博士論文

低遅延な遠隔モーション制御を実現する
アクセスエッジ技術に関する研究

北海道大学大学院情報科学院

小屋迫　優士

目次

第 1 章　　序論

1-1　研究背景

1-1-1　　IoT 端末のネットワーク越しの制御の現状

近年，IoT (Internet of Things) 領域に注目が集まっている．図 1-1 に示すように，2023 年には世界で 378 億台であった端末数は，2027 年には約 1.5 倍の 573 億台になると予想されている [1]．特に産業用分野の IoT 化が着実に進行しており，2023 年には 106 億台であったが，2027 年には 187 億台近くになるといわれている．これまで IoT 分野ではセンサ等を IoT 化することで大量のデータを収集し，集めたビッグデータを AI (Artificial Intelligence) 等を用いて解析することでデータに意味づけを行い，生産性の改善につなげるなどの取組が主であった．しかし，今後は得られた分析結果を IoT 端末自体にフィードバックし，制御する要求が増加していくことが予想される．このような制御を行う IoT 端末の例として，産業用ロボットやコネクテッドカー，屋外/屋内ドローン等があるが，これらのロボティクスの市場規模は図 1-2 に示すように特にサービスロボットを中心に年々拡大しており，2028 年には 655.9 億ドルに達することが予想される．また，それを踏まえた 2030 年代の Beyond 5G 時代のロードマップとして，低遅延・高信頼な IoT 基盤の確立が求められている [2].

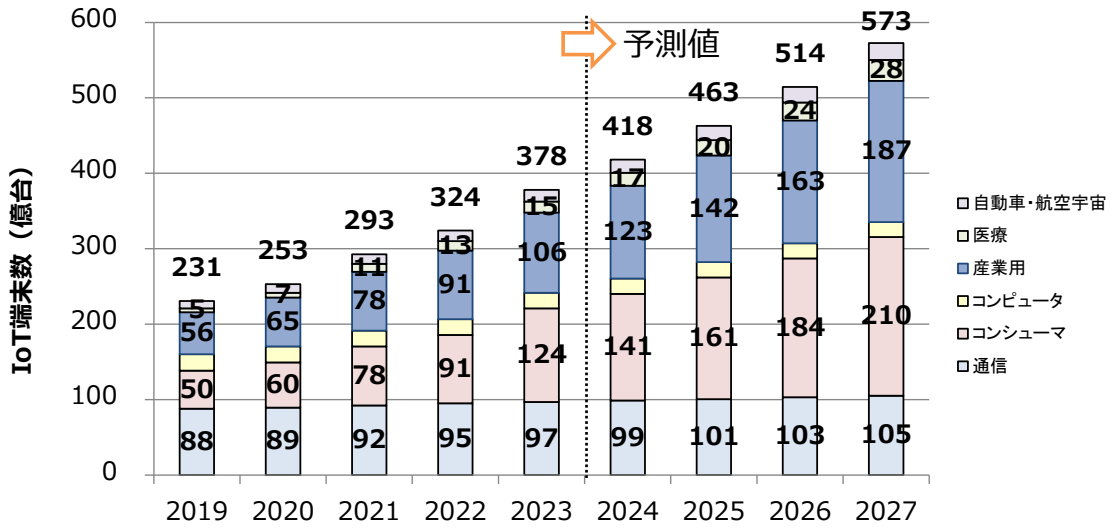


図 1-1　IoT 端末数の推移 [1]

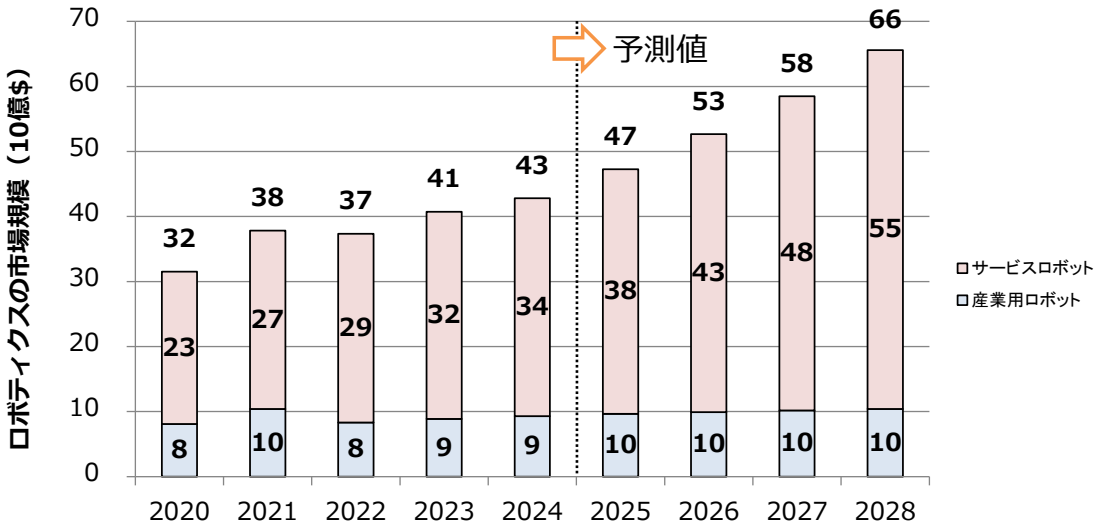


図 1-2　ロボティクスの市場規模 [1]

ドローンや産業用ロボットといった端末の位置や角度といった動きを制御することをモーション制御とよぶ．高速で詳細なモーション制御を実現するためには，センサや映像データ伝送のための高速広帯域のデータ伝送だけでなく，低遅延かつ高信頼なリアルタイム通信が要求される．そのため，これまで一般に端末上のプロセッサで行われてきた．しかし，プロセッサが各デバイス上に必要となるため，端末の増加に伴い，管理が難しくなり，またコストが大きくなる傾向にある．LAN (Local Area Network) 内サーバと言ったローカルの計算リソースからの制御も同様に，遅延や信頼性を保てる一方で，ユーザが LAN や制御システムの管理をする必要がある．

近年これらの端末をネットワーク越しに制御する需要が高まっている．特に，AI やビッグデータといった計算リソースを大量に必要とする処理を実現するために，大規模リソースをもつクラウドサービスを利用して遠隔制御するユースケースが広がっている [3], [4]. ネットワーク越しの制御は複数の工場などの拠点や端末を集約することにより管理を効率化できたり，システムの設置や入れ替え，拡大や管理を簡易に実現できたりする利点がある．また，豊富な計算リソースを活かし，リアルタイムなデータ収集・分析や AI などと連携した高度な監視・制御が実現できる．このようなネットワーク越しの制御はリソースのメンテナンスをサービス事業者に任せることができ，ユーザの負担を減らすことができる一方で，制御対象と制御器間の距離が大きくなることにより，遅延が発生し，結果として低遅延・高信頼性が要求されるモーション制御に適さなくなる可能性がある．図 1-3 に遅延の例を示す [5]-[7]．ファクトリーオートメーションの遅延要求は 0.25-10 ms であり，触覚通信の遅延要求は 1 ms であるが，クラウドからの制御遅延は 100 ms を超える [8]．ネットワーク遅延は制御性能を低下させ，結果として整定時間が長くなったり，制御系が不安定になったりする可能性がある．そのため，現状の遠隔制御では多くの場合，ネットワーク越しに端末近傍のローカル計算リソースに制御命令を送信し，ローカルで制御値を作

成して，端末のリアルタイム制御を実現する手法が取られている．また，特に工場において，クラウドとローカルリソースを連携させる手法が提案されている [9], [10]．この手法を用いることで，産業用機器のコントローラはローカルで端末を低遅延に制御することが可能であり，異常検知や故障予測といった高負荷処理を必要とするタスクはクラウド上で計算処理を行うといった連携が可能になる．ただし，ローカルリソースはユーザが管理する必要がある．

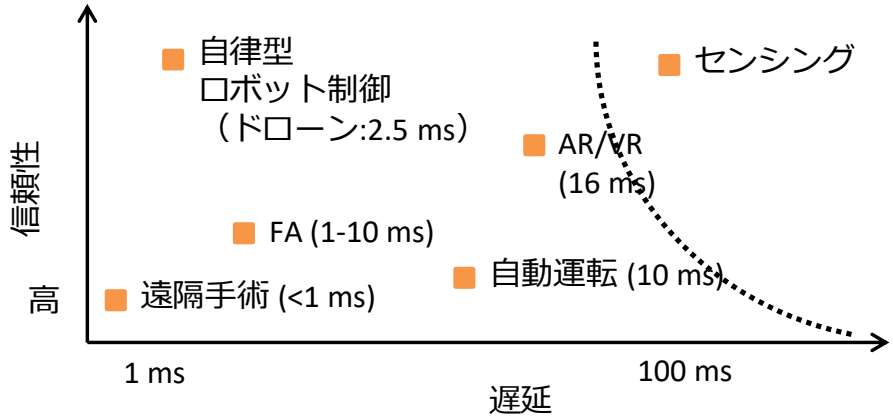


図 1-3　アプリケーションごとの要求遅延と信頼性

1-1-2　　ネットワークの概要

通信ネットワークは，ユーザと通信事業者ビルを結ぶアクセスネットワーク，通信事業者ビル間を結ぶコアネットワークに大別できる．図 1-4 にネットワークの概要を示す．アクセスネットワークは，一般家庭や企業等から通信を行う際に，利用者が通信事業者内のコアネットワークと接続するためのネットワークである．アクセスネットワークはユーザと近距離にあるネットワークであるため，発生する遅延は，コアネットワークが 100 ms 以上であるのに対し，数ミリ秒オーダである特徴を持つ [11]．クラウドはコアネットワークを介して提供されるため，遅延が大きい．

光ファイバが適用された光アクセスネットワークには，主に Point-to-Point 型と Point-to-Multipoint 型が用いられ，それぞれ異なる通信方式で実現されている．Point-to-Point 型は通信局舎の通信装置とユーザの通信装置を直接光ファイバで接続したもので，帯域を占有することから安定した通信品質が得られるため，主にビジネスユーザ向けに展開している．リアルタイム制御の想定されるユースケースとしては，ファクトリーオートメーションや屋内ドローン，遠隔手術等が挙げられる．Point-to-Multipoint 型は通信局舎の通信装置と多数のユーザの通信装置を光ファイバで接続したもので，効率的に帯域が利用できるため，

主にマスユーザ向けに展開している．想定されるユースケースとしては拡張・仮想現実 (Augmented Reality, AR / Virtual Reality, VR)，スマートハウス / シティ等が挙げられる．

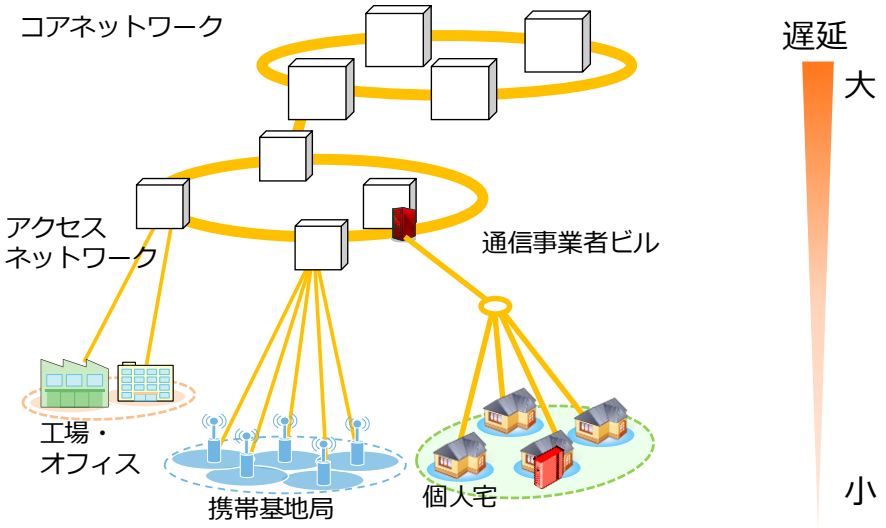


図 1-4　ネットワークの概要

1-1-3　　エッジコンピューティング技術

IoT 端末のネットワーク越しの制御を支援する技術として，エッジコンピューティングが注目を集めている [12], [13]．エッジコンピューティングは IoT 端末近傍のネットワークエッジに計算リソースを設置し，処理を行うもので，もともとモバイルデバイスの機能を強化するものとして提案された．エッジコンピューティングはクラウドに比べて短い遅延時間と十分な計算リソースを提供することができ，車間通信やビデオストリーミング，AR / VR や触覚通信といったアプリケーションを実現できる．欧州電気通信標準化機構 (European Telecommunications Standards Institute, ETSI) では Multi-access Edge コンセプト (MEC) が提案されている．MEC システムでは，モバイル通信事業者が無線アクセスネットワーク内の計算リソースを提供する．

1-2　本研究の目的および課題

1-2-1　　目標とするコンセプトとその効果

本論文では，光アクセスネットワーク上でのエッジコンピューティングを用いてリアルタイムにモーション制御を実現するアクセスエッジコンセプトの実現を目標とする．光アクセスネットワークは，ユーザの近傍にあり，既にインフラとして整っている低遅延・高信頼な基盤であるため，遅延要求が厳しくクラウドからでは難しいアプリケーションを実現でき，かつアクセスネットワーク上に計算リソースを配置することで，AI やビッグデータと連携した，より高度なサービスが実現できる可能性がある．

提案コンセプトではエッジサーバをアクセスネットワーク上の通信局舎に設置し，ユーザの IoT 端末の制御値をリアルタイムに計算する．提案コンセプトと従来技術との比較を図 1-5 に示す．また，図 1-6 に従来のネットワーク越しの制御とアクセスエッジによる制御の比較を示す．従来は外部からローカルリソースに制御命令を送信，ローカルのリソース上の制御アプリケーションが制御値を計算し，デバイスのモーション制御を行うものであったが，アクセスエッジからの制御では，計算リソースを通信局舎に集約し，通信局舎を低遅延制御が可能なデータセンタとして扱うことにより，ユーザは IoT 端末とアプリのみ用意するだけで高度な制御が実現できる．このとき，IoT 端末は高性能なプロセッサを必要とせず，通信機能のみを必要とする．制御アプリケーションを含めた計算リソースの集約は今後想定される大量の IoT 端末やセンサの接続を考慮すると，コストや拡張性，機能強化の面で利点をもたらす．また，アクセスエッジによる制御により現地での作業を簡易化することができる．このシステムは操作のために映像処理が必要なドローンや画像処理が必要な産業用機器といった計算リソースを必要とするアプリケーションに，より有用である．計算リソースはネットワーク上に集約されるため，必要に応じてスケールが可能であり，加えて，豊富な計算リソースを活用した AI やビッグデータといった高度なサービスと連携することも容易である．ユーザはシステムの管理を通信事業者に任せることができ，管理コストの低減が期待できる．

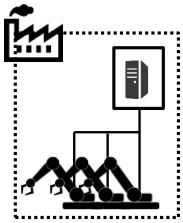
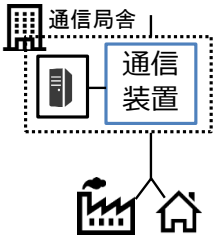
	デバイス	LAN内サーバ	**(提案)アクセスエッジ**	クラウド
構成				
遅延/信頼性	◎ (<1 ms)	◎ (<1 ms)	○ **(1-10 ms)**	× **(>100 ms)**
価格/リソース集約	×	△	○	◎
管理	△	△	○	◎

図 1-5　アクセスエッジと従来技術の構成比較

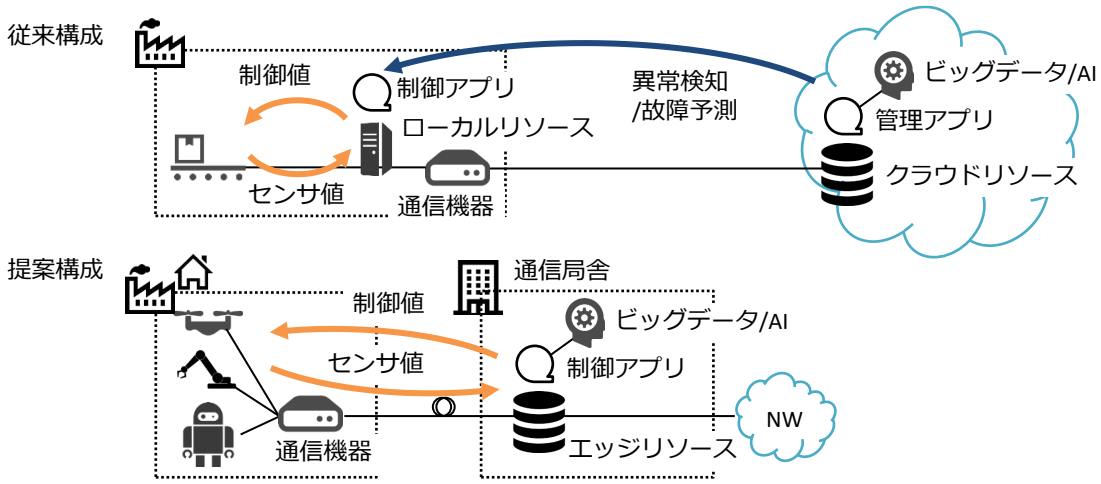


図 1-6　従来のネットワーク越しの制御とアクセスエッジによる制御の比較

1-2-2　　アクセスエッジにおける課題

アクセスエッジコンセプトで課題となるのは，アクセス区間で生じる遅延やジッタがクラウドに比べて十分小さいながら，制御性能を劣化させることである．アクセスシステムの遅延は伝搬遅延，伝送遅延，処理遅延，キューイング遅延に分類される [14]．図 1-7 に分類を示す．

伝搬遅延は物理媒体上の信号伝搬に依存する遅延である．1 km あたり約 5 us であり，最大伝送距離が 20 km の典型的な光アクセスシステムの場合，約 0.1 ms となる．遅延の大きさは固定値であり，一般に低減が難しい．

伝送遅延はフレームの全ビットを物理回線に乗せるまでにかかる時間のことで，フレームサイズやネットワークの帯域幅に依存する．1 Gbps の光アクセスを想定する場合，イーサフレームの最大サイズが 1518 byte のため，0.012 ms とほぼ無視できる値である．

処理遅延はアプリケーションにおいて，パケットを入力インタフェースから出力インタフェースに移行するまでにかかる時間である．遅延は単純なアプリケーションではマイクロ秒オーダであるが，複雑になるにつれ増大する．アプリケーションのアルゴリズムによりノード数，プロトコル，スケジューリング等によって遅延が時変の値をとる．IoT 端末の増加に伴い，遅延全体に与える影響の増加が予想される．

キューイング遅延はルータやスイッチ等でバッファに保持している時間であり，ネットワークのトラフィック量や混雑度に依存する時変の値である．また，バッファ溢れによる再送もあり，遅延への影響が大きい．

伝搬遅延と伝送遅延は固定値でかつ十分小さい値のため，アクセスネットワークでは無視できる．一方，処理遅延とキューイング遅延はデータ量やノード数等のネットワーク状態に依存して増大する時変の値で，また遅延要素の多くを占める．端末制御において，ジッタは遅延自体の大きさと同じくらい制御性能に悪影響を及ぼす．多くの IoT 端末がネットワークに接続することが想定されるため，対象を処理遅延とキューイング遅延とし，これらの遅延の影響を低減する．

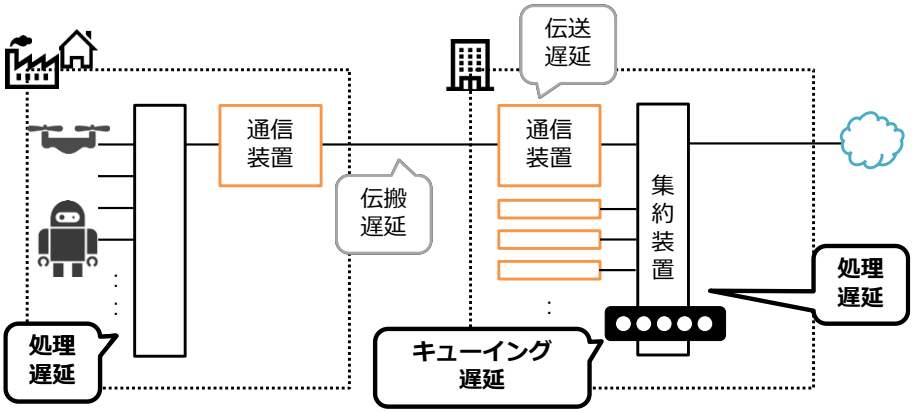


図 1-7　アクセスシステムでの遅延の分類

1-2-3　　提案手法の概要

アクセスシステムの遅延自体の絶対量を削減することは，アクセスシステムがすでに低遅延なネットワークであることや，また現状の通信方式は遅延とネットワーク効率のバランスを取っていることから，限界がある．一方でこれまで，通信技術とモーション制御技術はそれぞれ別々に研究がなされており，モーション制御を実現するためのネットワーク，ないしはネットワークの状態を考慮したモーション制御といった分野横断的な研究はあまり行われていない．

本論文では，以下の 2 つのアプローチにより，アクセスネットワークとモーション制御を連携させ，遅延の影響低減を実現する．

- アクセス区間で生じる遅延値等のネットワークの状態を正確に把握し，ネットワーク側から制御アプリケーションに提供することで，ネットワークを考慮した制御を行う．
- アクセス区間で生じる遅延値等を制御性能が向上する値になるように，ネットワーク情報の収集・分析を行ってネットワーク設定値を変更し，モーション制御に適切なネットワークを構築する．

これら 2 つのアプローチを実現するため以下の 4 つの手法を提案した．図 1-8 に提案手法の分類を示す．

1. アクセスシステムの Point-to-Point 構成を対象に，経路上の通信機器から直接処理時間を収集することで，経路全体の遅延値を算出し，遅延補償制御を行う手法
2. アクセスシステムの Point-to-Multipoint 構成を対象に，この構成特有のキューイング遅延である DBA (Dynamic Bandwidth Allocation) による遅延をモデル化して遅延値を予測し，遅延補償制御を行う手法
3. アクセスシステムの仮想化基盤を用いて，モーション制御アプリケーションから受け取った制御結果をもとに，通信機器の設定パラメータを動的に更新し，制御に特化したアクセスネットワークを構築する手法
4. 次世代の光アクセスシステムであるオールフォトニクスネットワークを対象に，経路全体の遅延を高速に算出し，ジッタバッファリングの設定値を動的に更新することで，遅延が固定値のジッタのないアクセスネットワークを構築する手法

従来の構成ではユーザが管理するモーション制御アプリケーションと通信事業者が管理するネットワークが分離し，連携していない．これら 4 つの手法では，ネットワークとモーション制御アプリケーションが連携可能なアーキテクチャを整備することで，リアルタイムな情報交換によるモーション制御の実現を可能にした．このアーキテクチャにより様々なアプリケーションに同様に活用できる．

最後に，モーション制御アプリケーションが様々なプロトコルに対応する必要があることから，実アプリケーションとして工場を想定し，産業用ネットワークにアクセスエッジ

を適用することで，光アクセスネットワーク上でのエッジコンピューティングによる End-to-End でのリアルタイムモーション制御を実現するアクセスエッジコンセプトの実証を行った．

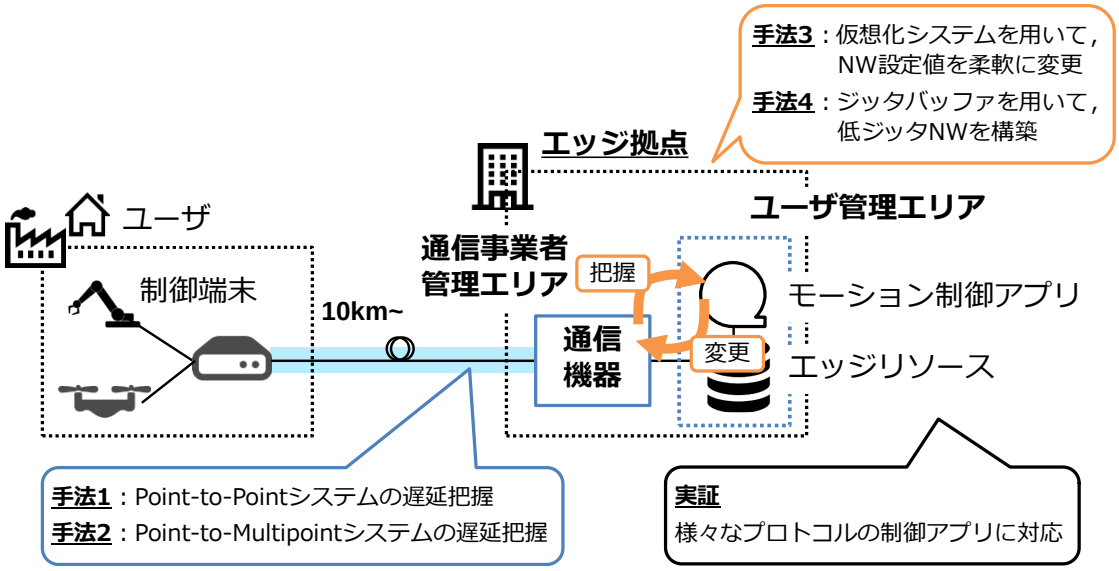


図 1-8　提案手法の分類

1-3　本論文の概要

本研究は全 7 章からなる．

第 1 章では，本研究の背景として我が国における IoT 領域の状況と，モーション制御の概要，また低遅延なネットワークであるアクセスネットワークと低遅延処理を可能にするエッジコンピューティング技術の概要について述べた．また，本研究の目的であるアクセスエッジコンセプトの概要と効果，それを実現するための 4 つの手法について説明した．

第 2 章では，アクセスエッジコンセプトを実現するために重要な要素技術である光アクセスネットワーク技術と遅延要因について述べる．続いて，ネットワークを介した IoT 端末のフィードバック制御の詳細と遅延補償制御手法について述べる．また，ネットワークの柔軟化に貢献する SDN (Software Defined Network) 技術とそのアクセスネットワークへの適用について説明する．最後に関連技術として，より遅延要件の厳しい産業用ネットワークと実現するプロトコル技術について述べる．

第 3 章では，光アクセスネットワークの Point-to-Point 構成を対象に，経路上の通信機器がすべて特定できることに着目して，通信機器それぞれから処理時間を収集，合算することで，ネットワーク全体の経路遅延を算出し，遅延補償制御に用いる手法を提案する．モータとドローンを用いた実機実験系により有効性を検証する．

第 4 章では，Point-to-Multipoint 構成を対象に，時変遅延の原因となる DBA に対して，OLT (Optical Line Terminal) から現在の設定情報を読み取り，新たに作成した遅延モデルによって，遅延を算出，遅延補償制御を行うことで，帯域使用効率を維持しながら高度な制御が実現できる手法を提案する．モータを用いて実機実験系を構築し，整定時間を評価指標とした従来手法との比較により有効性を検証する．

第 5 章では，光アクセスネットワークの仮想化基盤である SEBA (Software-Defined Network Enabled Broadband Access) を用いて，モーション制御結果をネットワーク設定にフィードバックし動的に変更することで，ユーザに設定を意識させることなく，制御に最適なネットワークを構築する手法を提案する．モータを用いた実機実験系を構築し，制御結果のフィードバックにより DBA 周期を変更，所定の整定時間内に制御が完了するか検証する．

第 6 章では，次世代のネットワーク基盤であるオールフォトニクスネットワークを対象に，通信機器間の区間遅延情報をリアルタイムに収集し，最適なジッタバッファリングを行うことで，所望の固定値で遅延補償制御を行う手法を提案する．FPGA (Field Programmable Gate Array) を用いた 100 Gbps 検証システムを構築し，制御に関連したデータのみミリ秒オーダでジッタバッファを行い，遅延が時変の場合でもモータ制御できることを確認する．

第 7 章では，これまで提案したアクセスエッジコンセプトをより要求の厳しい産業用ネットワーク上で実現し，End-to-End でのコンセプト実証を行うために，既存の産業用プロ

トコル機能をソフトウェアで実装し，汎用の計算リソースより制御する手法を提案する．モータを用いた実機実験により有効性を検証する．

以上のように，本研究では光アクセスネットワークとモーション制御の連携により，遅延の制御に及ぼす影響を低減させる手法を提案し，IoT 端末を対象にした制御性能の検証を行う．アクセスエッジコンセプトにより，低遅延・低ジッタなネットワークを構築し，より高度な制御が実現できることを示す．

1-4　参考文献

[1] 総務省，“令和 6 年版 情報通信白書”，2024

[2] 総務省，“令和 5 年版 情報通信白書”，2023

[3] L. Zhang, H. Gao and O. Kaynak, “Network-Induced Constraints in Networked Control Systems—A Survey,” IEEE Transactions on Industrial Informatics, vol. 9, no. 1, pp. 403–416, 2013.

[4] X. Luan, P. Shi and F. Liu, “Stabilization of Networked Control Systems With Random Delays,” IEEE Transactions on Industrial Electronics, vol. 58, no. 9, pp. 4323–4330, 2011.

[5] ITU-T, “The Tactile Internet,” 2014

[6] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat and H. Dai, “A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions,” IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3098–3130, 2018.

[7] S. Weibin, L. Yun, D. Yi, D. Yingguo, P. Mingbo and X. Gang, “Three-Real-Time Architecture of Industrial Automation Based on Edge Computing,” 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), pp. 372-377, 2019.

[8] M. Ahmadi, N. Khanezaei, S. Manavi, F. Fatemi Moghaddam and T. Khodadadi, “A Comparative Study of Time Management and Energy Consumption in Mobile Cloud Computing,” 2014 IEEE 5th Control and System Graduate Research Colloquium, pp. 199-203, 2014.

[9] “AWS IoT Greengrass - Amazon Web Services,” https://aws.amazon.com/jp/greengrass/

[10] “IoT Edge | Microsoft Azure,” https://azure.microsoft.com/en-us/services/iot-edge/

[11] N. Cvijetic, “Optical network evolution for 5G mobile applications and SDN-based control,” 16th International Telecommunications Network Strategy and Planning Symposium (Networks), pp. 1-5, 2014.

[12] M. T. Beck, M. Werner, S. Feld, and T. Schimper, “Mobile edge computing: A taxonomy,” Proc. 6th Int. Conf. Adv. Future Internet, 2014.

[13] H. Tanaka, M. Yoshida, K. Mori, and N. Takahashi, “Multi-access edge computing: A survey,” J. Inf. Process., vol. 26, pp. 87–97, 2018.

