

説明は8:50から開始します。
本資料は研究室Webページから
ダウンロードできます
<http://www.cn.riec.tohoku.ac.jp/>
トップページ → About

研究室紹介

長谷川研究室

Research Institute of Electrical Communication
東北大学 電気通信研究所 TOHOKU UNIVERSITY



長谷川研究室
Hasegawa Lab

<https://www.cn.riec.tohoku.ac.jp/>



研究室紹介

長谷川研究室

Research Institute of Electrical Communication
東北大学 電気通信研究所 TOHOKU UNIVERSITY



長谷川研究室
Hasegawa Lab

<https://www.cn.riec.tohoku.ac.jp/>

自己紹介: 長谷川 剛 (ごう)

- **東北大学 電気通信研究所 情報通信基盤研究部門 ネットワークアーキテクチャ研究室 教授**
 - go.hasegawa.a6@tohoku.ac.jp
 - 研究室Web <https://www.cn.riec.tohoku.ac.jp/>
 - 個人Web <https://www.riec.tohoku.ac.jp/~hasegawa/>
 - 52歳 (2026/9現在)
- **略歴**
 - 1992年～ 大阪大学 基礎工学部 情報工学科
 - 1995年～ 大阪大学 大学院基礎工学研究科 博士前期課程
 - 1997年～ 同博士後期課程 → 経済学部助手 (経済学部のネットワーク環境の運用管理)
 - 2000年～ 大阪大学 サイバーメディアセンター助手 (大学全体のキャンパスネットワーク管理)
 - 2002年～ 同助教授 (2007年より准教授)
 - 2019年～ 東北大学電気通信研究所 教授
- **研究テーマ: コンピュータネットワークアーキテクチャ**
 - 高性能・高可用ネットワーク、仮想化ネットワーク/システム設計・制御、5G/6Gモバイルネットワークアーキテクチャ、インターネット輻輳制御、情報ネットワークシステム性能評価、AI/機械学習のネットワーク応用、…

ネットワークアーキテクチャ研究室 (長谷川研究室)

- 発足: 2019年度
- 研究キーワード:
 - 仮想化ネットワークシステム
 - モバイルネットワークアーキテクチャ
 - AI/機械学習を用いたネットワーク制御
 - IoTのためのネットワーク技術
 - 超高速・超多量・低遅延/高信頼ネットワーク技術
 - ...
- 研究室Web:
 - <http://www.cn.riec.tohoku.ac.jp/>
- 連絡先 (質問など):
 - go.hasegawa.a6@tohoku.ac.jp



仮想化技術とAIで切り拓く、次世代ネットワークアーキテクチャ

より高速・高信頼・省エネルギーで、利便性の高い情報通信基盤の実現を目指して

現在のインターネットの課題

- ✓ 基本構造は40年以上変わらず
- ✓ 多様化するアプリケーションや要求品質に十分対応できない
- ✓ 輻輳や遅延、エネルギー消費、運用の複雑さが増大



研究のアプローチ

仮想化技術を基盤に、ネットワークの構成・制御を柔軟に実現

仮想化

サーバやネットワーク機能をソフトウェアで実現・配置

分散・柔軟制御

どこでも実行・変更できる自由度の高いネットワーク

AI・最適化

AIや数理最適化で状況に応じて賢く制御

次世代ネットワークアーキテクチャ

高性能・高信頼・低遅延・省エネルギーなネットワーク基盤

研究室の目標

ネットワークサービス最適化のための新しいネットワークアーキテクチャの確立



高性能・低遅延

輻輳を予測・回避し、快適な通信を実現



高信頼・可用性

障害に強く、安定したサービス提供を支援



省エネルギー・最適化

ICTインフラ全体のエネルギー消費を最適化



多様なサービスに柔軟対応

モバイル、IoT、クラウド、AI時代の要求に対応

研究テーマ例

1. インターネット輻輳制御の革新

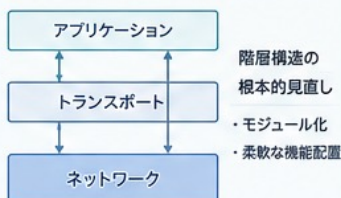
ネットワーク内での制御やAIの活用により、公平で高スループットな通信を実現



- ・ In-Network Congestion Control
- ・ 機械学習による状態推定と制御
- ・ BBR/CUBICの性能評価・改善

2. 次世代アーキテクチャ設計

プロトコル階層や制御の在り方を見直し、新しいインターネットの実現を目指す



3. 大規模ネットワークの運用・管理

膨大な管理情報やログを効率的に収集・解析し、安定運用と異常検知を支援



- ・ ログの圧縮転送技術
- ・ 遮隔管理・異常検知の高度化
- ・ 大規模ネットワークの性能評価

4. エネルギー最適化

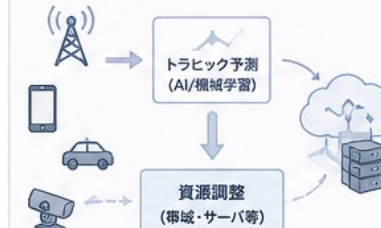
デジタルツインと最適化技術により、ICTインフラの消費エネルギーを最小化



- ・ クラウド/エッジ協調最適化
- ・ 動的な資源配分・経路制御
- ・ 計算コストと性能のトレードオフを考慮

5. モバイル・IoTネットワーク制御

トラフィック予測に基づき、先回りして資源を調整 多様なサービスを低コストで収容



研究手法



理論解析



シミュレーション



プロトコル設計



システム実装・評価

未来の情報通信基盤を支える
革新的なネットワーク技術を創出します

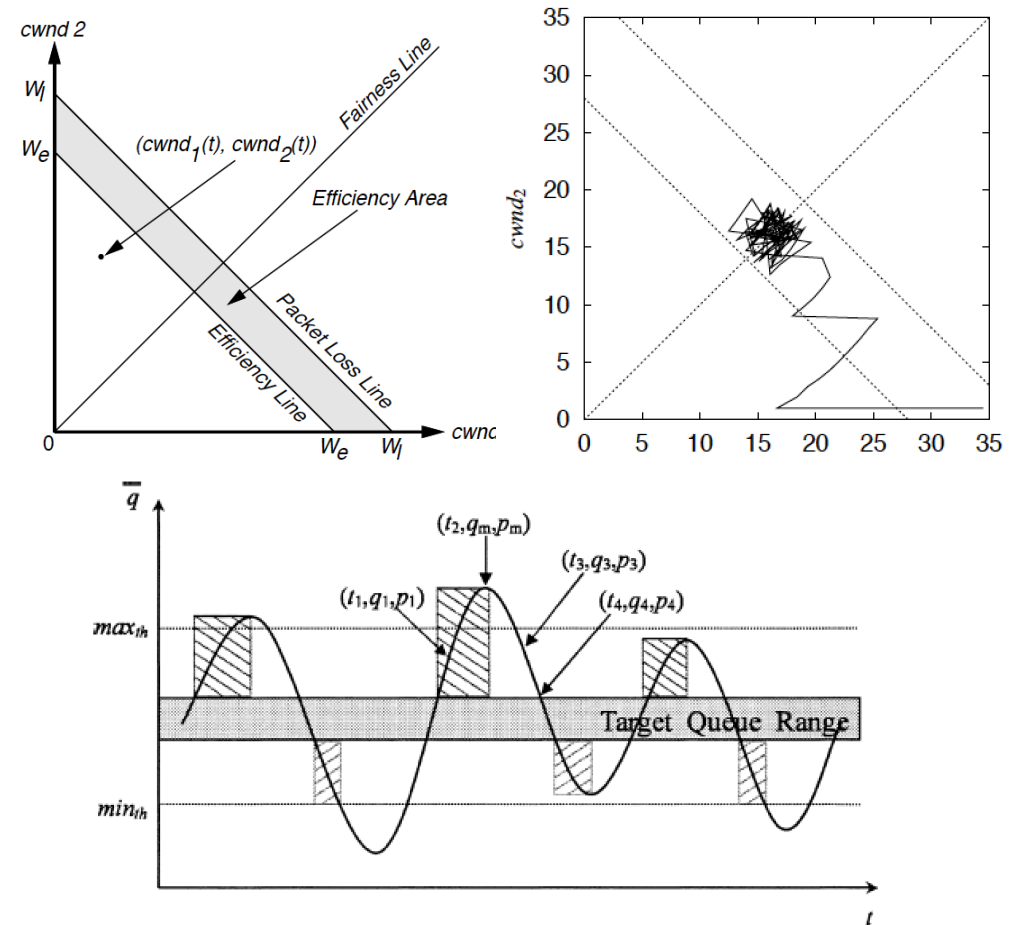
インターネット輻輳制御機構に関する研究 (1)

