチャ

| Adapter Daemon | |
|---|---|
| Runtime | Go ver 1.20 |
| Worker thread | Goroutines |
| Mutex | sync package |




**Event-driven model**

- Goroutine creates $M{:}N$ scheduler capable of processing $N$ concurrently for $M$ logical cores.
- Context switches are hidden from the OS.

**Sequencing processing scheme**

- A single packet gets a lock before being passed to a worker thread by mutex and is unlocked when processing is complete.
- Prohibit unauthorized access to packets being processed.

# オートスケールロジック (CPU指標の場合)

HPA: Horizontal Pod Autoscaler

HPA による Pod スケジュールは次の式で表される

*desiredReplicas*
*= ceil[currentReplicas * ( currentMetricValue / desiredMetricValue )]*

*currentReplicas*：現在の Pod 数
*currentMetricValue*：Pod のCPU 使用率
*desiredMetricValue*：スケール閾値

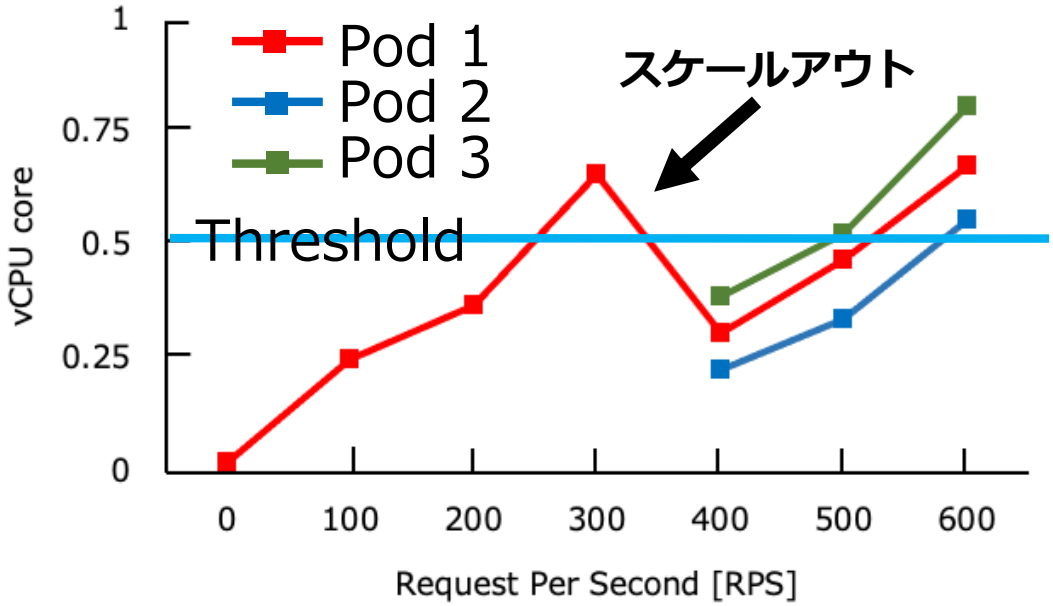
- HPA Controller は Pod のメトリクスを収集してオートスケールを実行
- Kubernetes Controller は Reconciliation によってPod を 規定数に維持

ex. 現稼働数 5 台, メトリクス指標を CPU 使用率として
以下の状況が発生した場合

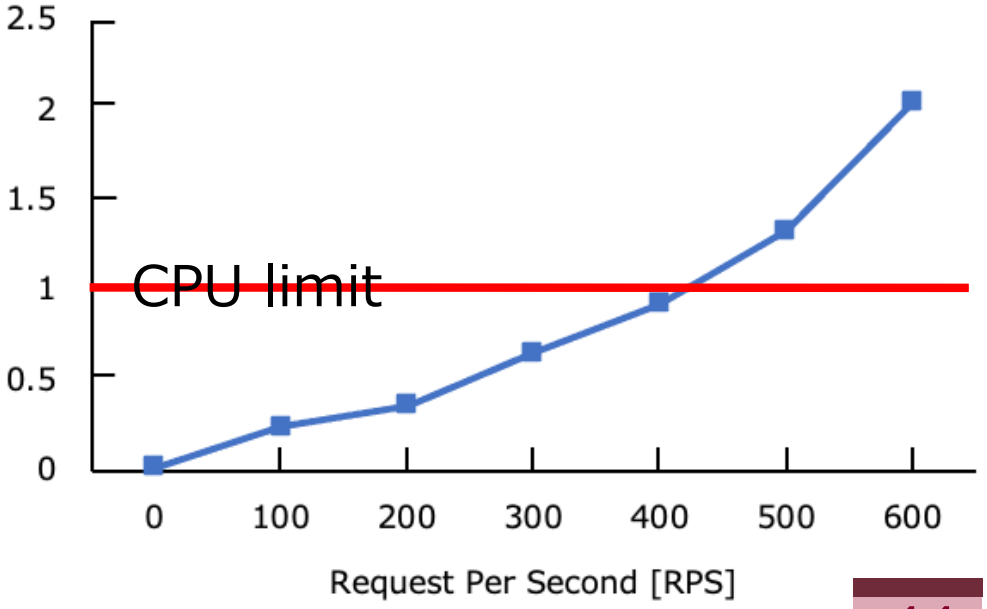
Pod の平均 CPU 使用率合計 90% / CPU 使用率閾値 70% = 1.2857...
➡ サービスサーバ Pod を 2 台スケールアウト

# スケーリング基礎評価

| 時間 (分) | リクエスト数 | 1 Pod コア | 2 Pod コア | 3 Pod コア | 全体使用コア |
|---|---|---|---|---|---|
| 0 | 0 | 0.023 | | | 0.023 |
| 1 | 100 | 0.25 | | | 0.25 |
| 2 | 200 | 0.37 | | | 0.37 |
| 3 | 300 | 0.66 | | | 0.66 |
| 4 | 400 | 0.31 | 0.23 | 0.39 | 0.93 |
| 5 | 500 | 0.47 | 0.34 | 0.53 | 1.34 |
| 6 | 600 | 0.68 | 0.56 | 0.81 | 2.05 |

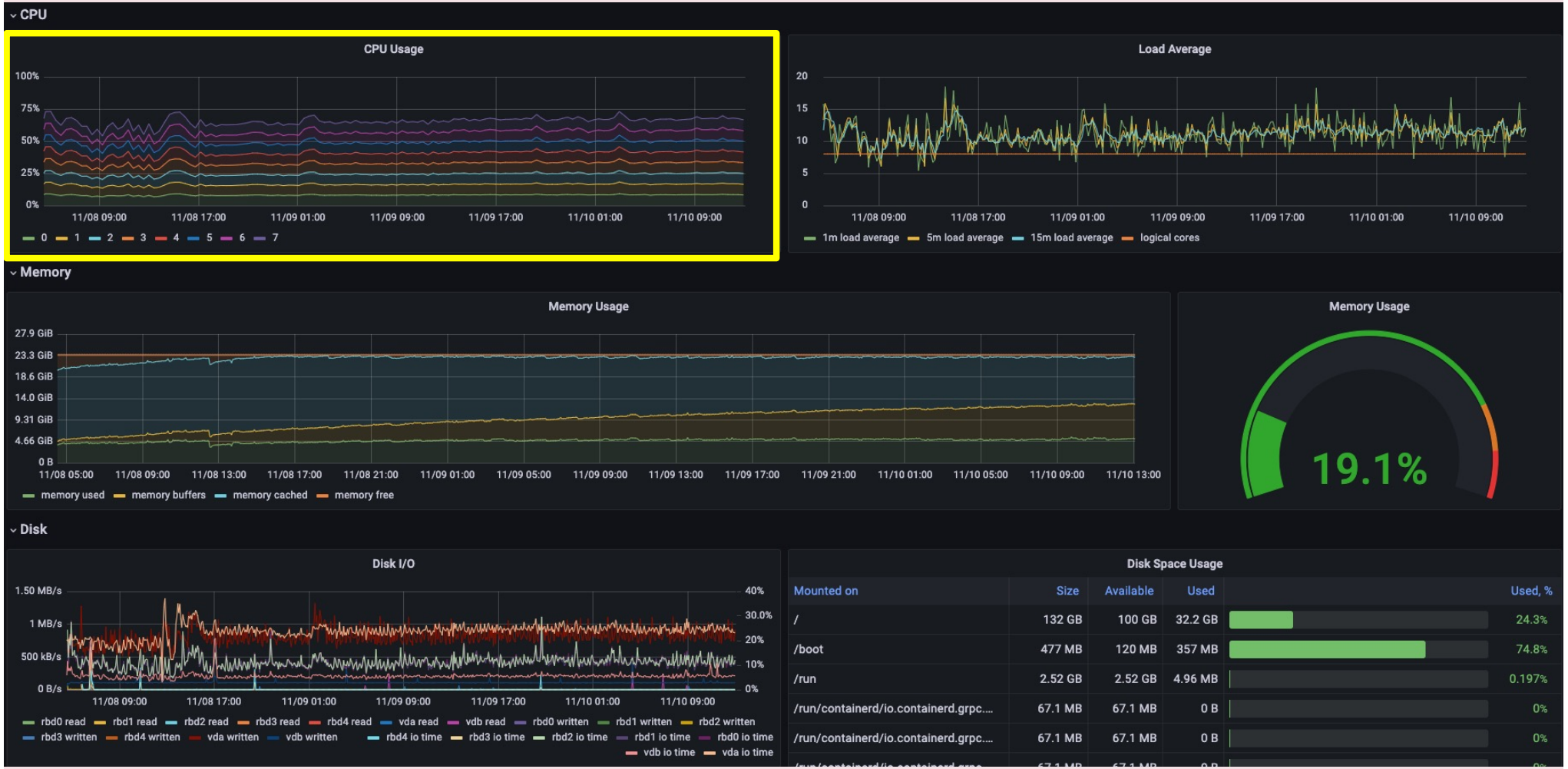



単一 Pod 辺りのCPUコア使用傾向




Pod 全体のCPUコア使用傾向

# Kubernetes 運用に伴うリソース消費傾向




## 構成コンポーネントの大半が Go ベースの実装

- マシンコアをフルに使用することでメモリ確保に伴うオーバーヘッドを抑制
- メモリを削減可能な一方で CPU 使用率が高くなる傾向にあると推測

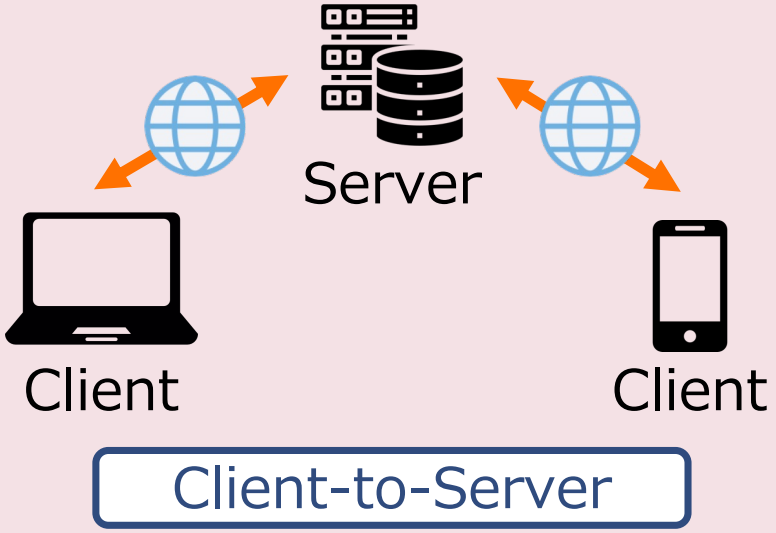
# CYPHONICに関する予備資料

# インターネットサービスモデル

**Client-to-Server (C2S) 型サービス**

(ex. Web サービス, メール配信サービス)

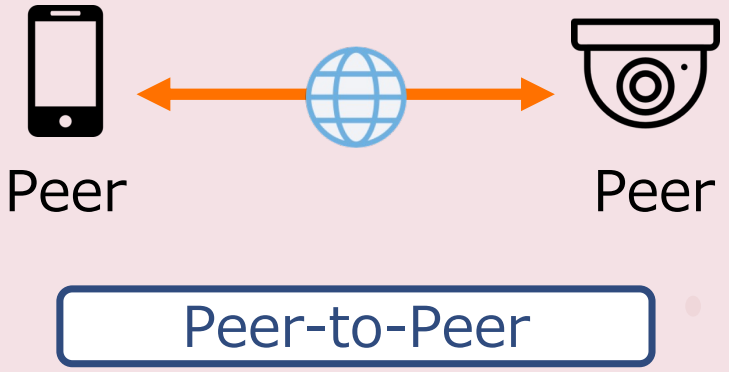
- クライアントはサーバと接続
- クライアントはサーバを介して
  データを送受信




**Peer-to-Peer (P2P) 型サービス**

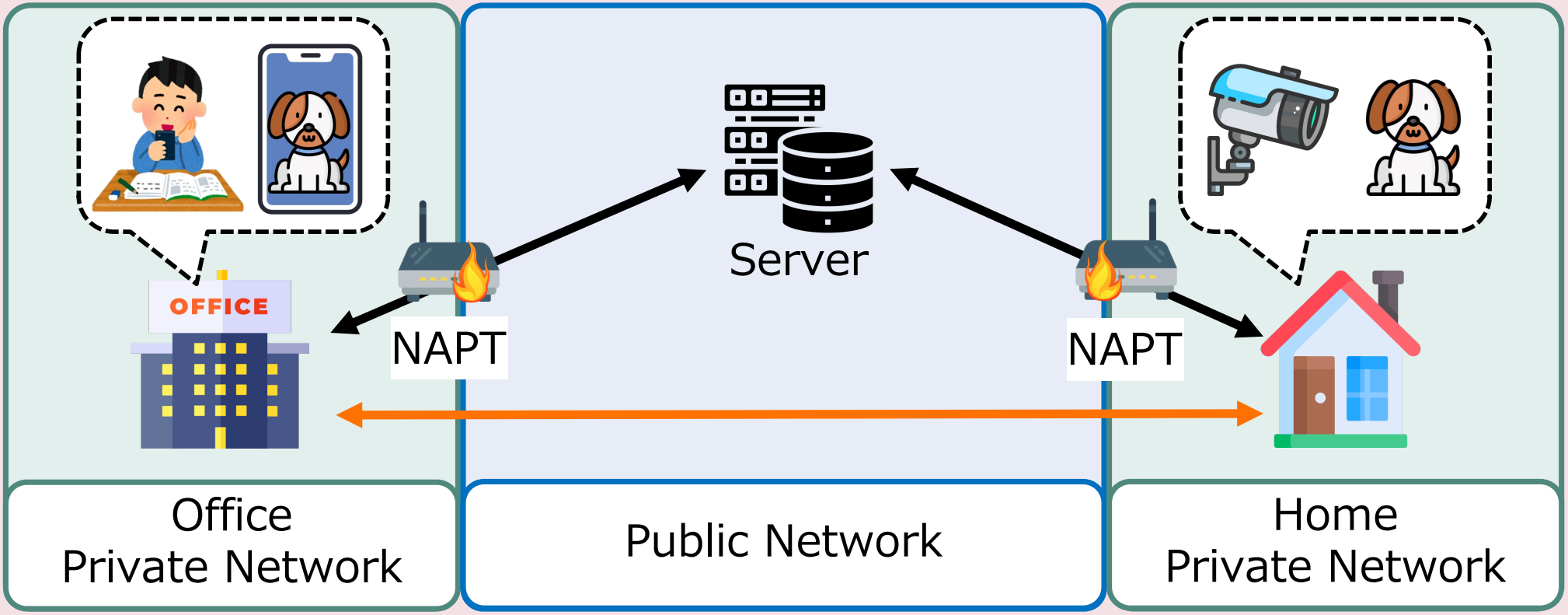
(ex. ビデオ会議・遠隔監視ソリューション)