[14] I. Grigorik and Y. Wada, “High Performance Browser Networking,” O'Reilly, 2014.

第 2 章　　関連研究と技術課題

2-1　はじめに

本章では対象とするアクセスエッジコンピューティングに関する技術である，光アクセスシステム，モーション制御手法，アクセスシステムの関連技術とその技術課題について述べる．

2-2　想定する広域光アクセスシステム

2-2-1　　アクセスネットワークの概要

コアネットワークが非常に多数のユーザ情報を少ない設備で効率的に伝送することで，ユーザあたりの設備コストを大幅に低減できる一方で，アクセスネットワークでは 1 ユーザ，ないしは少数のユーザが各システムを使用するため，システムコスト低減が最優先される．光ファイバが適用された光アクセスネットワークでは，通信設備のビル側装置を OLT (Optical Line Terminal)，ユーザ側装置を ONU (Optical Network Unit) と呼ぶ．光アクセスネットワークの網形態は OLT と ONU が 1 対 1 対応した Point-to-Point 型と，通信設備ビル装置とユーザ装置が 1 対 N 対応した Point-to-Multipoint 型の 2 つに大きく分類される [1], [2]．Point-to-Point 型は光ファイバ，局側装置を占有する方式で，伝送特性上の制約が少なく，光ファイバケーブルの広帯域性を活かした高速・広帯域なサービスへの対応が可能で，安定した通信品質が得られる一方，回線単位に通信設備ビル装置の光モジュール及び光ファイバが必要となることや，一般にアクティブ装置は高価であることから高コストとなってしまう特徴を持つ．ビジネスユーザ向け通信サービスに用いられることが多い．Point-to-Multipoint 型は OLT の 1 つの光モジュールを複数 ONU で共用するため，システムの低コスト化が可能であり，また統計多重効果による効率的な帯域利用ができる一方で，多重度合いによりユーザ一人当たりの帯域が制限され，混雑時に速度が低下する特徴を持つ．マスユーザ向け通信サービスに用いられることが多い．アクセス区間の途中に低コストな受動素子であるスプリッタを用いたパッシブ型の光ネットワークは PON (Passive Optical Network) ともよばれる．パッシブ型はコストの観点だけでなく，環境整備や電源も不要となることから，経済的かつ信頼性の高い設備構築が可能となる．

2-2-2　　Passive Optical Network

PON は，光-電気変換を行わず，スプリッタを用いて光信号を複数に分岐し，一心の光ファイバを複数のユーザで共有することで，経済的なネットワークを実現する．PON においては， OLT から ONU に向けた信号はすべての ONU へ同報型で送られ，ONU は自分宛

の信号のみを取り出し，他の信号は廃棄する．一方，ONU から OLT 向けの信号は，他の ONU からの信号と時間軸上でデータが重ならないように，OLT が ONU にデータ送信のタイミングやデータ量を指示する．そのため， OLT と ONU は同じ制御手順，速度で動作する必要がある．また，PON は共有型の方式であるため，固定的なタイミング制御では共有している ONU の数が多くなるにしたがって，ひとつの ONU に割り当てられる帯域・速度が低下する．PON システムには，データ衝突を防ぐためのアクセス制御プロトコルが必要であり，現在，TDMA (Time Division Multiple Access) プロトコルに基づく動的帯域割当 (Dynamic Bandwidth Allocation, DBA) アルゴリズムが，サービスリクエストとサービス品質 (Quality of Service, QoS) の要件に応じ，送信タイミング・データ量を動的に分配するために使用されている．図 2-1 に固定帯域割当と動的帯域割当の帯域利用効率の概念図を示す.

図 2-2 に DBA のメカニズムを示す [3]．ONU から OLT への信号を上りデータ，OLT から ONU への信号を下りデータという．ONU は IoT 端末からのデータ到着後，OLT に Request メッセージを送信する．OLT の DBA 機能が各々の ONU から届いた Request メッセージをもとに，アクセス権を計算し，ONU ごとに Grant メッセージを送信する．この Grant メッセージには ONU ごとの割当帯域と各 ONU からの上り信号が衝突しないように送信開始時間が含まれている．ONU は届いた Grant メッセージをもとに OLT にデータを送信する．DBA は上りデータを送信する前に，Request メッセージを送信し，Grant メッセージを受け取る必要がある．そのため，帯域利用効率は大きいものの，送信開始時の遅延は大きくなる傾向にある．

DBA は帯域利用効率，遅延時間，ONU 間の公平性などの指標でアルゴリズムが使い分けられる．ONU にデータが届いてから OLT にデータが届くまでの遅延時間に影響を与えるパラメータとしては，DBA 周期と OLT-ONU 間の距離，Grant メッセージの送信頻度がある．特に DBA 周期が支配的で，この値によって大きく変動する．なお，この場合，ONU からみた Grant 周期と DBA 周期は同一となる．

DBA 遅延 d_{DBA} は以下の式で与えられる．

$$d_{DBA} = d_{report} + d_{grant} + d_{start} \tag{2-1}$$

このとき， d_{report} はデータが ONU に到着してから Request メッセージを送信するまでの時間， d_{grant} は Grant メッセージを受信するまでの時間， d_{start} はデータを OLT に送信するまでの時間である．Request メッセージの送信権は各帯域割当周期において，すべての接続されている ONU に付与される．

DBA は標準で規定されている MPCP (Multi-Point Control Protocol) を利用しているが，割り当てられる送信許可量の計算方法等の詳細については標準で規定されていない．そのため，ONU の送信する送信要求の内容や受信した送信要求に対する割り当て方法，送信許可頻度，目標性能（帯域利用効率，遅延時間）は自由に設計可能である．

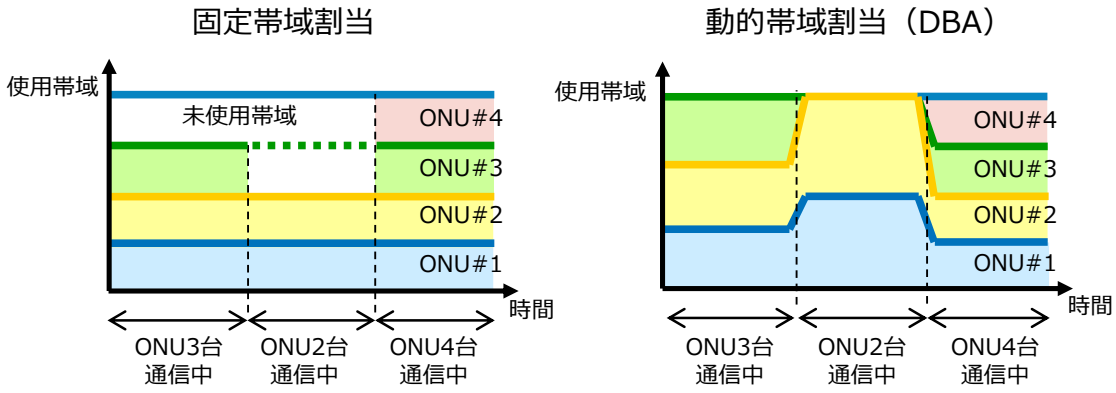


図 2-1　固定帯域割当と動的帯域割当の帯域利用効率

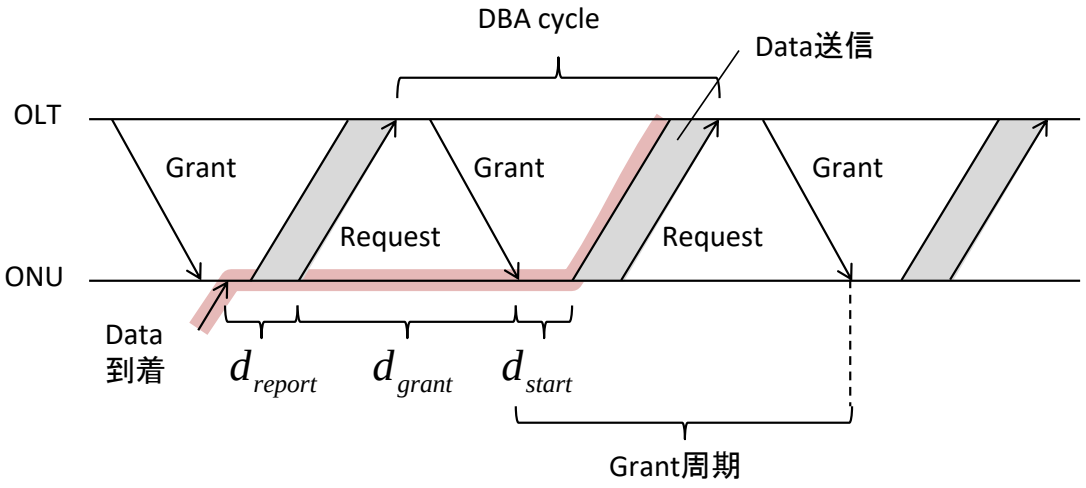


図 2-2　DBA メカニズム

2-2-3　　オールフォトニクスネットワーク

近年，サービスの多様化に伴うネットワークへの広帯域化・低遅延化の要求により，それを実現する新しいネットワーク基盤としてオールフォトニクスネットワーク (All-Photonics Network, APN) が検討されている．APN では，従来のネットワークで帯域制限や遅延の要因となっていたネットワークの階層間における集線や多重といった電気処理をなくし，End-to-End で光接続することで，極低遅延かつ高速大容量な光パスを実現する．APN では，様々なサービスからの通信要求に対して機能別のトポロジを設計し，有限の波長を最適に割り当てる技術や，オール光の通信を有限の光ファイバで収容可能にするための動的な波長パス接続技術が必要になる．このシステムを実現するため，APN アーキテクチャではそれぞれの光パスはネットワークコントローラによって管理される．図 2-3 に APN アーキテクチャを示す．ターミナルは光パスのエンドポイントである．それぞれのアクセスノードはアクセスノードと端末間，またはアクセスノード間の区間遅延情報を取得

し，ネットワークコントローラはすべてのアクセスノードからのネットワーク情報を継続的に収集する．

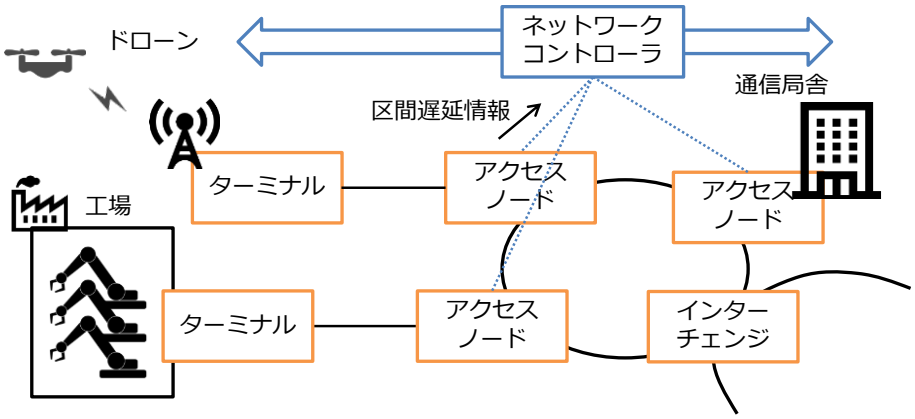


図 2-3　APN アーキテクチャ

2-2-4　　光ネットワーク上でのエッジコンピューティング

光ネットワーク上でエッジコンピューティングを行う研究が報告されているが，これらの研究は主にシステムアーキテクチャやタスクオフローディング，リソース管理に焦点を当てており，リアルタイムモーション制御に光ネットワーク上でのエッジコンピューティングを用いる手法は提案されていない [4].

コアネットワークを対象にモーション制御のための遅延補償を行う研究がいくつか行われている．コアネットワークは多くのスイッチ等の通信機器で構成されており，結果として全体で 100 ms 以上の遅延や，遅延ゆらぎにつながる場合がある [5]-[11]．5G の低遅延要件である URLLC (Ultra-Reliable and Low Latency Communications) を用いている研究もあるが，システムモデルを提案しており，モーション制御手法を提案している文献は見当たらない [12]-[14].

本研究では Point-to-Point 型と Point-to-Multipoint 型のアクセスネットワーク，および APN でのアクセスエッジコンセプト実現を目指す．

2-3　　ネットワークを介した IoT 端末のリアルタイム制御

2-3-1　　ネットワークを介したフィードバック制御の概要

モーション制御とは，モータやロボット，ドローンといった端末を，ある目標状態にするために入力を加えることをいう．入力の決定方法にはフィードフォワード制御とフィードバック制御の 2 種類がある．ここでは特に受信した位置情報などのセンサ情報をもとに，

制御値を作成し，動作させるフィードバック制御を対象とする．図 2-4 に一般的なフィードバック制御を示す．

図 2-4　フィードバック制御

このとき，G_C は制御器，G_P は制御端末を表す．$R(s)$, $U(s)$, $Y(s)$ は，それぞれ目標値 $r(t)$，制御器により作成される制御値 $u(t)$，制御対象の状態を表すセンサ値 $y(t)$ をラプラス変換したものである．フィードバック制御ではセンサ値が目標値と同じ値になるように制御器で制御値を作成し，制御対象に入力する．IoT 端末をネットワーク越しに制御する場合，制御器と制御端末間にネットワークの影響が挿入される．図 2-5 にネットワークが挿入された場合を示す．

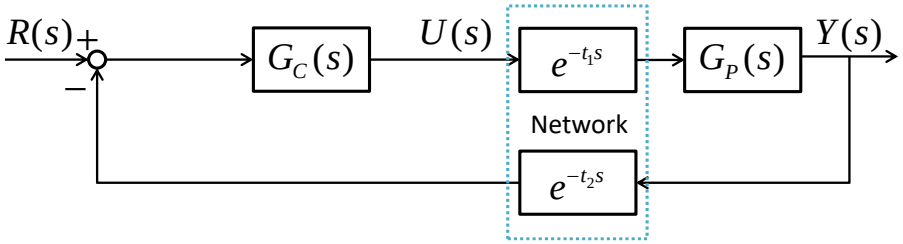


図 2-5　ネットワークを介したフィードバック制御

このとき，t_1，t_2 はそれぞれ，下りと上りの遅延を表す．ネットワーク遅延によって，制御端末に入力される制御値が $U(s)$ から $U(s)\cdot e^{-t_1 s}$ に変化してしまっていることがわかる．このとき，全体の伝達関数は以下の式で与えられる．

$$\frac{Y(s)}{R(s)} = \frac{G_C(s)G_P(s)e^{-t_1 s}}{1+G_C(s)G_P(s)e^{-(t_1+t_2)s}} \tag{2-2}$$

図 2-6 に遅延による制御性能への影響の概念図を示す．遅延の影響により，制御性能が劣化し，目標値に収束するまでの値である整定時間が伸びたり，最悪の場合には制御系が不安定になる場合がある．

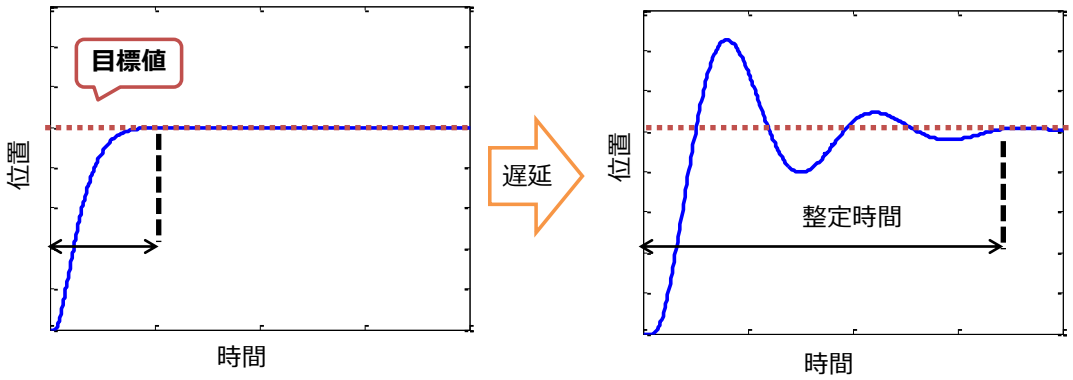


図 2-6　遅延による制御性能への影響

2-3-2　　遅延補償制御手法

遅延の影響による制御性能の劣化に対応するために，様々な手法が提案されている [5], [15]-[20]．スミス予測器はその中でも典型的な遅延補償制御手法である．図 2-7 にスミス予測器を入れた場合のブロック図を示す．

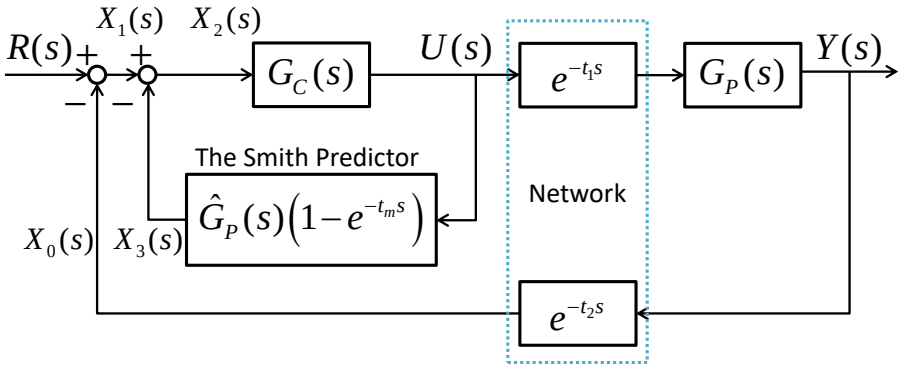


図 2-7　スミス予測器

スミス予測器 $P(s)$ は内部に制御対象を模擬したモデル $\hat{G}_p$ をもち，特定の遅延が生じた場合に実際のモータがどのように動作するかをシミュレートする．この手法では制御対象と制御器間の往復遅延を t_m と予測し，遅延を考慮した制御値を算出する．この遅延の予測値 t_m が実際の往復遅延である t_1+t_2 と等しいとき，図 2-8 のように遅延の影響をフィードバックループ外に切り出し，打ち消すことが可能である．

$$P(s)=\hat{G}_P(s)\left(1-e^{-t_m s}\right) \tag{2-3}$$

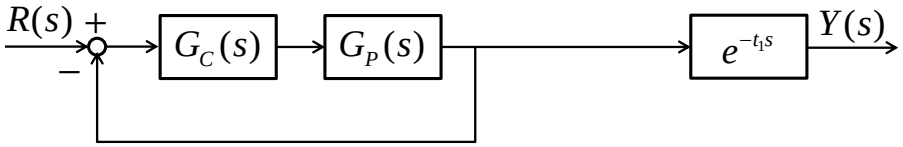


図 2-8　スミス予測器により遅延の影響が除かれたフィードバック制御

2-4　SDN 技術

ここでは，IoT 端末のリアルタイム制御をアクセスネットワークを介して実現する際に有用な技術である，SDN (Software-Defined Network) 技術について詳しく述べる．

2-4-1　SDN 技術

近年，多様なサービスのための増加する要求に迅速に対応するために，SDN 技術の導入が進んでいる．SDN 技術を用いることで，効率的な構成と大規模ネットワークの管理が可能になるため，大きな関心が寄せられている．SDN 技術により，制御プレーンとデータプレーンを分割することで，ソフトウェア定義のパケット処理を可能にし，遅延要件を満たす柔軟なネットワークを構築できる．通信に OpenFlow プロトコルといったインタフェースを用いることで，ネットワークデバイスのプログラマブルな制御を行うことができる．さらに，汎用のホワイトボックススイッチを用いることでシステムが迅速かつ安価に設計できる．さらなる柔軟性のために，データプレーン自体をソフト化する試みが行われている [21]-[23] ．システム全体を高速に開発することで新しいサービスの創出を加速することが目的である．

2-4-2　SEBA アーキテクチャ

これまでベンダから専用装置の形で提供されてきたネットワーク装置を，ハードウェアとソフトウェアに分離することで，ネットワークインフラを高速かつ柔軟にカスタマイズすることが可能である．SDN 技術はブロードバンドアクセスをサポートするために，光アクセスネットワーク機器の機能を追加したり置き換えたりする方法として用いられる [24], [25]．そのうえ，関連する製品やオープンソースソフトウェア (Open Source Software, OSS) も光アクセスネットワーク用に開発されてきた．

エッジコンピューティングは計算リソースの効率的な使用を保ちつつ，低遅延処理を可能にする．アクセスシステムにおいて，OLT を収容しユーザの接続の集約拠点となっている通信局舎は，そのユーザとの距離の近さからエッジコンピューティングに適した場所のひとつである．これまでいくつかの研究で通信局舎をデータセンタに転換するコンセプト

が提案されてきた [26], [27]．CORD (Central Office Re-architected as a Datacenter) は通信局舎をネットワークエッジのための俊敏なデータセンタとして用いることを目的としたプロジェクトである [28]．汎用サーバやホワイトボックス OLT を使って構築され，PON といったアクセスサービスを実現する．CORD プラットフォームはデータセンタのコスト効率とクラウドの俊敏性をキャリアネットワークにもたらす．従来のキャリアネットワークは専用装置で実現されてきたが，複雑で高価であった．CORD は OSS を用いて実現され，OpenFlow プロトコルで管理可能なリソースとして管理されることから，CAPEX (Capital Expenditure)，OPEX (Operating Expense) それぞれの観点から利点がある．SEBA (Software-Defined Network Enabled Broadband Access) は CORD の軽量プラットフォームで，ブロードバンドアクセス装置のためのハードウェア抽象化を提供するオープンソースプロジェクトである [29]．汎用サーバ上のソフトウェアとしてネットワークの制御・管理機能などを実装することで，通信局舎をデータセンタとして活用し，エッジコンピューティングのための計算リソースを提供することができる．複数の通信事業者が SEBA を導入しており，Telefonica は SEBA を通信局舎内にアプリケーションを設置することでエッジコンピューティングシステムとして利用することを検討している [30]．しかしながら，これまで SEBA の仮想リソースを用いてエッジコンピューティングによりリアルタイムモーション制御を行った研究はない．仮想化基盤である SEBA を用いることで，これまで専用の通信装置では実現の難しかった動的な設定変更が簡単に実現でき，ネットワーク状態に応じてアクティブに制御することで，モーション制御に特化したネットワークを構築できる可能性がある．

SEBA のアーキテクチャを図 2-9 に示す．SEBA は Kubernetes コンテナで動作する．NEM (Network Edge Mediator) と ONOS (Open Network Operating System)，VOLTHA (Virtual OLT Hardware Abstraction) で構成されている [31]-[33]．VOLTHA はホワイトボックスと PON ハードウェアデバイスのための PON の制御および管理システムを提供する．アクセスネットワークは抽象化され，SDN コントローラである ONOS に対してはプログラマブルイーサネットスイッチとして提示される．ONOS はネットワークの設定とリアルタイム制御を可能にするオープンソースの SDN コントローラである．NEM は上位のオペレーションシステムをコントローラに接続する．NEM は VOLTHA API (Application Programming Interface) からアクセスデバイスの状態と情報を収集することで作られるモデルを管理する．NEM はアクセスデバイスの状態に応じて，VOLTHA と ONOS にコマンドを送信する．

SEBA はテクノロジープロファイルとスピードプロファイルを用いて PON の設定を管理する．テクノロジープロファイルは PON タイプや優先度，DBA サイクルといった情報が含まれた YAML ファイルである．スピードプロファイルは固定帯域幅，保証帯域幅，ベストエフォート幅を定義する YAML ファイルである．テクノロジープロファイル，スピードプロファイルは上位レイヤから NEM 経由で ONOS に送信されたのち、VOLTHA に入力さ

れる．VOLTHA はネットワーク設定を読み込み，PON を構築するために OLT に入力する．OLT は OLT サービスを通じて起動される．

SEBA の起動ワークフローを図 2-10 に示す．(1) オペレータは NEM にテクノロジープロファイルを送信する．(2) NEM は VOLTHA に OLT の事前プロビジョニング指令を行い，(3) テクノロジーファイルを送信し，(4), (5) OLT を起動する．(6), (7) OLT が ONU を検出すると ONU ステータスや MAC (Media Access Control) アドレスといった情報は VOLTHA に送信される．(8) ONOS は OLT を論理スイッチとみなすため，VOLTHA は ONOS に ONU ステータスをポートステータスとして送信する．(9) ONOS は NEM に新しい UNI (User-Network Interface) ポートが追加されたことを通知する．(10) NEM は情報を保持する ONU の管理モデルを作成し，MAC アドレスが事前登録されたものを一致するかどうかを確認する．(11) 一致している場合は，モデルステータスをアップデートし，加入者情報を ONOS に追加する．以上の手順で SEBA の準備が完了する．SEBA は G-PON や XGS-PON だけでなく，10G-EPON にも対応している [34], [35].

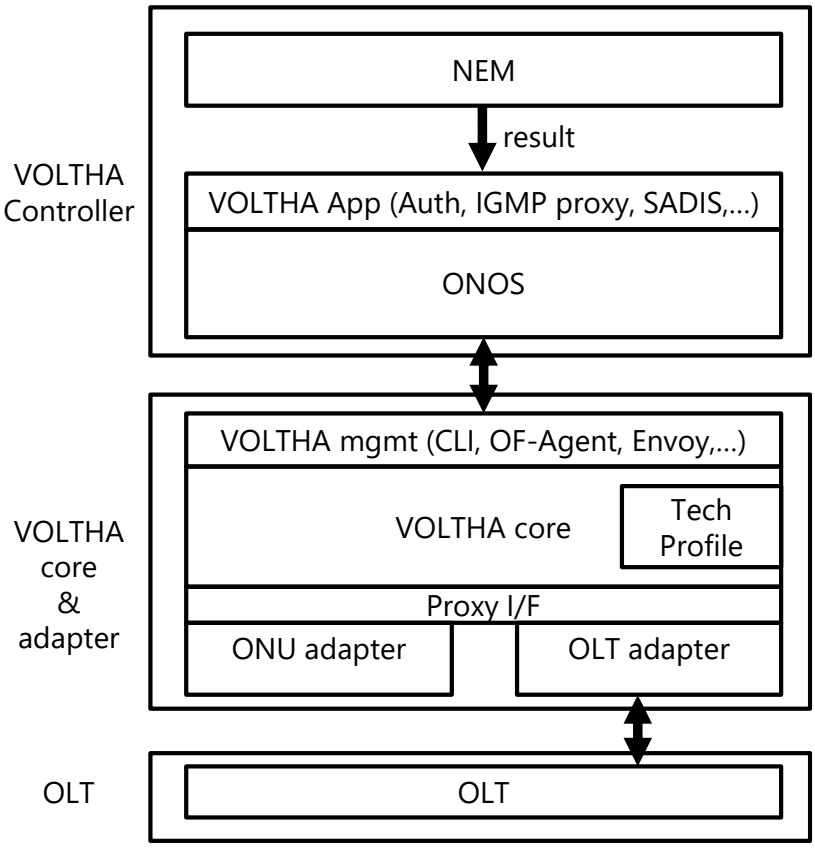


図 2-9　SEBA アーキテクチャ

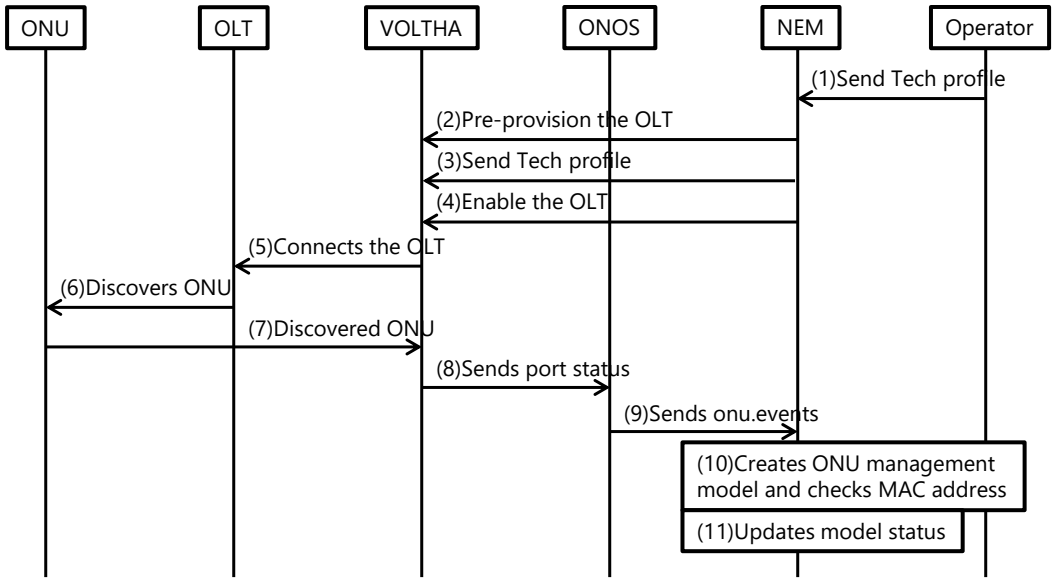


図 2-10　SEBA の起動ワークフロー

2-5　　産業用ネットワーク

最後に関連技術として，これまで特に工場の産業用機器といった遅延の存在が重大な影響を及ぼすようなシステムで用いられてきた産業用ネットワークについて述べる．

2-5-1　　概要

IoT 端末のモーション制御は，多数の IoT 端末からのセンサや映像データの高速大容量伝送だけでなく，高速かつ詳細な制御を実現するための低遅延かつ高信頼性が求められる．計算リソースをエッジやクラウドに置くことは高速大容量通信を実現するためのアプローチのひとつだが，IoT 端末と計算リソース間の距離が大きくなることから遅延が増加し，制御性能に悪影響を及ぼす．ファクトリーオートメーションは 0.25-10 ms の遅延を要求するが，クラウドからの制御は遅延が 100 ms を超えてしまう [36], [37]．そのため，これまで工場内の制御には産業用ネットワークとそれを実現する産業用ネットワークプロトコルが用いられてきた．

工場内の産業用ネットワークは低遅延や正確性，同期性や定時処理，信頼性など厳しい要件を満たす必要がある [38]．従来は専用装置のための専用プロトコルで，特別な転送ルールやシェーピング，スケジューリングやその他の機能が必要であった．そのため，遅延の影響が無視できるローカルでのモーション制御が実行されてきた．産業用ネットワークは一般に工場内に閉じたネットワークである．産業用ネットワークプロトコルはこれまでフィールドバスが用いられてきたが，近年，産業用イーサネットの導入が進んでいる．一般にファクトリーオートメーションの遅延要求を従来のイーサネットプロトコルで実現す

ることは難しく， PROFINET [39]や EtherCAT [40]，EtherNet/IP [41]といった様々な種類の産業用イーサネットプロトコルを用いて遅延性能を保証している [42], [43] .