● インターネット輻輳制御とは

- Black-boxed architecture
- PC、サーバ、スマホなどの端末がネットワークの輻輳状態を推定し、転送速度を自律分散的に調整
- インターネット誕生後40年あまり研究が続けられているが、決定的な手法は今だに皆無

● 研究例

- 輻輳制御へのAI/機械学習の適用
 - 空間的、時間的変動の大きなインターネットでは、1つで万能な輻輳制御は不可能
 - 適用箇所や時間に応じてアルゴリズムを選択・調整し利用する
- In-Network Congestion Control
 - 従来のエンド端末だけで行う輻輳制御から脱却し、ネットワーク内で輻輳制御を行う新しいネットワークアーキテクチャの探求



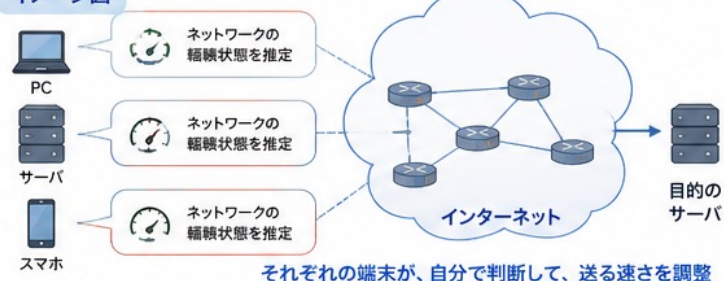
$$\begin{aligned}
 W_j^i &= \frac{W_j^i}{2} + \frac{1}{\tau_j} \frac{\tau_1 \tau_2}{2(\tau_1 + \tau_2)} W_s \\
 &= \frac{1}{\tau_j} \frac{\tau_1 \tau_2}{2(\tau_1 + \tau_2)} W_s - \left(\frac{1}{2}\right)^{i-1} \left(\frac{1}{\tau_j} \frac{\tau_1 \tau_2}{2(\tau_1 + \tau_2)} W_s - W_j^1 \right)
 \end{aligned}$$

インターネット輻輳制御機構に関する研究 (2)

インターネット輻輳制御とは

- PC、サーバ、スマホなどの端末がネットワークの輻輳状態を推定し、転送速度を自律分散的に調整
- Black-boxed architecture
- インターネット誕生後40年あまり研究が続けられているが、**決定的な手法は今だに皆無**

イメージ図



💡 ネットワークの内部の状態は直接わからない (Black-box) だからこそ、より良い推定や制御の技術が重要!

長谷川研究室のアプローチ

より速く・公平で・遅延の少ないインターネットを実現するための輻輳制御アルゴリズムを研究しています

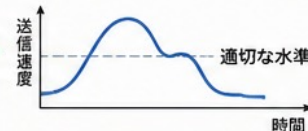
状態の観測と推定

遅延、パケット損失、ECNマークなどの情報から、輻輳状態や他の通信の様子を推定



賢い制御アルゴリズム

推定結果に基づいて、送信速度 (レート) を自律的に調整



- ・遅延を増やさない
- ・公平に帯域を使う
- ・スループットを最大化

実装と評価

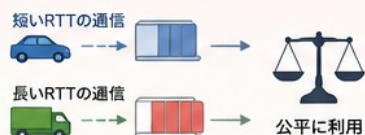
実ネットワークやシミュレーションで効果を検証し、改善をくり返す



研究テーマの例

BBRの改良による公平性と低遅延の実現

BBRの状態遷移や遅延検出の仕組みを改良し、異なるRTTの通信でも公平に、遅延を抑える



ECNがない環境でも遅延を減らす仕組み

eBPFを使って、端末側から“仮想的なECN信号”を作り出し、遅延を大幅に削減



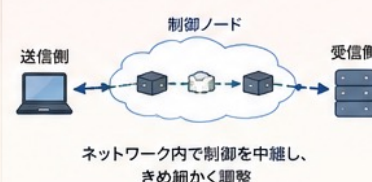
AI/機械学習を使った輻輳制御

競合状況やネットワーク状態を学習し、状況に応じて最適な制御を行う



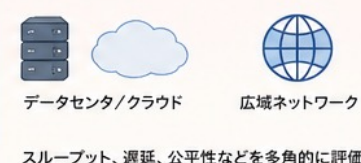
プロトコル設計と新しいアーキテクチャの検討

ネットワーク内での制御や、多段階の制御構造など、次世代インターネットの形を探る



大規模ネットワークでの実証実験と評価

データセンターやクラウド、広域ネットワークで実験を行い、実環境での効果を検証



実験結果の例

平均遅延の削減 (典型シナリオ)
BBRと比べて最大約30~90%削減



95パーセンタイル遅延の削減
BBRと比べて最大約50%削減



RTT公平性の向上

Jain's Fairness Indexが0.9以上を達成



リンク利用率の維持

多くの条件で98%以上を維持



スループットを落とさずに、遅延を減らし、公平に帯域を使うことを実現!