- 複数のピアを相互に接続
- ピア間で直接データを送受信




IoT の普及により P2P 型サービスが増加

# 現代の Peer-to-Peer 型サービス




: Network model　: Service model

NAPT: Network Address Port Translation

提供されるサービスと実際のネットワーク経路が乖離

- サービスモデル　　：端末間で行う P2P 型モデル
- ネットワークモデル：サーバを経由する C2S 型モデル

# 集中型処理モデルから分散型処理モデルへ

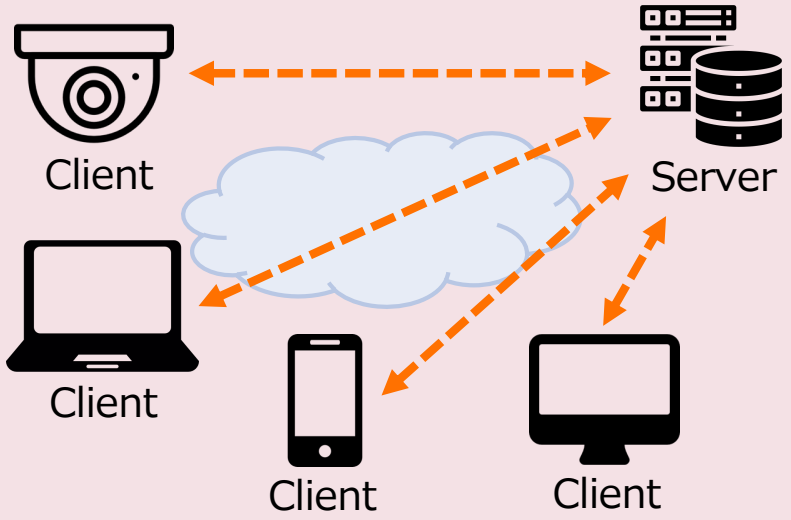
**集中型処理モデル：Client-to-Server**

サーバでの一極集中処理により

端末の管理や通信トラフィックを制御

➡ 通信量の増加に伴うネットワーク網や

サーバへの負荷集中・遅延の増大が懸念



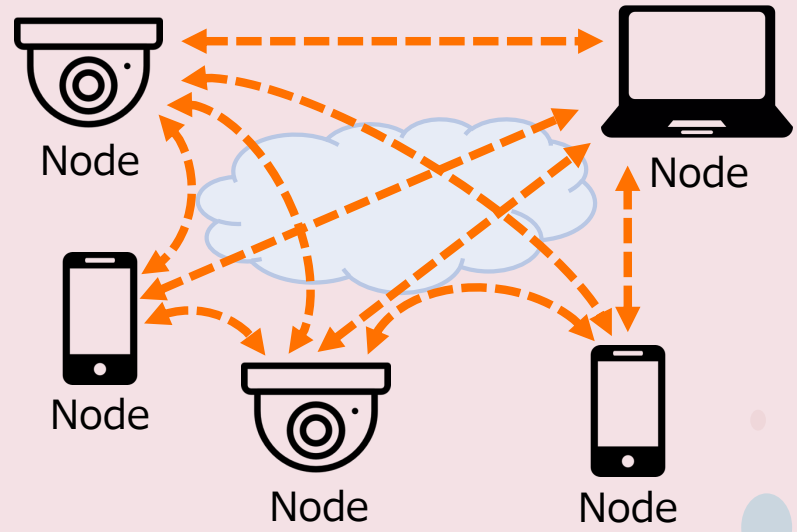

**分散型処理モデル：Peer-to-Peer**

一つのタスクを複数の端末に分散して

協調処理を行うことでサービスを実現

➡ クラウドサービスに対する負荷の軽減や

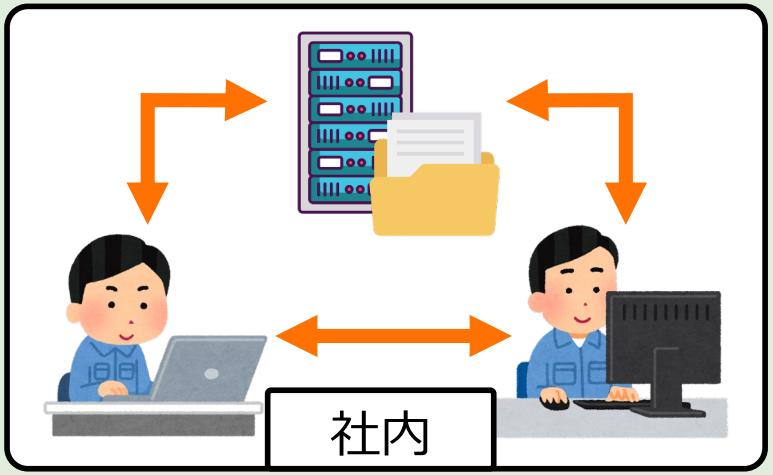
遅延の改善が可能



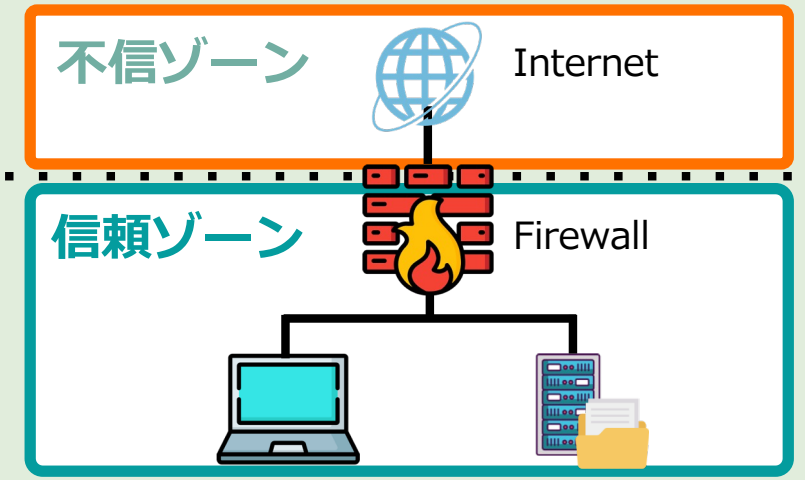

分散処理モデルの実現には Peer-to-Peer による
機器間の相互接続及び直接通信機構が必要

# ネットワーク利用形態とセキュリティモデル

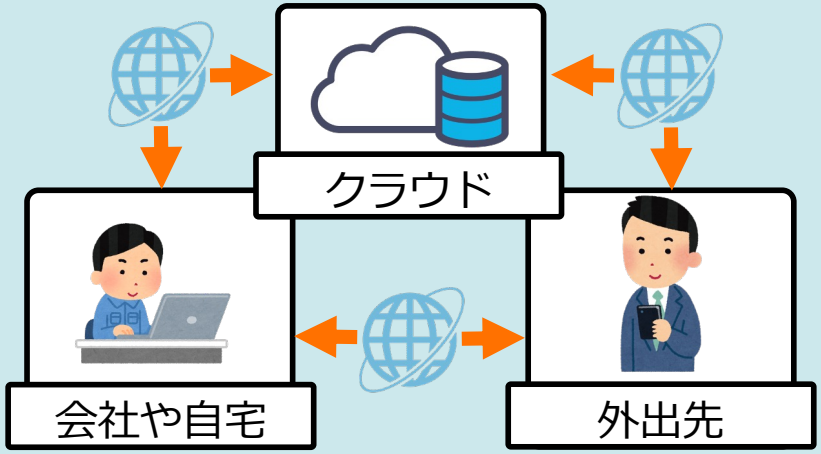
従来の利用形態



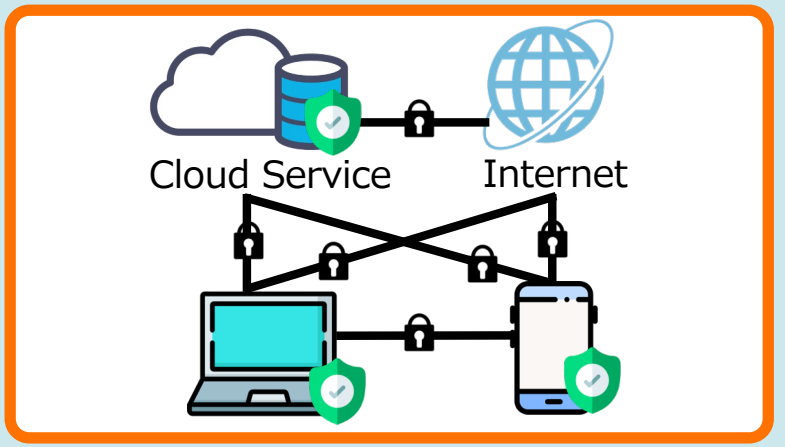

境界型モデル




今後の利用形態

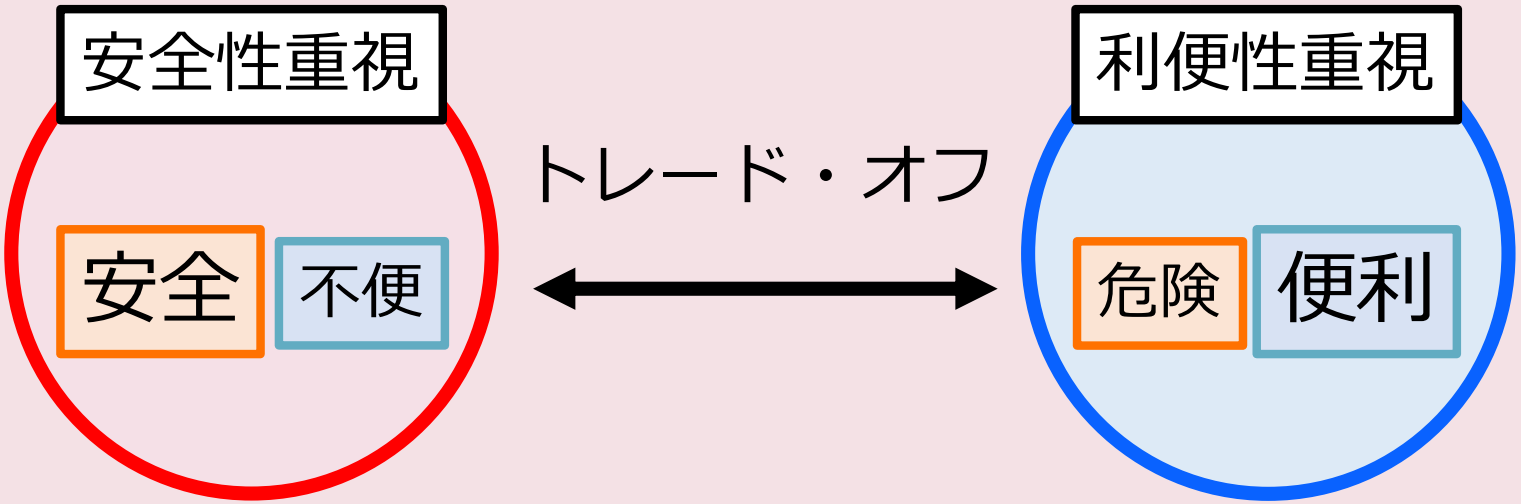



ゼロトラストモデル




インターネット利用形態の多様化に伴いセキュリティモデルも変化

# セキュリティトレード・オフ

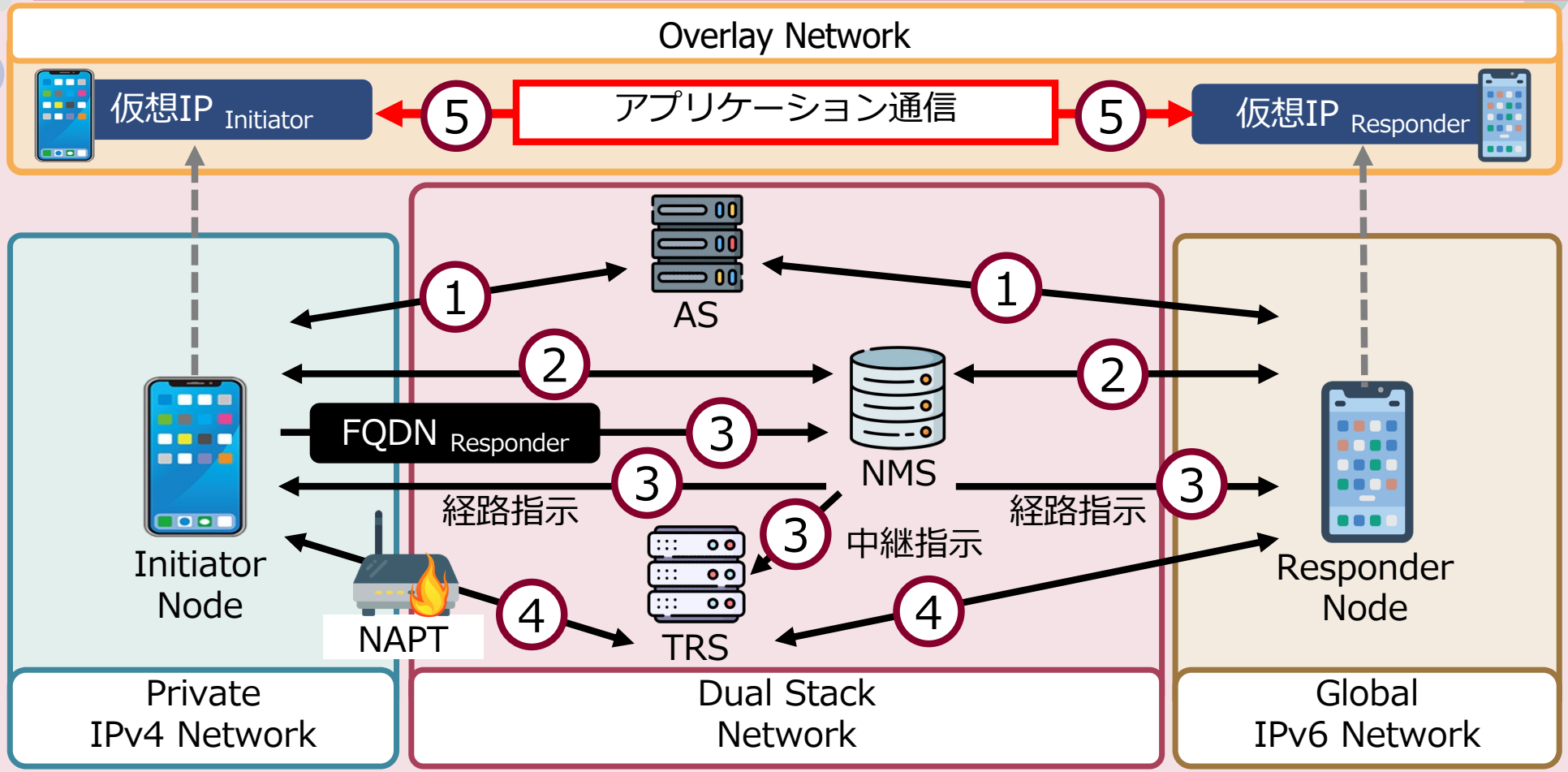



セキュリティ対策において **安全性** と **利便性** は
トレード・オフの関係

現在の複雑なIPネットワーク環境において
セキュリティ対策の複雑化が懸念

既存環境との共生を図りながら
効果的に導入することが重要

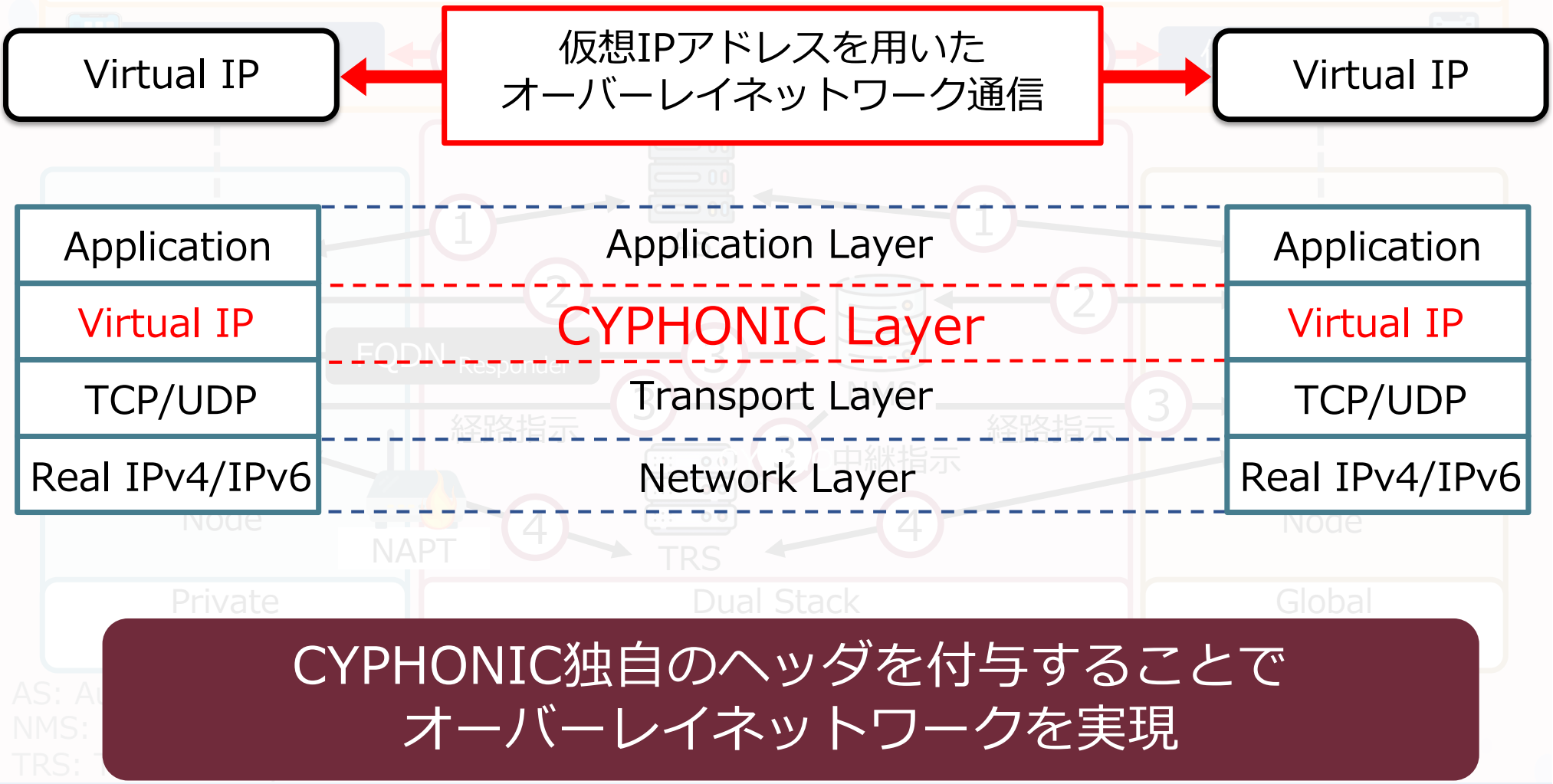
# CYPHONIC 全体概要




AS: Authentication Service
NMS: Node Management Service
TRS: Tunnel Relay Service
NAPT: Network Address Port Translation
FQDN: Fully Qualified Domain Name

1. 認証処理：端末の認証
2. 位置情報登録処理：ネットワーク情報登録
3. 経路選択処理：通信経路決定
4. トンネル確立処理：エンド間トンネル構築
5. データ通信処理：オーバーレイネットワーク上の通信