これらのプロトコルはそれぞれ独自の異なる方法で性能を保証しているため，スタック間で互換性がなく，複数のプロトコルを同じ制御システムで使ったり，統合することができない．産業用イーサネットプロトコルでは機器間の通信間隔が周期的に繰り返され，オフラインスケジュールが作成され，タイムスロットが通信機器に割り当てられる．結果として，スループットやネットワークの再構成にはほとんど重点が置かれない [44]．特に，これらのプロトコルは機器ベンダと結びついていることが多く，一度採用する機器ベンダを決めると，プロトコルも固定され，他のプロトコルを導入することは困難になる．現在の構成では専用の PLC (Programmable Logic Controller) が専用スイッチを介してモータやセンサと接続し，設定は専用の EMS (Equipment Management System) から投入される．インダストリー4.0 では生産システムのデジタル化と物理的なもの，人間，論理エンティティのシームレスな統合を目指している [45], [46]．今後，産業用ネットワークプロトコルの異なる機能を抽象化し，工場ネットワークをオープン化することが求められている．

産業用ネットワークは一般のネットワークと比べて，やり取りされるデータが主に数値データであること，通信経路が定まっており，頻繁なコネクション確率が必要ないこと，周期通信で，1:n，n:n 通信が基本であること，リアルタイム通信が要求され，一定時間通信がない場合は通信が終了することなどの違いがある．産業用ネットワークは装置を監視するための情報ネットワークとコントローラ間で情報を交換するための制御ネットワークと，モーション制御などのためにコントローラと IoT 端末を接続するためのフィールドネットワークとに分類される．これらは通常厳格な遅延要求のために遅延性能を保証する産業用イーサネットプロトコルによって実装される．図 2-11 に産業用ネットワークの分類を示す．情報ネットワークは一般に Ethernet が用いられ，流れるデータは運転計画データや実績データなどが多い．制御ネットワークは周期的なデータの他に，メッセージやアラームといった割り込みデータが多く流れるのが特徴である．速度はそれほど重要とされず，周期は 1 s 程度あれば十分とされる．情報ネットワークと制御ネットワークが外部から制御できる一方で，フィールドネットワークは閉じたネットワークとなっている．

この産業用ネットワークをキャリアネットワークと接続すれば，大量の計算リソースが必要なビッグデータや AI を活用した高度な制御が実現できる [47], [48].

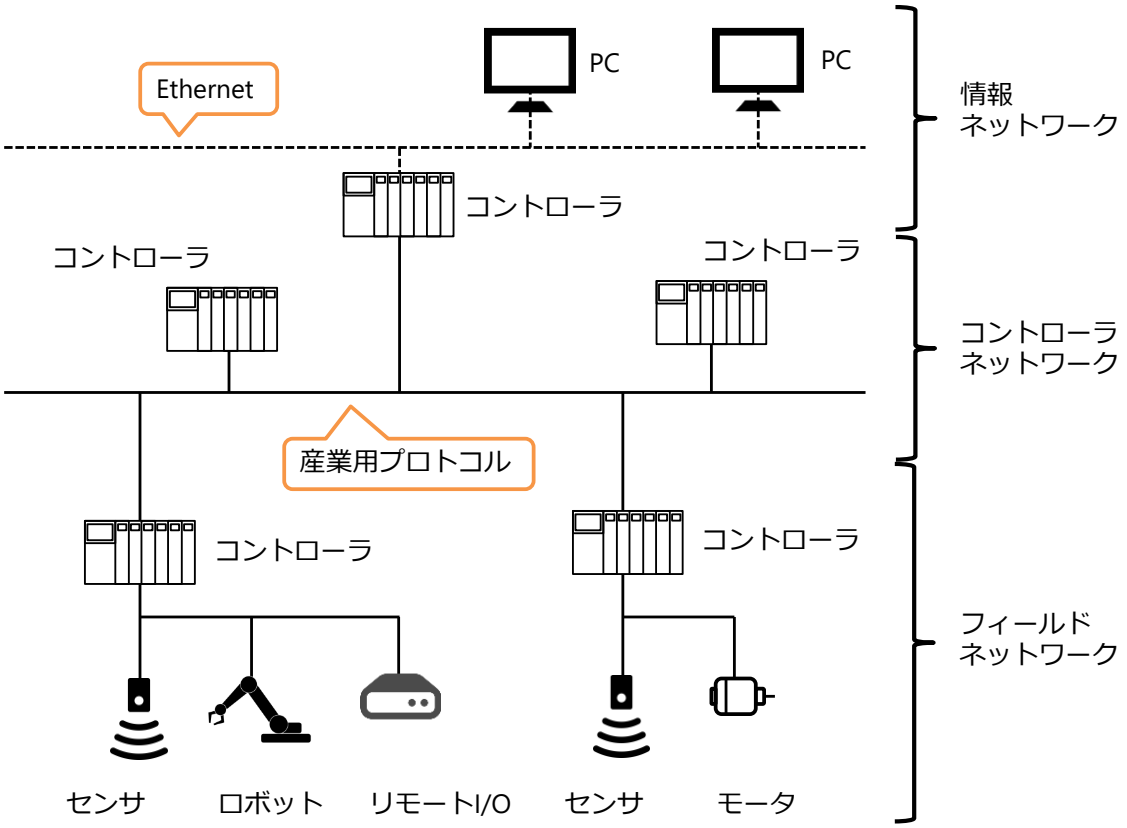


図 2-11　産業用ネットワークの分類

2-5-2　　産業用ネットワークプロトコルの詳細

ここでは PROFINET と EtherCAT を例に説明する．PROFINET プロトコルはシーメンス社が開発したプロトコルで，Non Real-Time (NRT)，Real-Time (RT)，Isochronous Real-Time (IRT) の 3 つのクラスに分けられる [39]．図 2-12 に PROFINET のクラスを示す．NRT はデータ交換に UDP (User Datagram Protocol) を用い，一般のイーサネット上で実装されるネットワークである．RT は標準のイーサネットコントローラを用い，TCP/IP (Transmission Control Protocol / Internet Protocol) や UDP/IP ヘッダなしに通信を行う．VLAN (Virtual Local Area Network) の設定を高い優先度にし，通常のパケットと差別化することで遅延 10 ms 以内を保証している．IRT は専用の ASIC (Application Specific Integrated Circuit) を用い，機器間で時刻同期を行い，遅延 1　ms 以内とジッタ 1 us 以内を保証する．図 2-13 に産業用プロトコルごとのクラスの分類を示す．また，図 2-14 に産業用プロトコルの市場シェアを示す [49].

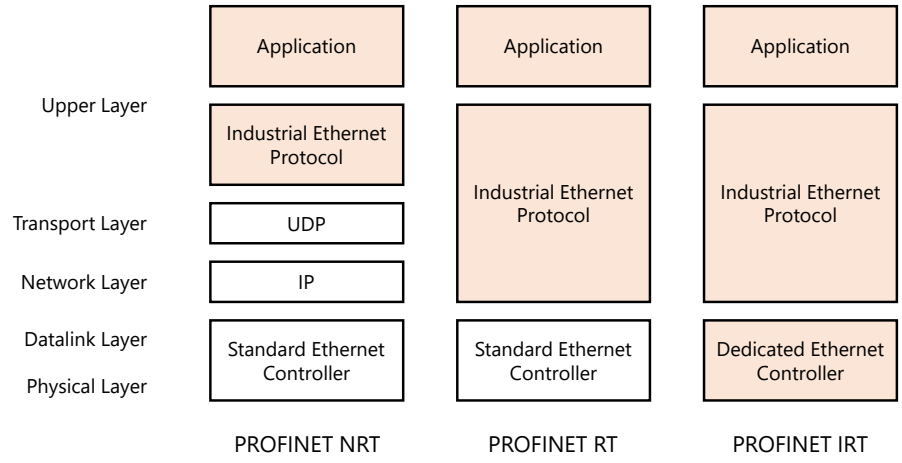


図 2-12　PROFINET プロトコルのクラス

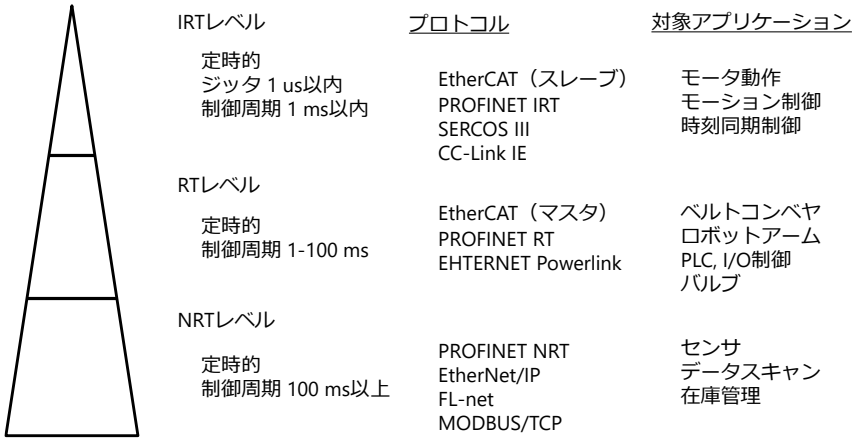


図 2-13　産業用ネットワークプロトコルのクラス

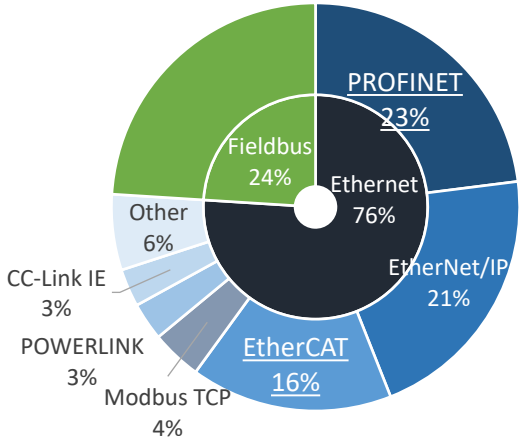


図 2-14　産業用プロトコルの市場シェア

2-6　参考文献

[1] 石田修，瀬戸康一郎，“改訂版 10 ギガビット Ethernet 教科書”，インプレス社，2005

[2] 天野博史，“この 1 冊で大丈夫！アクセスネットワークのすべて”，オーム社，2017

[3] “TR-402 Functional Model for PON Abstraction Interface,” The Broadband Forum, 2018.

[4] N. Yoshikane, “Applications of SDN-Enabled Optical Transport Network and Cloud/Edge Computing Technology,” Optical Fiber Communications Conference and Exhibition, pp. 1–3, 2019.

[5] L. Zhang, H. Gao and O. Kaynak, “Network-Induced Constraints in Networked Control Systems—A Survey,” IEEE Transactions on Industrial Informatics, vol. 9, no. 1, pp. 403–416, 2013.

[6] X. Zhang, Q. Han and X. Yu, “Survey on Recent Advances in Networked Control Systems,” IEEE Transactions on Industrial Informatics, vol. 12, no. 5, pp. 1740-1752, 2016.

[7] C. Lai and P. Hsu, “Design the Remote Control System With the Time-Delay Estimator and the Adaptive Smith Predictor,” IEEE Transactions on Industrial Informatics, vol. 6, no. 1, pp. 73–80, 2010.

[8] D. Yashiro and T. Yakoh, “Feedback Controller With Low-Pass-Filter-Based Delay Regulation for Networked Control Systems,” IEEE Transactions on Industrial Electronics, vol. 61, no. 7, pp. 3744–3752, 2014.

[9] K. Natori and K. Ohnishi, “A Design Method of Communication Disturbance Observer for Time-Delay Compensation, Taking the Dynamic Property of Network Disturbance Into Account,” IEEE Transactions on Industrial Electronics, vol. 55, no. 5, pp. 2152–2168, 2008.

[10] H. Yi, C. An. and J. Choi, “Compensation of Time-Varying Delay in Networked Control System over Wi-Fi Network,” International Journal of Computers, Communications & Control, vol. 12, no. 3, pp. 415–428, 2017.

[11] M. Ahmadi, N. Khanezaei, S. Manavi, F. Fatemi Moghaddam and T. Khodadadi, “A Comparative Study of Time Management and Energy Consumption in Mobile Cloud Computing,” 2014 IEEE 5th Control and System Graduate Research Colloquium, pp. 199-203, 2014.

[12] E. Markoval, D. Moltchanov, R. Pirmagomedov, D. Ivanova, Y. Koucheryavy and K. Samouylov, “Priority-based Coexistence of eMBB and URLLC Traffic in Industrial 5G NR Deployments,” 2020 12th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pp. 1-6, 2020.

[13] Č. Stefanović, “Industry 4.0 from 5G perspective: Use-cases, requirements, challenges and approaches,” 2018 11th CMI International Conference: Prospects and Challenges Towards Developing a Digital Economy within the EU, pp. 44-48, 2018.

[14] H. Yang, K. Zhang, K. Zheng and Y. Qian, “Leveraging Linear Quadratic Regulator Cost and Energy Consumption for Ultrareliable and Low-Latency IoT Control Systems,” IEEE Internet of Things Journal, vol. 7, no. 9, pp. 8356-8371, 2020.

[15] X. Zhang, Q. Han and X. Yu, “Survey on Recent Advances in Networked Control Systems,” IEEE Transactions on Industrial Informatics, vol. 12, no. 5, pp. 1740-1752, 2016.

[16] C. Lai and P. Hsu, “Design sthe Remote Control System With the Time-Delay Estimator and the Adaptive Smith Predictor,” IEEE Transactions on Industrial Informatics, vol. 6, no. 1, pp. 73-80, 2010.

[17] D. Yashiro and T. Yakoh, “Feedback Controller With Low-Pass-Filter-Based Delay Regulation for Networked Control Systems,” IEEE Transactions on Industrial Electronics, vol. 61, no. 7, pp. 3744-3752, 2014.

[18] K. Natori and K. Ohnishi, “A Design Method of Communication Disturbance Observer for Time-Delay Compensation, Taking the Dynamic Property of Network Disturbance Into Account,” in IEEE Transactions on Industrial Electronics, vol. 55, no. 5, pp. 2152-2168, 2008.

[19] H. Yi, C. An. and J. Choi, “Compensation of Time-Varying Delay in Networked Control System over Wi-Fi Network,” International Journal of Computers, Communications & Control, vol. 12, no. 3, pp. 415-428, 2017.

[20] M. Ahmadi, N. Khanezaei, S. Manavi, F. Fatemi Moghaddam and T. Khodadadi, “A Comparative Study of Time Management and Energy Consumption in Mobile Cloud Computing,” 2014 IEEE 5th Control and System Graduate Research Colloquium, pp. 199-203, 2014.
[21] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, et al., “P4: Programming Protocol-Independent Packet Processors,” ACM SIGCOMM Computer Communication Review, vol. 44, no. 3, pp. 87–95, 2014.
[22] T. Suzuki, S. Kim, J. Kani, and J. Terada, “Demonstration of Fully Softwarized 10G-EPON PHY Processing on a General-Purpose Server for Flexible Access Systems,” IEEE/OSA Journal of Lightwave Technology, vol. 38, issue 4, pp. 777–783, 2020.
[23] T. Suzuki, S. Kim, J. Kani, and J. Terada, “Real-Time Implementation of Coherent Receiver DSP Adopting Stream Split Assignment on GPU for Flexible Optical Access Systems,” IEEE/OSA Journal of Lightwave Technology, vol. 38, issue 3, pp. 668–675, 2020.
[24] G. Simon, P. Chanclou, M. Wang, D. Abgrall and D. Minodier, “Optical Access Evolutions Towards SDN and Disaggregated Hardware: an Operator Perspective,” 2021 European Conference on Optical Communication (ECOC), pp. 1-32, 2021.
[25] J. Kani, T. Suzuki, Y. Kimura, S. Kaneko, S. Kim and T. Yoshida, “Disaggregation and virtualization for future access and metro networks [Invited Tutorial],” Journal of Optical Communications and Networking, vol. 17, no. 1, pp. A1-A12, 2025.
[26] L. Peterson, A. Al-Shabibi, T. Anshutz, S. Baker, A. Bavier, S. Das, J. Hart, G. Palukar, and W. Snow, “Central office re-architected as a data center,” IEEE Commun. Mag., vol. 54, no. 10, pp. 96–101, 2016.
[27] S. Kim, T. Suzuki, J. Kani, and J. Terada, “Exploiting general purpose hardware in optical access systems [Invited],” Journal of Optical Communications and Networking, vol. 12, no. 2, pp. A182-A189, 2020.
[28] “ CORD Platform | Central Office Rearchitected as a Datacenter | ONF,” https://opennetworking.org/cord/
[29] S. Das, “From CORD to SDN Enabled Broadband Access (SEBA) [Invited Tutorial],” in Journal of Optical Communications and Networking, vol. 13, no. 1, pp. A88-A99, 2021.
[30] “Telefónica open access and edge computing,” White Paper, 2019.
[31] “ XOS and NEM – OpenCORD Wiki,” https://wiki.opencord.org/display/CORD/XOS+and+NEM
[32] “ONOS - Wiki,” https://wiki.onosproject.org/display/ONOS/ONOS
[33] “VOLTHA Docs,” https://docs.voltha.org/master/index.html
[34] T. Suzuki et al., “Demonstration of IEEE PON Abstraction for SDN Enabled Broadband Access (SEBA),” Journal of Lightwave Technology, vol. 39, no. 20, pp. 6434-6442, 2021.
[35] T. Suzuki et al., “Zero touch provisioning compliant with authentications of IEEE PON packages A and B for SDN-enabled broadband access,” Journal of Optical Communications and Networking, vol. 13, no. 11, pp. 244-252, 2021.
[36] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat and H. Dai, “A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions,” IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3098-3130, 2018.
[37] S. Weibin, L. Yun, D. Yi, D. Yingguo, P. Mingbo and X. Gang, “Three-Real-Time Architecture of Industrial Automation Based on Edge Computing,” 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), pp. 372-377, 2019.
[38] 元吉伸一, “工場通信ネットワーク入門: 製造現場で活躍するフィールドバスから産業用 Ethernet まで”, 日刊工業新聞社, 2006
[39] PROFIBUS & PROFINET International (PI), “PROFINET System Description,” Ohiostrasse 8 Karlsruhe Germany, 2014.
[40] “EtherCAT - The Ethernet Field Bus,” https://www.ethercat.org/

[41] “ EtherNet/IP™ | ODVA Technologies | Industrial Automation,” https://www.odva.org/technology-standards/key-technologies/ethernet-ip/
[42] I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, “A Survey on Low Latency Towards 5G: RAN, Core Network and Caching Solutions,” IEEE Communications Surveys & Tutorials, vol. 20, no. 4, pp. 3098–3130, 2018.
[43] S. Weibin, L. Yun, D. Yi, D. Yingguo, P. Mingbo, and X. Gang, “Three-Real-Time Architecture of Industrial Automation Based on Edge Computing,” 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), pp. 372–377, 2019.
[44] L. Moutinho, P. Pedreiras, and L. Almeida, “A Real-Time Software Defined Networking Framework for Next-Generation Industrial Networks,” IEEE Access, vol. 7, pp. 164468–164479, 2019.
[45] M. Wollschlaeger, T. Sauter, and J. Jasperneite, “The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0,” IEEE Industrial Electronics Magazine, vol. 11, no. 1, pp. 17–27, 2017.
[46] L. Silva, P. Pedreiras, P. Fonseca, and L. Almeida, “On the Adequacy of SDN and TSN for Industry 4.0,” 2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC), pp. 43–51, 2019.
[47] “ IoT Edge, Open Source Edge - AWS IoT Greengrass - AWS,” https://aws.amazon.com/greengrass/
[48] “ IoT Edge | Cloud Intelligence | Microsoft Azure,” https://azure.microsoft.com/en-us/products/iot-edge/
[49] HMS, “Market shares 2024 according to HMS Networks,” 2024.

第 3 章　　通信機器からの現在の正確な遅延情報の取得

3-1　はじめに

第 1 章，第 2 章ではアクセスエッジコンセプトの概要と，アクセスネットワークの概要と遅延原因について述べた．アクセスエッジコンセプトはユーザと近距離にある通信局舎やマイクロデータセンタ内にリソースを設置し，リアルタイムモーション制御のためのデータセンタとして扱うことで，低遅延と高信頼性を確保する．課題はアクセスネットワークの遅延がクラウドに比べて小さいながら制御性能を劣化させる可能性があることである．

本章では，Point-to-Point トポロジを対象に，アクセス区間で生じる遅延の影響を低減するための手法を提案し，アクセスエッジコンセプトの有効性を検証する．アクセスシステムではコアネットワークと異なり，通信事業者は局舎にある制御器から IoT 端末までの経路がわかり，経路上のすべての通信機器を特定することができる．遅延の原因であるキューイング遅延や処理遅延は通信機器で発生する遅延であるので，これらの通信機器全てから処理時間を収集することで，ネットワークでの全体の遅延が把握できる．提案手法では，遅延情報を積極的に取得し，遅延補償のために制御器に送信する．

提案手法により，通信局舎ではリアルタイム制御関連のデータセンタサービスを提供でき，またこれらのサービスをビッグデータや AI といった高性能なデータ処理アプリと簡単に接続できる．アクセスエッジを実現するためには，モーション制御と通信機能の連携が不可欠である．いくつかの従来手法ではローカルエリアネットワークパラメータをモーション制御のために設計する手法が提案されているが，光アクセスシステムについて扱った文献は見当たらない．また，アクセスシステムの仮想化基盤である SEBA はアクセスシステムの変数を管理するために汎用 API を用いるが，モーション制御とネットワーク管理は依然課題である [1], [2]．本手法では，アクセスエッジ構成をより柔軟にするために仮想化技術を用いて強化した．提案構成では新しいデータモデルが遅延とモーション制御アプリケーションを連携するために定義され，SEBA アーキテクチャが通信機器から遅延を収集し制御器に送信するために拡張されている．アクセスネットワークを模擬する実験系を構築し，遅延が増加し，パケットロスが生じるような高負荷環境下で実験を行い，従来手法が制御できない場合でも提案手法では制御が実現できることを示す．

3-2　遅延情報取得による遅延補償制御

提案手法では，遅延情報はリアルタイムに End-to-End の経路上の通信機器全てから取得し，End-to-End の総遅延を算出，時変遅延に対応した遅延補償制御に用いる．図 3-1 にこの手法の処理フローを示す．提案手法を実現するために，各通信機器のネットワーク機能の近傍に新たなコンポーネント（赤枠）を追加する．(1) 図 3-1 に示すように IoT 端末の状態を表すセンサデータを収集し，通信局舎のサーバ上の制御器に送信する．(2) 同時に，通

信機器はそれぞれのネットワーク機能の開始時刻と終了時刻のタイムスタンプを取得する．(3) 処理時間はこれらの時間の差分として求められ，この通信機器の遅延となる．(4) 続いて，遅延は通信局舎の制御器にフレーム処理により送信される．(5) 制御器では全ての通信機器からの遅延を合算して合計遅延を算出する．(6) 最後に，制御器はセンサデータと合計遅延を用い遅延補償制御値を算出する．制御値を IoT 端末に送信し，所望のモーション制御を実現する．

図 3-2 に提案手法の概念図を示す．ここでは遅延補償手法の例としてスミス予測器を採用する．遅延の予測値 t_m はすべてのネットワーク機器からの遅延情報 L_1, .., L_n を合算することで計算され，$x_3(t)$ が算出される．提案手法の機能フローは Algorithm 1 で表される．制御値 $u(t)$ は $r(t)-x_0(t)$ がしきい値より小さくなるまで $x_3(t)$ を用いて制御器によって繰り返し計算される．提案手法では合計遅延は t_m でリアルタイムに置き換えられる．

提案手法ではネットワーク状態が通信機器から制御器に繰り返し送信されるので，時変遅延に能動的に対応できる．また処理時間が長く，遅延の原因となっている通信機器を特定することも可能である．通信機器の適切な管理や経路構成を能動的に変更することで，リアルタイム制御のためな低遅延ネットワークを構築することが可能である．

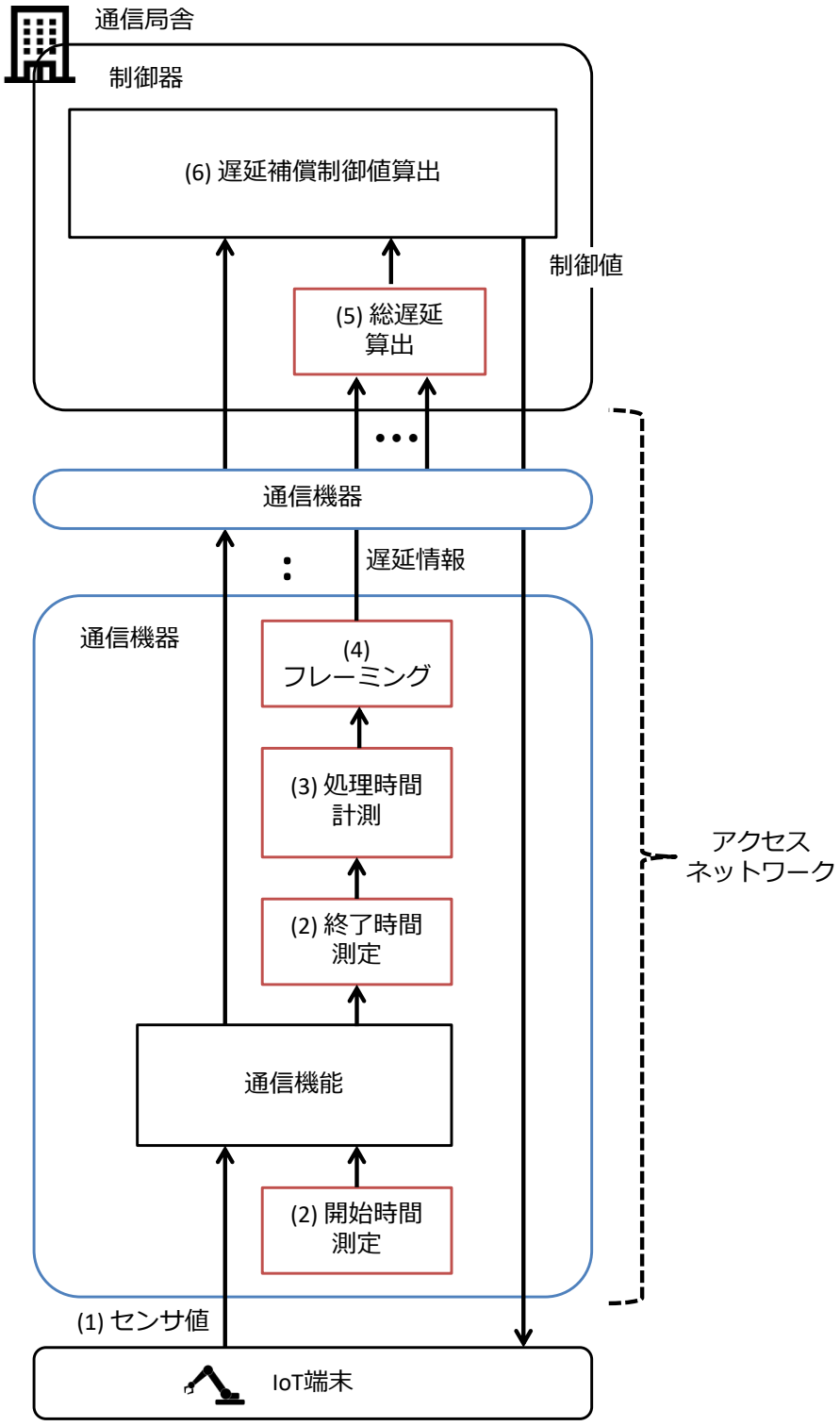


図 3-1　提案手法のフローチャート

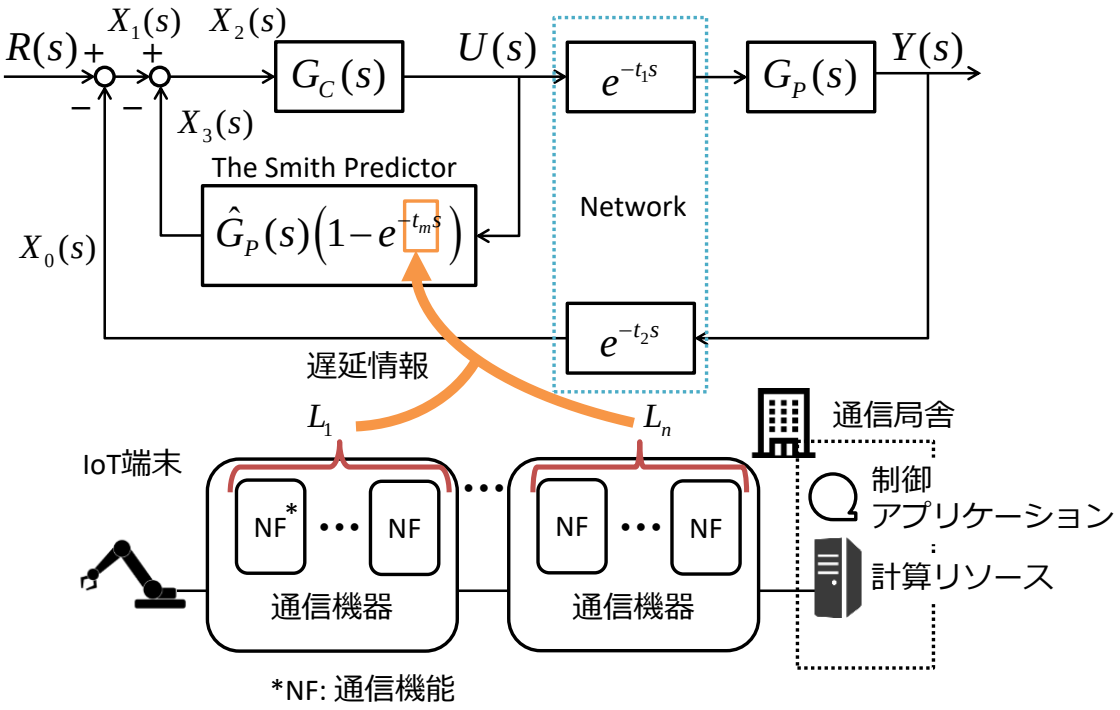


図 3-2　提案手法の概念図

Algorithm 1 Algorithm for the proposed method

Input: $r(t), x_0(t), L_1, .., L_n, threshold$
Output: $u(t)$

1: **while** $|r(t) - x_0(t)| < threshold$ **do**
2: Receive the delay information $L_1, .., L_n$ from all the communication devices
3: $t_m = \sum_{i=1}^{n} L_i$
4: $x_1(t) = r(t) - x_0(t)$
5: $x_2(t) = x_1(t) - x_3(t)$
6: $u(t)$ is calculated by the controller with $x_2(t)$
7: $x_3(t)$ is calculated by Smith predictor with t_m
8: **end while**
9: **return** $u(t)$

3-3　仮想化技術を用いた提案構成

アクセスエッジ構成はユーザの通信機器と通信局舎を接続する．エッジサーバは通信局舎に置かれ，通信機器を集約するスイッチに接続される．

アクセスエッジの実際の運用をサポートするために，エッジサーバ上のサービスを変更する必要がある．期待するサービス変更を実現するには，仮想化技術を用いることが効果

的である [3]．仮想化技術の進展によって，エッジコンピューティング技術のソフトウェアスタックである Akraino といった様々な技術が提案されている．VM (Virtual Machine) はエッジサーバ上で起動し，モーション制御を実現するアプリケーションを実装するために使用することができる．

近年の研究では，光アクセスネットワークの仮想化の一環として，アクセスネットワーク機能の追加や置き換えによって実現される柔軟性がターゲットとなっている．ONF (The Open Networking Foundation) は CORD コンセプト向けに，SEBA プラットフォームを導入した [4]．このコンセプトでは SDN のインスタンス化により，汎用サーバ上でサービスを実装することが可能となる．エッジサーバをホワイトボックススイッチのような汎用ハードウェアと接続したり，サーバの余剰リソースを使って汎用ハードウェア上にエッジコンピューティング技術を適用したりすることで，より柔軟な構成が実現できる．このような構成は複数のサーバ上の VM を管理できる Kubernetes のようなオープンソースソフトウェア (OSS) オーケストレーターを用いることで実現できる．いくつかの通信事業者が SEBA の導入を開始しているが，SEBA は主にネットワークエッジでのトラフィックオフローディングによるラストマイルの帯域幅の低減を行うコンテンツデリバリーネットワークのような遅延要件が厳しくないアプリケーションを対象としており，厳しい遅延要件が求められるモーション制御アプリケーションは対象としていない [5].