目指す未来

誰もが快適に、安定してインターネットを使える未来へ
動画、ゲーム、IoT、AIなど、あらゆるサービスを支える
高性能・低遅延・公平なインターネットを実現します



動画・ライブ配信



オンラインゲーム



IoT・スマート社会



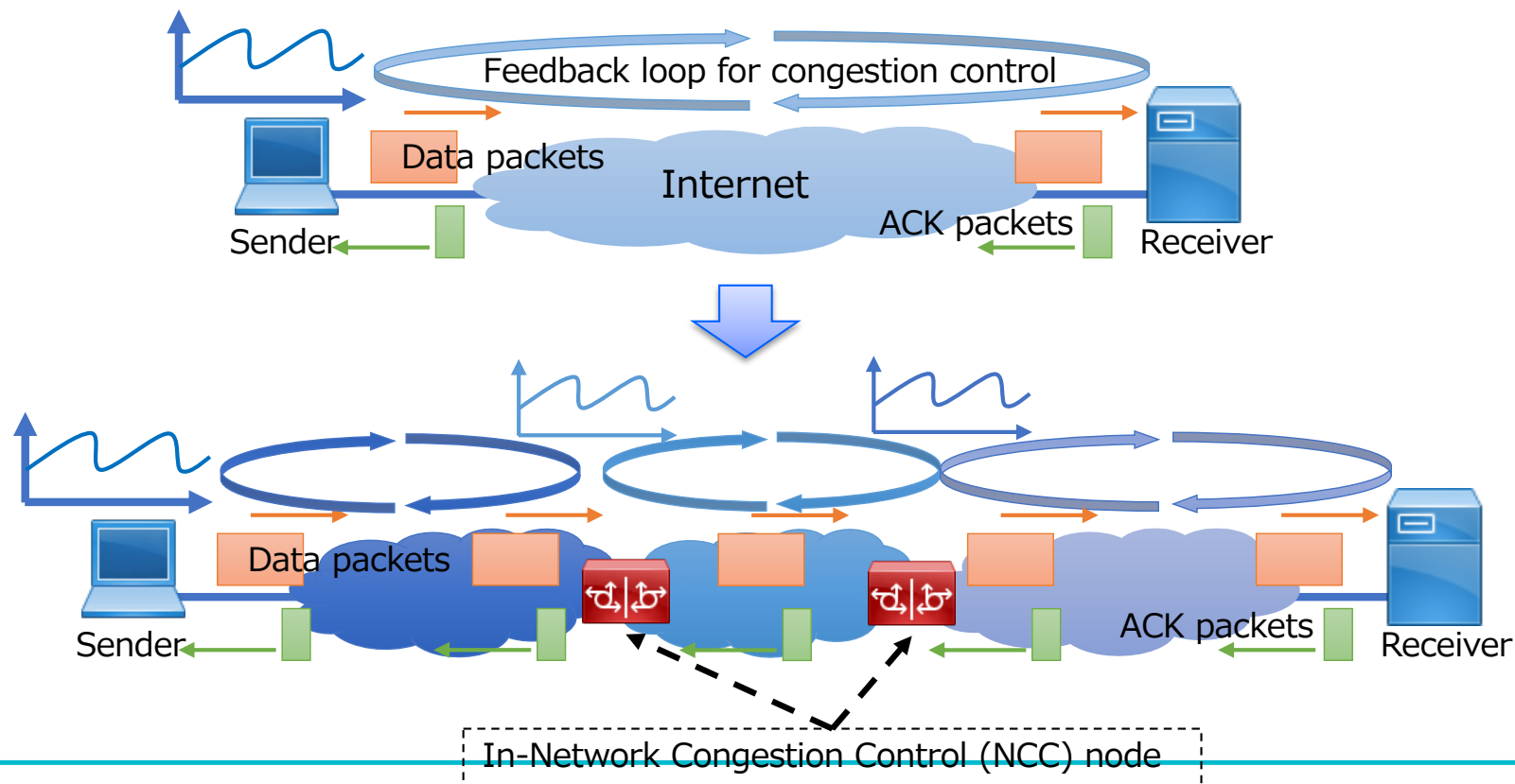
AI・クラウドサービス



理論の解析からアルゴリズム設計、実装、実験まで一貫して取り組み、インターネットの未来を支える技術を生み出します!

ネットワーク連動型輻輳制御アーキテクチャ (1)

- 従来は送受信端末間で1つである輻輳制御のフィードバックループを、ネットワーク内のノード (in-Network Congestion Control (NCC) ノード) で切断・中継
 - エンド端末間を複数のフィードバックループを用いてデータ転送を行う
- それぞれのフィードバックループで、ネットワーク特性やユーザ要求に応じた輻輳制御アルゴリズムを適用



ネットワーク連動型輻輳制御アーキテクチャ (2)

1. これまでの輻輳制御 (エンドツーエンド方式)

- 送信端末と受信端末だけで、ネットワークの混雑を推定
- パケットのやり取りやACKから、RTTやパケットロスを観測して、送信速度を自分で調整

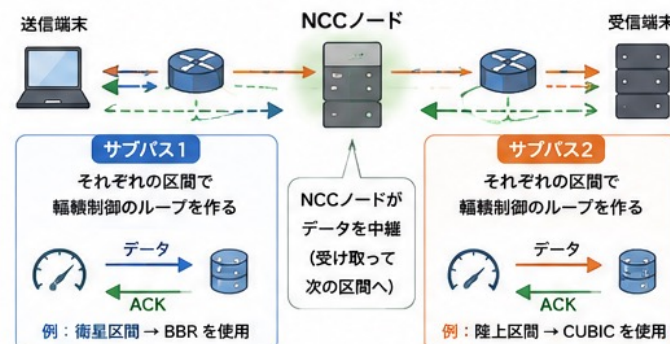


問題点

- 経路が長くなると遅延が大きくなり、スループットが下がる
- ネットワークの特性が違う区間があるとうまく制御できない
- 異なる環境の通信同士で、不公平が起きやすい

2. 提案するネットワーク内輻輳制御 (NCC) の考え方

- 経路の途中に「NCCノード」を置き、通信をいくつかの区間(サブパス)に分ける
- それぞれの区間で独立に輻輳制御を行い、区間の特性に合った方法を使う

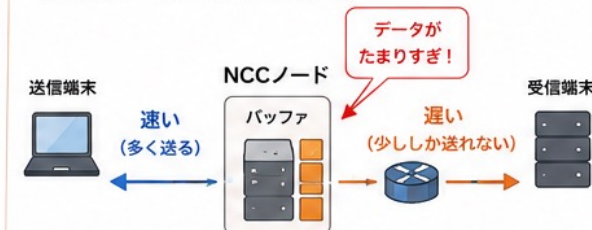


NCCのメリット



3. NCCノードで起こる問題：データのたまりすぎ

前の区間の送信速度が、次の区間の能力より大きいと、NCCノードにデータがたまり、遅延が増えたり、バッファがあふれてデータが失われたりする。



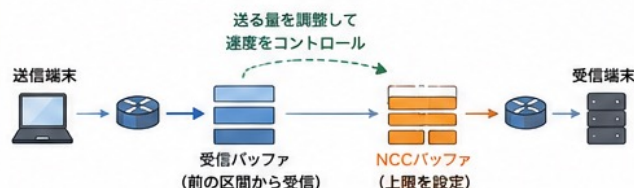
起こること

- 遅延が大きくなる
- スループットが不安定になる
- バッファがあふれてデータが失われる



4. 提案するしくみ：バッファ管理でデータのたまりを防ぐ

NCCノードで、前の区間の「受信バッファ」から「NCCバッファ」へ送る量を調整して、前の区間の送信速度をコントロールする。

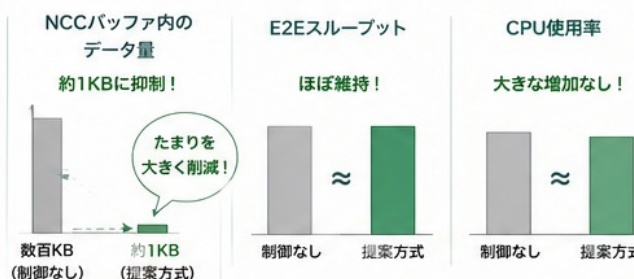


ポイント

- ✓ NCCバッファの上限を決めて、そこまでしかためない
- ✓ 受信バッファからNCCバッファへ送る量を調整
- ✓ 前の区間の送信速度を間接的にコントロール
- ✓ E2Eのスループットを落とさず、遅延を小さくできる

5. 評価結果 (実装による実験)

仮想ネットワーク環境で実装して評価しました。



実験条件の例

- ✦ 仮想マシンとLinuxを使用
- ✦ 4通りの遅延 (4/9/19/39ms)
- ✦ さまざまな遅延・帯域・ロス条件で多実験
- ✦ バッファ上限 (1KB~128KB) など

6. まとめと将来

まとめ

- NCCは、通信を区間に分けて、それぞれで最適な輻輳制御を行う新しいしくみです。
- NCCノードでのバッファ管理により、データのたまりを防ぎ、低遅延を実現できます。
- 実験により、E2Eの性能を落とさずに、NCCノードのバッファを約1KBに保てることを確認しました。