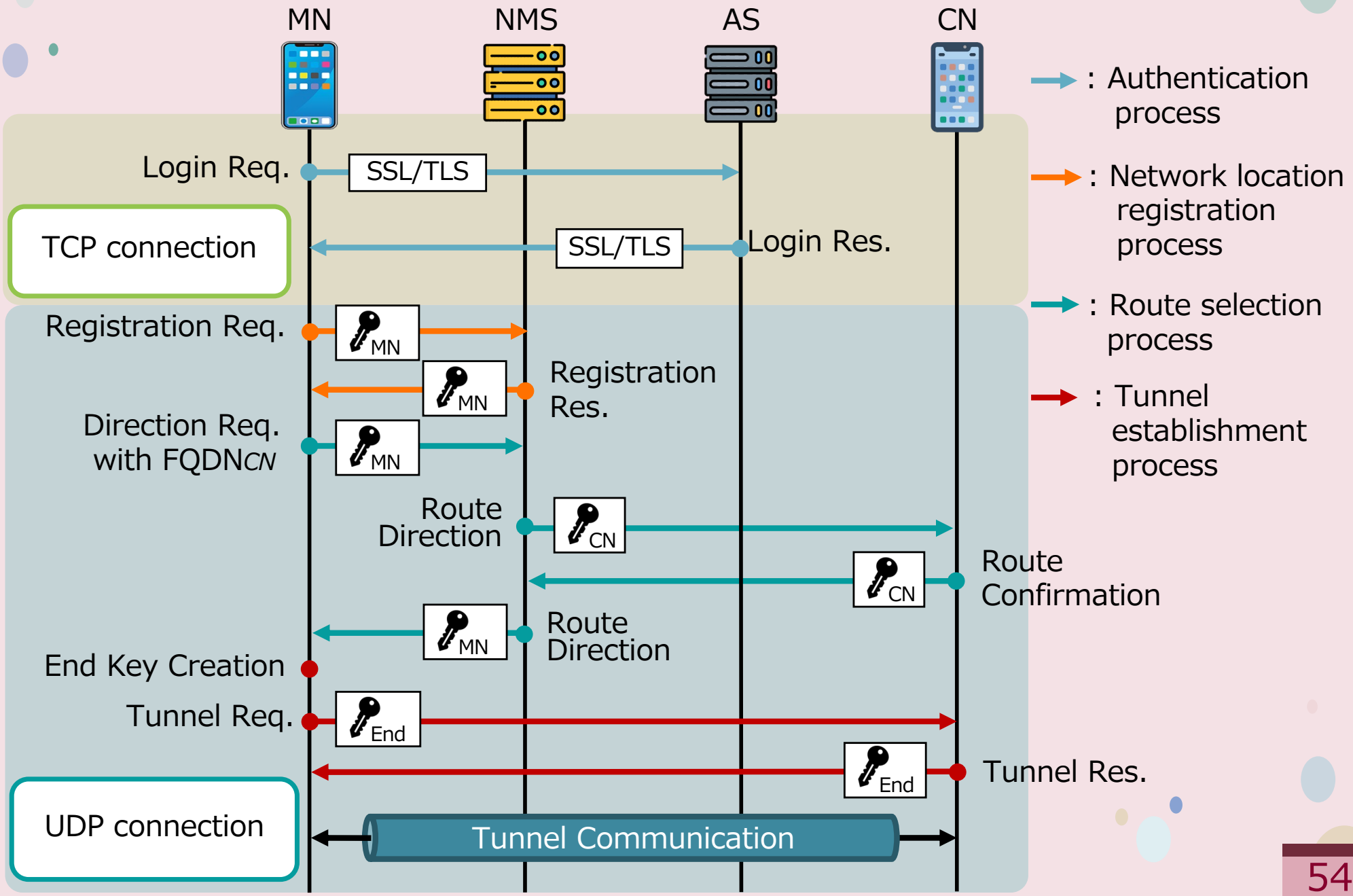
# CYPHONIC 全体概要



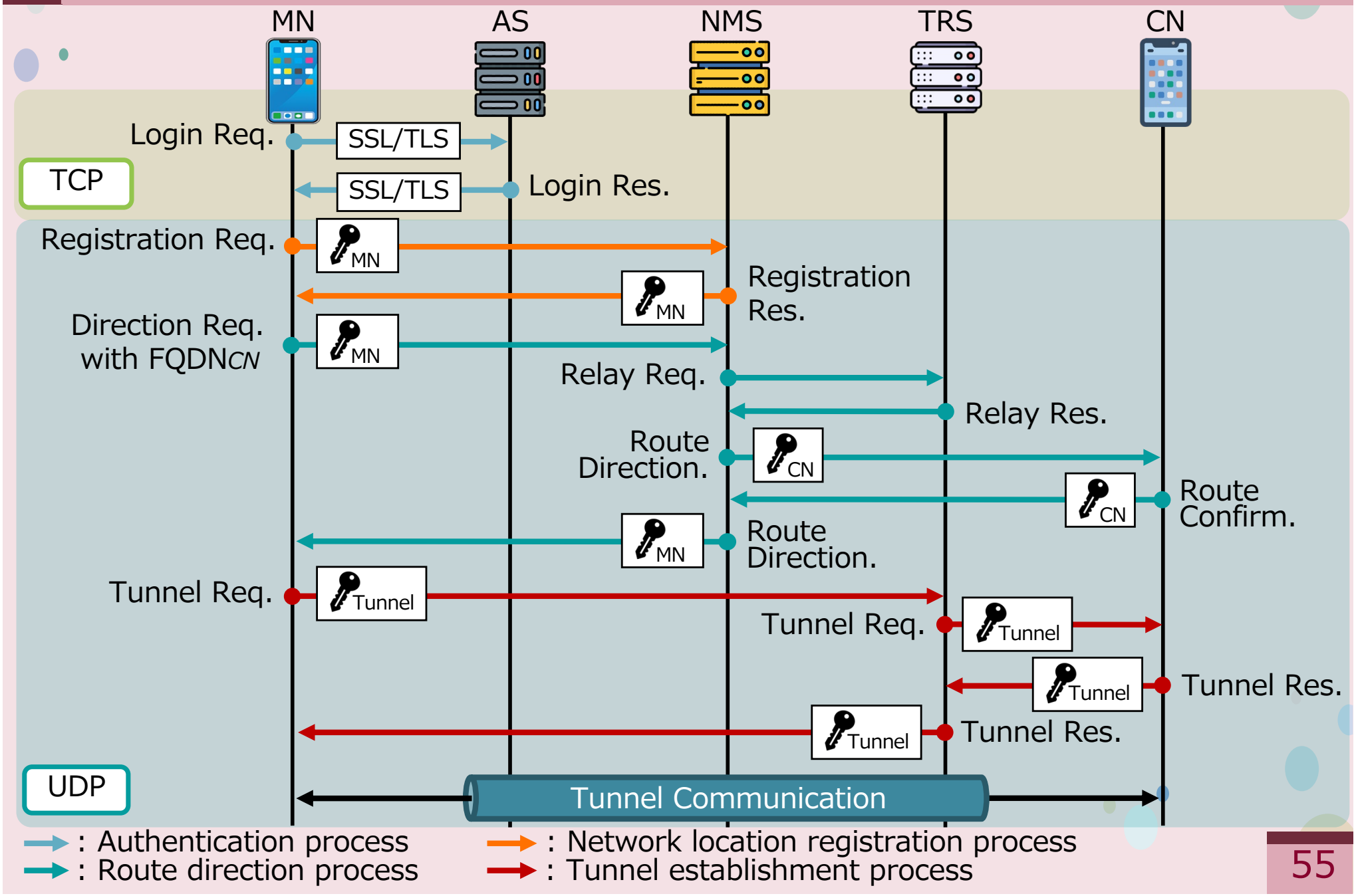

CYPHONIC独自のヘッダを付与することで
オーバーレイネットワークを実現

1. 認証処理：端末の認証　2. 位置情報登録処理：ネットワーク情報登録
3. 経路選択処理：通信経路決定　4. トンネル確立処理：エンド間トンネル構築
5. データ通信処理：オーバーレイネットワーク上の通信

# CYPHONICシグナリング 概要



# CYPHONICシグナリング（via TRS）概要



# ここから 中間報告

# セキュアオーバーレイネットワークシステムにおける サービス拡張性実現手法に関する研究

# Study on Methods for Achieving Service Extensibility in Secure Overlay Network Systems

研究者 後藤廉　　指導教員 内藤克浩

愛知工業大学大学院 経営情報科学研究科
修士論文中間報告会
2023年 11月 14日 (火)

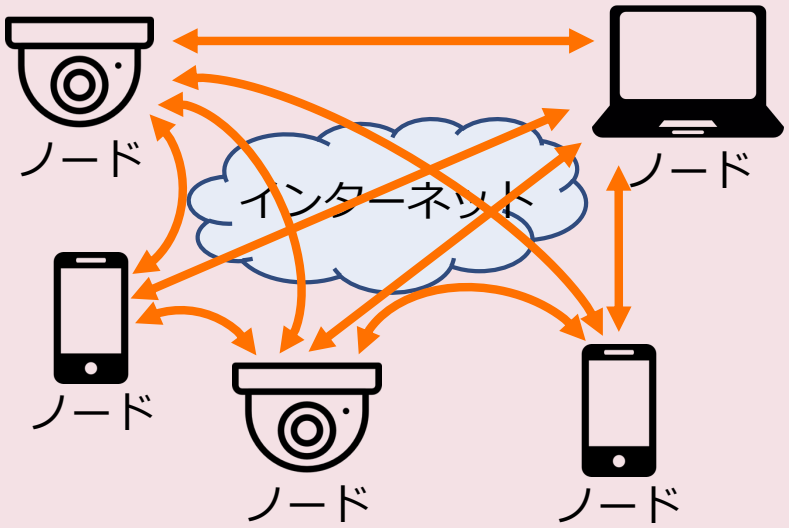
# 目次

# Peer-to-Peer サービス

## Peer-to-Peer (P2P) サービス

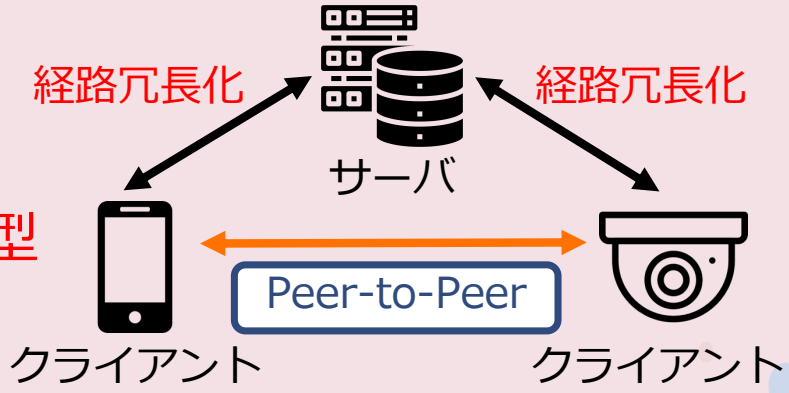
(ex. IoT 機器による分散処理)

- タスクを複数の端末に分散して
  協調処理を行うことでサービスを実現
- サービス障害の影響を受けにくい
- ネットワーク全体のトラフィック量を抑制




## 課題

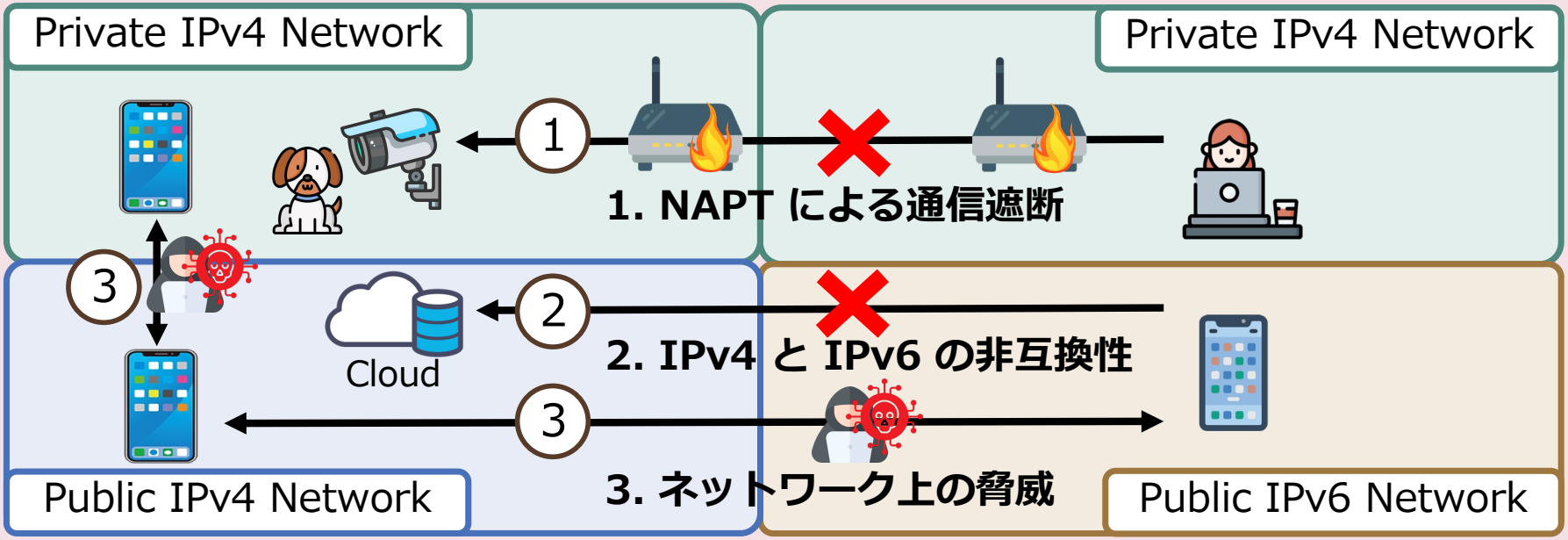
- 端末毎の 構成やセキュリティ管理 が複雑化
- サービスモデルに反して Client-to-Server 型
  ネットワーク によるサービス提供が一般的