図 3-3 は提案の具体構成を示す．既存の SEBA システム（黒枠）に，計算リソース上の制御器を SEBA に接続するための新たなコンポーネント（赤枠）が追加されている．また，各通信装置には新たにファームウェア（青枠）が新たにインストールされる．遅延情報とモーション制御アプリケーションを連携させるために，SEBA アーキテクチャを拡張し，遅延情報を収集，全体の遅延を計算できるようにした．また，新しく遅延モデルを定義した．この構成は NEM や ONOS，VOLTHA といったアクセスシステムの OSS や SDN スイッチを基盤としている [6]-[8]．ONOS は SDN の制御プレーンを提供し，VOLTHA はブロードバンドアクセス装置のためにハードウェアを抽象化する．既存の VOLTHA のデータモデルは，機器を識別する ID，機器の種類を示す TYPE，ADDRESS，STATE などが含まれる．SDN スイッチ (virtual Switch, vSW) は OpenFlow をサポートし，スイッチと制御器間の通信に用いられる．また，複数のメディアコンバータや OLT がエッジサーバに接続することが想定される．計算リソース上の制御器は IoT 端末を制御するユーザアプリケーションである．VOLTHA のデータモデルを拡張し，それぞれのデバイスの遅延情報を転送できるように拡張した．

処理のフローを示す．(1) 制御器と IoT 端末間のネットワーク経路を検出する．(2) この経路情報に基づいて，ONOS に経路上の各通信機器から遅延情報を収集するよう指示する．(3) 各デバイスは VOLTHA に遅延情報送信機能を通じて遅延情報を送信する．(4), (5) VOLTHA に格納された遅延情報は，経路全体の合計遅延を計算するために収集され，合算

されたのち，制御器に渡される．SEBA によりネットワーク装置の正確な識別と遅延情報の取得を容易に行うことができる．

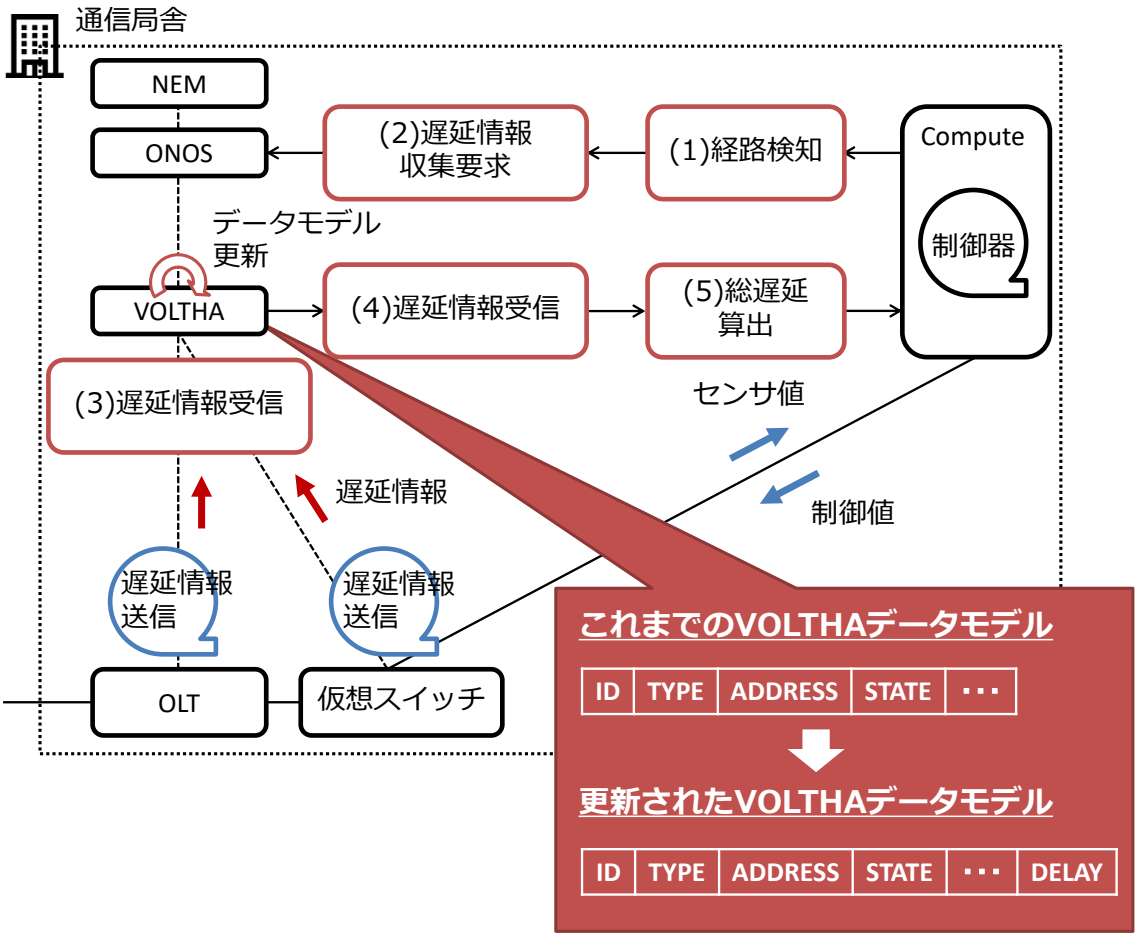


図 3-3　仮想化技術を用いたアクセスエッジコンセプトの構成

3-4　提案手法の実機検証

3-4-1　　実験構成

モータを用いた実機実験系を構築し，ネットワーク高負荷環境での制御性能を測定した．また遅延取得周期と制御周期を変更したときの提案手法の制御性能についても検証した．実験系を図 3-4 に示す．ユーザサイドの通信機器と通信局舎のエッジサーバを模擬するため，2 台のサーバを用いた．それぞれのサーバスペックは，Intel Xeon E5-2699 (2.20 GHz, 22 cores) CPU で 128 GB メモリである．また OS は Linux 16.04 を用いた．メディアコンバータは 10 Gbps の NIC と SFP+ (Small Form-Factor Pluggable Plus) モジュールを用いて模擬した．

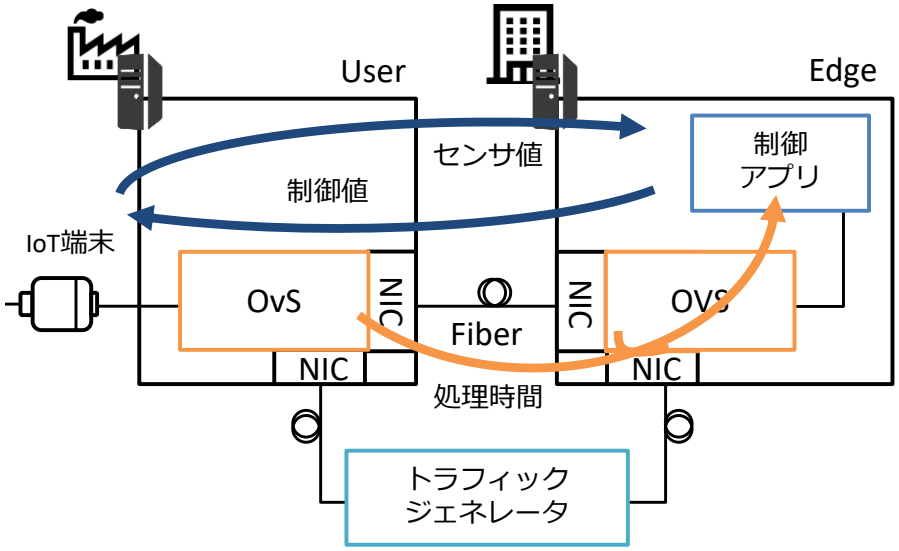


図 3-4　実験構成

通信機器はソフトウェアスイッチである Open vSwitch (OvS) を用いた．ソフトウェアスイッチを用いることで通信機器に機能を容易にインストールすることができる．OvS は処理時間を計測し，制御器に送信するようにプログラムを改変した．OvS は 2 台のサーバ両方にインストールされた．伝送プロトコルにはそのリアルタイム性能から UDP を用いた．制御器は通信局舎サーバにインストールされた．モータは Maxon EC-max 40 を用い，ユーザサーバとは USB ケーブルと EPOS2 24/5 を介したシグナルケーブルで接続した．計算された制御値は電流値に変換され，電流制御が行われる．

高負荷環境を構築するため，トラフィックジェネレータとして VIAVI MTS-5800 を用いた．トラフィックジェネレータとサーバは光ファイバで接続した．トラフィックは上り方向のみ，下り方向のみの 2 つのトラフィックパターンで 1 Gbps から 10 Gbps まで 1 Gbps 刻みで設定した．10 Gbps のスループットを実現するために，DPDK（Data Plane Development Kit）をそれぞれのサーバにインストールし，OvS と接続した．

制御手法として，工場でいまだ広く用いられている古典制御手法を用いた．以下の 3 つの手法で検証した．

- 手法 1：スミス予測器を用いない PI (Proportional-Integral) 制御
- 手法 2：スミス予測器の予測遅延を 10 ms の固定値に設定した PI 制御
- 手法 3：スミス予測器の予測遅延を通信機器から繰り返し送信される計測値を用いた PI 制御

スミス予測器のモータモデルは次式で示すように 2 次システムで近似した．数値はモータの仕様書に基づいて決定した．

$$\hat{G}_P(s) = \frac{195000}{s^2 + 114s + 150} \tag{3-1}$$

PI 制御のゲインはパラメータチューニングによって決定した．

$$G_C(s) = K_P E(s) + K_I E(s)/s \tag{3-2}$$

$$K_P = 0.002 \quad K_I = 0.001 \tag{3-3}$$

制御周期は 10 ms とし，位置制御の目標値をモータの 1 回転に相当する 4000 qc (quad counts) とした．整定時間は目標値の±5%以内に収束する時間とした．

3-4-2 モータを用いた実験

表 3-1 帯域ごとの遅延とフレームロスの関係

		Upstream [Gbps]		Downstream [Gbps]		
		1-9	10	1-8	9	10
Delay	Average [ms]	8.4	8.4	8.4	11.4	11.5
	Standard Deviation	0.90	0.90	0.90	1.35	1.45
Packet Loss [frames]		0	19000	0	0	8300

表 3-1 はシステム全体を通して発生した遅延とフレームロスの結果を示した．フレームロスは 30 s 間測定した．上りトラフィックの遅延はおよそ一定で 8.4 ms であった．下りトラフィックは 1-8 Gbps は平均遅延が 8.4 ms であったが，9 Gbps 導通したときには 11.4 ms, 10 Gbps 導通したときには 11.5 ms でパケットロスが生じた．遅延が下り方向のトラフィックで増加したため，高負荷環境下での制御性能は下り方向のトラフィックで測定した．

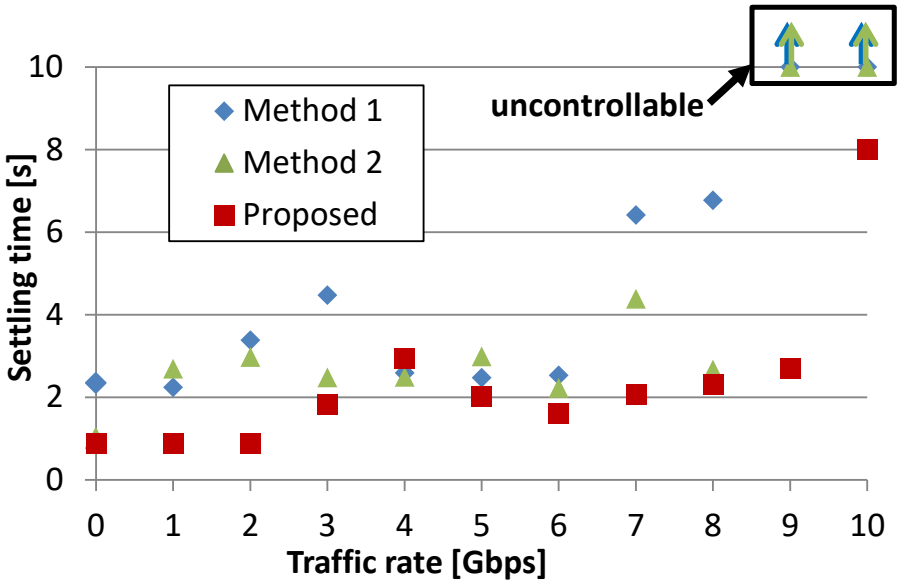


図 3-5　下り方向の帯域ごとの制御性能

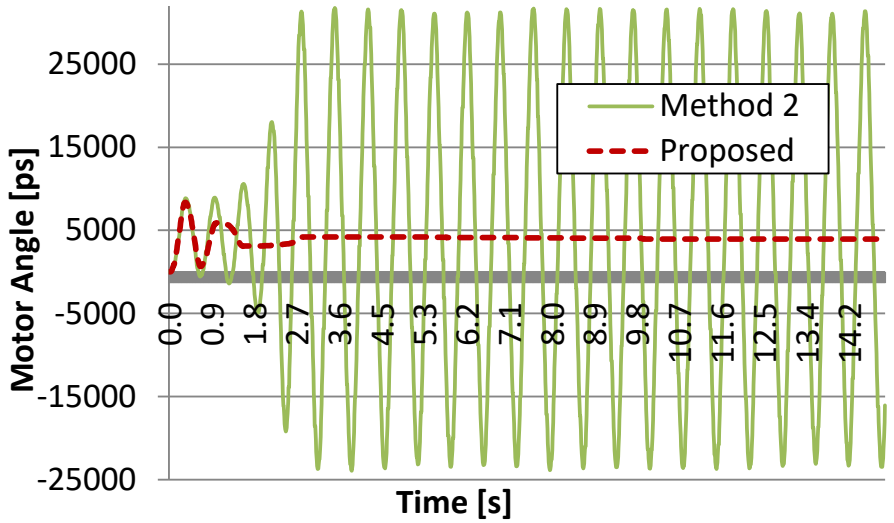


図 3-6　帯域が 9 Gbps のときの時刻歴

下りトラフィックが 1-10 Gbps の場合の制御性能を図 3-5 に示す．図 3-6 はトラフィックレートが 9 Gbps のときの，手法 2 と提案手法の時刻歴を示したものである．トラフィックが 9-10 Gbps のとき，従来手法である手法 1, 2 は遅延変化に対応できず，モータを制御することができない．一方で，提案手法では遅延変化に対応し，モータを制御することができた．整定時間は 9 Gbps のとき 2.7 s で，10 Gbps のとき 8.0 s であった．10 Gbps のときにはパケットロスも生じたため，制御性能が低下し，整定時間が増加したと考えられる [9]．提案手法は遅延変化にはうまく対応できたが，パケットロスには対応できなかった．パケットロスが発生すると，遅延情報やセンサ情報が正常に収集できず，また制御値が正しく送信されない可能性があるため，パケットロスが想定される環境では異なるアプローチを

取る必要がある．パケットロスを補償するひとつの方法は，スミス予測器に加えて外乱オブザーバを用いることである．外乱オブザーバはパケットロスの影響を測定し，フィードバック値に加えることで，その影響を低減することができる．

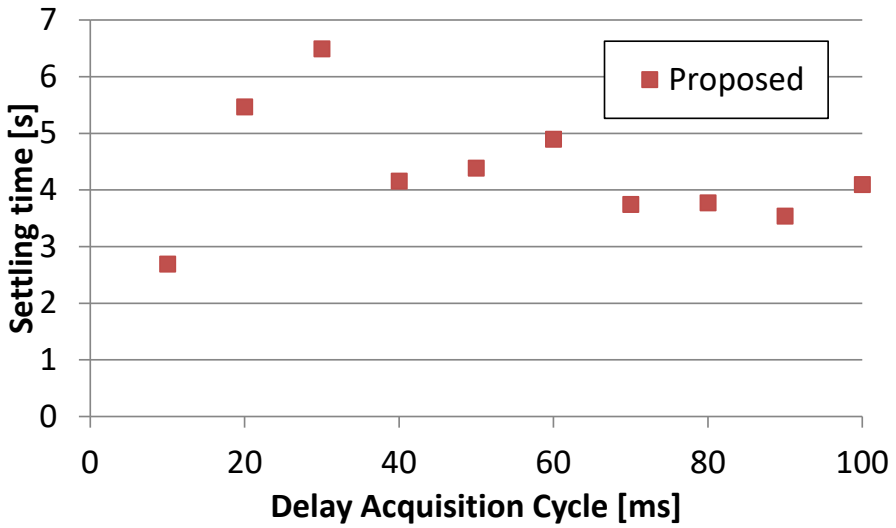


図 3-7　遅延取得周期と整定時間の関係

図 3-7 ではトラフィックレートが 9 Gbps のときの，制御性能と遅延取得周期の関係を示す．制御周期を 10 ms の固定値にし，遅延取得周期を変化させた場合の制御性能への影響を検証した．制御周期と遅延取得周期が 10 ms で等しいとき，整定時間は 2.7 s であった．しかしながら遅延取得周期が増加すると，整定時間も増加した．制御周期が 10 ms であったため，モータからのセンサ情報も 10 ms ごとに受信していた．制御値を計算する際に遅延情報が更新されていないと，古い値が使用され，遅延変動を細かく処理することができない．遅延情報と実際の遅延の差異によりシステムモデル誤差が引き起こされ，制御性能が劣化した．遅延変動がより大きなシステムではこの影響がより増大すると考えられる．

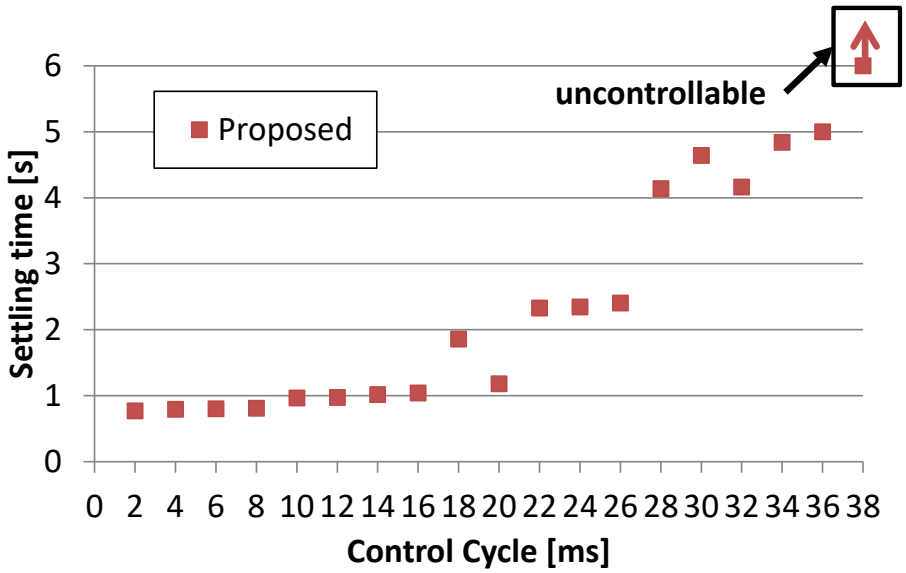


図 3-8　制御周期と制御性能の関係

図 3-8 では無負荷時のネットワーク環境における制御周期と整定時間の関係を示している．遅延取得周期を固定値にした状態で，制御周期を変更した場合の制御性能への影響を計測し，これまでの実験の制御周期 10 ms が適切な値であることを確認した．制御周期が 2-16 ms の場合は制御性能に大きな変化は見られないが，18 ms から整定時間が徐々に伸び，38 ms では制御不能となった．遅延情報が正しく送られていたとしても，制御周期が大きすぎると制御できないことがわかる．短い制御周期では，遅延情報と実際の遅延の間の差異から生じるモデル誤差への対応が容易であり，高い制御精度につながる．この実験では制御周期が 2-16 ms が適切であることが確認された．

3-4-3　　ドローンを用いた実験

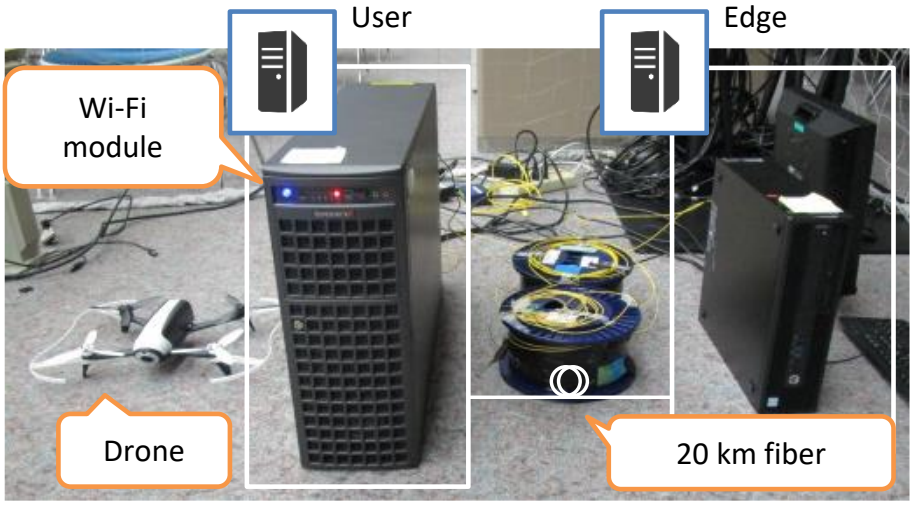


図 3-9　ドローン実験構成

実際のアプリケーションを想定し，ドローンを用いた実機実験を行った．ドローンとして Parrot Bebop 2 を用いた．目標整定時間を 10 s とした [10]．ドローンの高さは，搭載されている超音波センサを用いて測定した．手法は前節のモータ実験と同じである．図 3-9 に実験構成を示す．2 台のサーバを 20 km の光ファイバで接続した．初期状態を高さ 1 m，目標値を 2.5 m とする位置制御を行った．制御周期を 100 ms とした．この周期はセンサ情報が取得できる周期を測定することで決定した．

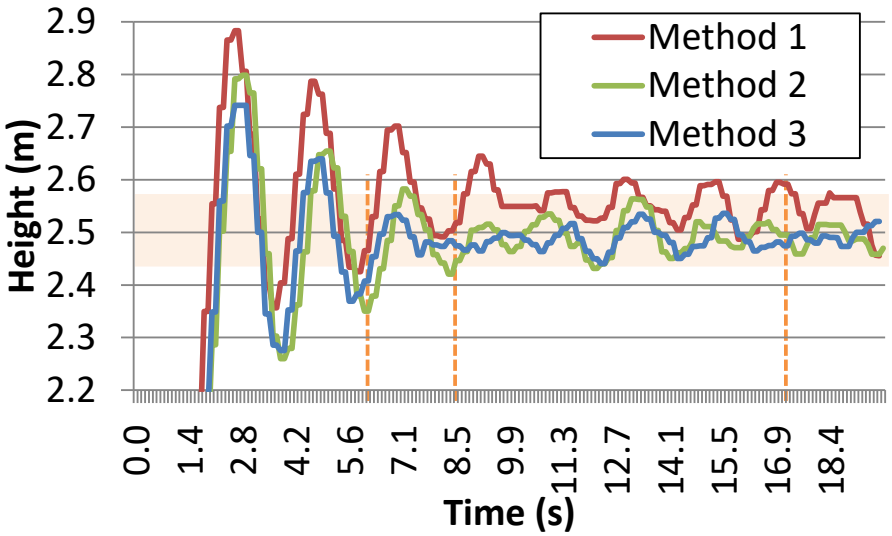


図 3-10　ドローン実験結果

図 3-10 に実験結果を示す．整定時間は手法 1 が 17.2 s，手法 2 が 8.5 s，提案手法が 6.3 s であった．提案手法は目標の整定時間を達成した．手法 2 と提案手法の間で有意な差は見られず，どちらも手法 1 より優れていた．これはアクセスネットワークの遅延がドローンの制御周期よりも小さく，遅延が大きく影響しなかったためであると考えられる．しかし，実際の制御では制御周期がより短くなり，遅延の影響が大きくなることが想定され，提案手法の制御性能がより良くなると考えられる．

3-5　結論

局舎に設置したエッジサーバ上でリアルタイムモーション制御を行うアクセスエッジコンセプトの詳細を説明し，通信機器から収集した経路全体の遅延を用いて時変遅延補償制御を行うことでアクセスネットワークで生じる遅延の制御性能への影響を低減する手法を提案した．また，仮想化技術に基づく柔軟性の高いアクセスエッジ構成を詳細に検討した．モータを用いた高負荷環境下での実験を行い，従来手法では下りトラフィックが 9, 10 Gbps では制御できない場合でも，提案手法では遅延に対応し，それぞれ 2.7 s, 8.0 s で制定できることを確認した．また，実際のアプリケーションを想定して，ドローンを用いた実機実験を行い，目標整定時間を達成した．

3-6 参考文献

[1] K. Huang, W. Liu, Y. Li, A. Savkin and B. Vucetic, “Wireless Feedback Control With Variable Packet Length for Industrial IoT,” IEEE Wireless Communications Letters, vol. 9, no. 9, pp. 1586-1590, 2020.

[2] Y. Maddahi, S. Liao, W. Fung, E. Hossain and N. Sepehri, “Selection of Network Parameters in Wireless Control of Bilateral Teleoperated Manipulators,” IEEE Transactions on Industrial Informatics, vol. 11, no. 6, pp. 1445-1456, 2015.

[3] A. C. Baktir, A. Ozgovde and C. Ersoy, “How Can Edge Computing Benefit From Software-Defined Networking: A Survey, Use Cases, and Future Directions,” IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2359–2391, 2017.

[4] L. Peterson et al., “Central office re-architected as a data center,” IEEE Communications Magazine, vol. 54, no. 10, pp. 96–101, 2016.

[5] “VOLTHA/SEBA status in Telefónica`s ONLIFE,” SEBA/VOLTHA Community Virtual Meeting, 2020.

[6] “XOS and NEM – OpenCORD Wiki,” https://wiki.opencord.org/display/CORD/XOS+and+NEM

[7] “ONOS - Wiki,” https://wiki.onosproject.org/display/ONOS/ONOS

[8] “VOLTHA Docs,” https://docs.voltha.org/master/index.html

[9] R. Imai and R. Kubo, “Experimental validation of communication disturbance observer for networked control systems with information losses,” IEICE Communications Express, vol. 5, no. 4, p. 102–107, 2016.

[10] V. Indrawati, A. Prayitno and G. Utomo, “Comparison of two fuzzy logic controller schemes for position control of AR.Drone,” 2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE), Chiang Mai, pp. 360-363, 2015.