将来の展望



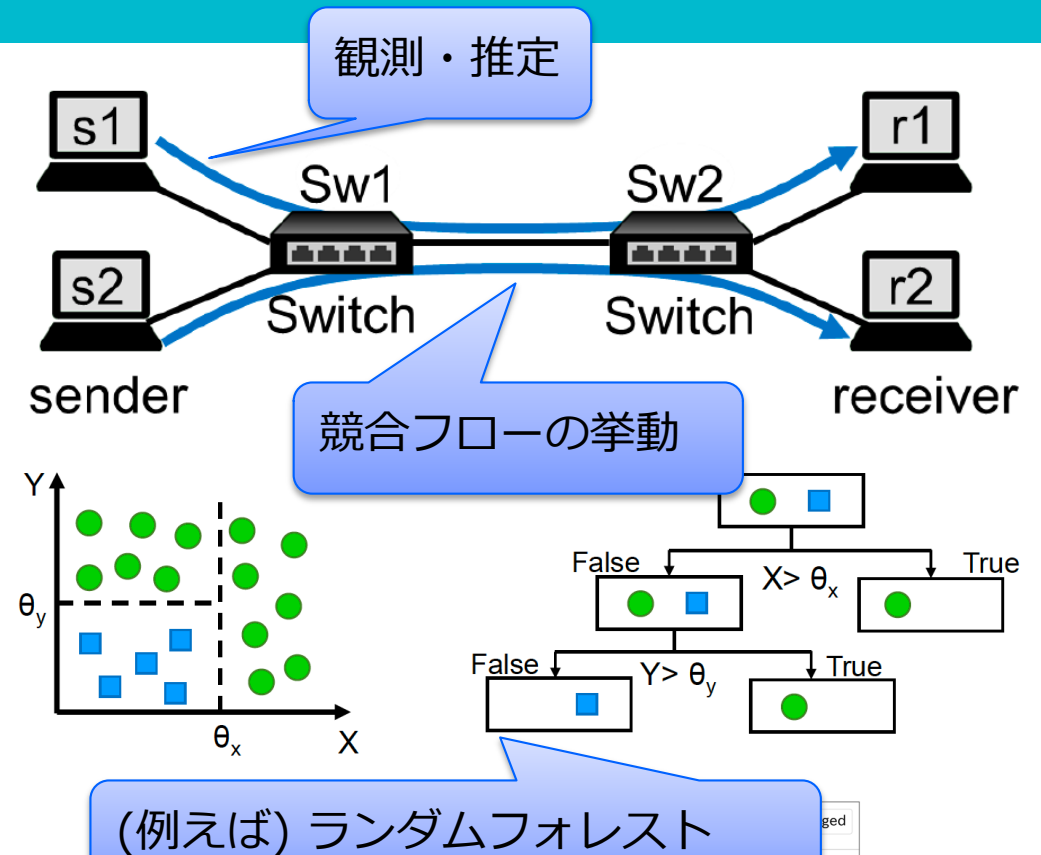
この研究が目指すもの

インターネットの多様化・大規模化に対応して、より速く・公平で・低遅延な通信を実現する、次世代の輻輳制御アーキテクチャの確立を目指します。

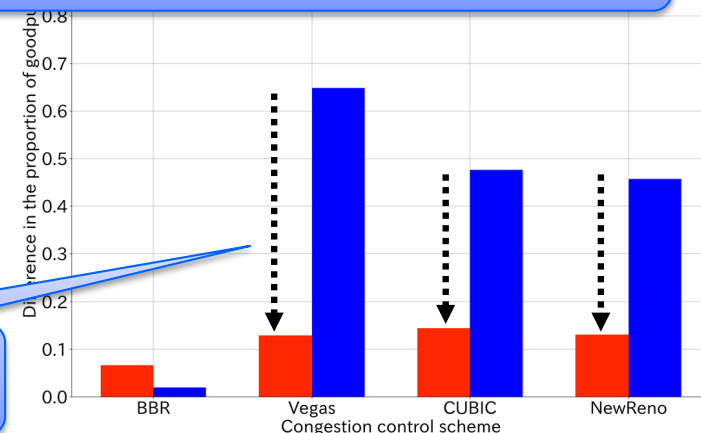


機械学習によるネットワーク状態推定と輻輳制御への応用 (1)

- 通常は知り得ない、競合関係にある他フローの挙動やネットワーク状態を推定
 - 輻輳制御アルゴリズム、競合しているフローの数、...
- 推定結果に応じて自身の挙動を変更することでフロー性能を改善



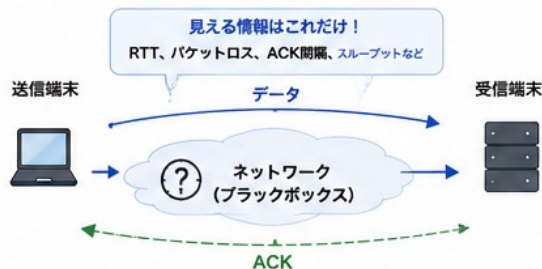
フロー間公平性の改善



機械学習によるネットワーク状態推定と輻輳制御への応用 (2)

① どんな問題があるの？

- インターネットでは、いろいろな「輻輳制御アルゴリズム」が同時に使われている（例：BBR、CUBIC、Vegas など）
- 送信端末は、RTTやパケットロスなど、限られた情報しか観測できない
- そのため、競合しているフローの数や、相手がどのアルゴリズムを使っているのかは分からない
- 結果として、スループットが下がったり、不公平が起きたりする



見えない情報の例

- ボトルネックで競合しているフローの数は？
- 相手のフローはどの輻輳制御アルゴリズムを使っている？

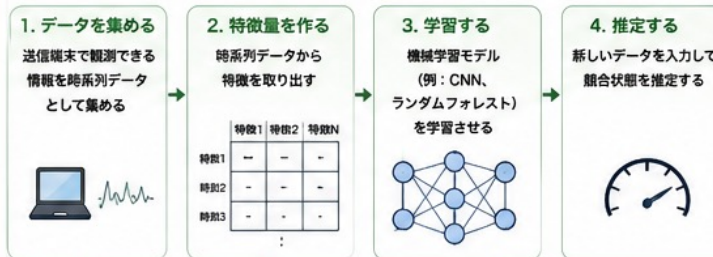
② 機械学習でネットワークの状態を推定する

推定するもの①

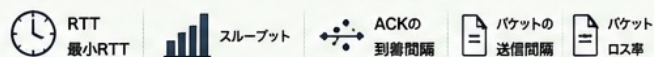
競合フローの数
(ボトルネックで何本のフローが競合しているか)

推定するもの②

競合フローの輻輳制御アルゴリズム
(相手はBBR? CUBIC? Vegas?)



使う情報の例 (送信端末で見えるもの)



送信端末だけで推定できるので、ネットワーク機器や受信端末を変えなくても使える！

③ 推定結果を輻輳制御に活用する

推定した状態を使って、よりよい制御を行う！

使い方の例①：競合フローの数を推定して送信速度を調整



使い方の例②：競合フローのアルゴリズムを推定して、自分のアルゴリズムを切り替える

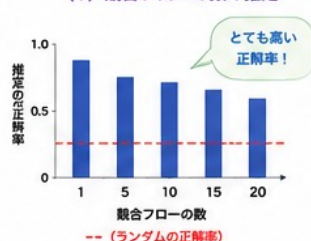


期待される効果

- ✓ 異なるアルゴリズム同士の不公平性を改善
- ✓ スループットを向上し、遅延を低減
- ✓ ネットワークの変化に柔軟に対応
- ✓ 送信端末だけで完結し、導入が簡単

④ 実験で確かめたこと

(1) 競合フローの数の推定



学習した競合フロー数に対して、正解率はほぼ100%！

(2) 競合フローのアルゴリズムの推定



競合相手のアルゴリズムを、高い精度で推定できることを確認！

⑤ 実験環境と評価項目

実験環境 (仮想ネットワーク)



評価するもの



いろいろな条件で実験

- ✓ いろいろな輻輳制御アルゴリズムの組み合わせ (BBR、CUBIC、Vegas、NewReno など)
- ✓ ボトルネックの帯域、遅延、バッファサイズを変える
- ✓ 学習データと異なる条件 (見たことのない状況) でもテスト