：Network model　：Service model

P2P サービスには P2P 型ネットワークを採用した
端末間の相互接続及び直接通信機構が必要

# P2P 型ネットワーク構築に伴う課題




NAPT：Network Address Port Translation

課題1：NAPT によるインバウンドトラフィックの遮断（通信接続性）
➡ 内部ネットワークに存在する端末を外部ネットワークから隠蔽

課題2：IPv4 と IPv6 間の非互換性（通信接続性）
➡ IPv4 と IPv6 ではパケットフォーマットが異なる

課題3：相互接続・直接通信に伴うネットワーク脅威の考慮（機密性・完全性）
➡ 暗号化やアクセス制御技術の導入による安全な通信の実現

# CYPHONIC 概要

CYber PHysical Overlay Network over Internet Communication

**P2P 型ネットワーク** に基づくセキュアなオーバーレイネットワーク通信技術

通信接続性：ネットワーク環境に応じた経路指示に従い通信

移動透過性：不変な仮想 IP アドレスを用いた通信によりコネクション維持

セキュリティ：認証の導入及びオーバーレイネットワーク上での暗号化通信

**CYPHONIC ノード**

➡クライアントプログラムを搭載した端末

**Authentication Service (AS)**

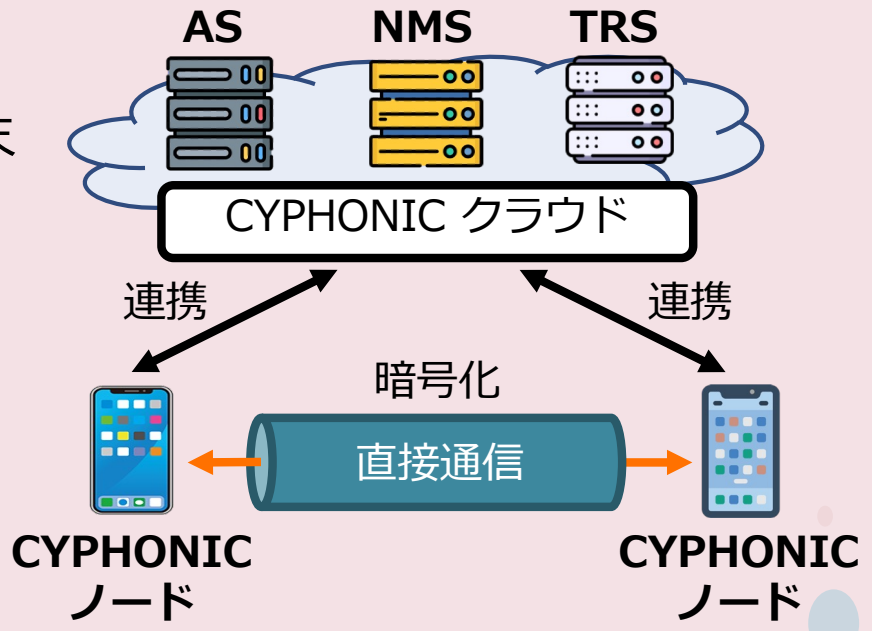
➡端末認証およびノード情報管理

**Node Management Service (NMS)**

➡ネットワーク情報管理と通信経路指示

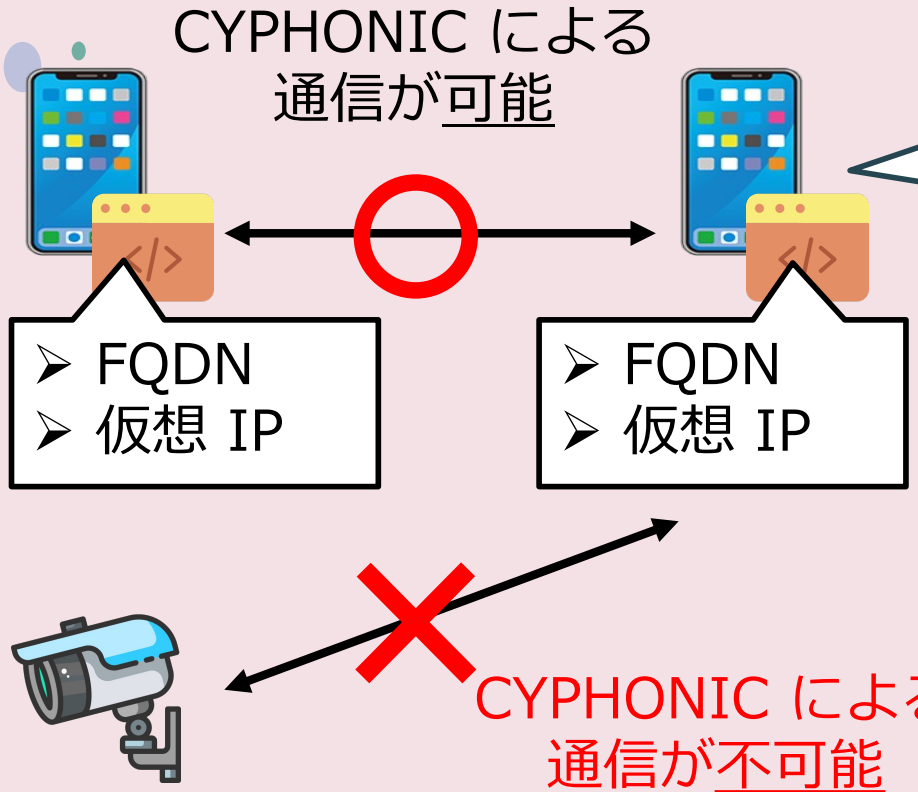
**Tunnel Relay Service (TRS)**

➡直接通信が困難な場合に通信を中継




端末間で自律的にトンネルを
構築して直接通信を確立

# CYPHONIC 現状：エンドノード




**FQDNによる相手ノードの識別**
➡ CYPHONICドメインを管理する
DNSリゾルバ

**仮想IPアドレスによる通信**
➡ 仮想インタフェースを介した
データのカプセル化

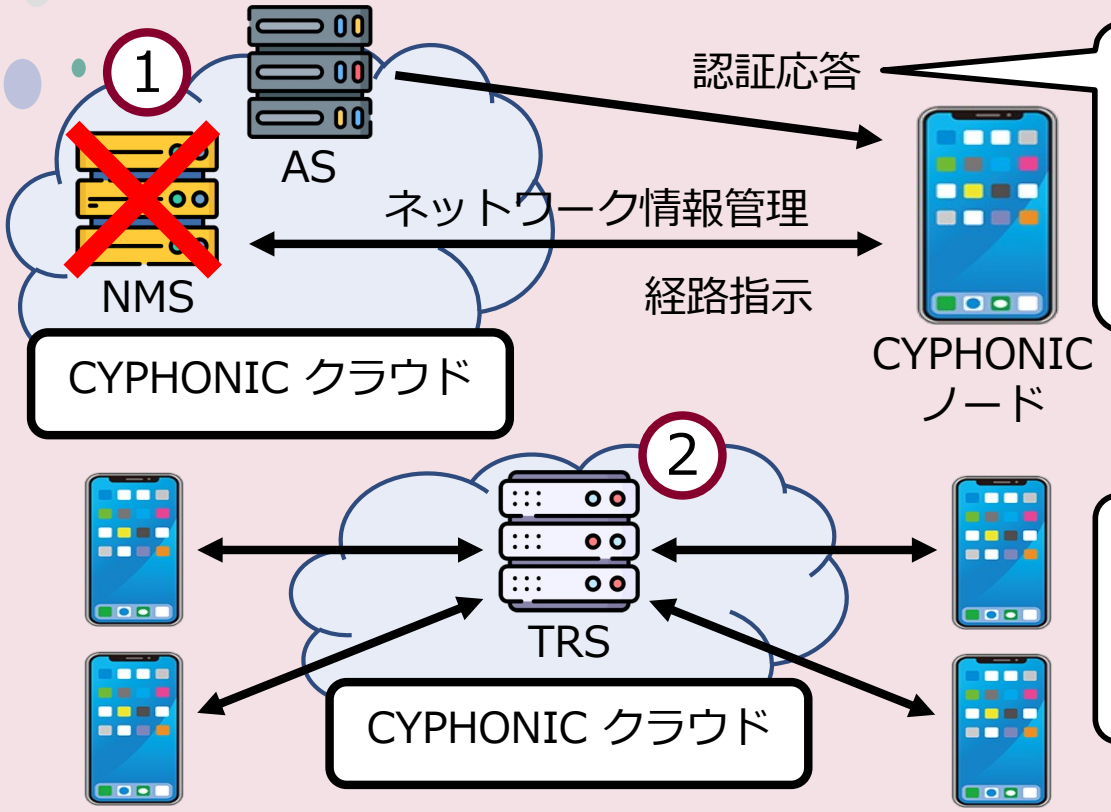
**暗号化通信**
➡ 仮想IPパケット暗号化/復号処理機能

FQDN: Fully Qualified Domain Name

全ての端末にクライアントプログラムの追加が必要
➡ IoT 機器や専用サーバをはじめ一部の端末は既存システムの変更が困難

改良困難な端末を含む多様なエンドノードの包括的なサポートが必要

# CYPHONIC 現状：クラウドサービス




認証応答に含まれる情報
- CYPHONIC ノードの FQDN
- NMS の宛先 (1台分)
- NMS との通信に用いる暗号鍵

TRS が中継を行うケース
- 両ノードが NAPT 配下に存在
- IPv4 – IPv6 間の通信

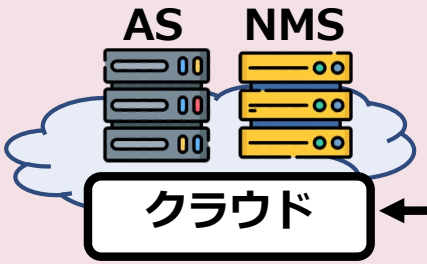
1. CYPHONIC ノードに割り当てられる NMS が 1台のみ
   ➡ NMS 障害発生時に端末の経路構築が不可能
2. 中継が必要な端末間通信において TRS に負荷が集中
   ➡ 高トラフィック・高負荷により TRS が停止する恐れ

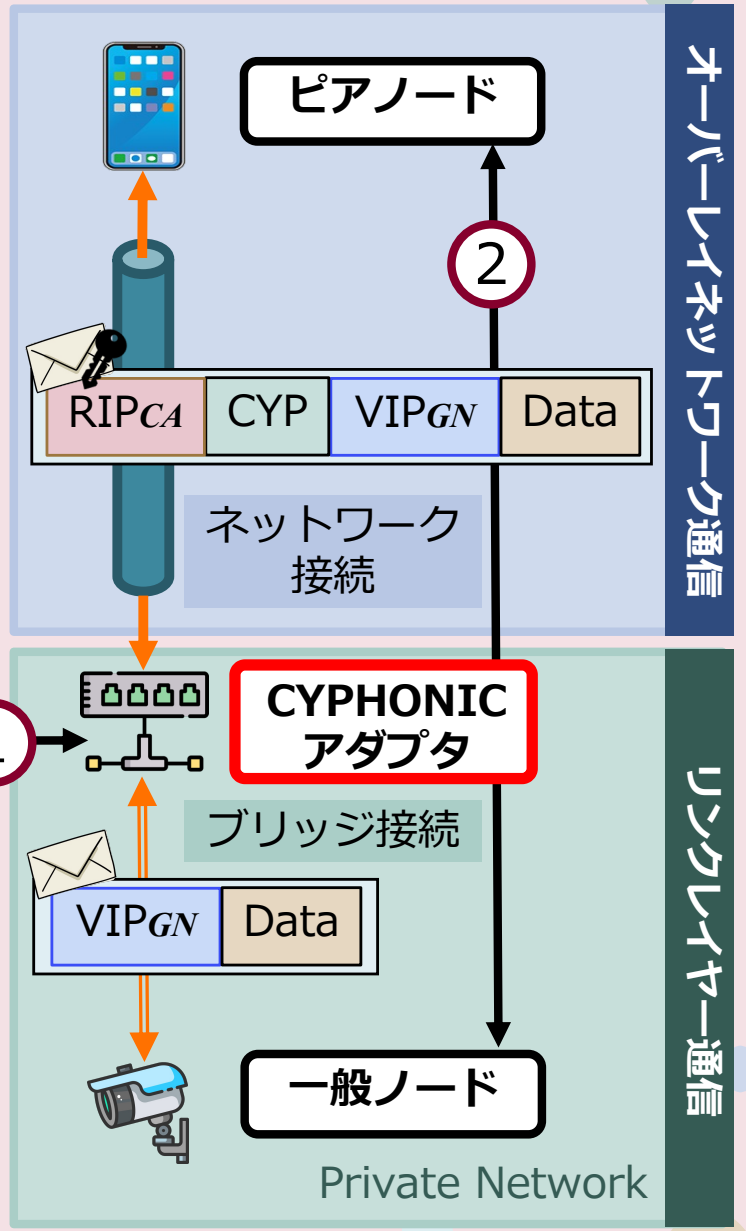
単一障害点の解消及び処理負荷に応じた拡張性の考慮が必要

# アダプタ端末による代行処理

## CYPHONIC アダプタ

- システム改良が困難な端末 (一般ノード) に CYPHONIC の通信機能を提供
- 一般ノードは隣接設置されるアダプタ端末に接続して CYPHONIC を利用



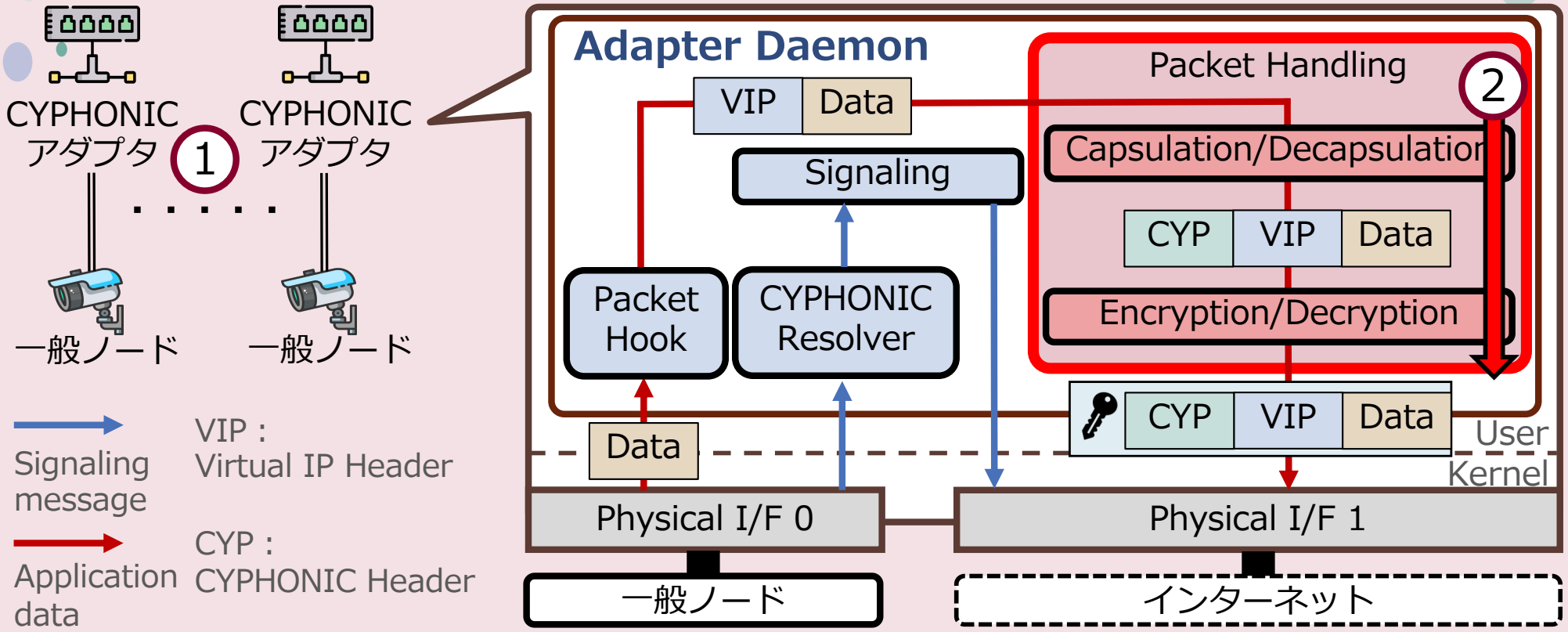




## 機能概要

1. **シグナリング機能**
   トンネル確立に伴うシグナリングを代行処理
2. **オーバーレイネットワーク通信機能**
   一般ノードと CYPHONIC 上のピアノードに介在して所定のパケットへ相互に変換

CA: CYPHONIC Adapter　VIP: Virtual IP Header
GN: General Node　RIP: Real IP Header　CYP: CYPHONIC Header

# CYPHONICアダプタ 課題



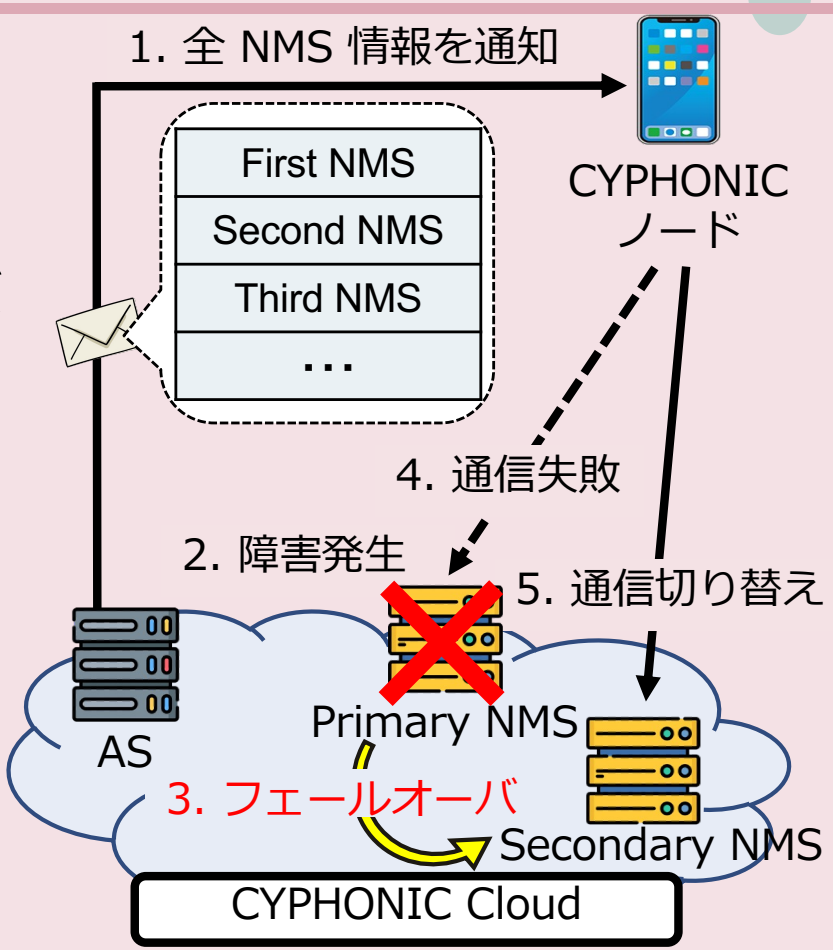

1. 一般ノード と CYPHONIC アダプタが 1 対 1 対応
   ➡ ネットワーク規模の拡大及び端末管理の煩雑化やコストの増大
2. パケット処理モジュールはシングルスレッドで受信パケットを直列処理
   ➡ 単一モジュールに負荷がかかることによる通信品質の劣化が懸念

複数一般ノードを想定した CYPHONIC アダプタの拡張設計が必要

# Active-Standby での単一障害点の解消

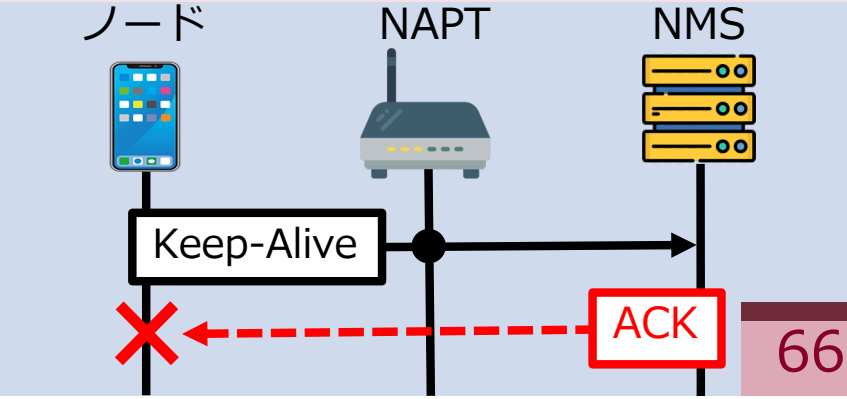
## 多重化とフェールオーバ

- NMS のみ Master-Slave 方式による多重化により障害発生時にフェールオーバ
- 認証応答の際に複数台分の NMS 情報を CYPHONIC ノードに通知
- 割り当てられた複数の NMS に対してネットワーク環境情報の登録を実行