第 4 章　遅延モデルを用いた PON ネットワークでの遅延情報の取得

4-1　はじめに

第 2 章で述べたように，光アクセスシステムの構成は主に，Point-to-Point トポロジと Point-to-Multipoint トポロジの 2 つに分けられる．PON システムは一般的に Point-to-Multipoint トポロジで用いられる．PON システムは 1 波長を複数のユーザに分配する方式で，工場内の IoT 端末の制御に対し，大量の端末を接続できる，大容量である，低コストである，電磁耐性がある，柔軟な帯域割当ができるなど多数の利点がある．しかしながら，Point-to-Point トポロジと異なり，遅延ゆらぎや帯域制限の影響が重大になりうる．PON システムはデータ衝突を防ぎながら多数のデバイスと接続するため，アクセス制御プロトコルを必要とし，現状，サービス要求と QoS 要求に応じて帯域幅を動的に分配するために，TDMA プロトコルに基づく DBA アルゴリズムが用いられる．DBA アルゴリズムは効率的な帯域の使用を可能にする一方で，設定によっては制御性能を劣化させる重大な遅延ゆらぎを発生しうる．

この課題に対し，DBA の遅延を低減するアルゴリズムが提案されている．それぞれのユーザからの帯域要求を収集し，帯域割当を計算する必要があるため，DBA では一般に低遅延と帯域使用効率の間にトレードオフの関係が存在する [1]-[3]．参考文献[4]の手法はモバイル端末の使用を前提としており，工場の IoT 端末には使用できない．一方で，IoT 端末制御の観点からは，DBA の遅延自体を削減せずとも，PON 区間を含めた End-to-End の遅延情報を正確に収集できれば，より詳細な制御が実現できる可能性がある．

PON を用いてエッジコンピューティングを行ったいくつかの従来研究が存在するが，リアルタイムモーション制御に用いている文献はあまり見当たらない [5]-[7]．参考文献[8]は，超多分岐 PON を用いて，工場のコントロールセンタと産業用機器を直接接続できる最大の許容分岐数について述べている．しかしながら，光パワーの損失に基づいて，最大 PON 分岐数が計算されており，産業用機器の制御性能を評価していない．参考文献[9]は，光アクセスエッジコンピューティングのための長距離 10G-EPON を用いたモータ制御のための低遅延アプリケーションを実証している．しかし，これまで DBA アルゴリズムによる遅延変化を考慮したモーション制御システムは提案されていない．

第 3 章では，Point-to-Point トポロジを対象に，アクセスエッジコンセプトを実現する手法を提案した．本章では，Point-to-Multipoint トポロジを対象とし，PON システムの DBA 遅延の影響を低減する手法を提案する．提案手法では DBA の遅延変化を考慮し PON の遅延モデルを用いて遅延を補償し，リアルタイムで高精度な制御を実現する．PON においての遅延は，DBA によるものが支配的なので，DBA の遅延を適切にモデル化することで詳細な制御が期待できる．ONU からの要求に応じて決定される DBA アルゴリズムによる遅延には変更を加えず，帯域使用効率を維持しながら制御性能を向上することができる．提

案手法は PON の設定パラメータを OLT から取得することで，設定パラメータによる遅延を推測し，この遅延値を用いることで遅延補償制御を行う．XGS-PON による実験で提案手法が従来手法では制御不可能な場合でも制御可能なことを示す．

4-2　PON システムの遅延モデルを用いた制御手法

4-2-1　　全体構成

PON を用いてアクセスエッジコンセプトを実現した場合の概念図を図 4-1 に示す．IoT 端末とエッジサーバは PON を介して直接接続され，工場と通信局舎は Point-to-Point 方式で接続される．PON によるカスケード接続は大量の IoT 端末を接続するために用いられる．PON によるアクセスエッジを実現するためには DBA の遅延変化に対応する必要がある．

提案手法では，遅延が発生している通信装置から遅延情報を取り出し，制御器に受け渡す．制御器は通信機器の設定情報を入力とする遅延モデルを保持する．このモデルにより制御器と制御対象間で発生する遅延を予測し，遅延補償に利用することができる．設定情報は IoT 端末数やトラフィック量に影響を受けるが，PON の設定値が変化した場合でも高性能な制御が実現できる．提案手法では XGS-PON を用いた [10].

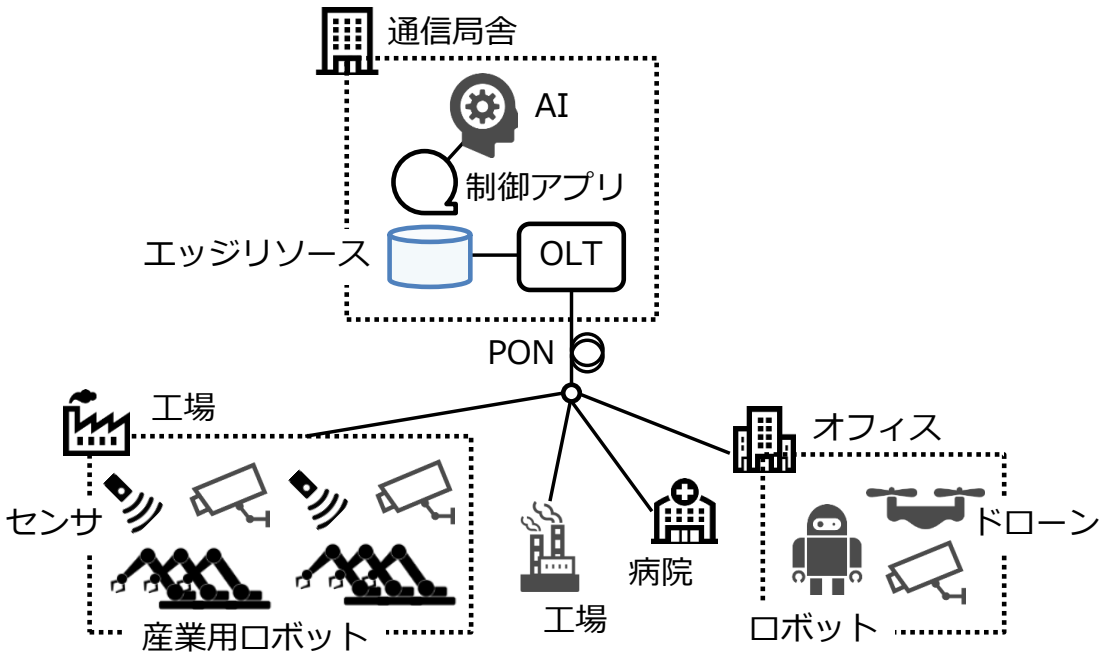


図 4-1　PON を用いたアクセスエッジ構成

4-2-2 PON 遅延モデル

XGS-PON では OLT から ONU への連続した Grant メッセージ間の最大時間である最大 Grant 周期 P_{MAX} の設定値が PON システムで生じる遅延を変化させる．上り方向の遅延は以下のように決定される．

$$t_{2MAX} = P_{MAX} + 2 \cdot \frac{T \cdot P_{MAX}}{L} \tag{4-1}$$

第 2 項は伝送遅延である．トラフィックレート T は上り方向のトラフィックレート，ラインレート L はネットワークの最大トラフィック量である．平均遅延はこの値の半分になると想定される．

$$t_m = t_1 + \frac{t_{2MAX}}{2} \tag{4-2}$$

このとき，下り方向の遅延 t_1 は上り方向の遅延 t_2 よりも十分小さく，また DBA 周期に関係なく固定値であるため，無視できるか，測定によって計測できる．

PON システムでは遅延は DBA によるものだけでなく，トラフィック要求によるキューイングでも発生する．トラフィック要求がトラフィックレートを超えると，余剰トラフィックは伝送バッファに蓄積され，伝送前にキューイング遅延が発生する．トラフィック量はランダムであるため，この動作もランダムに発生し，予測は困難である．

センサデータや制御値といった制御トラフィックは通常テキストデータであるため，1 台の IoT 端末の制御に求められるスループットは通常数百 kbps 以下である [11]．ここでは，すべての機器の制御トラフィックが 10 Gbps を超えず，トラフィック要求によるキューイング遅延は発生しない場合を考える．

しかしながら，実際のネットワークでは制御トラフィック以外にも監視用の映像データなどが流れることが想定される．全体のトラフィックが 10 Gbps を超過したとき，トラフィック要求によるキューイング遅延が発生し，制御トラフィックに影響する．この問題を解決するため，既存の産業用プロトコルではモーション制御に必要なデータのようなリアルタイムトラフィックをそれ以外のトラフィックよりも高い優先度を割り当て，このキューイング遅延を回避する [12]．この手法を提案手法に用い，トラフィック要求によるキューイング遅延を予測するのではなく回避を行う．制御データのデータサイズは通常大きくないため，効果的な結果が得られると想定される．

処理手順を図 4-2 と Algorithm 1 に示す．(1) まず，最大 Grant 周期やトラフィックレート，ラインレートといった PON の設定パラメータを OLT から取得し，PON 遅延モデルに受け渡す．(2) 遅延モデルではこれらの値を用いて上りの遅延値 t_{2MAX} を計算したのち，t_2 と下りの遅延値 t_1 を足し合わせることで全体の予測遅延値 t_m を決定する．この遅延値は制御値を算出する制御器に受け渡される．(3) 続いて制御器は IoT 端末から送られてきたセンサ

値 $x_0(t)$ を受信する．(4) 制御器はセンサデータ $x_0(t)$ と予測遅延値 t_m を用いて遅延補償制御値 $u(t)$ を算出する．このとき，例えば遅延補償制御手法としてスミス予測器などを用いる．(5) 制御器はこのデータの優先度を設定する．(6) 最後に，制御器は制御値 $u(t)$ を IoT 端末に送信する．遅延は最初に一度だけ受信され，(3)-(6)は制御を実現するために繰り返し実行される．

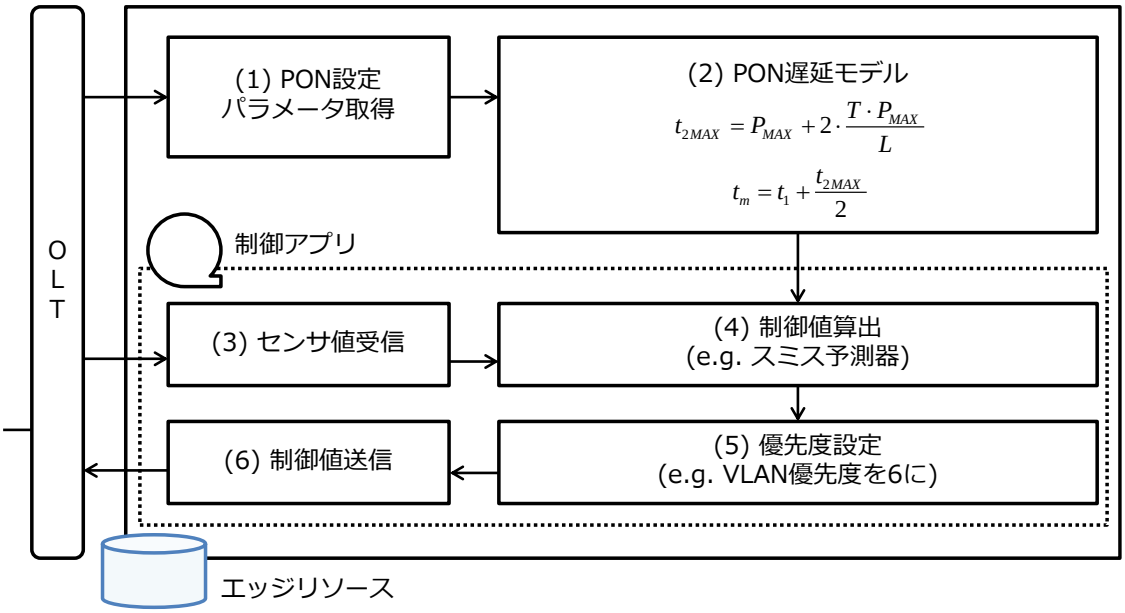


図 4-2　提案手法のアーキテクチャ

Algorithm 1 Algorithm for the proposed method

Input: $r(t), x_0(t), P_{MAX}, T, L, threshold$
Output: $u(t)$

$t_{2MAX} = P_{MAX} + 2 \cdot \frac{T \cdot P_{MAX}}{L}$
$t_m = t_1 + \frac{t_{2MAX}}{2}$
while $|r(t) - x_0(t)| < threshold$ **do**
    $x_1(t) = r(t) - x_0(t)$
    $x_2(t) = x_1(t) - x_3(t)$
    $u(t)$ is calculated by the controller with $x_2(t)$
    $x_3(t)$ is calculated by Smith predictor with t_m
end while
return $u(t)$

4-3　提案手法の実機検証

4-3-1　　実験構成

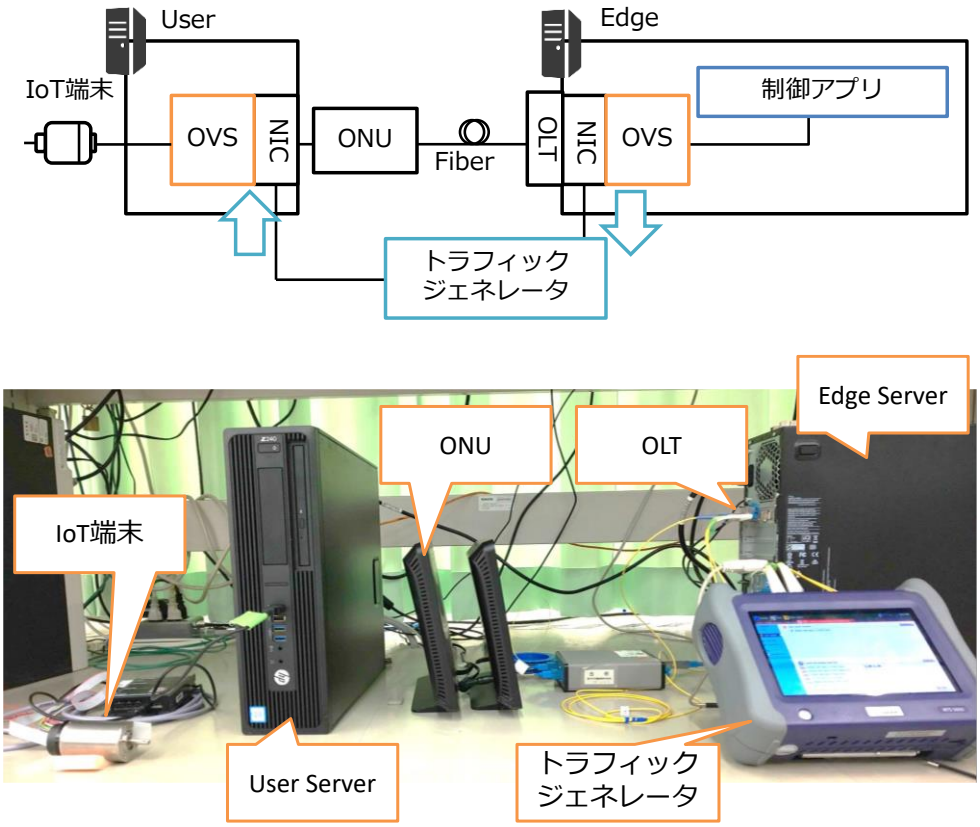


図 4-3　実験構成

提案手法の有効性を検証するため，モータを用いた実機実験系を構築した．図 4-3 に実験系を示す．ユーザ側と局舎側を模擬するため，2 台のサーバを用いた．それぞれのサーバスペックは Intel Xeon E3-1270 (3.80 GHz, 4 core) CPU で 32 GB メモリである．OS は Ubuntu 16.04 を用いた．OLT には Tibit Micro Plug OLT を，ONU には T&W ONU を用いた．トラフィックジェネレータは VIAVI MTS-5800 を用い，サーバとは光ファイバで接続した．スイッチ機能はソフトウェアスイッチである Open vSwitch (OvS) を用いて実現した．制御プログラムは局舎サーバにインストールした．モータは Maxon EC-max 40 を用い，ユーザサーバとは USB ケーブルと EPOS2 24/5 を介したシグナルケーブルで接続された．計算された制御値は電流値に変換され，電流制御を行った．

以下の 3 つの手法が比較検証された．

- 手法 1：スミス予測器の想定遅延値を 10 ms の固定値とした PI 制御
- 手法 2：PON 遅延モデルの最大 Grant 周期の設定を固定値の 20 ms に設定したスミス予測器を用いる PI 制御

- 提案手法：PON 遅延モデルの最大 Grant 周期の設定を OLT の設定値と連携したスミス予測器を用いる PI 制御

手法 1 は PON 区間を含めた End-to-End の遅延値が 10 ms の固定値としたもので，手法 2 は遅延モデルで用いる最大 Grant 周期を最大値に設定したものである．

次の 2 つの実験を行った．

- 実験 1：無負荷ネットワーク環境
- 実験 2：40 Mbps の負荷環境

モータ特性は次の式で与えられる．数値はモータの仕様書に基づいて決定した．

$$\hat{G}_P(s) = \frac{195000}{s^2 + 114s + 150} \tag{4-3}$$

PI 制御のゲインはパラメータチューニングによって決定した．

$$G_C(s) = K_P E(s) + K_I E(s)/s \tag{4-4}$$

$$K_P = 0.001 \quad K_I = 0.0016 \tag{4-5}$$

制御周期は 2 ms とし，4000 qc の位置制御を行った．整定時間は目標値の±5%に到達したときの時間とした．

4-3-2　　実験結果

実験 1

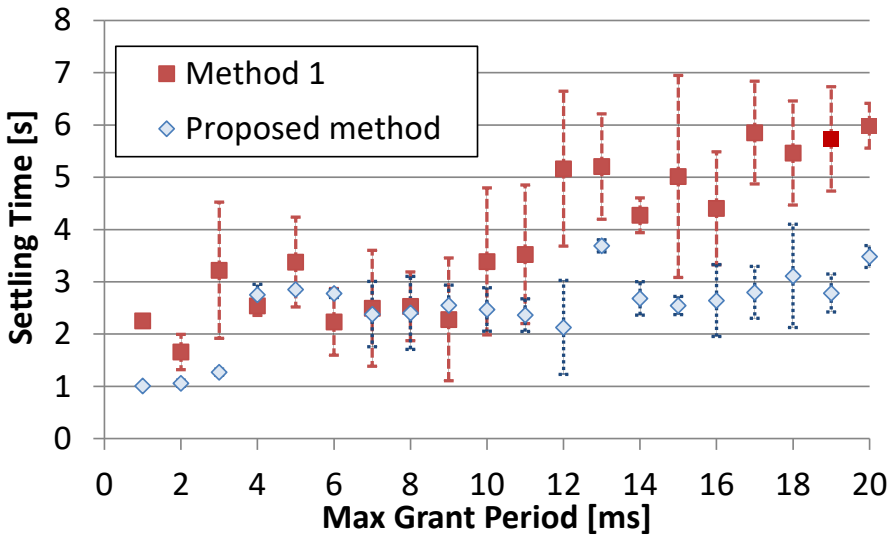


図 4-4　手法 1 と提案手法の結果比較

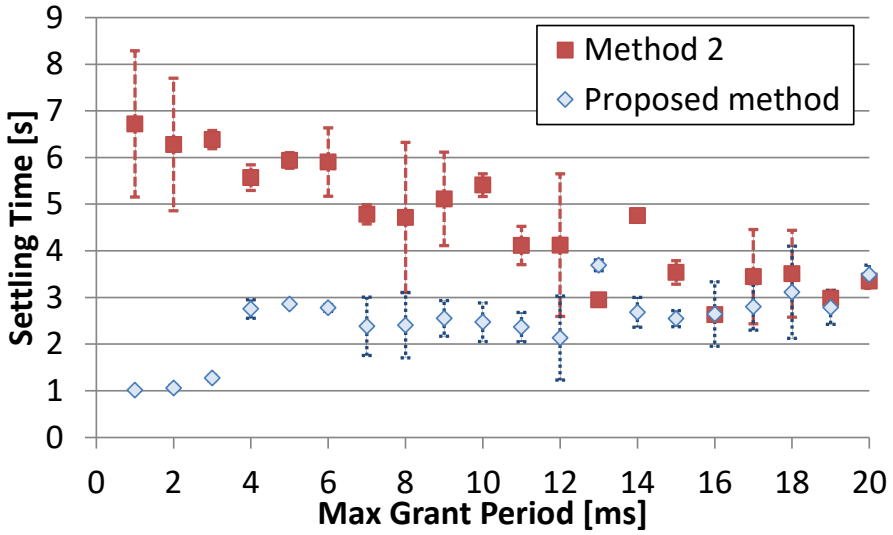


図 4-5　手法 2 と提案手法の結果比較

最大 Grant 周期の変化による遅延変化が制御性能に与える影響を評価するため，無負荷ネットワーク環境で整定時間を計測した．提案手法を手法 1, 2 と比較した結果を図 4-4, 4-5 にそれぞれ示す．それぞれの結果は 4 回試行し，標準偏差も示している．図 4-4 では最大 Grant 周期が 12 ms より小さいとき，手法 1 と提案手法の間に有意な差は見られない．しかし，12 ms を超えると，手法 1 では整定時間が長くなる．これは想定遅延値と実際の遅延値の間の差異に対応できていないことから生じると考えられる．一方で，提案手法では遅延モデルが遅延値の変化に対応できるため，最大 Grant 周期によらず同等の制御性能を維持できることがわかる．

図 4-5 に示すように手法 2 では遅延モデルを最悪値に設定しているため，すべての最大 Grant 周期において制御が成功していることがわかる．しかし，最大 Grant 周期が 1 ms のとき，実際の遅延値に対応していないため，提案手法と最大 6 s の整定時間の差異が生じている．設定遅延値と実際の遅延値の間の差異が小さくなるにつれ，提案手法との整定時間の差異も小さくなるが，実際の遅延に合わせて遅延モデルを設計することでより詳細な制御が可能になる．これらの実験から提案手法では DBA 周期の変化に柔軟に対応でき，詳細な制御が実現できることを確かめられた．

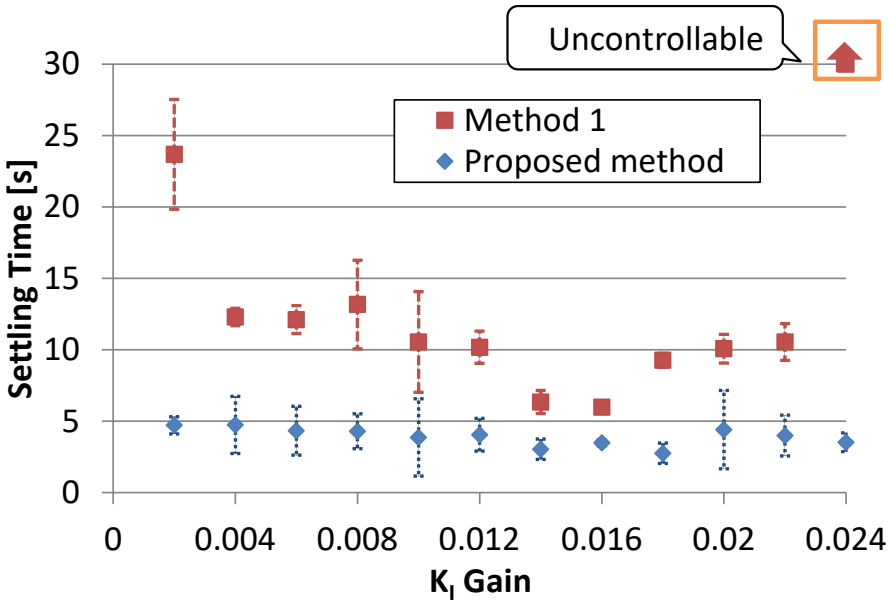


図 4-6　手法 1 と提案手法の Ki ゲインを変化させたときの結果比較

続いて，制御パラメータにかかわらず提案手法が有効であることを検証するため，ゲインを変化させた場合のそれぞれの手法の制御結果を調査する．図では，K_P ゲインを固定し，K_I ゲインを 0.002 から 0.024 まで 0.002 刻みで変化させた場合の，手法 1 と提案手法の整定時間の比較を行った．一般に K_I ゲインが小さいと，目標値との誤差を修正する効果が弱まり，K_I ゲインが大きすぎると誤差に敏感になり振動的な挙動を示す．手法 1 では K_I ゲインが 0.002 のとき整定時間は 23 s であったが，K_I ゲインが 0.024 のとき発散して収束しなかった．一方で，提案手法では遅延を考慮に入れているので，どのような K_I ゲインを選んでも手法 1 よりも高速に制御できることを確認できた．

これらの実験では制御アプリケーションがセンサデータを受け取ってから，制御値を送信するまでの平均的な処理時間は 69 us であった．この値は制御周期とした 2 ms の 28 分の 1 の値であるので，提案アルゴリズムは 28 倍遅い性能の安価な検証プラットフォームでもリアルタイム性能を保証できると期待できる．

実験 2

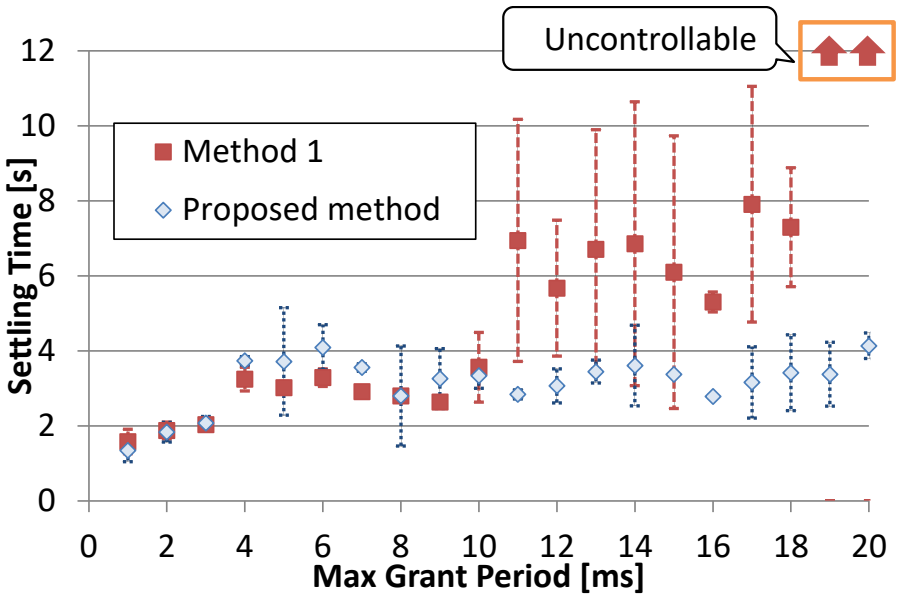


図 4-7　手法 1 と提案手法の 40 Mbps 帯域下での結果比較

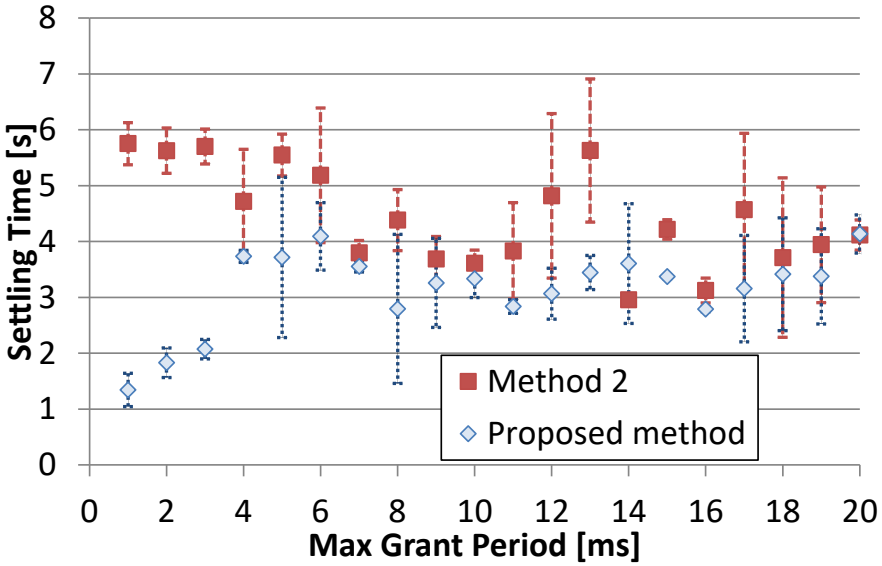


図 4-8　手法 2 と提案手法の 40 Mbps 帯域下での結果比較

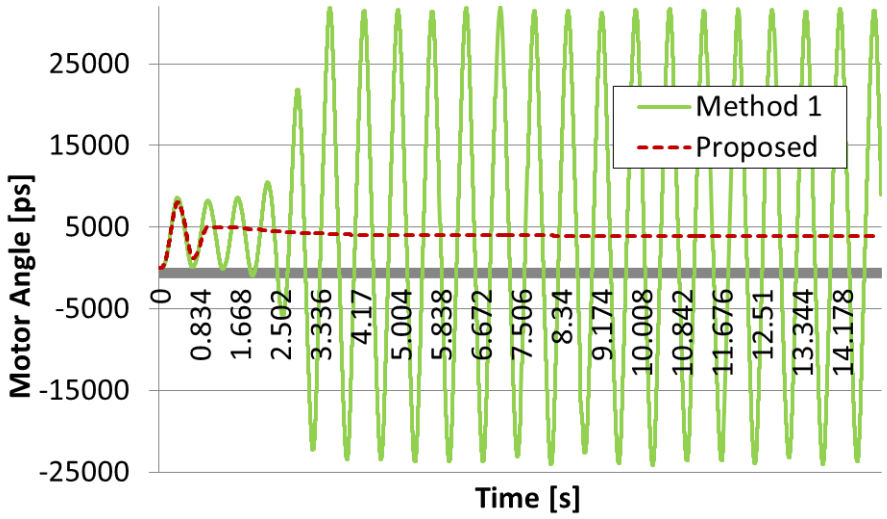


図 4-9　手法 1 と提案手法の 40 Mbps 帯域下で最大 Grant 周期が 20 ms のときの時刻歴

高負荷環境での提案手法の有効性を検証するために，制御性能を測定し，提案手法である PON の遅延モデルの出力と実際の遅延の差異を評価した．図 4-7, 4-8 に示すように，提案手法を手法 1, 2 と比較した．また，図 4-9 は最大 Grant 周期を 20 ms としたときの手法 1 と提案手法の結果を比較した時刻歴である．図 4-7 では最大 Grant 周期が 11 ms 以下では手法 1 と提案手法の間に大きな差異は見られなかった．しかし，最大 Grant 周期が 11 ms を超えると，手法 1 では整定時間が伸び，19, 20 ms では制御することができなかった．これは上りトラフィックの遅延増加によるものである．実際の工場ではカメラ等からの映像トラフィックなど制御以外のデータが同時に発生することがあるため，これらの遅延に柔軟に対応することが必要である．PON 遅延モデルを用いることで提案手法は従来手法では制御できなかった場合でも制御できることを確認した．

図 4-9 に示すように，最大 Grant 周期が増加するに連れ，手法 2 と提案手法の整定時間の差は減少していくが，最大 Grant 周期が 1 ms のとき，整定時間の差は最大で 4 s ほどになった．手法 1 と異なり，手法 2 はある程度の制御が実現できたが，提案手法はより整定時間が短かった．

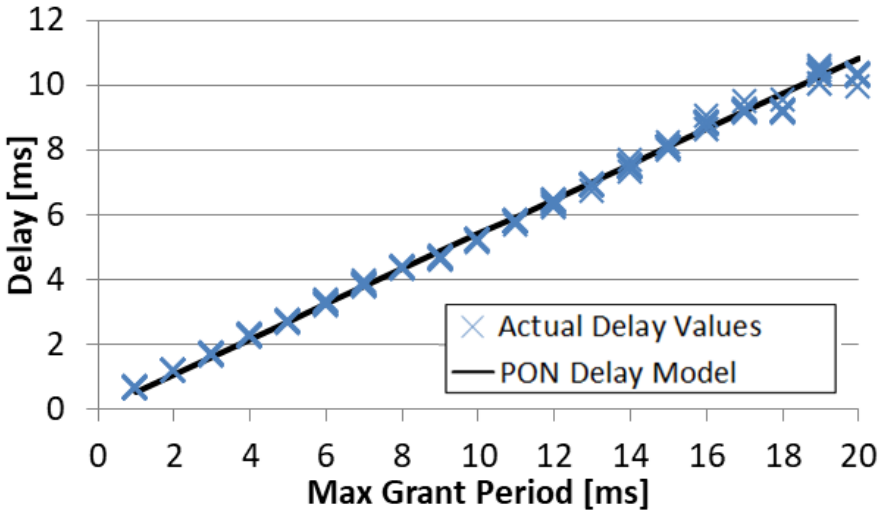


図 4-10　実際の遅延値と遅延モデルの出力の結果比較

図 4-10 は PON の遅延モデルの出力と 10 回の試行から得られた実際の遅延値を示している．予測不可能な遅延は制御性能を劣化させ，整定時間の差異として反映される．この実験では PON 区間で生じる遅延はトラフィックジェネレータで計測し，PON 遅延モデルと比較した．結果より遅延モデルと実際の遅延はほぼ等しく，提案の遅延モデルが遅延を正しく予測できていることが確認できた．

提案手法は最大 Grant 周期が 20 ms のときも 4 s 以内の整定時間を達成している．これらの実験から提案手法は要求を満たすために設定された DBA 周期を変更することなく，帯域利用効率を維持しながら，制御性能を向上できることを確認した．

4-4　結論

PON 遅延モデルに基づいて，PON システムで予想される様々な遅延値を能動的に補償するリアルタイムモーション制御手法を提案した．この PON 遅延モデルは DBA 周期によって決定される遅延を予測し，制御に用いる．提案手法により，アクセスエッジコンピューティングで大量の IoT 端末の高精度な制御を実現できる．モータを用いた実機実験を構築し，提案手法の有効性を検証した．実験により，提案手法は PON の設定パラメータによらず正確な制御性能を保ち，従来手法では遅延変化に対応できず制御が難しい場合でも，4 s 以内のモータ制御を満たすことを確認した．これらにより，PON システムにおけるエッジコンピューティングを用いたファクトリーオートメーションのためのリアルタイムモーション制御の実現可能性を示した．

4-5 参考文献

[1] L. Ruan, M. P. I. Dias and E. Wong, “Machine Learning-Based Bandwidth Prediction for Low-Latency H2M Applications,” IEEE Internet of Things Journal, vol. 6, no. 2, pp. 3743-3752, 2019.