⑥ まとめ



送信端末だけで見える情報から、競合フローの数や相手のアルゴリズムを機械学習で推定できる！



推定した結果を使って、自分の輻輳制御をうまく調整することで、不公平を改善し、より良い通信を実現できる！



誰でも導入しやすく、将来のインターネットで重要な役割を果たす技術になる！



将来の展望



NCC ノードと組み合わせたスマートなネットワーク制御



モバイル通信や衛星通信など多様なネットワークへの適用



さらなる精度向上とリアルタイム制御への応用



他の技術 (SDN、NFVなど) との連携で未来のネットワークへ

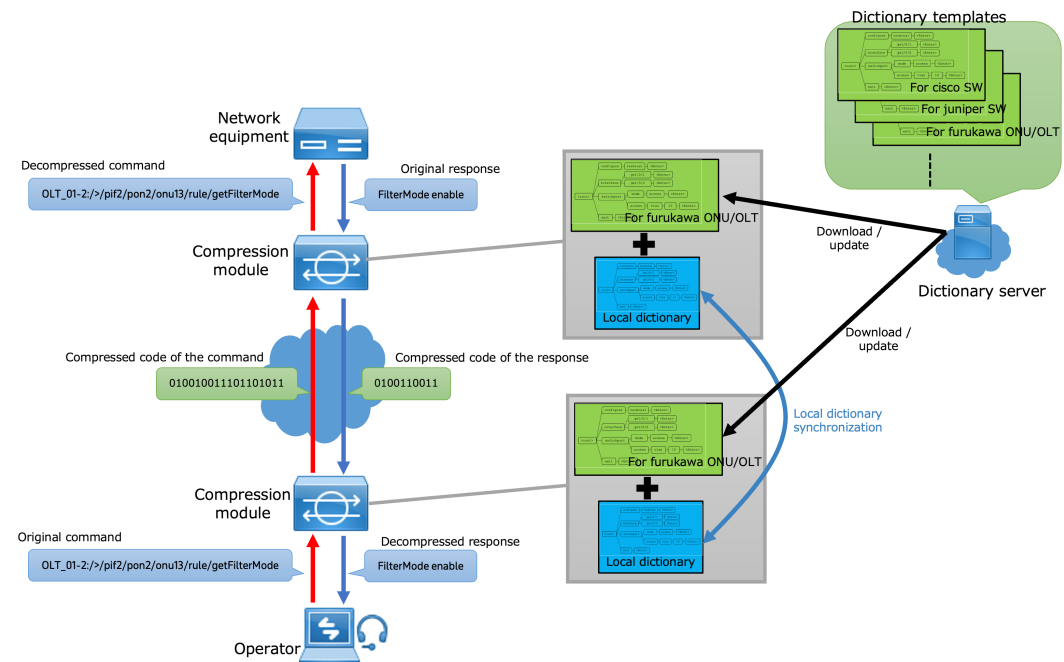
大規模ネットワーク管理のための情報圧縮転送技術 (1)

● 大規模ネットワークの遠隔管理、ログ収集、AI技術による異常検知

- オペレータやログ収集サーバと、多数のネットワーク機器間で膨大な情報が交換される
- 特にネットワーク障害による性能低下時の収容が課題

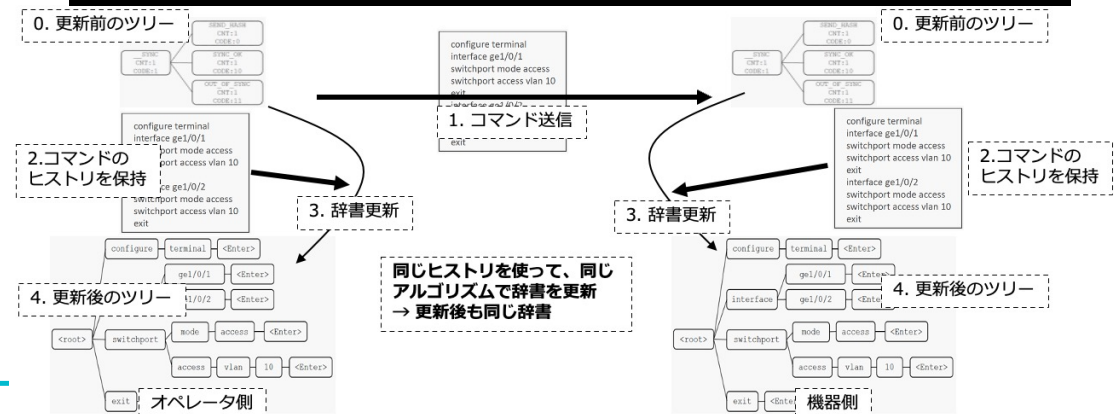
● トラフィック圧縮技術

- 辞書分離型の全く新しい圧縮アルゴリズムにより、情報量を90%以上削減
- 大規模ログ管理システムへの応用



```

2024/03/31 13:37:09.832102 sv_cswd: 1523: main.c: send_port_command: 161: open request pom12
2024/03/31 13:37:10.548771 sv_sequence: 1258: Telnetc.c: logging_atcmd: 344: sv => pom06 : [8Sv_getTemperature#012]
2024/03/31 13:37:10.649081 sv_sequence: 1258: Telnetc.c: logging_atcmd: 346: pom06 => sv : [8Sv_getTemperature#015#012575#015#015#012#015#012#> ]
2024/03/31 13:37:10.832494 sv_sequence: 1264: Telnetc.c: logging_atcmd: 344: sv => pom12 : [8getPortInfo 1#012]
2024/03/31 13:37:10.834540 snmptrapd[1170]: 2024-03-31 13:37:10 169.254.10.112(via UDP: [169.254.10.112]:53737->[169.254.10.10]:162) TRAP, SNMP v
erprises.246.4.2.24.1.11.1.0 = INTEGER: 26#011SNMPv2-SMI::enterprises.246.4.2.24.1.11.2.0 = INTEGER: 12#011SNMPv2-SMI::enterprises.246.4.2.24.1.1
0.00.00.00 #011SNMPv2-SMI::enterprises.246.4.2.24.1.11.6.0 = STRING: "4064A4F0237E" #011SNMPv2-SMI::enterprises.246.4.2.24.1.11.7.0 = STRING: "ope
2024/03/31 13:37:10.906182 sv_traphandler: 5152: mux_csw.c: writeCswPom1GPort: 495: start to OPEN the pom12 port1 on CSW for 1G
2024/03/31 13:37:10.906935 sv_traphandler: 5152: mux_csw.c: writeCswPom1GPort: 441: start to OPEN the pif12 port1 on CSW for 1G
2024/03/31 13:37:10.907400 sv_traphandler: 5152: mux_csw.c: writeCswPom1GPort: 609: start to OPEN the pom12 port1 on CSW for 10G
2024/03/31 13:37:10.908700 sv_traphandler: 5152: mux_csw.c: writeCswPom1GPort: 549: start to OPEN the pif12 port1 on CSW for 10G
2024/03/31 13:37:10.932794 sv_sequence: 1264: Telnetc.c: logging_atcmd: 346: pom12 => sv : [8getPortInfo 1#015#012#015#012#> ]
  
```

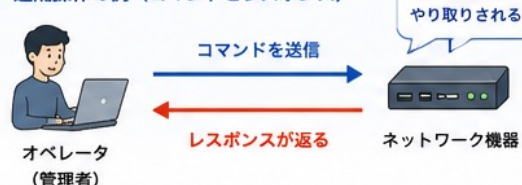


大規模ネットワーク管理のための情報圧縮転送技術 (2)

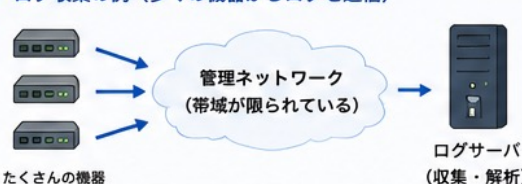
1. どんな問題があるの？

ネットワーク機器の遠隔操作やログ収集では、
大量の情報を送る必要があり、
限られた**帯域**を**圧迫**してしまう。

遠隔操作の例 (コマンドとレスポンス)



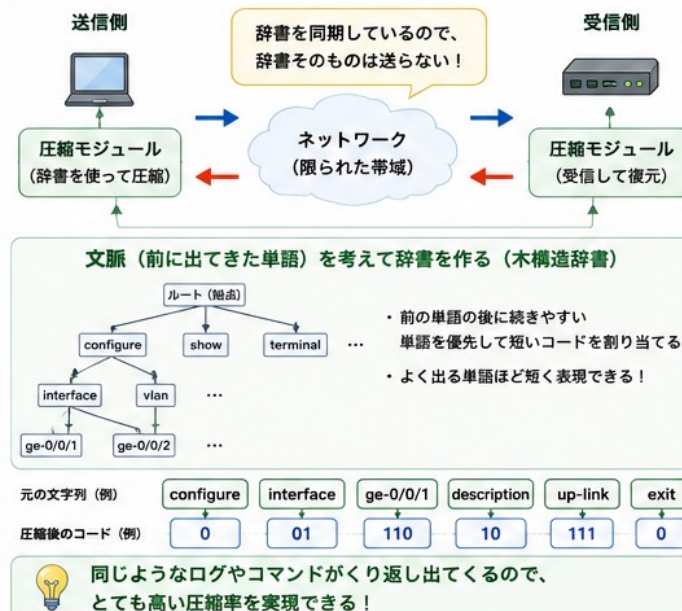
ログ収集の例 (多くの機器からログを送信)