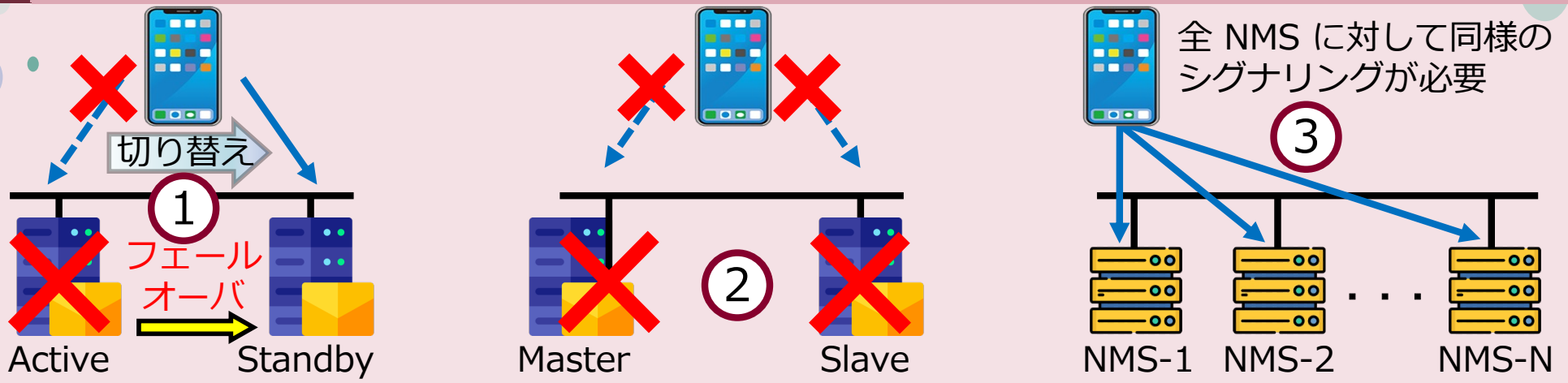



## 通信先の切り替え

- NAPT マッピングテーブル保持のための Keep-Alive を利用して NMS を死活監視
- 一定期間 ACK が受信されない場合は別の NMS に通信先を変更



# 既存の単一障害点解消手法に関する問題点




1. 障害を前提としたクラウド設計
   - 障害を未然に防ぐ仕組みやトラフィックに応じたスケーリングは未搭載
2. 障害からの復旧機能について未考慮
   - Master サーバに続き Slave サーバも停止する恐れ
   - サーバ管理者が常時監視する必要性
3. 単一障害点解消に伴うエンドノード機能の冗長化
   - サーバの冗長度に応じて実行シグナリングが増加
   - クラウド死活監視に関するパケット処理機能が必要

耐障害性 (Fault Tolerance) と 規模拡張性 (Scalability) を兼ね備えた
CYPHONIC クラウドサービス全体としてのスケール設計が必要

# 本研究の目的

CYPHONIC におけるサービス利用多様性の実現と
エンドノードの増加を想定したスケーラブルなクラウドシステムを提案

【 サービス としての拡張性 】

**改良困難な端末を含む多様なエンドノードの包括的なサポート**

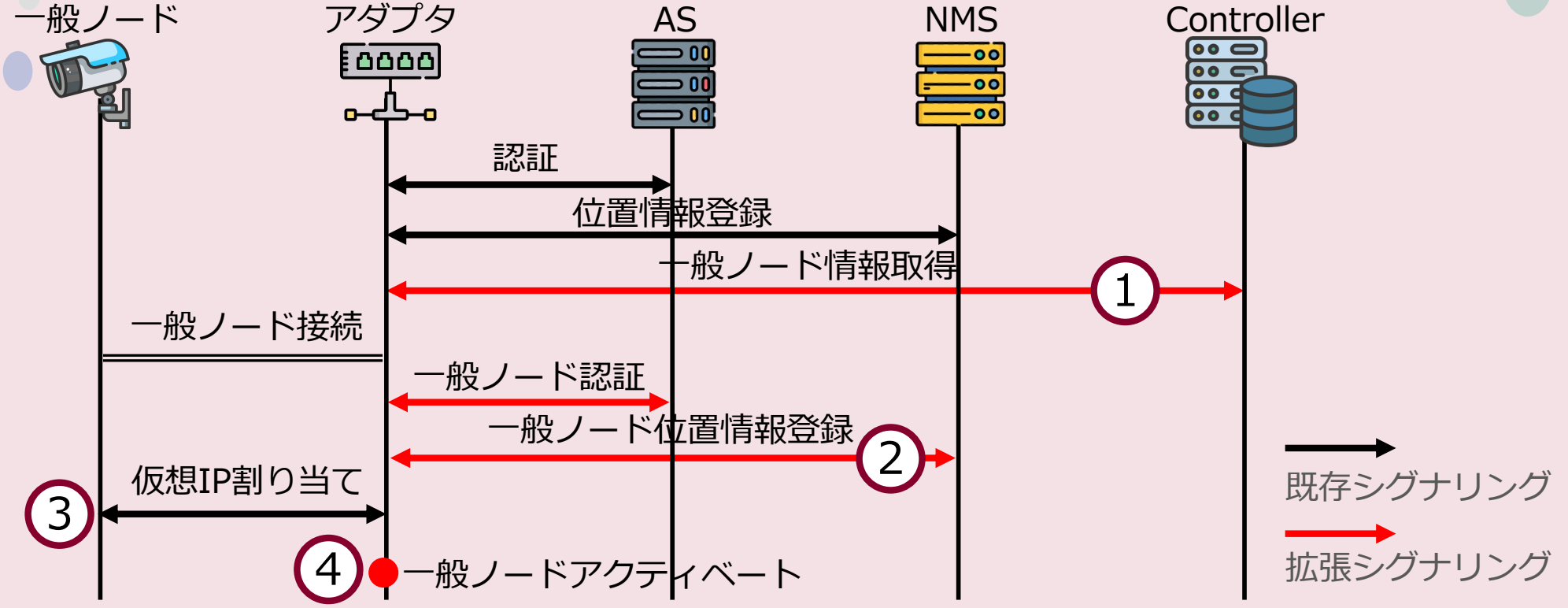
- CYPHONIC アダプタの処理機能マルチスレッド化による性能向上の実現
- 複数一般ノードのオーバーレイネットワーク同時接続の実現

【 システム としての拡張性 】

**クラウドサービスにおける単一障害点の解消と処理負荷に応じたスケール設計**

- クラウドにおける各サービスのクラスタリング及び機能多重化の実現
- トラフィック量に応じた水平スケーリングと負荷分散の実現

# アダプタ実行シグナリングプロセスの拡張




1. 複数一般ノードの情報を CYPHONIC クラウドから取得
   - 一般ノードが利用するネットワーク I/F の MAC アドレス
   - 一般ノード認証方式 (Basic 認証, MAC アドレス照合, IEEE 802.1X 認証)
   - 一般ノードの実行プロトコルバージョン（IPv4 モード or IPv6 モード）
2. 一般ノードの認証処理及び位置情報登録処理を実行
3. DHCPv4 / Stateful DHCPv6 により仮想 IP アドレスを一般ノードへ付与
4. アダプタ内部の一般ノード起動ステートを UP

# マルチスレッド化に伴う追加機能

パケット処理モジュールにおける暗号化/復号及びカプセル化/デカプセル化にマルチスレッド処理を採用した同時実行モデルが必要

- メインスレッドから分離したワーカースレッドを準備
- 受信パケットの処理ジョブを各ワーカースレッドに移譲
- 各ワーカースレッドの処理状況は言語ランタイム及び OS のリソース使用状況に依存
- 受信パケットと処理済みパケット・送信パケットの順序が異なることによる通信スループットの低下が懸念

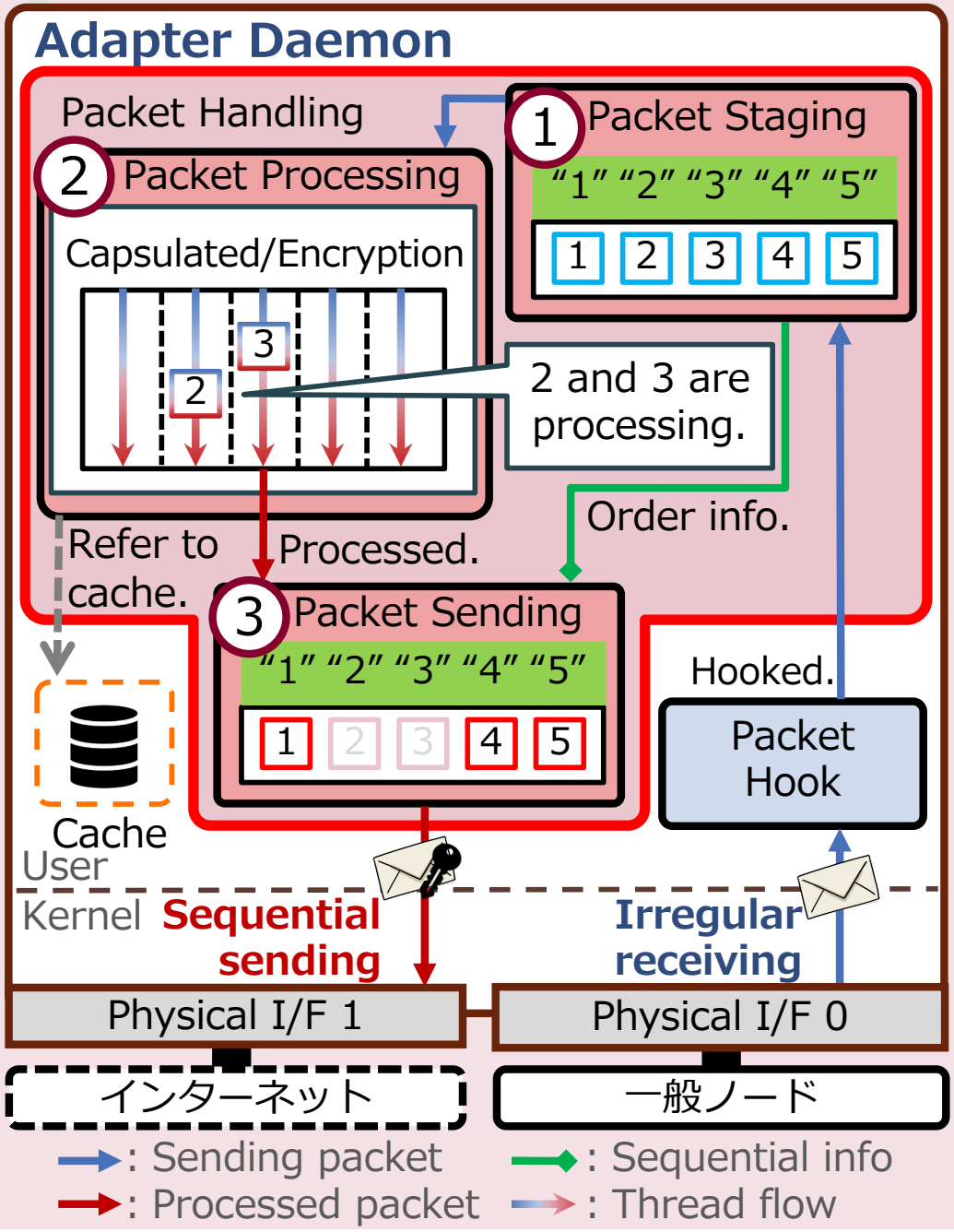
送受信パケットの順序を制御する順序付け機構が必要

➡ 既存のパケット処理モジュール内部を 3 つの機能に分離

1. **処理ステージ機能**：親スレッドからパケットを受け取り受信順序を保存
2. **パケット処理機能**：受信パケットをワーカースレッドに割り当て並行処理
3. **送信ステージ機能**：処理済みパケットを受け取り処理ステージ機能が保存した受信順序に従って逐次的に送信

マルチスレッド化とパケット順序付け及び逐次処理機構を導入することで処理の高速化と送受信時点におけるパケット順序の整合性維持を検討

# パケット順序付け及び逐次処理モデル




## 1. Packet Staging Module

受信パケットをバッファに格納し

受信順序を保存

## 2. Packet Processing Module

処理をワーカースレッドに割り当て

カプセル化・暗号化を非同期実行

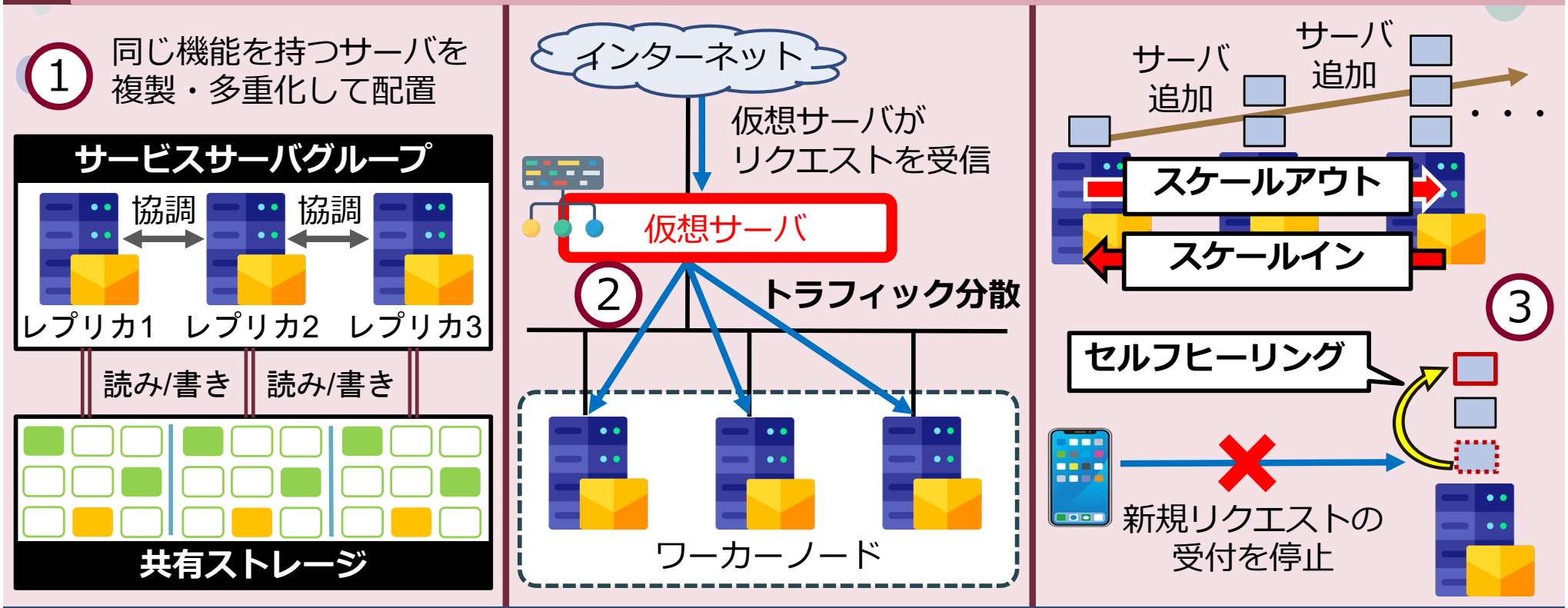
## 3. Packet Sending Module

処理済みパケットをキューに

追加すると共に受信順序に基づき送信

非同期実行及びワーカースレッドの
処理状況に依らず送受信において
パケットの順序維持が可能

# 耐障害性及び規模拡張性の実現手法




1. サーバクラスタリングによる機能の多重化
   - サーバを複数台準備して Peer-to-Peer 構成を取ることで機能を多重化
2. リクエスト及びトラフィックの分散
   - 単一サーバあたりに流入するジョブをクラスタの前段で分散
3. オートスケーリング・オートヒーリング
   - トラフィック量に応じて自動的にスケールアウト・スケールインを実行
   - 障害時には新規リクエストの受付を停止すると共に自動的に復旧