[2] R. Raad, E. Inaty and M. Maier, “Improving Latency and Jitter Performance in CDMA-Based Next-Generation Ethernet Passive Optical Networks for 5G Applications,” 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC), pp. 1-8, 2019.

[3] K. Kim, K. Doo, H. H. Lee, S. Kim, H. Park, J. Oh and H. S. Chung, “High Speed and Low Latency Passive Optical Network for 5G Wireless Systems,” Journal of Lightwave Technology, vol. 37, no. 12, pp. 2873-2882, 2019.

[4] T. Tashiro, S. Kuwano, J. Terada, T. Kawamura, N. Tanaka, S. Shigematsu and N. Yoshimoto, “A novel DBA scheme for TDM-PON based mobile fronthaul,” OFC 2014, pp. 1-3, 2014.

[5] T. Suzuki, S. Kim, Y. Koyasako, J. Kani and J. Terada, “Demonstration of Adaptive Image Transmission That Meets Various Application Requirements in 10G-EPON,” IEEE Access, vol. 8, pp. 186433-186440, 2020.

[6] A. Helmy and A. Nayak, “Toward parallel edge computing in long-reach PONs,” IEEE/OSA Journal of Optical Communications and Networking, vol. 10, no. 9, pp. 736-748, 2018.

[7] A. M. Alqahtani, S. H. Mohamed, T. E. H. El-Gorashi and J. M. H. Elmirghani, “PON-Based Connectivity for Fog Computing,” 2020 22nd International Conference on Transparent Optical Networks (ICTON), pp. 1-6, 2020.

[8] Z. Guo, K. Zhang, H. Xin, M. Bi, H. He and W. Hu, “An optical access network framework for smart factory in the industry 4.0 era supporting massive machine connections,” 2017 16th International Conference on Optical Communications and Networks, pp. 1-3, 2017.

[9] R. Kubo, T. Michigami, K. Yamada and M. Yoshino, “Demonstration of Wide-Area Networked Motion Control over Long-Reach 10G-EPON Based on Optical Access Edge Computing,” 2019 24th OptoElectronics and Communications Conference (OECC) and 2019 International Conference on Photonics in Switching and Computing (PSC), pp. 1-3, 2019.

[10] “G.9807.1 : 10-Gigabit-capable symmetric passive optical network (XGS-PON),” ITU-T Recommendation G.9807.1, 2016.

[11] A. B. D. Kinabo, J. B. Mwangama and A. A. Lysko, “An Overview of Time-Sensitive Communications for the Factory Floor,” 2021 IST-Africa Conference (IST-Africa), pp. 1-9, 2021.

[12] “PROFINET System Description - Technology and Application,” PROFIBUS Nutzerorganisation e.V., 2018.

第 5 章　仮想化システムを用いたネットワーク設定の最適化

5-1　はじめに

アクセスシステムにおいて，OLT を収容し，ユーザ接続の集約ポイントである通信局舎は，そのユーザとの距離の近さからエッジコンピューティングの最適な場所のひとつである．通信局舎をデータセンタに変えるコンセプトの研究がこれまでに行われてきた [1], [2].

SEBA はこのコンセプトを実現する仮想化基盤である [3]．SEBA は制御と管理機能を汎用サーバ上にソフトウェア実装することで通信局舎をデータセンタのように扱うことができるため，エッジコンピューティングのための計算リソースを用意できる．いくつかのオペレータで SEBA の導入が進んでおり，例えば Telefonica はアプリケーションを通信局舎に配置することでエッジコンピューティングシステムとして利用することを検討している [4]．しかしながら，これまで SEBA の仮想化リソースを用いたエッジコンピューティングによるリアルタイム制御に関する研究は行われていない．

4 章で提案した手法は，独自仕様のプラットフォームを用いており，遅延の影響がより重大で実用的な Point-to-Multipoint 方式の仮想化基盤は用いていなかった．アクセスエッジコンセプトがこの問題に対処し，仮想化された構成で正確なリアルタイムのモーションコントロールを実現できることは重要な課題である．特に工場内の IoT 端末の制御はそれぞれのデバイスのトラフィックは小さいながら，大量のデバイスと接続することが想定されるため，PON システムの利点が十分に活用される．DBA 周期に遅延は支配されるが，最小の遅延は数十マイクロ秒であり [5]，これは 0.25-10 ms のファクトリーオートメーションの遅延要件を満たしうる．このような制御を達成するためには，モーション制御中にネットワーク設定を変更する必要がある．

これまで受信データに基づいてネットワーク設定を変更する研究は様々行われてきた．OLT と ONU 間のトラフィックを低減するため，ONU 配下にある監視カメラといった IoT 端末から送られてくるメディアデータの圧縮率を解析し，帯域割当を決定することができる．参考文献[6]では，メディアデータの品質を計算し，画像品質評価指標であるピーク信号対雑音比 (Peak Signal to Noise Ratio, PSNR) が閾値より大きくなるように伝達関数を最小化する．しかしながら，この手法はもともと画像品質に関するパラメータを設定するためのものであり，低遅延リアルタイムモーション制御を目的としたものではない．参考文献 [7], [8]ではモーション制御のためのローカルエリアネットワークのパラメータを設計する方法が提案されているが，これまで光アクセスシステムを対象としたものは存在していない．

この章では，仮想化システムである SEBA ベースの 10G-EPON 上でのエッジコンピューティングを用いたリアルタイムモーション制御を実証する．提案システムでは，SEBA とモーション制御アプリケーションを連携させるドライバを用いる．ドライバは SEBA から取得したネットワーク設定と制御結果を連携し，過去の値と比較することで，ネットワー

ク設定による制御の影響を分析し，必要に応じてネットワーク設定を更新する．提案手法ではユーザのアプリケーションと通信事業者のネットワークを，SEBA の柔軟性を活用することで統合する．これにより，ネットワークパラメータは遠隔モーション制御に特化した仮想ネットワークを構築するために，制御アプリケーションの要求から直接決定される．このような，制御アプリケーションの要求をもとにネットワーク端末の構成を自動的に変更するような手法は，これまで提案されていない．この手法により，ユーザがネットワーク設定を意識することなく，モーション制御に最適なネットワークを構築できる．10G-EPON の実験系を構築し，提案手法の有効性を検証する．

5-2　ドライバを用いたネットワークパラメータの最適化

5-2-1　　全体構成

モーション制御における遅延の対応には主に 2 つのアプローチがある．ひとつはモーション制御を行うアプリケーション側で遅延の影響を補償する方法 [9]-[14]，もうひとつはネットワーク側で遅延の影響を低減する方法である．通信機器の設定を変更することでアクセスネットワークに起因する遅延を低減できる．

提案システムを図 5-1 に示す．提案システムでは SDN コントローラ経由で得られたネットワーク設定とモーション制御アプリケーションから得られた制御結果を解析し，最適なネットワーク設定を計算し，低遅延モーション制御や大量の IoT 端末の接続，高負荷ネットワークといった，ネットワークの利用状況に応じて更新する．このシステムは様々なネットワークを構築するために普遍的に適用できる．

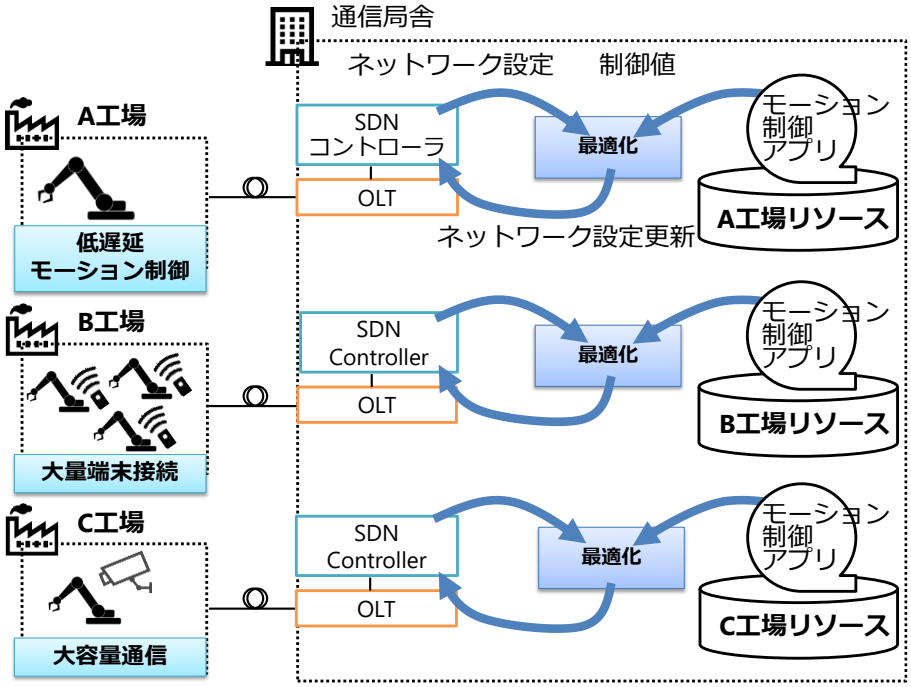


図 5-1　提案手法の概念図

図 5-2 に構成を示す．提案手法は SEBA とモーション制御アプリケーションを連携するドライバを用い，遠隔モーション制御に特化した仮想ネットワークを構築する．SEBA とドライバが動作するエッジコンピューティングプラットフォームは通信事業者が管理する形で局舎内に存在する．モーション制御アプリケーションはユーザが管理できる形でエッジサーバ上に存在する．通信事業者は API をユーザに提供し，整定時間といった制御結果を収集する．これらの値は受信データを処理することによって得られ，ユーザのみが持っている情報である．ドライバは制御アプリケーションからの制御結果と SEBA からのネットワーク設定を受信する．これらの値を結びつけ，過去の値と比較することで，ネットワーク設定が制御性能に影響を及ぼす影響を分析することができる．制御性能を向上させるネットワーク設定が特定され，テクノロジープロファイルが書き換えられる．書き換えられたテクノロジープロファイルを SEBA に入力し，OLT は最適な設定を採用する．

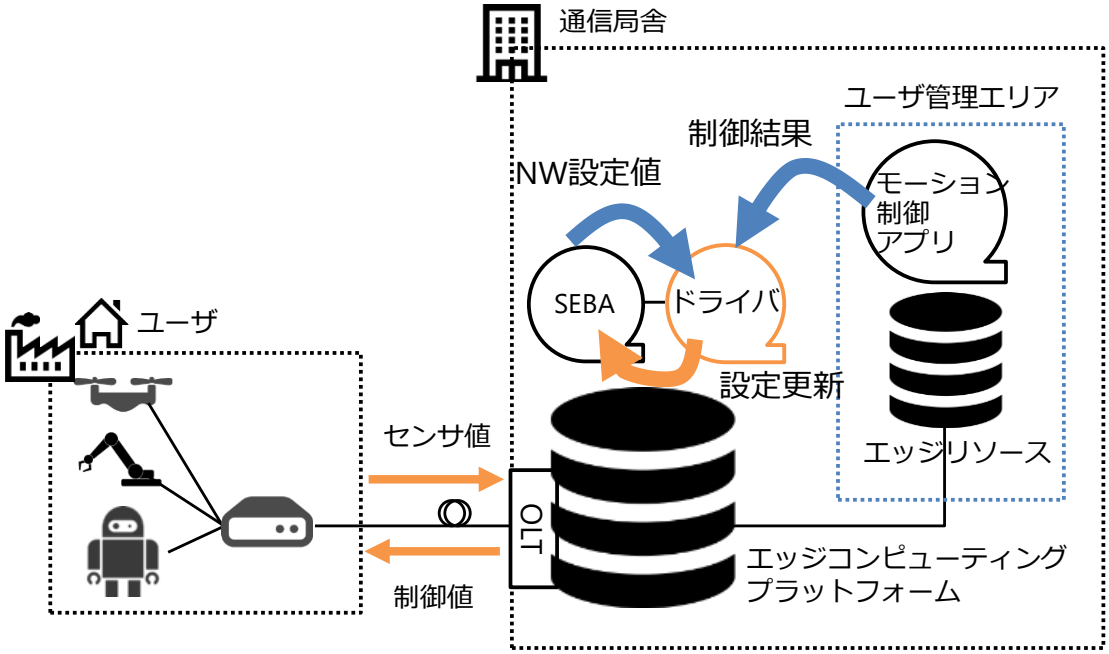


図 5-2　提案構成

5-2-2　　シーケンス

ここでは例として，PON システムのパラメータである DBA 周期を変更するネットワーク設定を想定する．図 5-3 にこのシステムのワークフローを示す．モーション制御が繰り返し実行される状態を想定する．モーション制御が実行されると，(1) ドライバがモーション制御開始要求を受け取り，(2) ユーザの制御下にある制御アプリケーションから前回のモーション制御の整定時間と行った制御結果を取得する．なお，最初の制御である場合は何も受信しない．(3) ドライバは NEM から OLT の DBA 周期といった現在のパラメータを取得する．(4) 取得したネットワーク設定と制御結果を紐づけ，過去のデータと比較することで，ネットワーク設定の変化による制御結果の差異を特定可能である．これらの結果に基づき，ネットワーク設定が変更される．このとき，PON の遅延モデルは 4 章のものを用いる．(5) テクノロジープロファイルは新しいネットワーク設定を用いて書き換えられ，NEM に投入される．(6), (7) その後，ドライバがテクノロジープロファイルを NEM に投入する前に，OLT の無効化と削除処理が実行される．(8)-(17) 起動ワークフローにおいて，SEBA の準備が行われるが，ONU 接続処理はスキップされる．(18) 設定パラメータが OLT に反映されると，制御アプリケーションに通知する．(19), (20) その後，制御アプリケーションは位置制御や速度制御といったモーション制御を実行する．

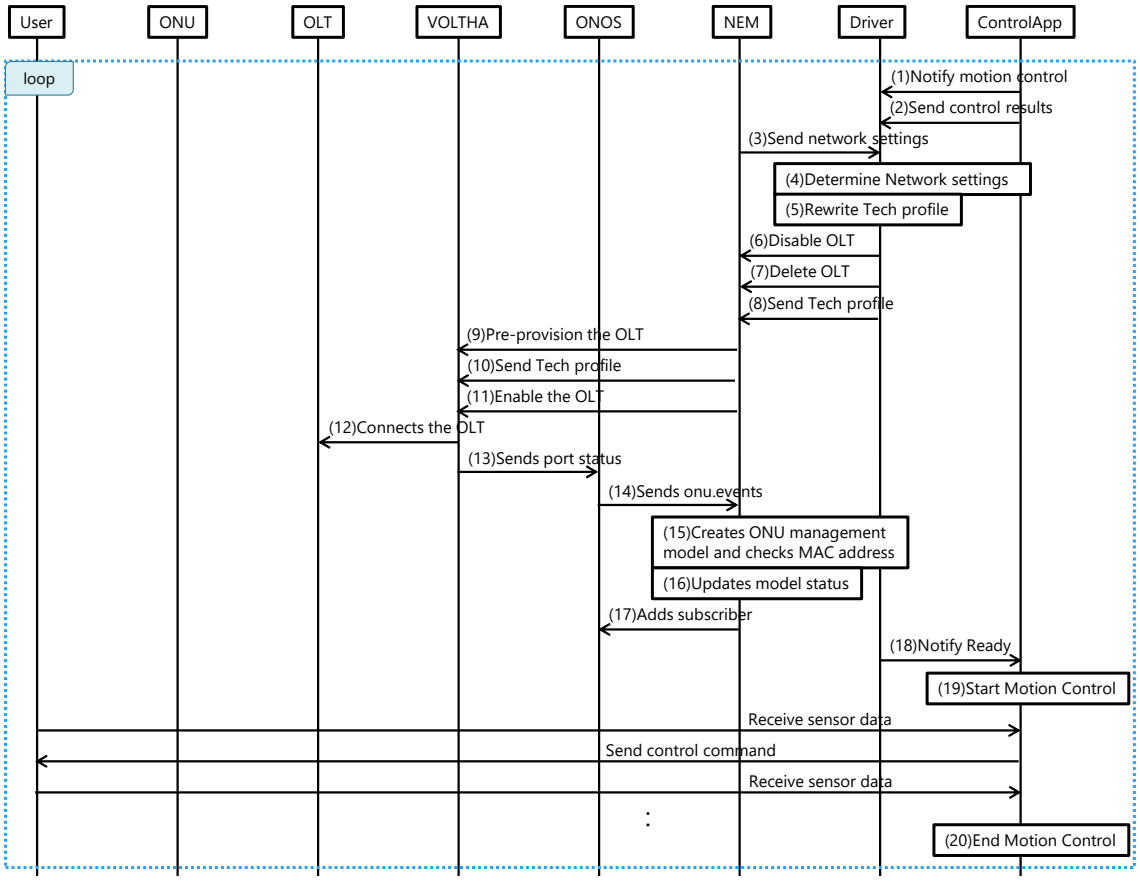


図 5-3　提案システムのフローチャート

5-2-3　　評価関数

アプリケーションの整定時間は DBA 周期によって変動する．評価関数 $f_{(DBA)}$ は下記のように表現される．

$$f_{(DBA)} = \left(t_{(DBA)} - t_r\right)^2 \tag{5-1}$$

$t_{(DBA)}$ は DBA 周期ごとの整定時間，t_r は整定時間の目標値である．この評価関数は二次関数であるため，最小値が存在する．この評価関数を最小化する DBA 周期を見つけることが必要であり，それはアプリケーションごとに異なる．そのため，提案した操作を繰り返すことで，ネットワーク設定の決定プロセスにおいて，勾配降下法を用いることで，目標整定時間を満たすモーション制御のための最適な DBA 周期を決定することが可能となる．次の DBA 周期 DBA_{n+1} は現在の DBA 周期 DBA_n と重み W から決定される．勾配降下法は本研究で想定する条件下では解を導くのに十分実用的であり，計算複雑性も低い．

$$DBA_{n+1} = DBA_n - W \cdot \left(t - t_r\right) \cdot DBA_n \tag{5-2}$$

5-3　提案手法の実機検証

5-3-1　　実験構成

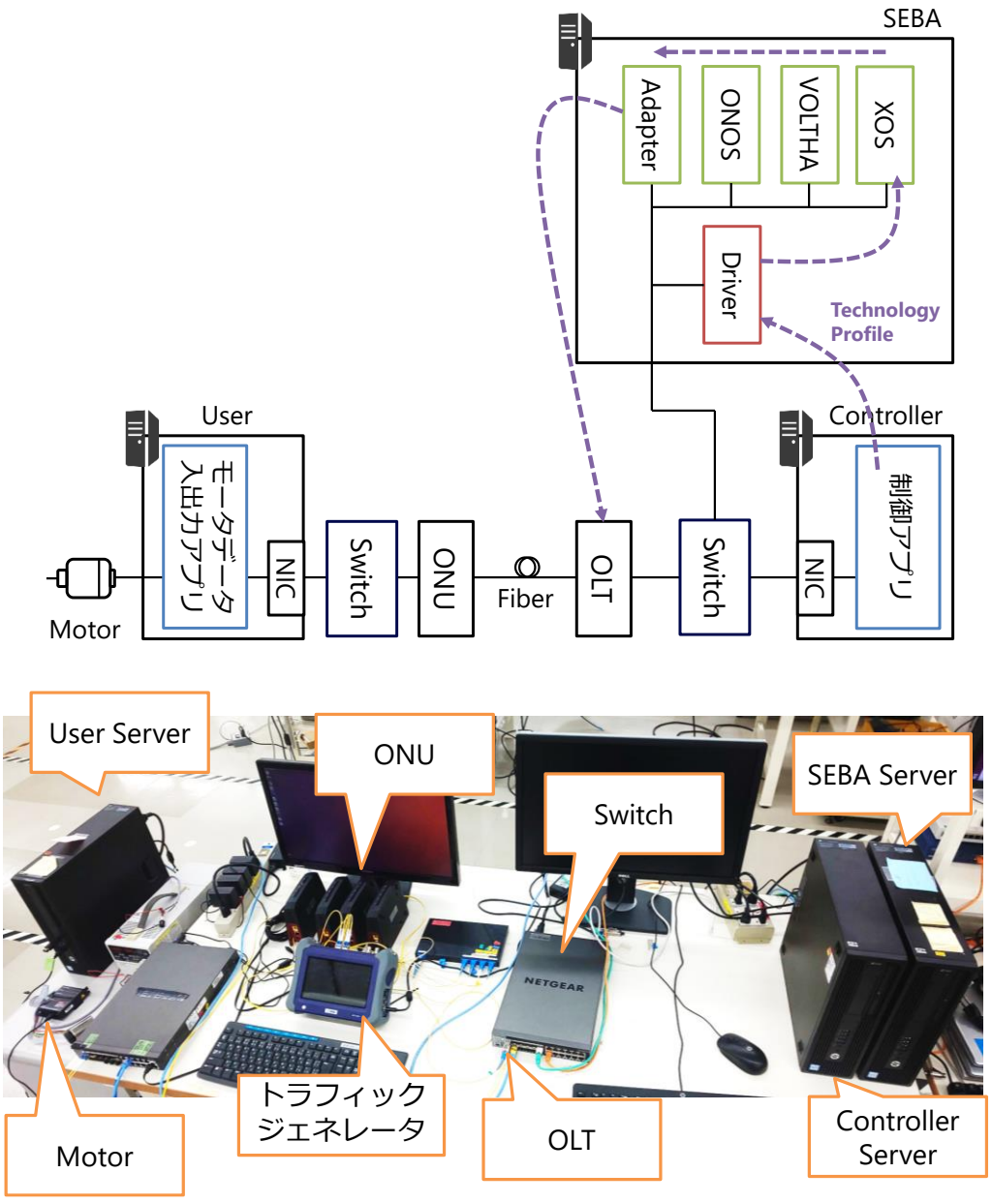


図 5-4　実験構成

ネットワーク設定がモーション制御アプリケーションからの要求通りに，ドライバを介して SEBA に正しく投入されるか実験により検証した．加えて，ネットワーク設定が正しくハードウェアに投入され，ネットワーク設定がモーション制御のための最適な値に収束するかどうか検証した．図 5-4 に示すような一般的なモータ実験系を構築した．2 台のサー

バは局舎側を模擬するために用いられ，1 台は制御アプリケーションを実行し，もう 1 台は SEBA とドライバを実行した．1 台のサーバでユーザ側を模擬し，モータデータを入出力するアプリケーションを実行した．このアプリケーションは DBA 周期とは独立してモータに対して周期的にデータを入出力する．サーバスペックは全て，Intel Xeon E3-1270 (3.80 GHz, 4 core) CPU の 32 GB メモリである．OS は Ubuntu 16.04 を用いた．OLT は Tibit Micro Plug OLT を用い，ONU は Furukawa Electric ONU を用いた．スイッチは NETGEAR M4300-12X12F である．モータは Maxon EC-max 40 を用いた．トラフィックジェネレータは VIAVI MTS-5800 でサーバと光ケーブルで接続した．モータとユーザサーバは USB ケーブルと EPOS24/5 を介してシグナルケーブルで接続された．計算された制御値は電流制御を行うため，電流値に変換される．予測遅延値を 10 ms としたスミス予測制御の PI 制御を行った．

SEBA サーバ上の SEBA 実験系は Docker と Kubernetes を用いて構築した．EPON を用い，ntt-workflow ドライバが OLT アダプタとして用いられた [15]．テクノロジープロファイル内では DBA 周期は“nominal interval”というパラメータで表され，ドライバを用いて書き換えられた．

ドライバは Python を用いて開発した．ドライバはサーバとして起動し，モーション制御アプリケーションから開始要求を受け取ると，制御結果とネットワーク設定から YAML ファイルの形でテクノロジープロファイルを作成する．ドライバは既存のテクノロジープロファイルの削除と新しいテクノロジープロファイルの投入・有効化に curl コマンドを用いる．その後，モーション制御アプリケーションに処理完了を通知する．

初期設定時の DBA 周期を 20 ms，重み W を 0.1 とした．目標整定時間を 3.0 s とした．

モータ特性は次式で与えられる．数値はモータのスペックシートをもとに決定した．

$$\hat{G}_P(s) = \frac{195000}{s^2 + 114s + 150} \tag{5-3}$$

PI 制御のゲインはパラメータチューニングによって決定した．

$$G_C(s) = K_P E(s) + K_I E(s)/s \tag{5-4}$$

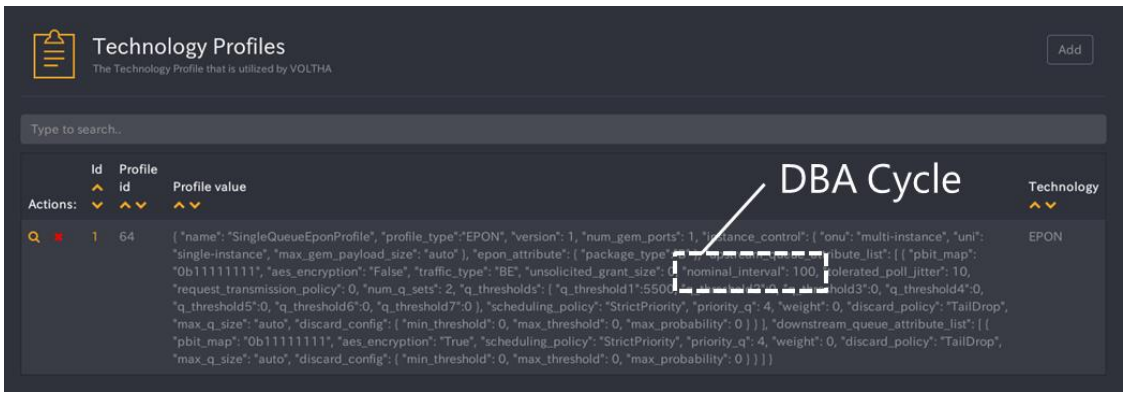
$$K_P = 0.001 \quad K_I = 0.0016 \tag{5-5}$$

制御周期を 2 ms とし，4000 qc を目標値とする位置制御を行った．上りトラフィックを 40 Mbps とした．整定時間は目標角度の±5%以内に収まったときの時間とする．

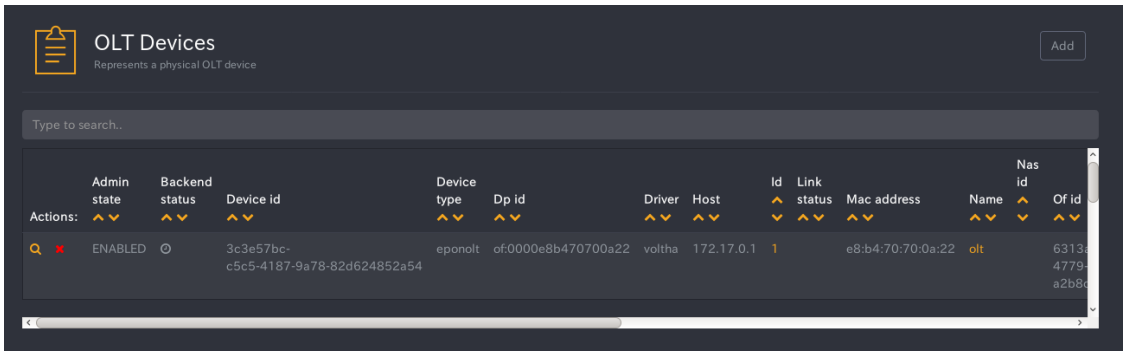
5-3-2　　実験結果

(a) モーション制御前のテクノロジープロファイル画面

(b) モーション制御前の OLT デバイス画面



(c) モーション制御後のテクノロジープロファイル画面



(d) モーション制御後の OLT デバイス画面

図 5-5　SEBA の動作画面

図 5-5 はネットワーク設定がモーション制御アプリケーションからドライバを介して SEBA に送信されたときの SEBA の挙動を示している．図 5-5 (a), (b)はそれぞれモーション制御開始前のテクノロジープロファイルと OLT の SEBA 上の画面を示している．テクノロ

ジープロファイルはまだ投入されていない．図 5-5 (c), (d)はネットワーク設定を受信したあとの画面を示している．DBA 周期が書き換えられたテクノロジープロファイルが適切に投入されている．テクノロジープロファイルを反映するため，OLT と ONU が再有効化されたことを確認した．