そのままだと、通信が遅くなったり、
リアルタイムの監視や解析が難しくなる！

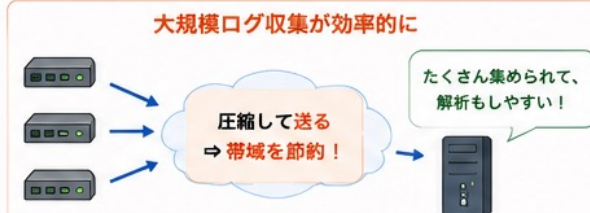
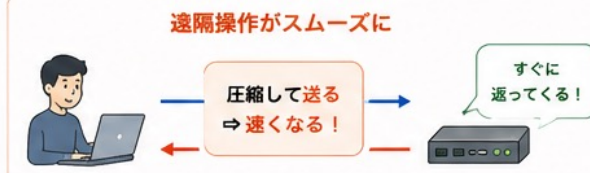
2. 私たちの提案する圧縮のアイデア

送る情報を“単語のつながり”として覚え (辞書を作り)、
短いコードに置き換えて送る！



3. 圧縮した情報をどう使うの？

圧縮することで、限られた帯域でも安心して使える！

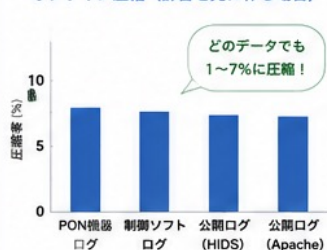


期待できる効果

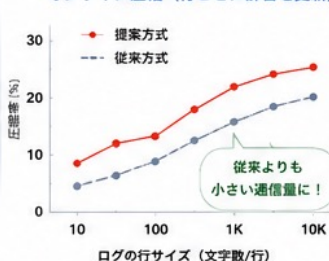
- ✓ 通信量を大幅に削減
- ✓ 操作のレスポンスが速くなる
- ✓ リアルタイム監視や障害検知がしやすくなる
- ✓ 多くの機器を効率よく管理できる

4. 実験結果 (例)

オフライン圧縮 (辞書を先に作る場合)



オンライン圧縮 (行ごとに辞書を更新)



✓ 短い行のログでも、従来方式より高い圧縮性能を達成！

5. 提案方式のポイント



文脈を考えた木構造辞書

前に出た単語の後に続きやすい単語を優先して登録



送受信側で辞書を同期

辞書を送らないので、通信量が増えない



時刻を数値で扱う

日時の文字列を数値にして、辞書の肥大化を防ぐ



長い文字列を細かく分割

ファイルパスやIDを小さな単語に分けて再利用しやすく



オンライン学習に対応

新しいログが来るたびに辞書を更新して使える

6. 全体の流れ



将来の展望



さらに多くの種類の機器や
システムに適用



AIと組み合わせて
異常検知や自動運用へ



暗号化された情報への
圧縮にも対応



より大規模・複雑なネットワークを
安全かつ効率的に運用

マイクロサービスアーキテクチャに基づくソフトウェア実装構成の最適化

1. どんな問題を解決したいの？

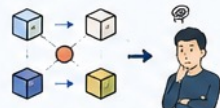
アプリケーションをマイクロサービスという
小さな機能の集まりで作るのが**主流**になっています。
でも、作り方を間違えると...

ソフトをまとめすぎると...



- ・1つが壊れると全部が止まる
- ・拡張にくい

細かく分けすぎると...



- ・壊れる場所が増える
- ・通信やサーバ資源のムダが増える



どの機能をまとめるか(実装構成)と
どのソフトを何台に増やすか(冗長構成)を
うまく決めて、**高い可用性**を実現したい！

2. 私たちの考え方(モデル化)

アプリケーションを、機能・ソフト・サーバの3つで考える

機能(サービス)

アプリの機能の
つながり



ソフトウェア
(マイクロサービス)

機能をまとめた
ソフトのかたまり



サーバ

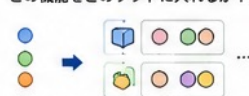
CPU・メモリなどの
資源と故障する可能性



2つの設計を決める(最適化の対象)

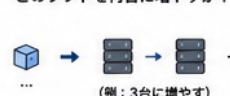
① 実装構成

どの機能をどのソフトに入れるか？



② 冗長構成

どのソフトを何台に増やすか？



目的：資源の制約(CPU・メモリ・予算など)の中で、
システムの可用性を最大にすること！

3. 提案する方法(2段階で最適化)

たくさんの候補の中から、良い設計を効率よく探す！

ステップ1: 実装構成を決める(資源のムダを減らす)

いろいろな
候補を作る
(多量スタート)



3つの操作で
近くの候補へ移動

- ・機能を別のソフトへ移す
- ・1つのソフトを分ける
- ・2つのソフトをまとめる



一番良い候補を
選ぶ(RCEが高い)



ステップ2: 冗長構成を決める(可用性を高める)

最小の台数から
スタート



ヒルクライミングで
台数を増減



一番可用性が高い
構成を選ぶ



※ RCE(資源利用効率)という指標を使って評価します。
RCEが高いほど、同じ資源でより多くの可用性を実現できます。

4. 探索の3つの操作(ステップ1で使う)

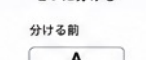
① 機能を移す

ある機能を別の
ソフトへ移動する



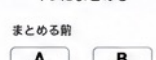
② ソフトを分ける

1つのソフトを
2つに分ける



③ ソフトをまとめる

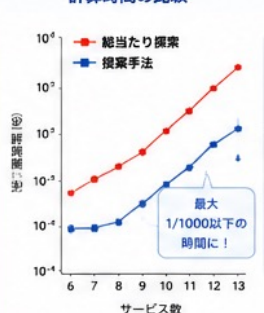
2つのソフトを
1つにまとめる



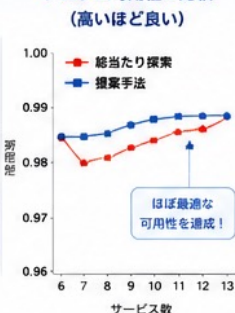
この3つをくり返して、より良い実装構成を探します！

5. 評価結果(例)

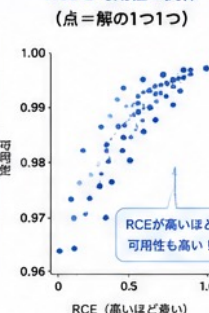
計算時間の比較



システム可用性の比較 (高いほど良い)



RCEと可用性の関係 (点=解の1つ1つ)



✓ 提案手法は、計算を大幅に速くしながら、ほぼ最適な解を見つけられます！

6. どんな場面で役立つの？

連環ネットワーク運用システム



- ・障害に強いシステム設計
- ・必要な資源をムダなく使用
- ・広いネットワークでも安定運用

クラウド/データセンターのアプリケーション



- ・サービスの急な増減に対応
- ・コストを抑えて高い可用性
- ・自動で最適な構成を決定

ポイントまとめ



機能のまとめ方と、
増やす台数を同時に最適化！



独自のヒルクライミング法で、
速くて良い設計を見つける！



限られた資源の中で、
システムの可用性を最大化！

今後の展望



ネットワーク遅延や通信費用
も考えた最適化へ



AIと組み合わせて、
自動運用につなげる



より大規模・複雑な
システムへ適用

エッジ・クラウド協調システムの最適化

1. どんな問題を解決したいの？

クラウドだけだと、遠くて遅くなったり
ネットワークが混雑したりします。

クラウドだけの場合



- ・距離が遠くて遅い
- ・データが集中して混雑
- ・リアルタイム処理が苦手

エッジを使うと？



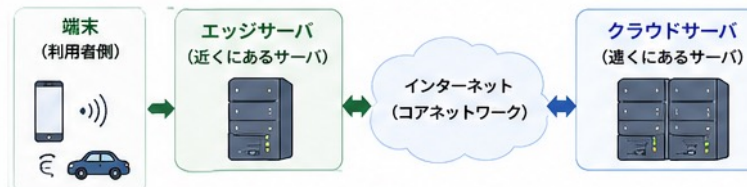
- ・近くで処理できて速い
- ・データ量を減らせる
- ・ネットワークの負荷を軽くできる



でも、いつもエッジを使えばいいとは限りません。
状況によっては、クラウドに直接送った方が
早いこともあります。

2. 想定している仕組み（エッジ・クラウド環境）

端末から出たデータを、エッジサーバかクラウドサーバで処理します。



データの流れ（2つの選択肢）

① エッジを使う場合（2段階）



エッジで前処理してからクラウドへ送るので、
データ量を減らせて速くなることがある！

② クラウドへ直接送る場合（1段階）



エッジをはさむと、逆に遅くなることもある...