# 提案するCYPHONICクラウドの構成概要

各サービスサーバを Kubernetes (分散コンテナ実行基盤) 上に構築
➡コンテナ化されたサービスを Pod 単位でインスタンスに展開して稼働

1. サービスサーバのレプリケーション

ステート管理用の内部キャッシュが存在

➡ステート及びデータの共有機構が必要

**2. リクエスト分散機能**

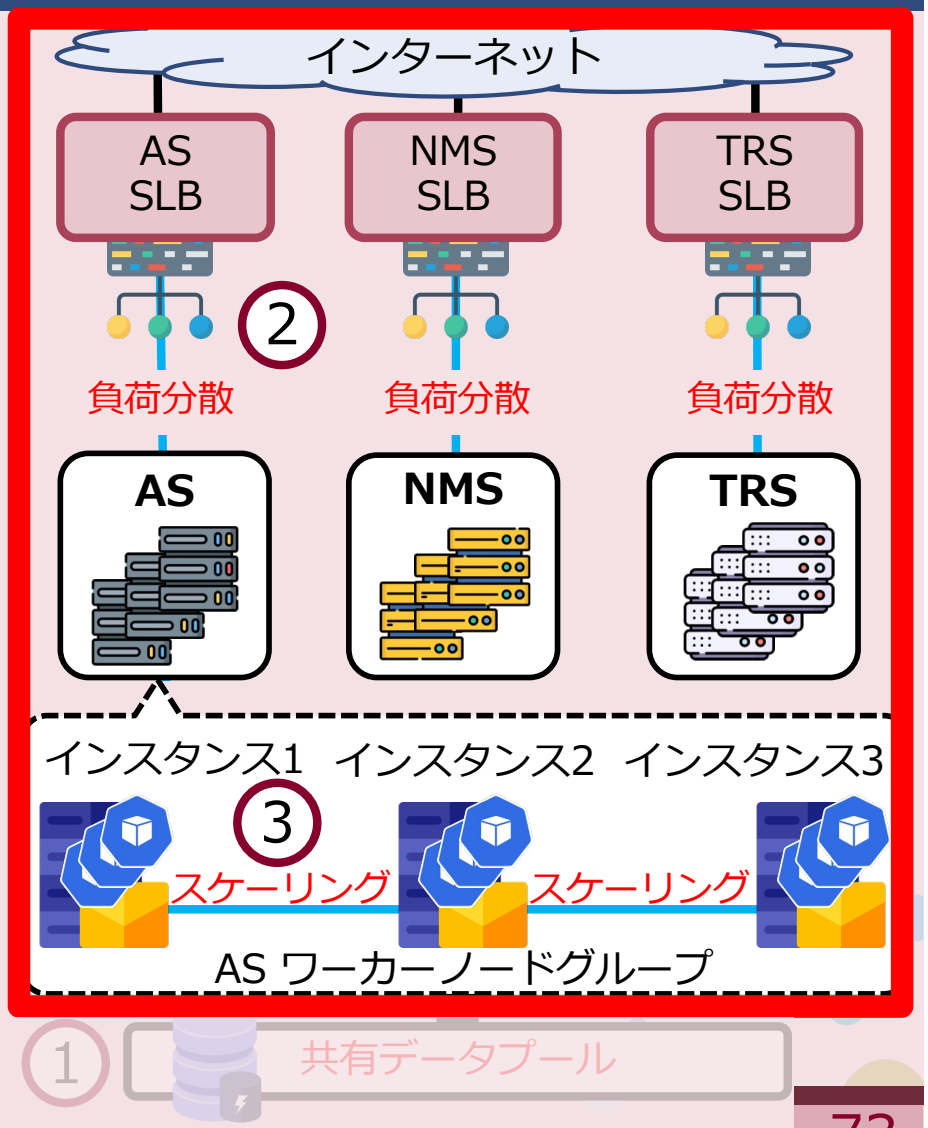
リクエスト転送の際に送信元 IP を SNAT

➡送信元 IP を保存する仕組みが必要

**3. リクエスト数に応じたスケール機能**

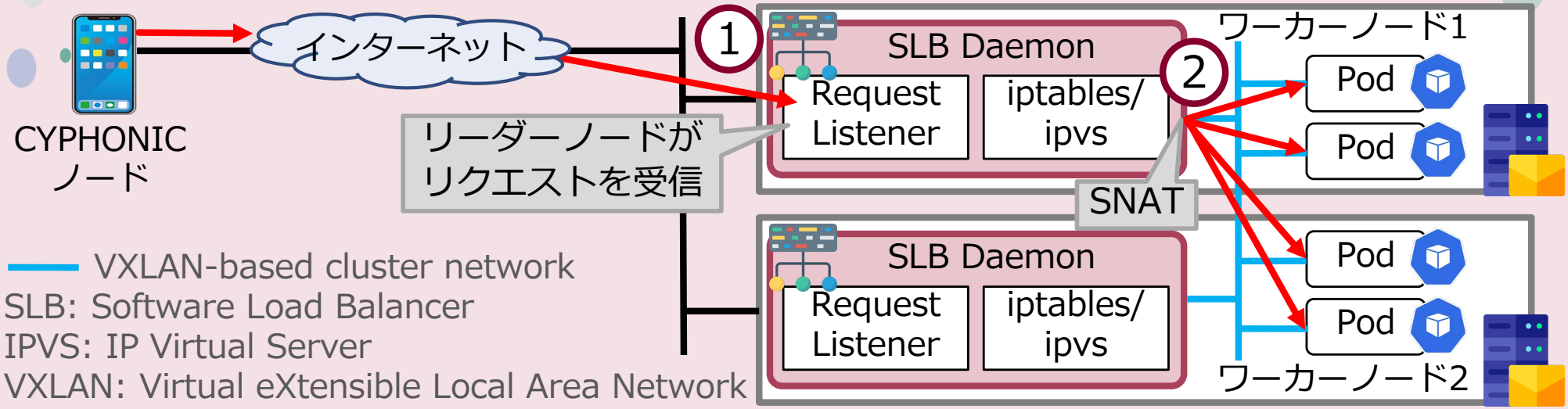
フェールオーバではダウンタイムが発生

➡オートスケール・オートヒールによる
システムレジリエンスの向上が必要




SNAT: Source Network Address Translation　SLB: Software Load Balancer

# Kubernetes における負荷分散の基本原理




CYPHONIC ノードからのリクエスト

- 各シグナリングは SLB を宛先として送信
- SLB の実体は各ワーカーノードで起動する iptables/ipvs 制御用デーモン

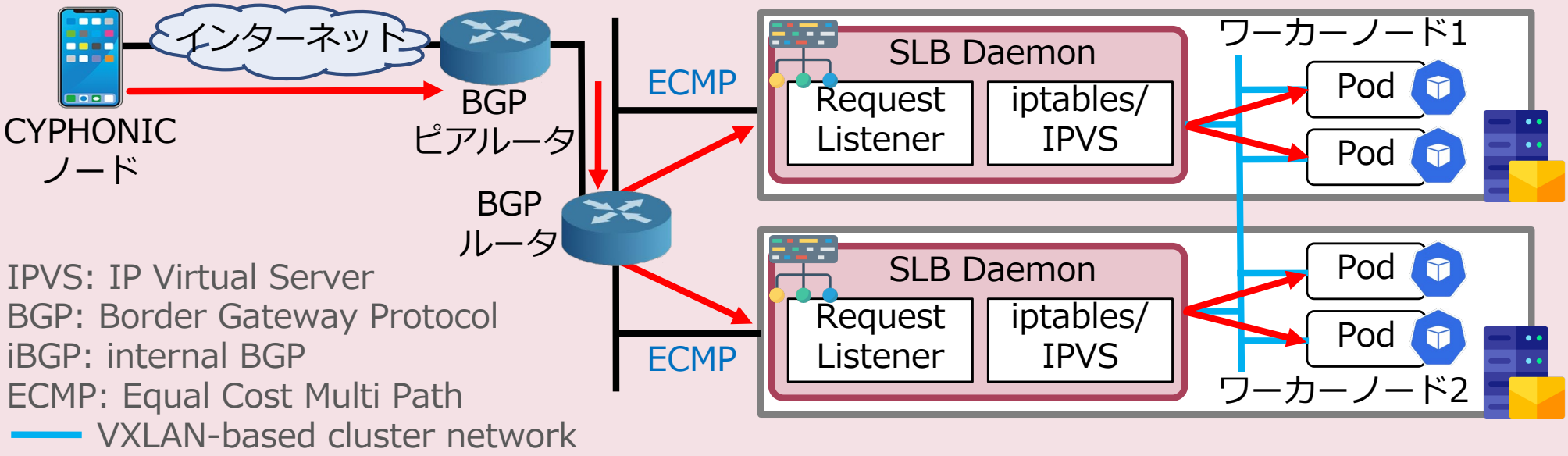
クラスタ内における Pod 間通信

- 各ワーカーノードに配置された Pod は VXLAN ベースのクラスタ内オーバーレイネットワークを通じて連携

1. 任意のワーカーノードがリーダーノードとして全トラフィックを受信
   ➡ 特定のワーカーノードにトラフィックが集中しボトルネックになる恐れ
2. SLB Daemon は SNAT によってクラスタ内の全 Pod にリクエストを分散
   ➡ CYPHONIC ノードのネットワーク情報を正確に取得することが困難

# BGP によるクラスタワイドな負荷分散

SLB Daemon と BGP ルータを連携した
ECMP による等コストパスベースの負荷分散




CYPHONIC ノードからのリクエスト

- 各シグナリングは BGP ルータを経由して SLB に送信
- パケットを受信したワーカノード内の Pod にリクエストを分散

➡ **ワーカーノードを跨ぐ負荷分散を防止することで SNAT を回避**

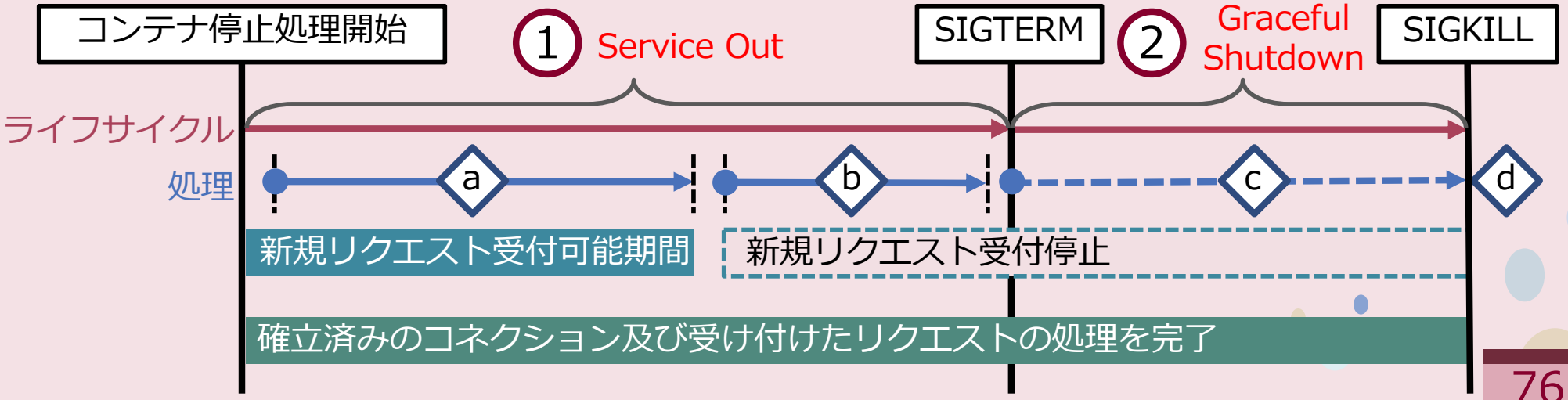
# サービスサーバ群のオートスケーリング

トラフィック量に応じたスケーリング

➡Pod のメトリクスを収集してサービスサーバをオートスケール

ケールイン及び障害予兆に伴うコンテナ削除

1. **スケールイン命令により PreStop フック処理を実行**
   a. SLB に登録された削除対象 Pod の IP アドレスリストを削除
   b. SLB Daemon は iptables を更新して新規コネクションの受付を停止
2. **SIGTERM をトリガにコンテナ終了処理を開始**
   c. CYPHONIC クラウドサービスは猶予期間内に既存のリクエストを処理
   d. 処理が終了した後 SIGKILL によりコンテナを強制停止



# CYPHONICアダプタ 性能評価

一般ノード と CYPHONIC ノード間の
通信性能を測定

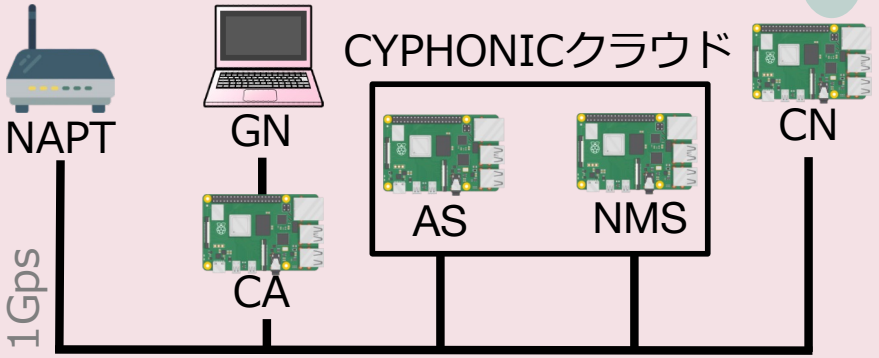
**通信遅延時間の計測**

➡ ping を使用

**通信スループットの計測**

➡ iperf を使用

- シングルスレッドベースの処理と
  マルチスレッド対応した CYPHONIC
  アダプタの 2 通りを計測
- 双方のシナリオで通信性能を比較




| Raspberry Pi 4 Model B (CYPHONIC Cloud, Adapter, Node) | |
|---|---|
| OS | Raspbian GNU/Linux 10.0 |
| CPU | Quad Core 1.5GHz<br>Broadcom BCM2711 |
| Memory | 4GB |

| MacBook Air 2017 (General Node) | |
|---|---|
| OS | macOS Monterey Ver 12.2 |
| CPU | Dual Core 2.20GHz<br>Intel(R) Core i7-5650U |
| Memory | 8GB |

CA: CYPHONIC Adapter　GN: General Node　CN: CYPHONIC Node

# 一般ノードにおける通信性能の測定結果

| シングルスレッドベースの CYPHONICアダプタ | | | |
|---|---|---|---|
| Round-trip time | UDP throughput | Jitter | TCP throughput |
| 3.47 [ms] | 29.7 Mbit/sec | 0.40 [ms] | 1.68 Mbit/sec |

| マルチスレッド対応した CYPHONICアダプタ | | | |
|---|---|---|---|
| Round-trip time | UDP throughput | Jitter | TCP throughput |
| 3.27 [ms] | 30.0 Mbit/sec | 0.50 [ms] | 37.3 Mbit/sec |

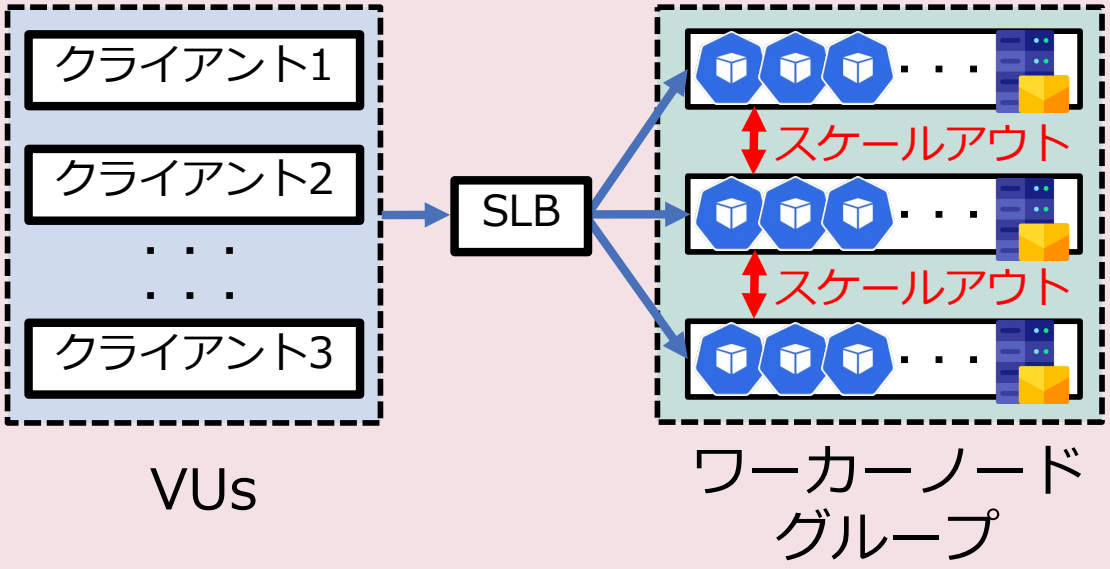
※ UDP 帯域は 30 Mbps に設定

- シングルスレッド方式と比較して TCP スループットは 約 22.2 倍 向上
- 断片化されたパケットの順序を維持しつつ高速な内部処理を実現

# CYPHONICクラウド スケーリング基礎検証

- 毎分 100 RPS ずつリクエストを増加
- スケールアウト指標 CPU 使用率 50%
- に設定

- 閾値を元にオートスケーリングを検証
- スケールアウトした Pod に対する負荷分散を検証




ホストマシン

| | |
|---|---|
| CPU | Intel(R) Core(TM) i9-13900 CPU @ 5.60GHz 24 cores 32 threads |
| RAM | 128 GiB |

ゲストマシン (Data-Plane)

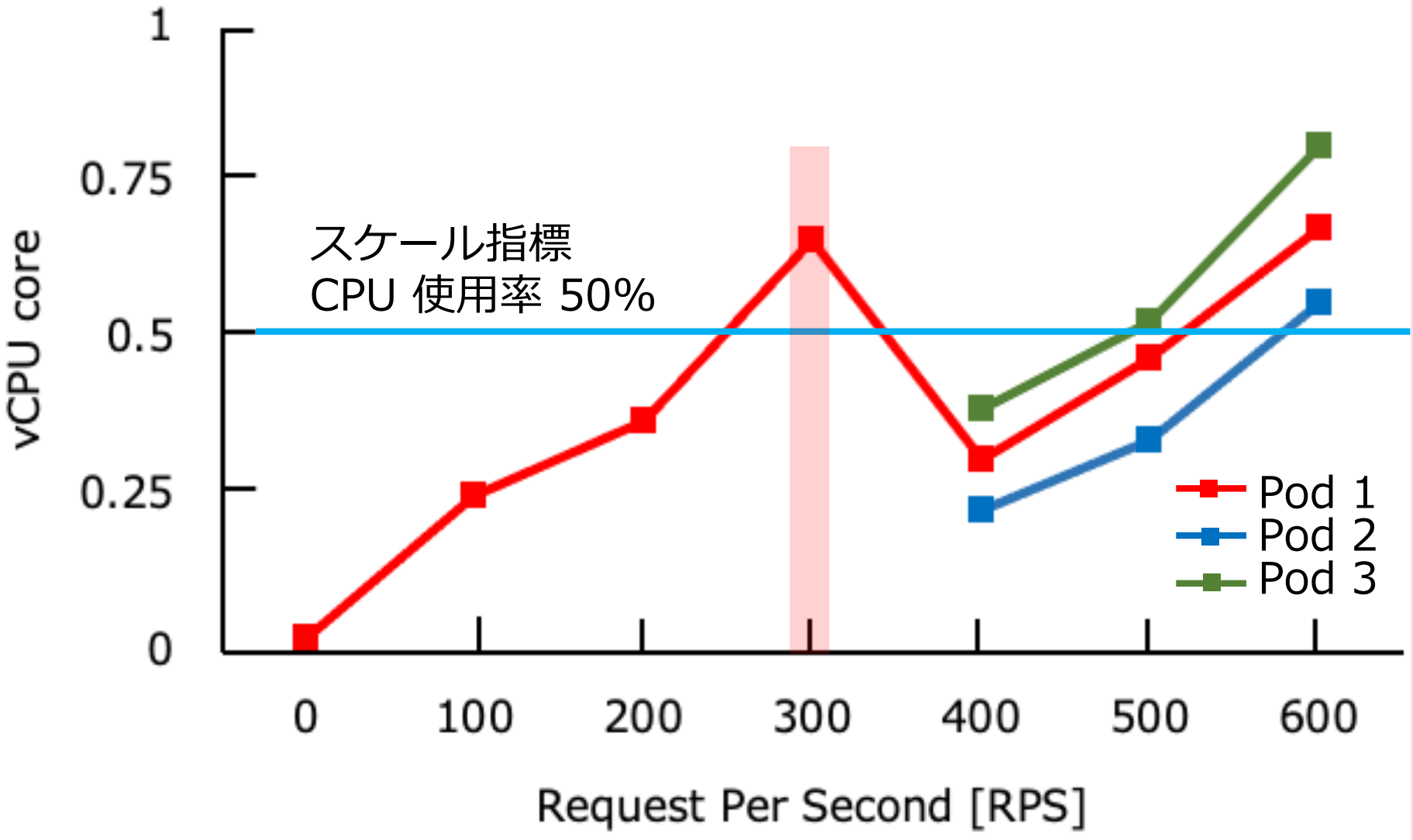
| | |
|---|---|
| ノード | 3 台 |
| OS | Ubuntu 20.04 (Focal Fossa) |
| CPU | 8 cores 8 threads |
| RAM | 24 GiB |