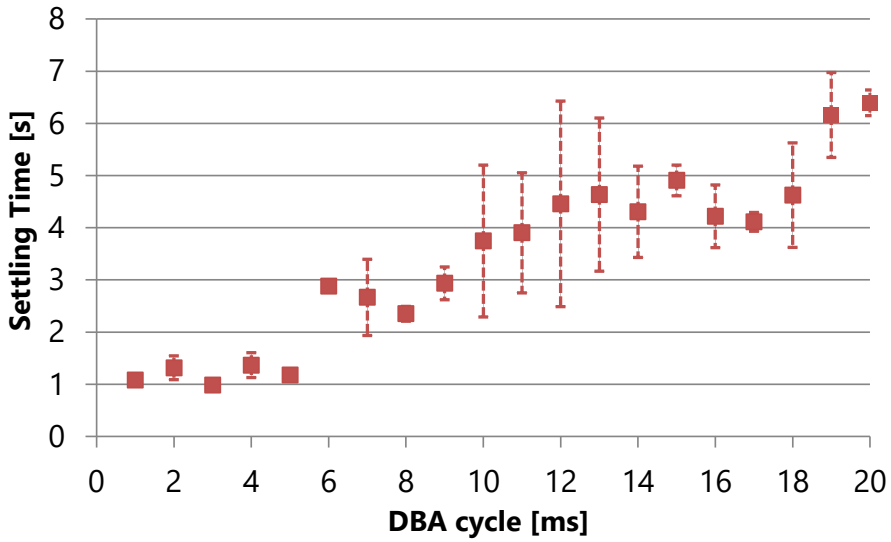


図 5-6　モーション制御結果

モーション制御の結果を図 5-6 に示す．この図は本実験系での DBA 周期と整定時間の関係を示す．各結果は 4 回の測定の平均値であり，標準偏差も示している．DBA 周期が長くなるにつれ，整定時間が長くなっている．これは想定遅延時間である 10 ms と実際の遅延値の差異に対応できないため発生すると考えられる．例えば本実験系では 9 ms のような適切な DBA 周期を選択することでより詳細な制御が期待できる．

DBA 周期とモータデータ取得周期は非同期である．モータからのデータ取得タイミングと DBA 周期が一致しているとき，整定時間は短くなるが，一致していないとき，整定時間は長くなる．DBA 周期が 15-20 ms のとき，データ収集周期に比べて十分大きいため，整定時間は常に長く，ばらつきが小さい．

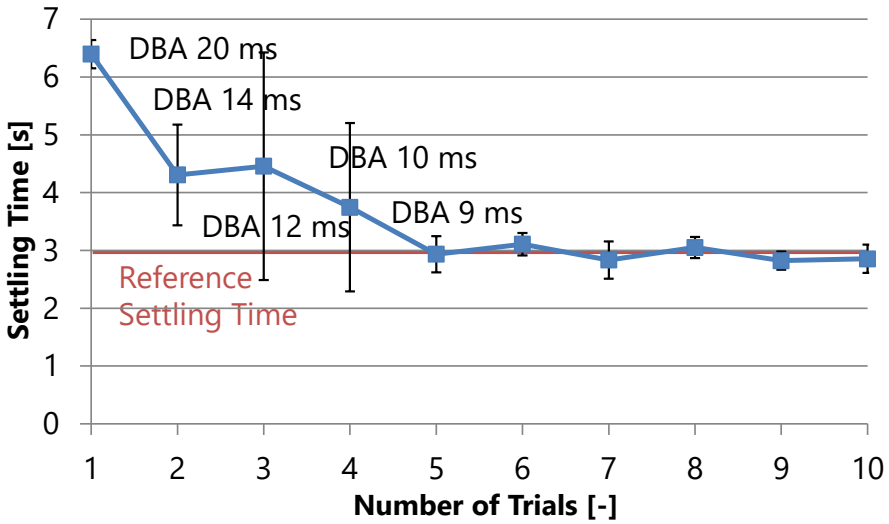


図 5-7　試行回数と整定時間の関係

図 5-7 はドライバが最適な DBA 周期を見つけるまでの試行回数と整定時間の結果を示す．提案手法は勾配降下法を用いて目標整定時間を満たすようにモーション制御結果からネットワーク設定を最適化する．最適な DBA 周期は 5 回の試行ののち，9 ms に収束した．この場合では，提案手法は，DBA 周期が 20 ms で最適化が行われていない場合と比べて，整定時間を 53 %低減することができた．

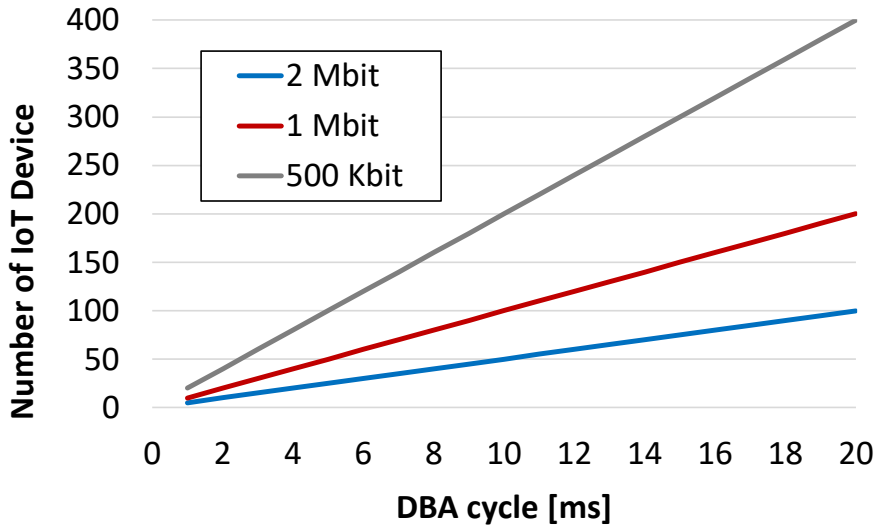


図 5-8　DBA 周期と接続可能な IoT 端末数の関係

図 5-8 は DBA 周期に対する，データサイズごとの接続可能な IoT 端末数の計算結果を示している．それぞれのデバイスは DBA 周期ごとに特定のデータサイズを送信することを想定する．接続できる端末数は次式で表される．

$$NumberOfDevice = \frac{LineRate \cdot DBAcycle}{DataSize} \tag{5-6}$$

それぞれのデバイスが DBA 周期ごとに 1 Mbps のデータを送信し，ラインレートが 10 Gbps とする．データが 1 Mbit を占有し，DBA 周期が 20 ms のとき，DBA 周期ごとの送信可能なデータサイズは 200 Mbps であるため，200 台接続可能である．DBA 周期が 9 ms のとき，90 台接続可能である．結果から DBA 周期が小さい時，遅延は小さくなるが，接続可能な IoT 端末数も制限されることがわかる．工場には大量の IoT 端末が存在するため，制御性能を保ちながら接続可能な IoT 端末数を増やすことが必要である．

本検討では，IoT 端末との接続に有線ネットワークを想定し，無線ネットワークは想定していない．工場では通常，信頼性の観点から有線ネットワークが使用されているが，無線ネットワークを使う場合には無線遅延が追加される可能性がある．無線遅延が数ミリ秒オーダであるとき，提案手法は DBA 周期を短くすることでこの遅延の増加に対応できる．しかし，無線遅延が大きい場合は，ネットワーク構成を変更する必要がある．

また，このシステムは工場で用いることを想定している．ネットワーク負荷が変動してもネットワーク容量の範囲内であれば，遅延は揺らがず，このシステムを用いることができる．ネットワーク容量を超えてネットワーク負荷が変動し，遅延が増加するときでも，QoS 制御により制御データの優先度を向上させることで，その他のデータに大きな遅延やパケットロスが発生しても制御データには追加遅延が発生しないようにし，本手法を適用できる．

これらの結果から，システム全体が連携することで，モーション制御のために特化した仮想ネットワークを構築できることを確認した．モーション制御アプリケーションからの要求に基づいてテクノロジープロファイルが作成され，ネットワーク設定が変更された．また，提案のネットワーク設定最適化手法はリアルタイムに高精度なモーション制御を実現した．たとえば，PON で局舎と工場を接続し，局舎に置いたモーション制御アプリケーションから IoT 端末を制御することでモーション制御に適したネットワークを構築することができる．

5-4　結論

SEBA に基づく 10G-EPON 環境でのエッジコンピューティングを用いてリアルタイム制御を行った．提案手法では SEBA とモーション制御アプリケーションを連携し，ネットワーク設定をモーション制御からの要求に基づいて，自動的に更新する．システムはユーザにネットワーク設定を意識させることなく，特定のモーション制御アプリケーションに最適なネットワークを提供できる．モータを用いた実機実験系を構築し，有効性を検証した．勾配降下法を用いた提案手法により，目標整定時間 3 s を満たすようにネットワーク設定が更新されることを確認した．モーション制御に最適な DBA 周期は 5 回の試行ののち，9

ms に決定され，整定時間は DBA 周期を 20 ms とし最適化を行わなかった場合に比べて，53%短縮された．実験により SEBA に最適なネットワークっていが適切に反映されていることを示した．このとき 1 Mbps のデータ通信を行う IoT 端末が 90 台接続可能であることも検証した．

5-5 参考文献

[1] L. Peterson, A. Al-Shabibi, T. Anshutz, S. Baker, A. Bavier, S. Das, J. Hart, G. Palukar, and W. Snow, “Central office re-architected as a data center,” IEEE Commun. Mag., vol. 54, no. 10, pp. 96–101, 2016.

[2] S. Kim, T. Suzuki, J. Kani, and J. Terada, “Exploiting general purpose hardware in optical access systems [Invited],” Journal of Optical Communications and Networking, vol. 12, no. 2, pp. A182-A189, 2020.

[3] S. Das, “From CORD to SDN Enabled Broadband Access (SEBA) [Invited Tutorial],” in Journal of Optical Communications and Networking, vol. 13, no. 1, pp. A88-A99, 2021.

[4] “Telefónica open access and edge computing,” White Paper, 2019.

[5] D. Zhang, D. Liu, X. Wu and D. Nesset, “Progress of ITU-T higher speed passive optical network (50G-PON) standardization,” Journal of Optical Communications and Networking, vol. 12, no. 10, pp. D99-D108.

[6] T. Suzuki, S. Y. Kim, Y. Koyasako, J. Kani and J. Terada, “Demonstration of Adaptive Image Transmission That Meets Various Application Requirements in 10G-EPON,” IEEE Access, vol. 8, pp. 186433-186440, 2020.

[7] K. Huang, W. Liu, Y. Li, A. Savkin and B. Vucetic, “Wireless Feedback Control With Variable Packet Length for Industrial IoT,” IEEE Wireless Communications Letters, vol. 9, no. 9, pp. 1586-1590, 2020.

[8] Y. Maddahi, S. Liao, W. Fung, E. Hossain and N. Sepehri, “Selection of Network Parameters in Wireless Control of Bilateral Teleoperated Manipulators,” IEEE Transactions on Industrial Informatics, vol. 11, no. 6, pp. 1445-1456, 2015.

[9] X. Zhang, Q. Han and X. Yu, “Survey on Recent Advances in Networked Control Systems,” IEEE Transactions on Industrial Informatics, vol. 12, no. 5, pp. 1740-1752, 2016.

[10] C. Lai and P. Hsu, “Design the Remote Control System with the Time-Delay Estimator and the Adaptive Smith Predictor,” IEEE Transactions on Industrial Informatics, vol. 6, no. 1, pp. 73-80, 2010.

[11] D. Yashiro and T. Yakoh, “Feedback Controller with Low-Pass-Filter-Based Delay Regulation for Networked Control Systems,” IEEE Transactions on Industrial Electronics, vol. 61, no. 7, pp. 3744-3752, 2014.

[12] K. Natori and K. Ohnishi, “A Design Method of Communication Disturbance Observer for Time-Delay Compensation, Taking the Dynamic Property of Network Disturbance Into Account,” IEEE Transactions on Industrial Electronics, vol. 55, no. 5, pp. 2152-2168, 2008.

[13] H. Yi, C. An. and J. Choi, “Compensation of Time-Varying Delay in Networked Control System over Wi-Fi Network,” International Journal of Computers, Communications & Control, vol. 12, no. 3, pp. 415-428, 2017.

[14] M. Ahmadi, N. Khanezaei, S. Manavi, F. Fatemi Moghaddam and T. Khodadadi, “A Comparative Study of Time Management and Energy Consumption in Mobile Cloud Computing,” 2014 IEEE 5th Control and System Graduate Research Colloquium , pp. 199-203, 2014.

[15] T. Suzuki et al., “Zero touch provisioning compliant with authentications of IEEE PON packages A and B for SDN-enabled broadband access,” Journal of Optical Communications and Networking, vol. 13, no. 11, pp. 244-252, 2021.

第６章　　ジッタバッファを用いた低ジッタネットワークの構築

6-1　はじめに

産業用イーサネットにおいて，通信は固定された周期で行われることが重要である．これまで提案してきたアクセスエッジコンセプトでは，コアネットワークと比べて小さいながら，遅延やジッタの影響を受ける．特に，遅延を予測し補償する遅延補償制御では想定した遅延と実際の遅延の差異が制御性能に悪影響を及ぼすため，ジッタの影響を抑える必要がある．これらのジッタの問題に対し，特にコアネットワークを対象に，ジッタバッファを用いて遅延補償制御を行ういくつかの研究が提案されてきた [1]-[4]．ジッタバッファは特定の期間，パケットを一時的に保存することで，一定の遅延の増加と引き換えに，遅延の値を固定化し，制御ループ内のジッタの影響を低減する方法である．バッファリング時間は，大きな値の場合遅延を徒に増やし，小さい値の場合パケットロスを引き起こすことから，バッファリング時間の決定方法は重要である．

バッファリング時間の設定を決定するため，正確な遅延情報を収集する手法が提案されている．参考文献[5]は，ラウンドトリップタイム (Round Trip Time, RTT) を用いて通信遅延をユーザが計測する．参考文献[6]では PTP (Precision Time Protocol) や GPS (Global Positioning System) を用いた時刻同期システムが送信時刻をパケット内に格納し，受信時刻との差を計算することで正確な遅延値を計測する．だが，システム全体で時刻同期が必要なことから構築コストが高い．

近年，オールフォトニクスネットワーク (APN) といった，正確な遅延情報をネットワークから常時収集できる先進的なネットワークが提案されている [7]-[10]．このようなアーキテクチャではそれぞれの光パスはネットワークコントローラによって管理される．それぞれのアクセスノードはアクセスノードと端末間やアクセスノード間の区間遅延情報を取得することができ，ネットワークコントローラは継続的にすべてのアクセスノードからネットワーク情報を収集できる．参考文献[11]では，この情報をネットワーク状態を最適に保つためのパス切り替えに用いる手法が提案されている．APN を用いることで，End-to-End の経路を構築するパスそれぞれから，遅延情報をリアルタイムに収集し合算することで End-to-End 全体の正確な遅延情報を取得することが可能である．このようなネットワークは有線・無線区間両方の正確な遅延情報を収集でき，遅延ゆらぎに強い高性能なジッタバッファリングを実現できる．しかしながら，これまでネットワーク遅延を制御し，収集した遅延情報を用いてモーション制御を行った研究はない．

本章では APN を対象とし，パス自体からリアルタイムに区間遅延情報を収集，最適なジッタバッファリングを行い，所望した値で遅延補償制御を行う手法を提案する．提案手法では制御器と端末間を構成する End-to-End の経路を構成するそれぞれのパスの区間遅延情報をアクセスノードから収集し，エッジリソースに送信する．エッジリソースでは受信した情報をもとに End-to-End の遅延を計算し，リアルタイムにジッタバッファ設定値を決

定・更新する．到着したパケットは解析され，対象のアプリに必要な場合のみジッタバッファが行われる．これにより，IoT 端末のモーション制御といったアプリケーションに適した大容量で低ジッタなネットワークを構築できる．この方法はユーザがネットワーク情報を収集する必要がない利点がある．また，収集した値を用いてモーション制御に最適なネットワーク設定に更新することも可能である．本章では典型的な低遅延アプリケーションであるモーション制御に焦点を当てているが，このアーキテクチャは他のアプリケーションの様々な要求に合わせたネットワークを構築するのにも有用である．提案手法の有効性を検証するため，FPGA (Field Programmable Gate Array) ベースのエッジリソースを用いてモータを制御する 100 Gbps の実験系を構築した．実験では映像データを模した 70 Gbps のデータを IoT 端末のセンサデータとともに送信した．この実験では大容量データ通信下で，制御に関連したデータのみミリ秒オーダでジッタバッファできることを確認する．

6-2　ジッタバッファを用いた遅延補償制御

6-2-1　　全体構成

提案手法では，ジッタバッファリングのために必要な End-to-End の経路のネットワーク遅延情報を，経路を構成するそれぞれの経路から区間遅延情報としてリアルタイムに収集・合算し，算出する．これにより，End-to-End の遅延を固定し，スミス予測器といった遅延補償制御を実現することで，現状のネットワークを反映したジッタ制御を実現できる．

この目的のため，リアルタイムに遅延情報を収集可能なネットワークを対象とする．図 6-1 に提案手法の全体図を示す．ネットワークコントローラはすべてのアクセスノードからリアルタイムに区間遅延情報を受け取り，遅延収集デバイスに受け渡す．遅延収集デバイスはセンサ値にかかる遅延を計算する．この値はジッタバッファ値を設定値に更新するために用いられる．制御対象からのセンサパケットは他のパケットと区別され，設定遅延値になるまでジッタバッファされる．この操作はそれぞれのセンサ値ごとに繰り返される．

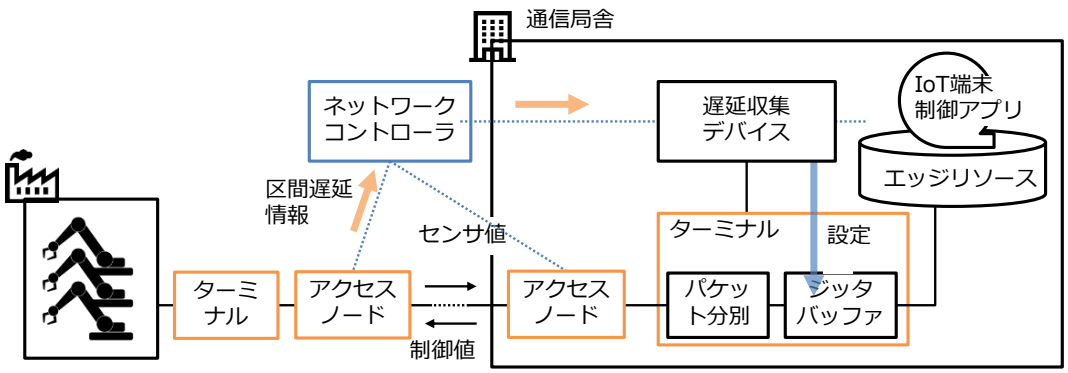


図 6-1　提案アーキテクチャ

6-2-2　　シーケンス

図 6-2 は提案手法の詳細なフローチャートを示す．(1) まず，IoT 端末制御アプリケーションは，遅延固定要求と，そのために必要な制御端末との間の End-to-End の経路情報を送信する．この情報は例えば，制御対象と制御アプリケーションの IP アドレスである．(2), (3) 遅延収集デバイスは，この要求を受信したのち，End-to-End の経路情報を経路を構成する区間情報に変換する．(4), (5) 遅延収集デバイスはネットワークコントローラに区間遅延情報を要求し，リアルタイムに受信する．(6) この遅延情報は End-to-End の遅延情報を計算するために合算される．(7) この値を用いて，センサ値が所望の値になるよう，ジッタバッファ設定値を決定する．ジッタバッファ値は遅延値から決定しても良いし，ユーザが決定しても良い．(8) 制御アプリケーションはこの値を通知される．(9), (10) 受信したパケットはセンサパケットと映像パケットといった制御に関係のないパケットに分別される．(11) センサ値は設定値になるまでバッファされ，制御アプリケーションに受け渡される．(12) 制御アプリケーションでは，遅延は設定値と等しいものと想定し，遅延補償制御が行われる．(13) 適切な制御値が IoT 端末に送信される．

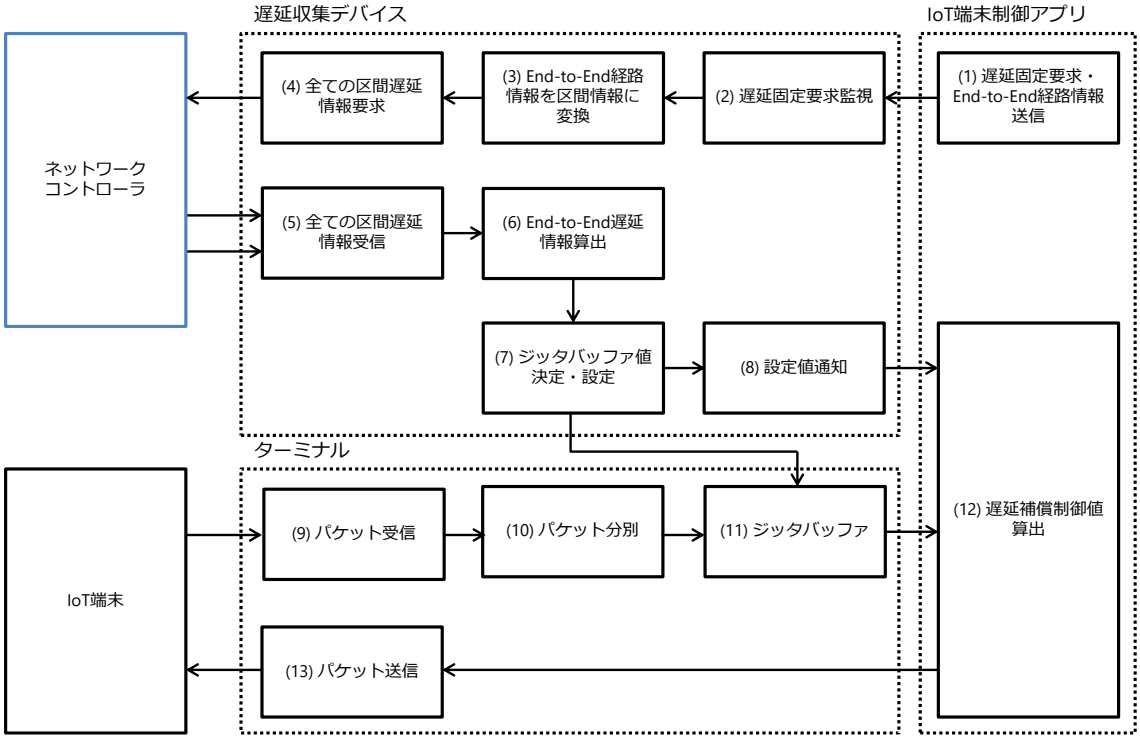


図 6-2　提案アーキテクチャのフローチャート

この処理によりフィードバックループの遅延値が固定され，遅延補償制御のための遅延値が決定される．遅延値は制御周期ごとにネットワークコントローラが収集する．制御周

期は受信したセンサ値から制御値を算出する周期であり，アプリケーションごとに適切に設定される．

制御アプリケーションは一般に IoT 端末のユーザが管理する．ユーザはネットワークの区間情報を知ることができないため，従来手法では End-to-End の経路の遅延情報を収集することには限界があった．しかし，本手法を用いることで，ユーザは遅延情報を収集することなく，ネットワーク遅延値が所望の値と等しいと想定し，より詳細な制御が実現できる．

6-2-3　　FPGA を用いた実装

図 6-3 に提案手法を FPGA で実装する際の具体的な構成を示す．パケット分別はヘッダ検出，ヘッダ解析，パケットセレクタから構成される．ジッタバッファはジッタバッファ，レジスタ，衝突検知から構成される．受信パケットはヘッダ検出とヘッダ解析を行ったのち，パケットセレクタで，送信元 IP アドレスやポート番号，イーサタイプやプロトコル番号などを監視することで，センサパケットとそれ以外のパケットに区別される．センサパケットはジッタバッファに送られ，その他のパケットは衝突検知機能に送られる．センサパケットはレジスタで決められた値だけバッファリングされ，出力される．バッファされたセンサパケットとその他のパケットの出力タイミングが重なったとき，センサパケットが優先され，その他のパケットは衝突検知機能においてセンサパケットが出力されたあとに，出力される．

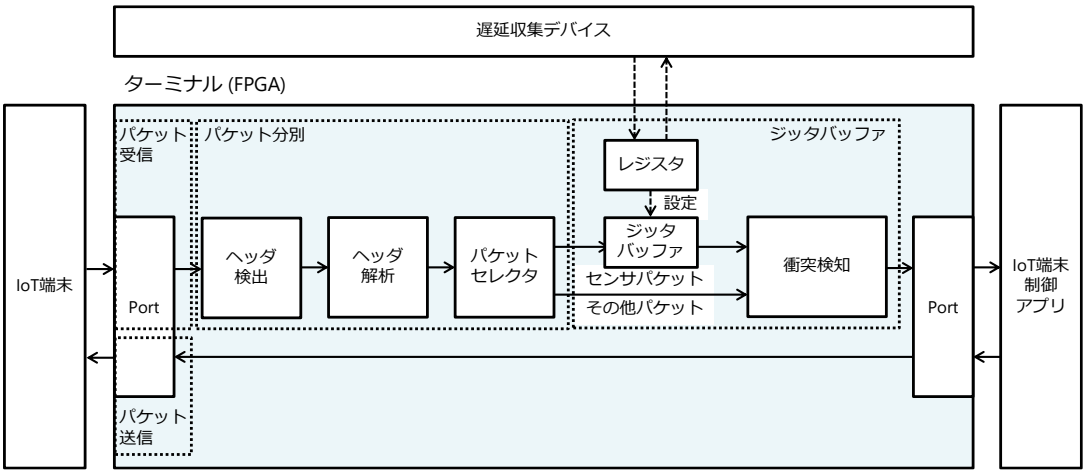


図 6-3　FPGA 構成

送信元 IP アドレスやポート番号，イーサタイプやプロトコル番号といったセンサパケットを分別するために必要な情報は，レジスタに格納される．これらの設定テーブルはセンサパケットを優先付けするために前もって構築され，受信パケットのヘッダがテーブルエ

ントリと一致したとき，パケットはセンサパケットとして識別される．これらの設定は遅延収集デバイスによって動的に変更される．イーサタイプとプロトコル番号は，産業用イーサネットプロトコルの場合は固有の値に設定されることが多く，特に重要な情報である．例えば，PROFINET のイーサタイプは 0x8892 で EtherCAT のイーサタイプは 0x88A4 である．このシステムはまた監視フィールドの場所をずらすことで，VLAN や Q-in-Q パケットも対応する．

ジッタバッファ値はネットワークの状態によってリアルタイムに更新される．また，同じメカニズムをユーザ端末の近くに設置することで，制御値のジッタバッファも実現できる．

この構成では，ネットワークコントローラは区間遅延情報をそれぞれの経路から収集する．そのため，ジッタバッファで吸収できない大きな遅延が発生したときには，通信事業者で遅延要求を満たす，より小さな遅延の経路に切り替えることで，ユーザは継続的に制御端末の制御を実現できる．

6-3　提案手法の実機検証

実験を 3 つ行った．実験 1 で，ジッタバッファが想定通りに動作するか検証した．実験 2 で，高負荷ネットワーク環境下でもパケット分別が正しく実行され，ジッタバッファが制御関連のパケットにのみ適用され，詳細な制御が実行されるか検証した．実験 3 で，複数のモーション制御イベントの間でもジッタバッファが遅延変化に対応できるか検証した．

6-3-1　　実験構成

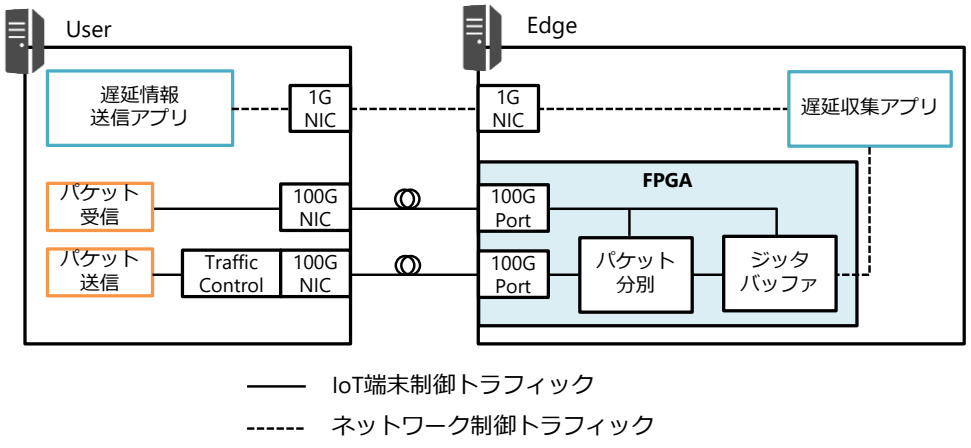
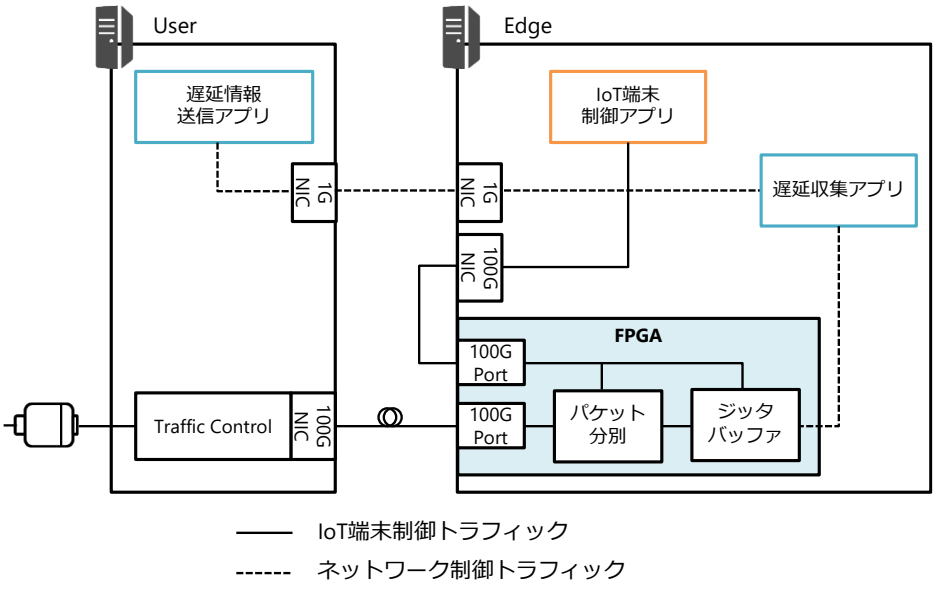