影響する主な要素

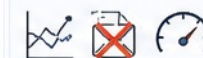


3. 私たちの提案する方法

観測した情報をもとに、エッジを使うかどうかを自動で判断します。

① 今の状況を観測

- ・通信の遅さ（RTT）
- ・パケットロス率
- ・データ量
- ・サーバの負荷や性能
などを集める



② 機械学習モデルで判断

過去のデータを学習した
モデルが、どちらが早くて
効率的かを予測！



エッジを使う



クラウドへ直接送る



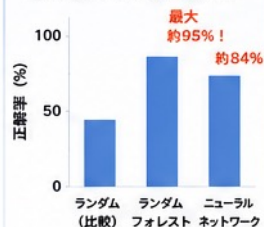
③ 判断に従って処理を実行



4. 実験・評価の結果

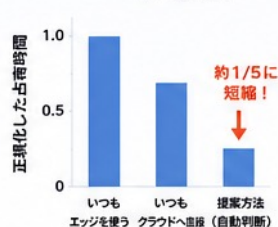
判断の正しさ（例）

エッジを使うべきと判断できた割合



無線ネットワークの占有時間

短いほど良い！



いろいろな条件でも安定して効果あり！



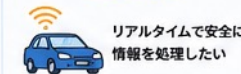
提案方法は、ほぼ最適な判断ができて、無線ネットワークを効率よく使えます！

5. この研究のポイント

- エッジとクラウドを「状況に応じて協調的に使い分ける」ことで、全体の性能を最大化できる。
- ネットワークやサーバの状態をリアルタイムで観測して判断に活かす。
- 機械学習を使うことで、複雑な条件の中でも素早く判断できる。
- 無線ネットワーク資源の占有時間を大きく減らせるので、より多くの端末が快適につながる。

6. どんな場面で役立つの？

自動運転



リアルタイムで安全に
情報を処理したい

VR / AR



遅れがあると気持ち悪く
なるので低遅延が重要

工場のIoT



たくさんのセンサーを
つないで効率よく監視

防災・監視システム



カメラ映像などを
すばやく分析したい

まとめ・将来の展望



エッジとクラウドを上手に
協調させることで、
通信も処理も効率的に！



5G / Beyond 5G の進化に合わせて、
もっと多くの場面で使える
技術を目指す！



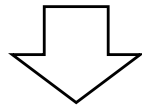
さらに賢い判断や、
複数エッジの協調などの
研究を進める！



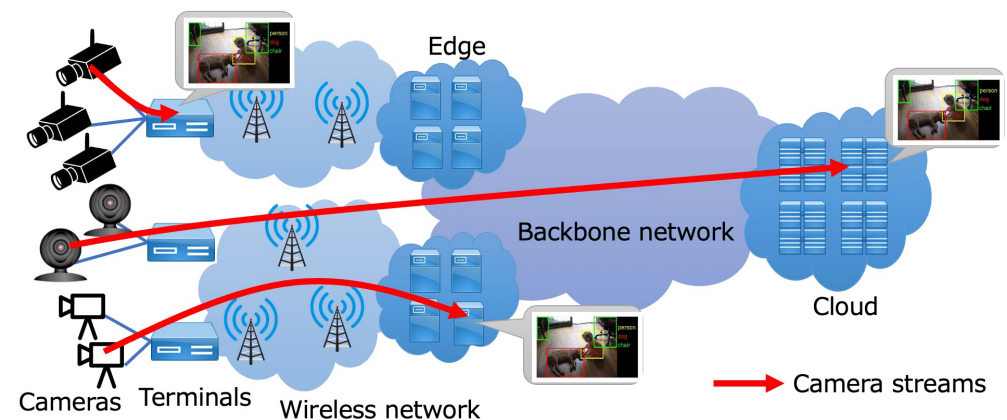
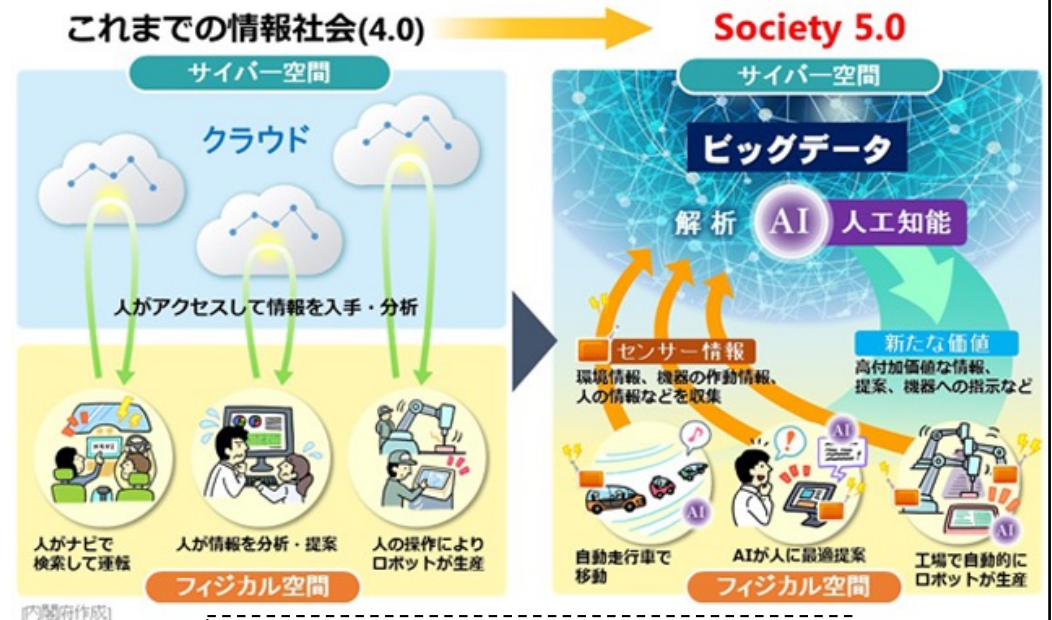
誰もが安心・快適に使える
つながる社会の実現に
貢献します！

デジタルツイン構築のエネルギー最適化

- **デジタルツイン: 現実世界であるフィジカルシステムから様々な情報を収集し、サイバーシステムを構築**
 - データ量が膨大となり、データ流通のためのネットワーク、データ処理のためのコンピューティング負荷が増大 → ICTインフラのエネルギー消費の増大
- **デジタルツイン構築のためのICTインフラ**
 - クラウド+エッジ、端末+ネットワーク
 - 最適化性能と計算コストのトレードオフ
 - システム状態の変動に応じた動的な最適化