Kubernetes

| | |
|---|---|
| Kubernetes (kubeadm) | v1.25.3 |
| CRI (containerd) | v1.6.24 |
| CNI (flannel) | v0.20.1 |
| SLB (Metal LB) | v0.13.12 |

Vus: Virtual Users
CRI: Container Runtime Interface
CNI: Container Network Interface
SLB: Software Load Balancer

# 基礎評価結果




- 単一 Pod の平均 CPU 使用率に基づきスケールアウトすることを確認
- 全ての Pod にリクエストが分散することを確認

# 今後の方針

**CYPHONIC アダプタ**

- 複数一般ノード接続時における通信性能の評価が不十分

➡ コネクションの増加に伴う一般ノードのスループット測定を実施

- アダプタを長時間稼働させ続けた際のリソース動向の調査 (ヒート試験)

➡ マシンリソースの消費量及び APM メトリクスの収集・評価

APM: Application Performance Management

**CYPHONIC クラウド**

- トランザクション (TPS) の計測及び負荷試験の実施

➡ サービス全体としての処理性能の確認及び C10K 検証

- クラスタネットワークの性能測定

➡ クラスタのスペックとネットワークリンクを強化して検証を実施

# まとめ

CYPHONIC におけるサービス利用多様性の実現と
エンドノードの増加を想定したスケーラブルなクラウドシステムを提案

【 サービス としての拡張性 】

**改良困難な端末を含む多様なエンドノードの包括的なサポート**

- CYPHONIC アダプタの処理機能マルチスレッド化による性能向上の実現
- 複数一般ノードのオーバーレイネットワーク同時接続の実現

【 システム としての拡張性 】

**クラウドサービスにおける単一障害点の解消と処理負荷に応じたスケール設計**

- クラウドにおける各サービスのクラスタリング及び機能多重化の実現
- トラフィック量に応じた水平スケーリングと負荷分散の実現

- アダプタのマルチスレッド化により通信性能が大幅に向上したことを確認
- クラウドサービスにおけるオートスケーリングの基礎動作検証を実施

# ここまで 中間報告

# ここから 他大合同報告MTG

# セキュアオーバーレイネットワークシステムにおける サービス拡張性実現手法に関する研究

# Study on Methods for Achieving Service Extensibility in Secure Overlay Network Systems

研究者 後藤廉[1)]　指導教員 内藤克浩[2)]

[1)] 愛知工業大学大学院 経営情報科学研究科
[2)] 愛知工業大学 情報科学部 情報科学科

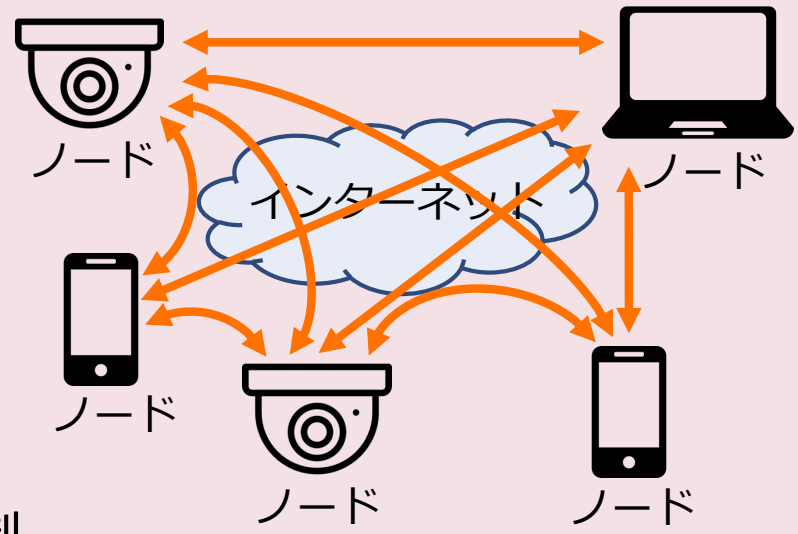
愛知工業大学・名古屋大学 合同すまーとミーティング
2023年 10月 16日 (月)

# Peer-to-Peer (P2P) サービスと課題

**Peer-to-Peer (P2P) サービス**

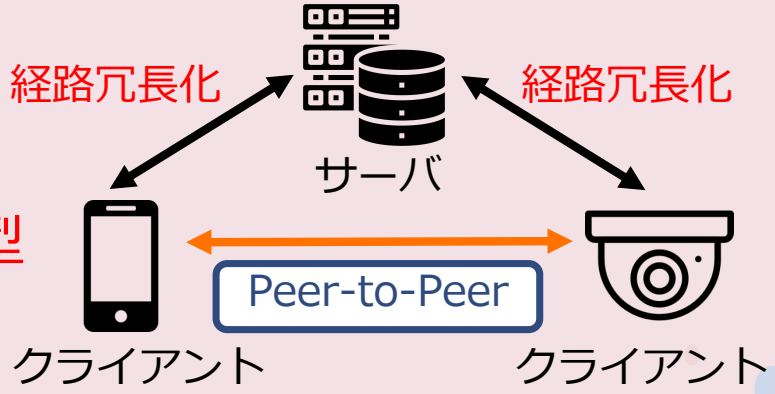
(ex. IoT 機器による分散処理)

- タスクを複数の端末に分散して
  協調処理を行うことでサービスを実現
- サービス障害の影響を受けにくい
- ネットワーク全体のトラフィック量を抑制




**課題**

- 端末毎の 構成やセキュリティ管理 が複雑化
- サービスモデルに反して Client-to-Server 型
  ネットワーク によるサービス提供が一般的




：Network model　：Service model

P2P サービスには P2P 型ネットワークを採用した
端末間の相互接続及び直接通信機構が必要

# CYPHONICの概要

CYber PHysical Overlay Network over Internet Communication

**P2P 型ネットワーク** に基づくセキュアオーバーレイネットワーク通信技術

通信接続性：ネットワーク環境に応じた経路指示に従い通信

移動透過性：不変な仮想 IP アドレスを用いた通信によりコネクション維持

セキュリティ：認証の導入及びオーバーレイネットワーク上での暗号化通信

**CYPHONIC ノード**

➡クライアントプログラムを搭載した端末

**Authentication Service (AS)**

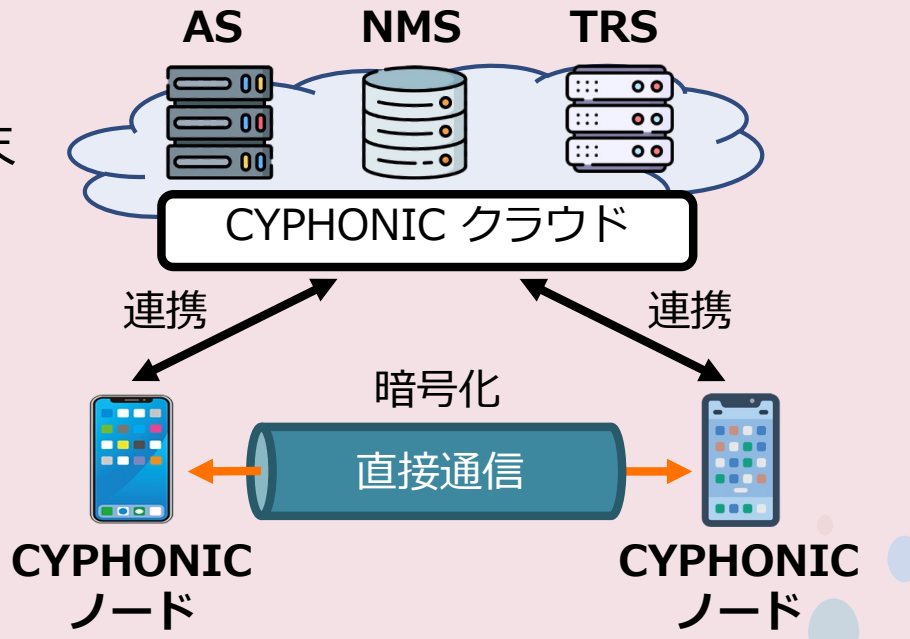
➡端末認証およびノード情報管理

**Node Management Service (NMS)**

➡ネットワーク情報管理と通信経路指示

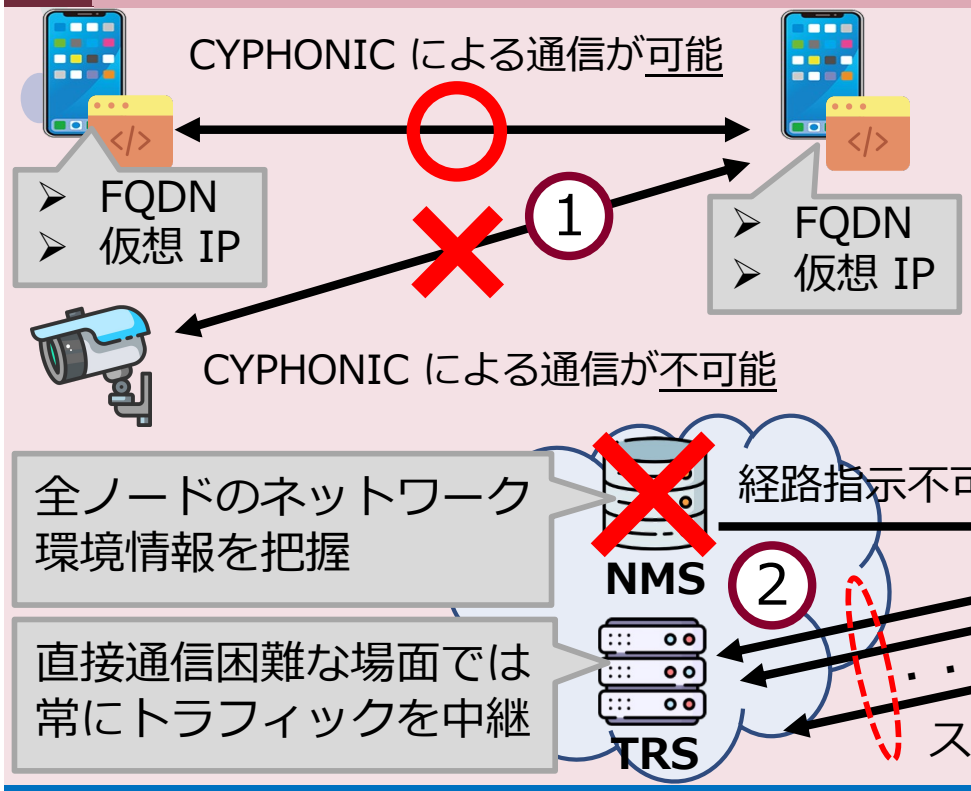
**Tunnel Relay Service (TRS)**

➡直接通信が困難な場合に通信を中継




端末間で自律的にトンネルを
構築して直接通信を確立

# 既存CYPHONICの課題




FQDN: Fully Qualified Domain Name

クライアントプログラム (CYPHONIC Daemon) が提供する機能
- シグナリングメッセージ交換機能
- CYPHONIC 独自ドメインのフィルタリング機能
- 仮想インターフェース生成機能
- CYPHONIC 独自パケットへのカプセル化機能

ネットワーク環境の把握及び経路指示を行う NMS は1台のみ

中継サービスを担う TRS に膨大な負荷が集中する恐れ

NMS: Node Management Service
TRS: Tunnel Relay Service

1. エンドノード：全ての端末にクライアントプログラムの追加が必要
   ➡ IoT 機器や専用サーバをはじめ一部の端末は既存システムの変更が困難
2. クラウドサービス：各サーバは１インスタンスでの運用を前提に設計
   ➡ 障害発生時に通信経路指示や中継処理が不可能となる単一障害点が存在

- 改良困難な端末を含めた多様なエンドノードの包括的なサポートが必要
- 単一障害点の解消及び処理負荷に応じた拡張性の考慮が必要

# 本研究の目的

CYPHONIC におけるサービス利用多様性の実現と
エンドノードの増加を想定したスケーラブルなクラウドシステムを提案

改良困難な端末を含めた多様なエンドノードの包括的なサポート

- CYPHONIC 接続用の外部端末 (ゲートウェイ装置) を導入
- 複数の既存端末を集約して同時にオーバーレイネットワークへ接続

単一障害点の解消及び処理負荷に応じた規模拡張性設計

- 各クラウドサービスのクラスタリングと機能の多重化
- トラフィック量に応じた水平スケーリングと負荷分散

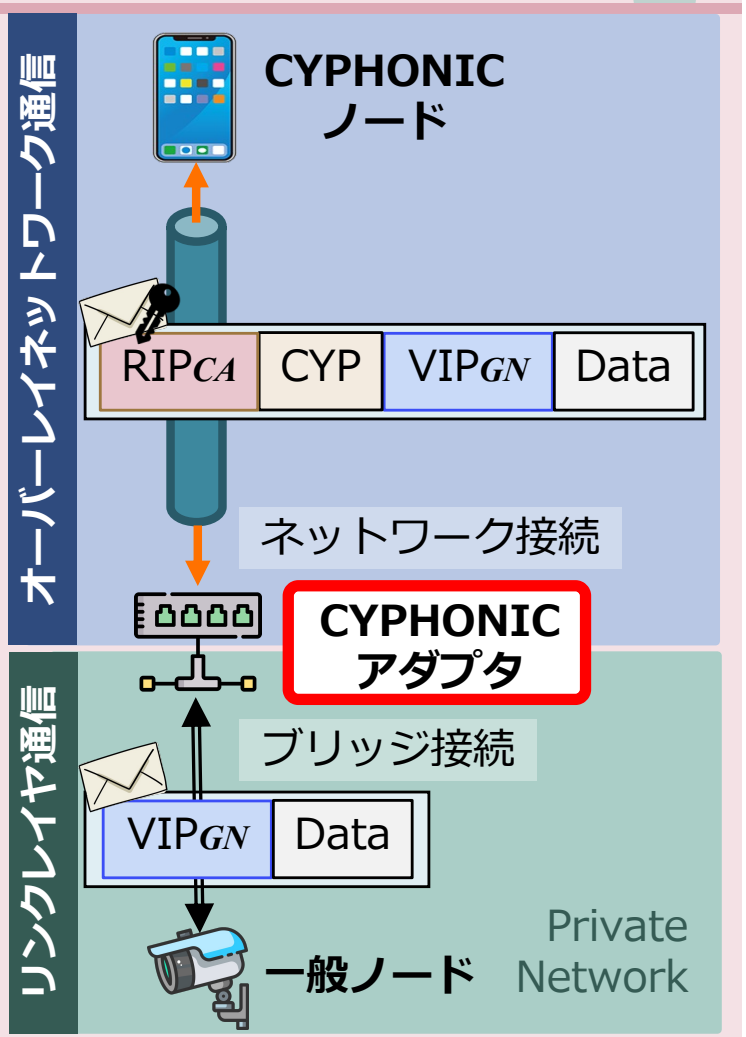
# 先行研究と課題点：エンドノード

**従来のアプローチ**

エンドノード：アダプタ端末による代行処理

- 一般ノードは隣接設置されるアダプタに接続して CYPHONIC サービスを利用
- 一般ノードと CYPHONIC ノードに介在して所定のパケットフォーマットに相互変換