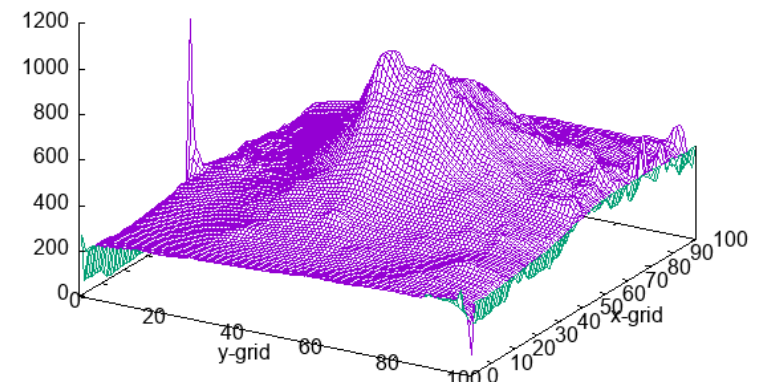
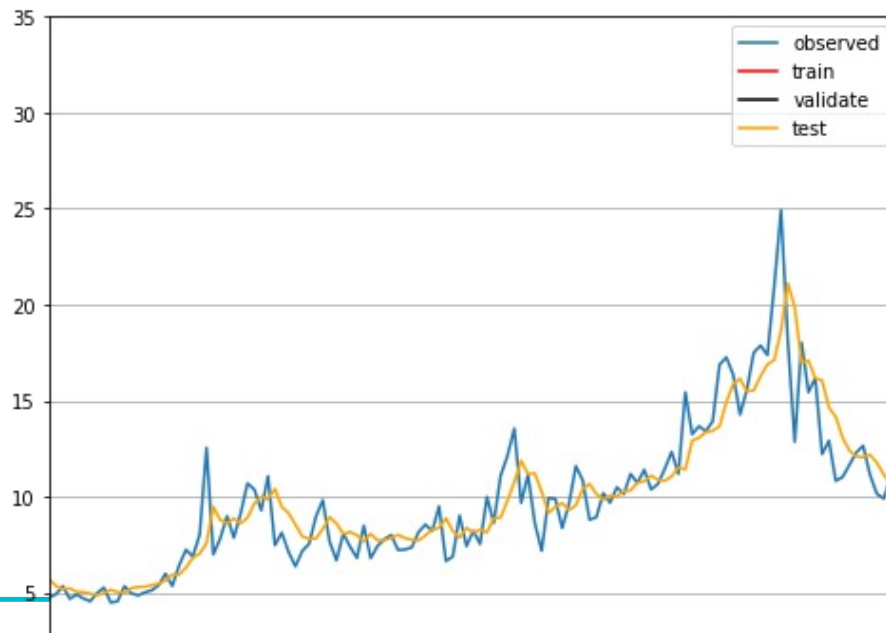
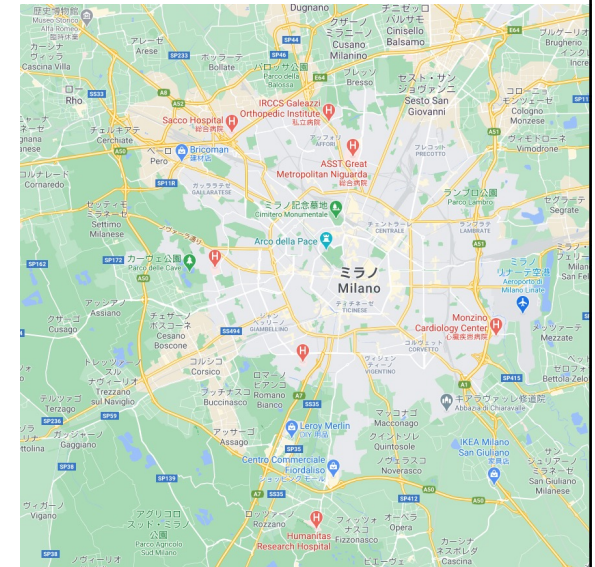


- **目的: サイバーシステムの構築のためのセンサデータ収集・処理の消費電力最適化**
 - エッジクラウド環境を想定
 - カメラからの映像データを処理する場所と処理品質を決定
 - ネットワークの消費電力とデータ処理の消費電力の総和を低減
- **手段: 高速かつ(準)最適な、環境変動に応じたシステム再構成手法**
 - ILPソルバと発見的手法の組み合わせにより、解の性能と環境変動への追随性を両立



モバイルトラフィック予測に基づく仮想ネットワーク制御

- 過去のデータを使って将来のトラフィック量を予測し、ネットワークに必要な資源 (帯域、サーバ台数等) を推定
- 推定結果を利用して先回りでネットワーク、サーバ容量を調整する
 - 多種多様なアプリケーションを、省電力、低コストでネットワークに収容



研究テーマ例

1. Beyond 5G ネットワークスライシング

利用者ごとに最適なネットワークを提供する仕組みを研究

- ・ サービスではなくユーザ単位でネットワークを構成
- ・ 利用状況に応じてリアルタイムに再構成

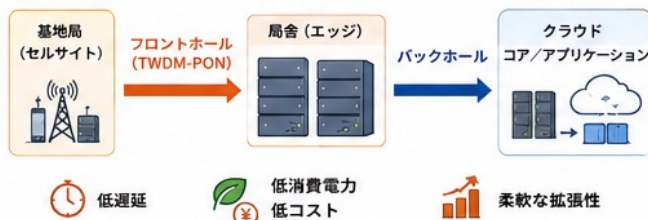


(参考論文: UONA: ユーザ指向ネットワークスライシングアーキテクチャ)

3. ローカル5G・無線ネットワーク最適化

ローカル5Gを低コストで展開するための最適配置を研究

- ・ ベースバンド処理やコア機能を最適な場所に配置
- ・ 消費電力の削減と遅延の最小化を両立



(参考論文: TWDM-PON フロントホールにおける機能配置最適化)

未来の情報通信ネットワーク

高速・大容量 低遅延 高信頼・耐災害 省エネルギー

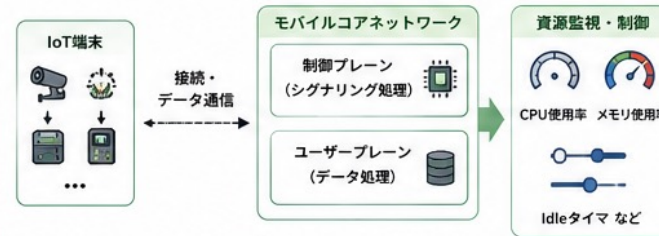


多様なサービスや端末を支えるネットワークを、賢く設計・運用するための基盤技術を研究しています。

2. モバイルネットワーク資源管理

IoT端末の増加に対応するための資源管理を研究

- ・ CPUやメモリの使用状況に応じた効率的な制御
- ・ シグナリングパラメータの調整で資源利用を最適化

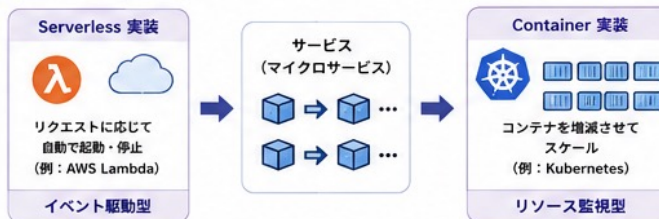


(参考論文: 修士論文 / IoT通信におけるモバイルコアノード資源管理)

4. マイクロサービス実装評価

サーバレス実装とコンテナ実装の性能を比較・評価

- ・ スケーリング性能や応答時間を解析
- ・ 負荷特性に応じた最適な実装方式を提案

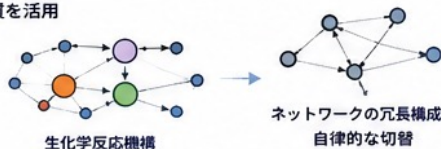
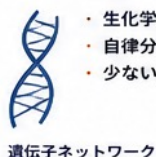


(参考論文: マイクロサービスのサーバレス/コンテナ実装評価)

5. 生物に学ぶネットワーク設計

生物のメカニズムをヒントにした冗長構成・切替方式を研究

- ・ 生化学反応や遺伝子ネットワークの性質を活用
- ・ 自律分散的に障害を回避・回復
- ・ 少ない冗長で高い信頼性を実現

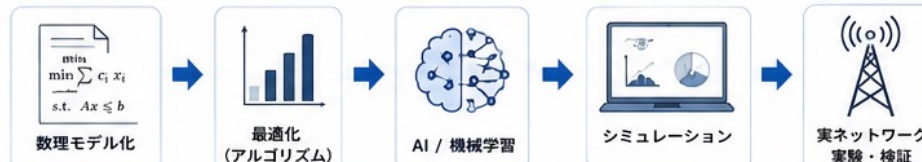


ネットワークの冗長構成・自律的な切替

(参考論文: 生化学・生物機構に着目した冗長構成手法)

6. 共通する研究アプローチ

数理モデル・最適化・AI・シミュレーション・実ネットワーク実験を組み合わせる研究



(共通の研究基盤・手法)



将来のネットワークを支えるために、多様な技術を組み合わせて「最適設計」と「賢い運用」を実現します！



多様なサービス・端末に対応



高い信頼性・耐災害性



低遅延・大容量通信



低コスト・省エネルギー