※一般ノード：既存システムの改良が困難な端末




**課題**

一般ノード と CYPHONIC アダプタが 1対1 対応

➡ ネットワーク規模の拡大や端末の管理が煩雑化

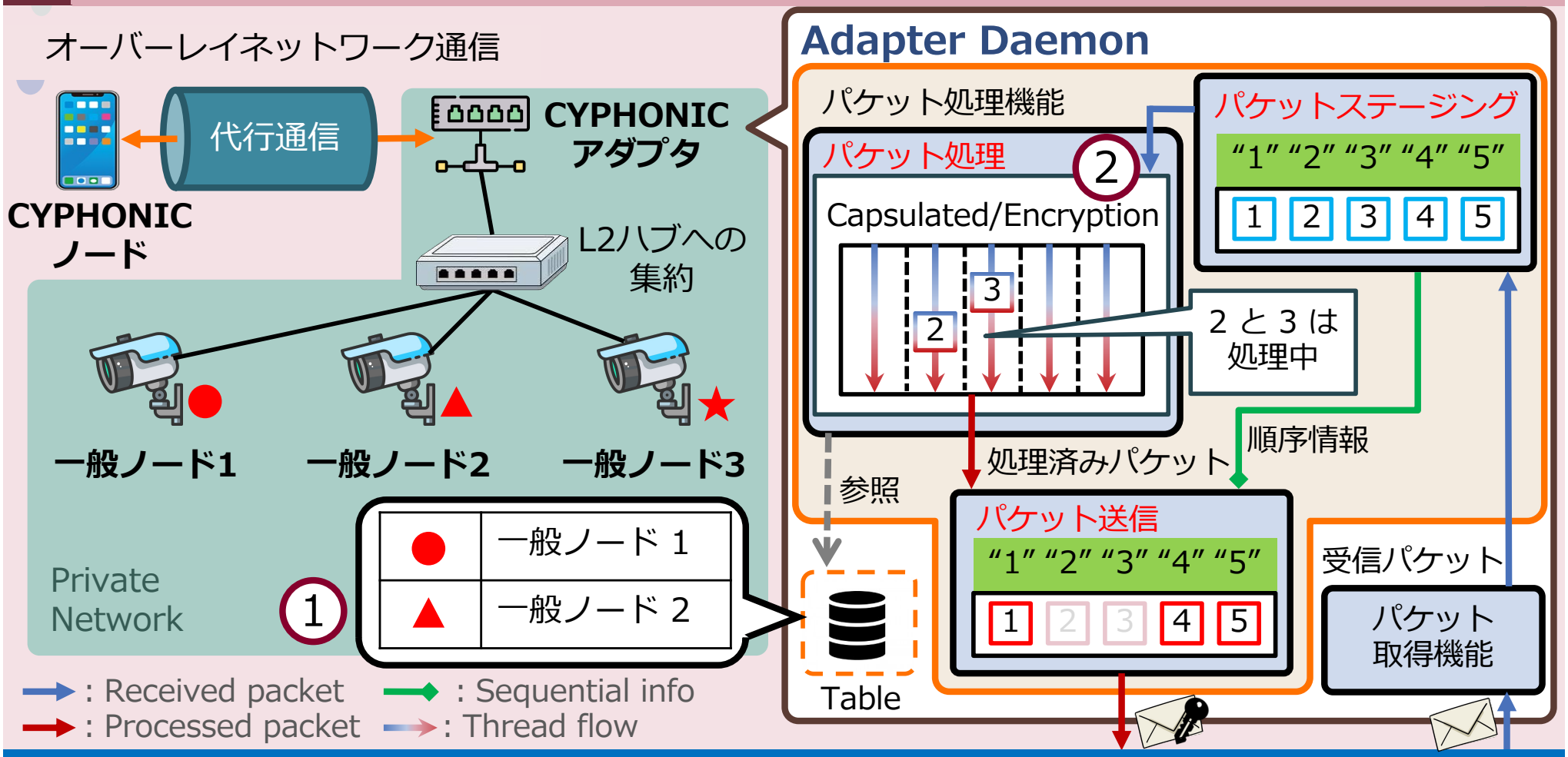
➡ 実運用におけるコストの増大が懸念

VIP: Virtual IP Header
RIP: Real IP Header
CA: CYPHONIC Adapter
GN: General Node
CYP: CYPHONIC Header

複数一般ノードを１台の CYPHONIC アダプタに集約する仕組みが必要

➡ 既存アダプタを拡張して複数一般ノードの同時接続をサポート

# 複数一般ノードを想定したアダプタ拡張設計



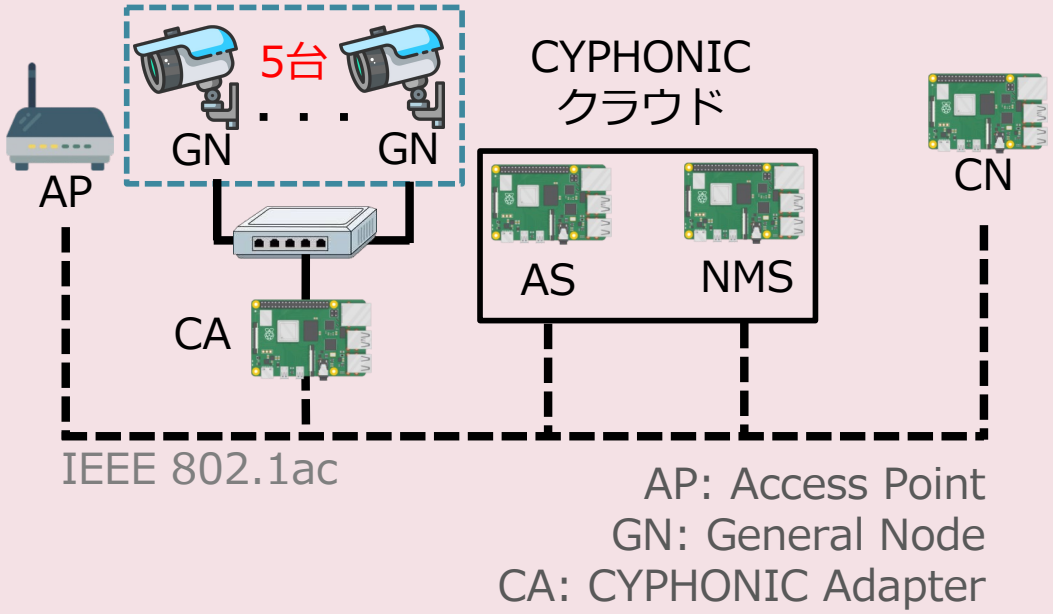

1. 複数一般ノードの管理機能を追加
   - Outgoing/Incoming において特定の値をキーとして管理テーブルを参照
2. パケット送受信処理に伴う同時実行性とパケット順序処理機能の追加
   - パケット処理機能のマルチスレッド対応と並行・非同期処理を搭載

# CYPHONIC アダプタの動作検証

CYPHONIC アダプタを使用した映像配信デモンストレーション
- 5台の一般ノード（カメラ）から映像を配信
- CYPHONIC ノードとして動作する端末から映像を確認

**検証環境**

- 映像配信：MJPEG-Streamer
- 通信遅延測定：ping
- 通信スループット測定：iperf3




**基礎評価**

オーバーレイネットワーク上で
30 FPS 程度の映像配信を確認

一般的な防犯カメラで用いられる
フレームレートで配信可能

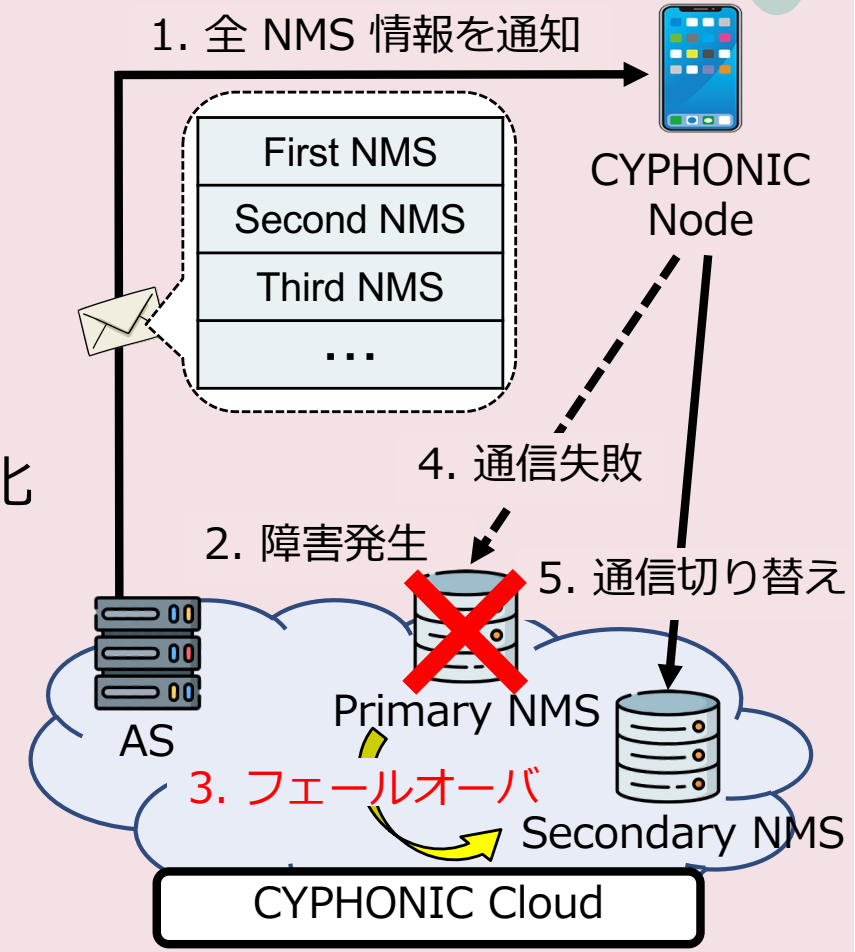
| 一般ノード一台あたりの通信性能 | |
|---|---|
| UDP Throughput | 6.92 Mbits/sec |
| TCP Throughput | 7.82 Mbits/sec |
| Round-trip time | 5.43 ms |

# 先行研究と課題点：クラウドサービス

**従来のアプローチ**

CYPHONIC クラウド：単一障害点の解消

- 複数台分の NMS 宛先情報をメッセージに含めて CYPHONIC ノードに通知
- NMS のみ Master-Slave 方式による多重化により障害発生時に稼働システムを変更




**課題**

冗長度に応じてパケットサイズが肥大化

➡ エンド端末のパケット処理機能が複雑化

障害を前提としてクラウドシステムを設計

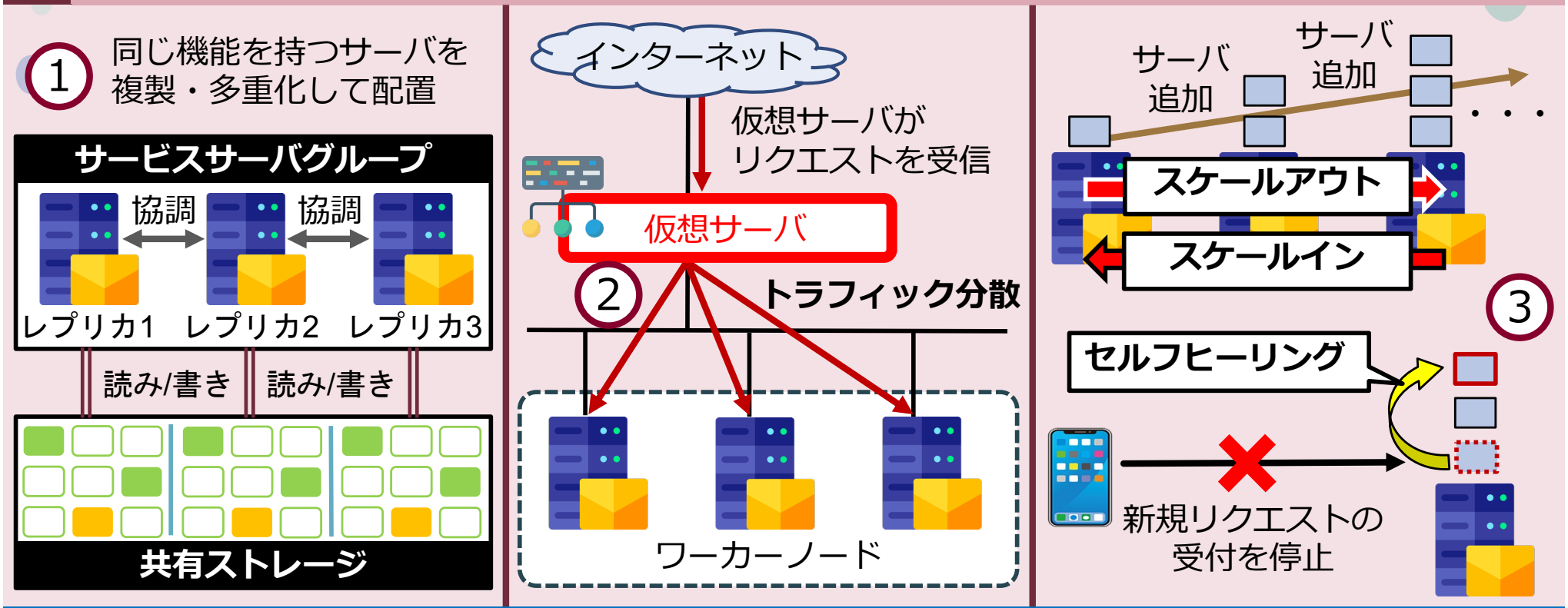
➡ 障害を未然に防ぐ仕組みや設計について未検討

AS: Authentication Service
NMS: Node Management Service

耐障害性 (Fault Tolerance) と 規模拡張性 (Scalability) を兼ね備えた設計

➡ 機能の多重化とトラフィックに応じたスケーリング機能の搭載

# 耐障害性及び規模拡張性の実現手法




1. サーバクラスタリングによる機能の多重化
   - サーバを複数台準備して Peer-to-Peer 構成を取ることで機能を多重化
2. リクエスト及びトラフィックの分散
   - 単一サーバあたりに流入するジョブをクラスタの前段で分散
3. オートスケーリング・オートヒーリング
   - トラフィック量に応じて自動的にスケールアウト・スケールインを実行
   - 障害時には新規リクエストの受付を停止すると共に自動的に復旧

# 提案するCYPHONICクラウドの構成概要

各サービスサーバを Kubernetes (分散コンテナ実行基盤) 上に構築
➡ コンテナ化されたサービスを Pod 単位でインスタンスに展開して運用

**サービスサーバの複製と多重化**

- AS, NMS, TRS を分散・多重化して配置

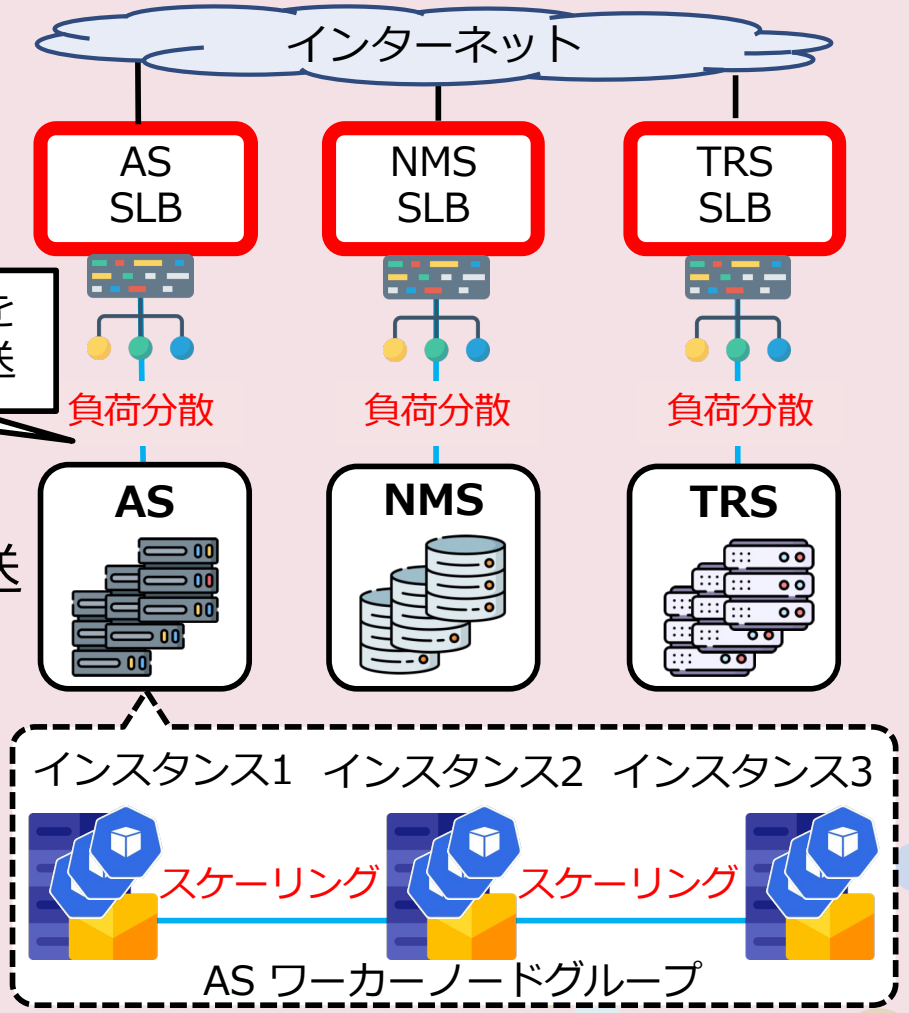
**通信リクエストの分散**

- サービス毎にリクエストを受け付けるロードバランサを配置してジョブを転送

**オートスケーリング・オートヒーリング**

- HPA 及び 宣言的な構成 に基づき Pod の台数を動的に変更

➡ ex. ある Pod の CPU 使用率が80%を超えた場合に１つ追加する




SLB: Software Load Balancer
HPA: Horizontal Pod Auto-scaler

# まとめ：研究進捗及び今後の方針

CYPHONIC におけるサービス利用多様性の実現と
エンドノードの増加を想定したスケーラブルなクラウドシステムを提案

改良困難な端末を含めた多様なエンドノードの包括的なサポート
➡ 多様な IoT 端末を同時にサポート可能な CYPHONIC アダプタの拡張設計

- 進捗：5 台の一般ノードを同時にオーバーレイネットワークへ接続可能
- 課題：単一ノードあたりのスループットが低くく検証・評価が不十分

クラウドサービス単一障害点の解消及び処理負荷に応じた規模拡張性設計
➡ 耐障害性と規模拡張性を兼ね備えたスケーラブルなクラウド設計

- 進捗：分散コンテナ実行基盤によって CYPHONIC クラウドを構成
- 課題：負荷分散に伴いエンド端末の IP アドレスの取得が困難

# ここまで 他大合同報告MTG