図 6-4　実験 1 の実験構成



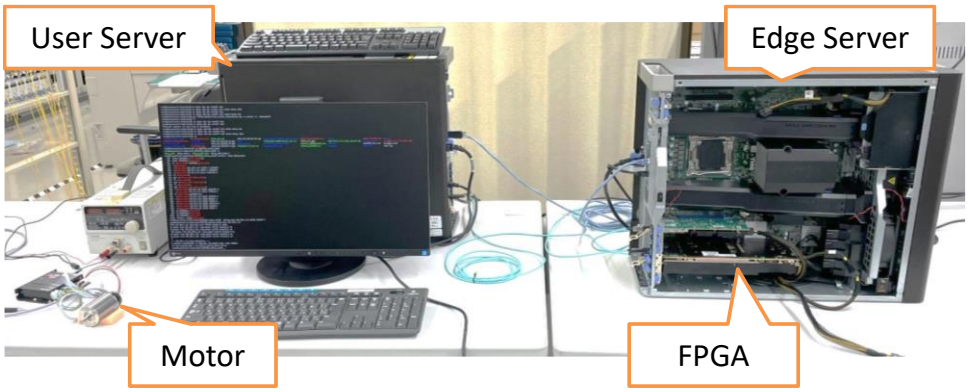


図 6-5　実験 2, 3 の実験構成

実験 1 の実験系を図 6-4 に，実験 2 と 3 の実験系を図 6-5 に示す．2 台のサーバを用い，1 台は制御アプリケーションと遅延収集アプリケーションを格納し，エッジサーバを模擬した．もう 1 台にはネットワークコントローラとアクセスノード機能を格納し，ユーザサーバを模擬した．2 台のサーバスペックは，Intel Xeon E5-2670 v3 (2.30 GHz, 24 core) CPU と 32GB メモリである．FPGA は Silicom N5010 を用いた．OS はエッジサーバには Cent OS 8.2，ユーザサーバには Ubuntu 16.04 を用いた．モータは Maxon EC-max 40 で，ユーザサーバとは USB ケーブルと EPOS2 24/5 モータコントローラを介してシグナルケーブルで接続した．計算された制御値は電流値に変換される．センサ値や制御値といった IoT 端末制御トラフィックは 100 Gbps のリンクレートで転送され，遅延情報といったネットワーク制御トラフィックは 1 Gbps のリンクレートで転送した．

ユーザサーバの 100G NIC では Linux Traffic Control を用いて出力データにネットワーク遅延を付加した．設定遅延値はユーザサーバからエッジサーバに送信される．エッジサーバ上の遅延収集アプリケーションは送信された遅延情報を収集し，センサ値をバッファリ

ングするために必要な値を算出し，FPGA に設定する．FPGA では設定値に基づいて，センサ値を一定時間バッファリングし，モーション制御アプリケーションに受け渡す．モーション制御アプリケーションでは設定値を用いて遅延補償制御を行い，モーション制御のための制御値を算出する．

これらの実験ではジッタバッファは，遅延ゆらぎが特に大きくなりやすい上りトラフィックに対して行い，下りトラフィックでの遅延は一定値とした．到着したセンサ値はヘッダ情報から，制御に関連するパケットと映像データといったそれ以外のパケットに分別された．センサ値は設定値をもとにジッタバッファされ，制御アプリケーションに受け渡される．一方で，その他のデータはジッタバッファされずに出力されるが，センサ値と衝突する場合は破棄された．

次の 3 つの実験を行った．

- 実験 1：ジッタバッファの検証
- 実験 2：高負荷ネットワーク環境での検証
- 実験 3：遅延が時間変動する複数モーション制御イベント下での検証

実験 2 と 3 では次のふたつの手法が比較された．どちらの手法でも遅延値は 20 ms と想定している．

- 従来手法：ジッタバッファのない，スミス予測器を用いた PI 制御
- 提案手法：ジッタバッファの設定値を 20 ms とした，スミス予測器を用いた PI 制御

スミス予測器のモータモデルは次式より導出される．数値はモータのスペックシートから決定した．この値はモータに依存する固定値である．

$$\hat{G}_P(s) = \frac{195000}{s^2 + 114s + 150} \tag{6-1}$$

PI 制御のゲインは次の式で決定した．

$$G_C(s) = K_P E(s) + K_I E(s)/s \tag{6-2}$$

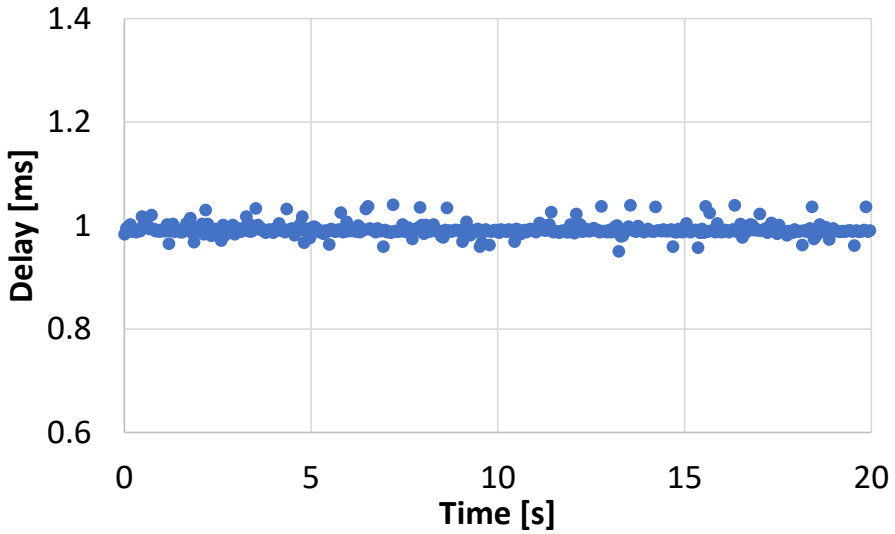
$$K_P = 0.001 \quad K_I = 0.0012 \tag{6-3}$$

K_p と K_I は比例ゲインと積分ゲインをそれぞれ表す．これらのゲインを大きくすると，入力に対する変化量が大きくなり，収束が早くなる可能性があるが，誤差も発生しやすくなり，制御系が不安定になる可能性がある．逆に，これらの値を小さくすると，整定時間が長くなる．本実験では両手法で同じゲインを用い，ジッタバッファの効果のみが現れるようにした．

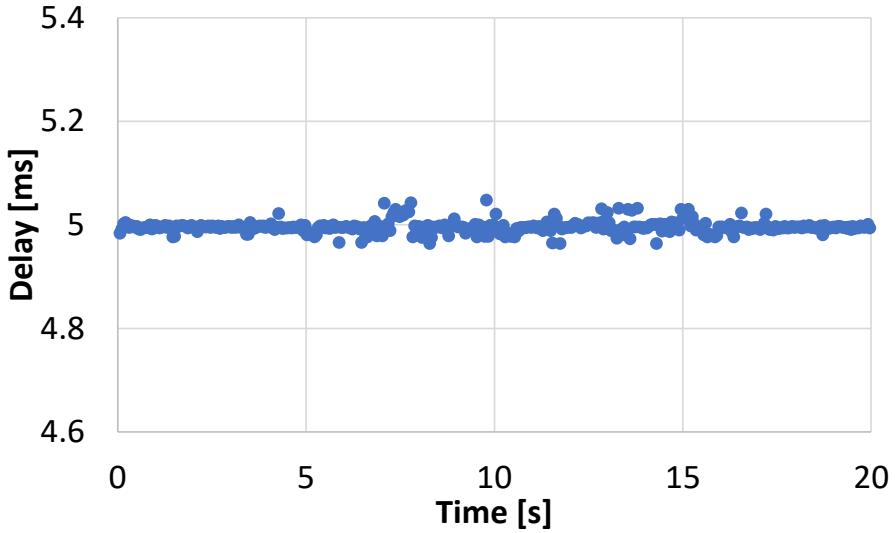
この実験では制御周期は 10 ms に設定し，遅延収集周期も同じく 10 ms に設定した．目標値を 4000 qc とする位置制御を行った．整定時間は目標角度の±5%以内に収まったときの時間とした．

6-3-2　　実験結果

実験 1



(a) ジッタバッファ設定値 1 ms の時刻歴



(b) ジッタバッファ設定値 5 ms の時刻歴

図 6-6　ジッタバッファ結果

表 6-1　ジッタバッファの設定値と平均遅延の結果

Jitter Buffer Set [ms]	Average Delay Time [ms]
1	0.99
2	1.99
5	4.99
10	10.00

まずジッタバッファ性能を検証した．模擬されたセンサパケットは 50 ms ごとに送信された．図 6-6 にジッタバッファの設定値をそれぞれ 1 ms と 5 ms としたときの遅延の時刻歴を示す．各パケットは設定値にしたがって遅延していることから，ジッタバッファがうまく機能していることがわかる．表 6-1 にはジッタバッファの設定値と平均遅延の結果を示す．結果からジッタバッファは 1 ms レベルでも可能であることが確認できた．結果から，提案アーキテクチャは 1 ms 以下の超低遅延が求められる産業用アプリケーションにも適用できることが示された．

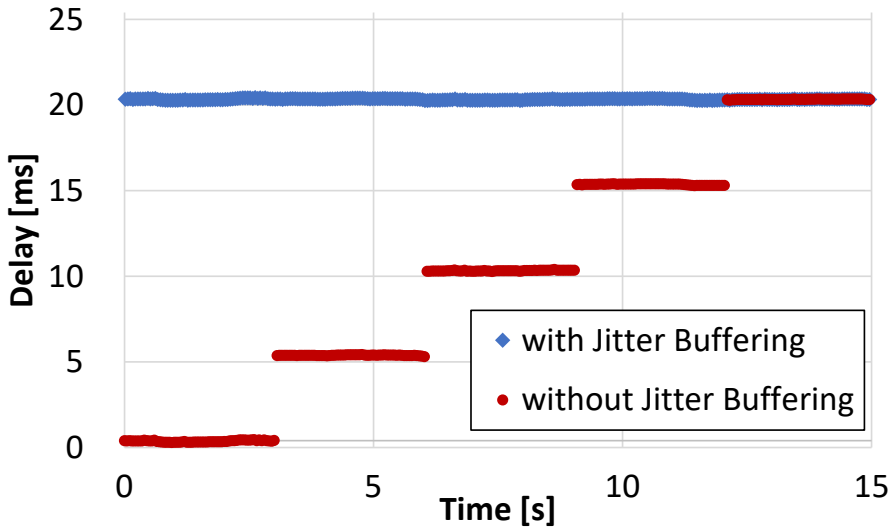


図 6-7　ジッタバッファの有無による End-to-End の遅延値の比較

続いて，制御中にバッファからの読み出し時間を変更できる FPGA の新しい外部インタフェースが正常に動作し，入力タイミングが変わった場合でも固定遅延で正しく出力できることを確認した．図 6-7 はネットワーク遅延を 3 s 毎に 0 ms から 20 ms まで 5 ms ずつ増加させたときのジッタバッファがない場合と 20 ms のジッタバッファを行った場合の End-to-End の遅延の時刻歴を示す．結果から，ジッタバッファがない場合では End-to-End の遅延が変動しているが，ジッタバッファがある場合ではジッタバッファ設定値がすぐさま更新され，ネットワーク遅延が変動しても，End-to-End の遅延値は 20 ms の一定値に適切に保たれていることを確認した．

実験 2

映像データといった大容量トラフィックがセンサ値とともに伝送されたとき，FPGA がパケット分別を期待したとおりに実現し，センサ値にだけジッタバッファを適用することでモーション制御が実現できることを検証した．TCP パケットを用いた高負荷ネットワーク環境を iperf で作成し，映像データを模した 70 Gbps トラフィックを IoT 端末のセンサ値とともに伝送した．

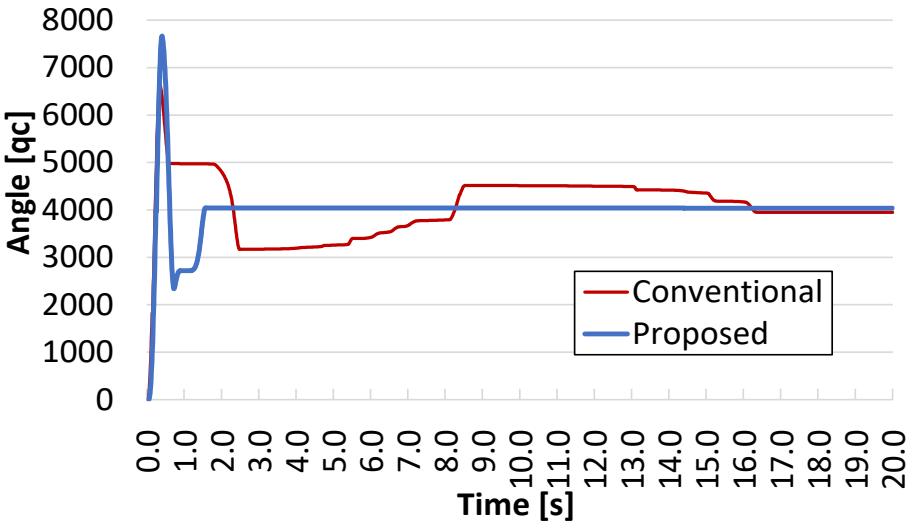


図 6-8　想定遅延値が 20 ms で実際の遅延値が 0 ms であった場合の時刻歴

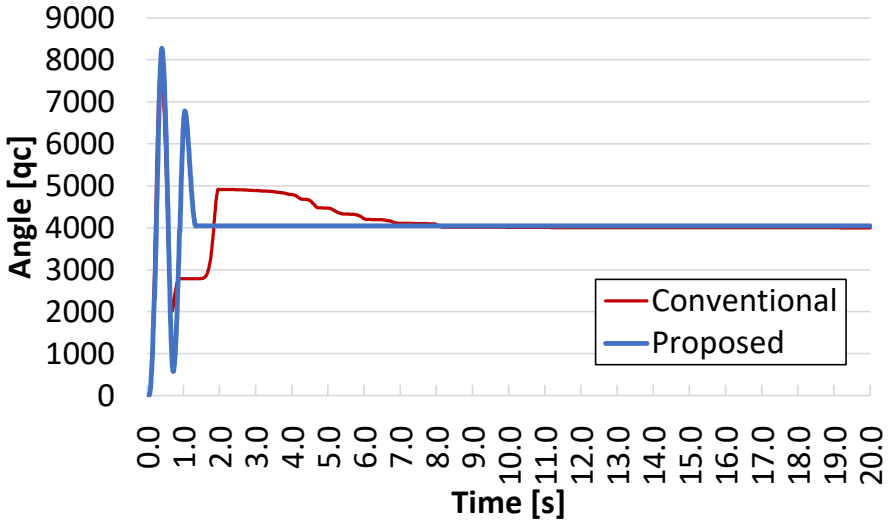


図 6-9　想定遅延値が 20 ms で実際の遅延値が 12 ms であった場合の時刻歴

図 6-8 はスミス予測器の想定遅延値が 20 ms で実際の遅延値が 0 ms であった場合の時刻歴を，図 6-9 は想定遅延値が 20 ms で実際の遅延値が 12 ms であった場合の時刻歴をそれぞれ示す．図 6-8 では提案手法の整定時間が 1.5 s だったのに対し，従来手法では 15.2 s であった．また，このとき，従来手法では制御に失敗することがあった．図 6-9 では提案手法の整定時間が 1.3 s であったのに対し，従来手法では 6.0 s であった．結果から提案手法は 70 Gbps の大容量フローに対して，パケット分別により制御データにのみジッタバッファが実現でき，ジッタバッファを行わず輻輳が発生する可能性のある従来手法と比べて，制御性能が向上することがわかる．従来手法は想定遅延値と実際の遅延値の差異により，スミス予測器で遅延を補償できていないため，収束に時間がかかっている．ジッタバッフ

ァが行われず，遅延が変動するとき，遅延想定値 t_m と実際の遅延値 $t_1 + t_2$ が異なり，モータ出力のシミュレーション値が実値からずれる．このずれが外乱となり，制御性能に悪影響を及ぼす．

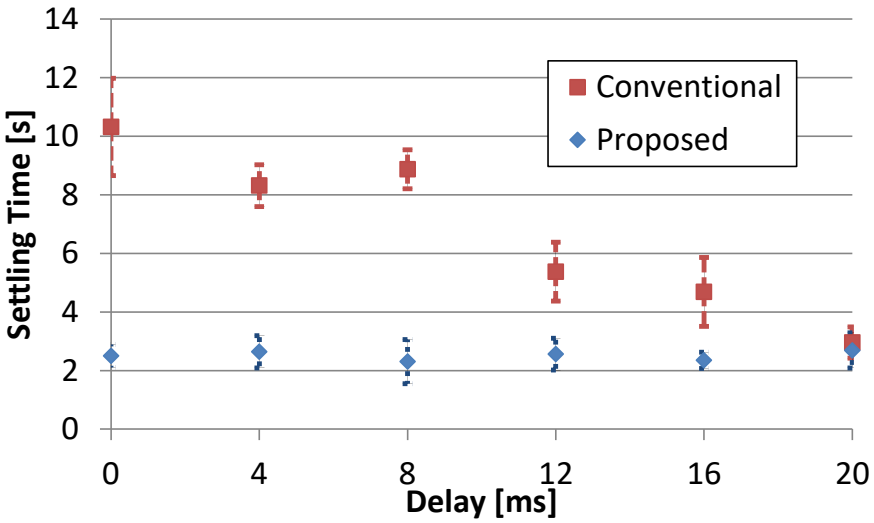


図 6-10　実際の遅延の変化に対する整定時間の結果比較

図 6-10 は実際の遅延の変化に対する整定時間の結果をまとめたものである．それぞれの結果ごとに 4 回試行し，標準偏差も示した．提案手法では遅延値が 20 ms になるように応答性の高いジッタバッファを実現したため，整定時間はあまり変化しない．従来手法では，想定と実際の遅延との差異が大きいとき，整定時間が増加し，その差が小さくなるにつれ，整定時間も減少した．

実験 3

遅延値が複数制御イベント中に変動しても，ジッタバッファ値が即座に書き換えられ，高い制御性能を保つことを検証した．遅延が 5 s 毎に 0 ms から 20 ms まで 10 ms ずつ増加するとき，モーション制御を 3 回連続して行った．

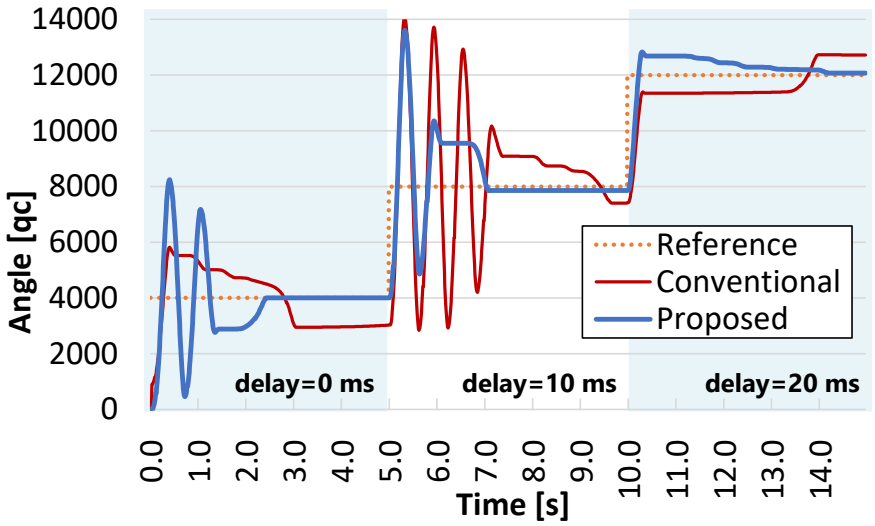


図 6-11　複数制御イベントの時刻歴

図 6-11 に複数制御イベントの時刻歴を示す．従来手法では 1 回目と 2 回目のイベントでは目標値に収束できず，3 回目のイベントの整定時間は 3.7 s であった．一方で，提案手法では各イベントで 2.4 s，2.0 s，3.4 s の整定時間で制御に成功している．これらの結果から従来手法では複数のモーション制御イベント中，遅延変化に対応できず制御できないが，提案手法では制御可能なことを示した．

6-4　結論

本検討ではアクセスノードから収集したネットワーク遅延情報を用いて，ユーザが管理する IoT 端末に適したジッタバッファを行うアーキテクチャを提案した．提案手法では End-to-End の遅延をリアルタイムに収集し，得られた値からジッタバッファを実施，所望の遅延値に基づいてより詳細な遅延補償制御を実行する．モータを用いた実験系を構築し，提案手法の有効性を検証した．実験 2 では提案手法がパケット分別により高負荷ネットワーク環境下でもミリ秒オーダで意図したとおりにセンサ値にのみジッタバッファを行い，1.5 s 以内のモーション制御を実現することを示した．実験 3 では，遅延が変化する複数モーション制御イベントにおいて，提案手法が制御性能を保持し，整定時間 3 s 程度を実現することを示した．これらの実験から，提案手法では遅延変動に対応し，高い制御性能を保持できることを示した．

6-5 参考文献

[1] W. Grega and A. Tutaj, “Active buffer design method for networked control systems,” 2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 1157-1162, 2015.

[2] L. Fu, Z. Zhang, Q. Kong and J. Mao, “The design of adaptive fuzzy PID controller with Smith compensator for Network Control Systems,” 2015 IEEE International Conference on Information and Automation, pp. 3037-3040, 2015.

[3] D. Yashiro and T. Yakoh, “Feedback Controller With Low-Pass-Filter-Based Delay Regulation for Networked Control Systems,” IEEE Transactions on Industrial Electronics, vol. 61, no. 7, pp. 3744-3752, 2014.

[4] Y. Nagai, T. Watanabe, T. Sato, R. Nakamura and H. Hadama, “An Evaluation of a State-Predictive Controller and a Jitter Buffer for Remote Controlled Autonomous Vehicles via the Internet,” 2019 International Conference on Information and Communication Technology Convergence (ICTC), pp. 444-449, 2019.

[5] R. Imai and R. Kubo, “Introducing jitter buffers in networked control systems with communication disturbance observer under time-varying communication delays,” IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society, pp. 002956-002961, 2015.

[6] L. Repele, R. Muradore, D. Quaglia and P. Fiorini, “Improving Performance of Networked Control Systems by Using Adaptive Buffering,” IEEE Transactions on Industrial Electronics, vol. 61, no. 9, pp. 4847-4856, 2014.

[7] R. Muñoz et al., “Dynamic Reconfiguration of WDM Virtual Network Topology over SDM Networks for Spatial Channel Failure Recovery with gRPC Telemetry,” 2022 Optical Fiber Communications Conference and Exhibition, pp. 1-3, 2022.

[8] M. F. Silva, A. Pacini, A. Sgambelluri, L. Valcarenghi and F. Paolucci, “Bringing Disaggregated Telemetry and ML to the Transceiver for Autonomic Signal Adaptation,” 2022 Optical Fiber Communications Conference and Exhibition, pp. 1-3, 2022.

[9] A. Itoh, “Innovative Network for 2030 (Beyond 2020),” NTT technical review, 2019.

[10] “Open All-Photonic Network Functional Architecture,” NTT Whitepaper, 2022.

[11] H. Ou, K. Asaka, T. Shimada and T. Yoshida, “SLA-Aware Real Time Control Technology across Optical and Mobile Networks,” 2022 Optical Fiber Communications Conference and Exhibition (OFC), 2022.

第 7 章　産業用ネットワークを対象とした End-to-End でのアクセスエッジ技術実証

7-1　はじめに

第 3 章から第 6 章まで，アクセス区間で生じる遅延の影響を低減し，アクセスエッジコンセプトを実現する手法について提案してきた．本章では，アクセスエッジコンセプトの End-to-End での実証を行うために，モーション制御の重要なユースケースである，工場内の産業用ネットワークを対象とする．第 2 章で述べたように，産業用ネットワークは遅延や遅延ゆらぎに厳しい要求があることから，工場内に閉じたネットワークとなっており，また専用のハードウェアと産業用プロトコルで実現されている．産業用ネットワーク上でアクセスエッジコンセプトを実現するためには，既存の産業用ネットワークを SDN 化してアクセスネットワークと接続する必要がある．本章では，産業用ネットワークプロトコルの機能をソフト化して汎用サーバ上で実現，SDN 化することで，工場内の具体的なアプリケーションを想定したアクセスエッジコンセプトの有用性を示す．

これまで産業用のタイムクリティカルなネットワークに SDN 技術を適用する研究がいくつか行われてきた [1]．参考文献[2]では，OpenFlow をリアルタイムデータプレーンに拡張し，それをサポートするリアルタイム SDN コントローラを提案した．参考文献[3]では，様々な産業用イーサネットアプリケーションを TSN (Time-Sensitive Networking) を通じて統合し，SDN により TSN スイッチを制御することを目的としている．参考文献[4]では，PROFINET の SDN 対応である SDNPROFINET を提案した．中央のコントローラがネットワークの情報を収集し，PROFINET のデータチャンネルルートにしたがってデータプレーン機器を設定する．しかしながらこの方法は PROFINET に限られる．管理の柔軟性を高めるために，異なる要求の産業用プロトコルの集中管理を行う研究がいくつか提案されている．参考文献[5]では，異なるプロトコルのコントローラを管理し，通常手動で行われるエンジニアリングや構成などのネットワーク関連の処理の大部分を動的に自動化するコントローラを作成している．しかし，これらの手法は OpenFlow のプロトコル管理といったコントロールプレーンのソフト化に焦点を当てており，データプレーン含めた産業用ネットワーク全体のソフト化は達成していない．

産業用ネットワークの柔軟性を高めるために，いくつかの研究では，PLC (Programmable Logic Controller) アプリケーションを計算リソースに設置し，産業用ネットワーク越しに IoT 端末の制御を行っている．参考文献[6]では，制御アプリケーションをクラウド上に実装する SoftPLC のコンセプトを提案，評価を行った．参考文献[7]は，PLC 機能を VM に実装する vPLC のコンセプトを提案し，ジッタ性能を検証した．参考文献[8]では，フィールドレベルの通信に確定的 IP ネットワークを使用できることを実証した．そのアーキテクチャでは vPLC が Ubuntu の VM にインストールされ，確定的 IP ネットワークにハイパバイザ上の仮想スイッチ経由で接続される．参考文献[9]では，PLC 機能をコンテナに実装した．

参考文献[10]では，SoftPLC と vPLC を比較し，リアルタイム処理の確認と応答時間の比較を行った．これらの研究はほとんどがリソースに焦点を当てており，多くの実験では制御機能に既存の機器ベンダのソフトウェアを用いている．そのため，他のアプリケーションと接続するような PLC 機能の柔軟性は考慮されていない．

産業用プロトコル機能をソフト化することは，各プロトコルが低遅延やリアルタイム性能を保証するためそれぞれ独自の技術を用いているため，課題である．しかし，産業用ネットワーク全体の SDN 化が達成されれば，採用したプロトコルの変更やシステムの拡張が容易になり，プロトコルフリーなネットワークが構築できる．また，このネットワーク上でアクセスエッジコンセプトを実現することで，経済的な利益だけではなく，現代制御のような複雑なモーション制御の実現や AI やビッグデータといった高度なデータ処理アプリケーションとの連携制御が行える制御システムを作ることができる．

この章では，従来の産業用イーサネットプロトコル機能をソフト化し，産業用機器の機能を汎用サーバ上で実現する新しいアーキテクチャを提案する．提案手法では，産業用イーサネットプロトコル機能をソフト化して，ドライバで実現する．モーション制御機能は汎用言語でソフト実装することで，従来の PLC では困難，または専用のハードウェアが必要であった複雑な制御を容易に実現する．また，ネットワークコントローラをもちいて，データプレーンの柔軟なネットワーク制御を行う．既存の産業用イーサネットプロトコル機能をソフト化により実現することで，システム構成やモータといったアプリケーションを変更することなく，ソフトウェア定義の産業用ネットワークを実現することができる．本章では PROFINET NRT プロトコルをソフト化し，汎用サーバ上で PROFINET PLC，およびスイッチの機能を実現することを目指す．モータを用いた実機実験系を構築し，モータ制御によりドライバ動作と提案アーキテクチャの有効性を検証する．システムは産業用ネットワークプロトコルの処理，制御値の計算，ネットワークコントローラとの通信が可能であり，産業用ネットワーク機能の SDN 化を実現する．

7-2　産業用ネットワークの SDN 化

7-2-1　　全体構成

汎用サーバ上で既存の専用コントローラ機能を実装し，ソフトウェア定義の産業用ネットワークを構築することを目的とする．

図 7-1 は現在の構成と提案構成を示す．現在の構成では，専用 PLC が専用スイッチやモータ，センサ等に接続している．しかし，これらの構成は専用 PLC が特定のプロトコルに縛られていることから，他のプロトコルの産業用デバイスに拡張できない．この問題に対処するため，制御プレーンとデータプレーン，モーション制御機能をソフト実装することで様々なプロトコルをサポートするプロトコルフリーのアーキテクチャを提案する．VM,

もしくはコンテナが汎用サーバ上で起動し，PROFINET や EtherCAT といった定時性を管理する産業用イーサネットプロトコルの機能がドライバの形で格納される．ソフトウェアスイッチが用いられ，制御関連のパケットのみ優先度を上げるといった転送ルールは，REST や OpenFlow などのエージェントを用いてネットワークコントローラから投入される．

このアーキテクチャにより，モーション制御といったアプリケーションを修正する必要なしに，様々なベンダの多様な産業用機器を制御することが可能となる．これは，異なるプロトコルのデータを各プロトコル専用のドライバが仲介することで実現している．その結果，複数のプロトコルを同じネットワーク上に共存できる．これにより，これまで工場で採用しているプロトコルに縛られていた産業用機器の選択を，実際の性能やコストに基づいてより柔軟に決定できる．加えて，このアプローチにより，プロトコル機能の柔軟な更新やバージョンアップが容易になる．

このアーキテクチャを通信局舎に実装することで，通信局舎をリアルタイムモーション制御に特化したデータセンタのように扱うことができる．

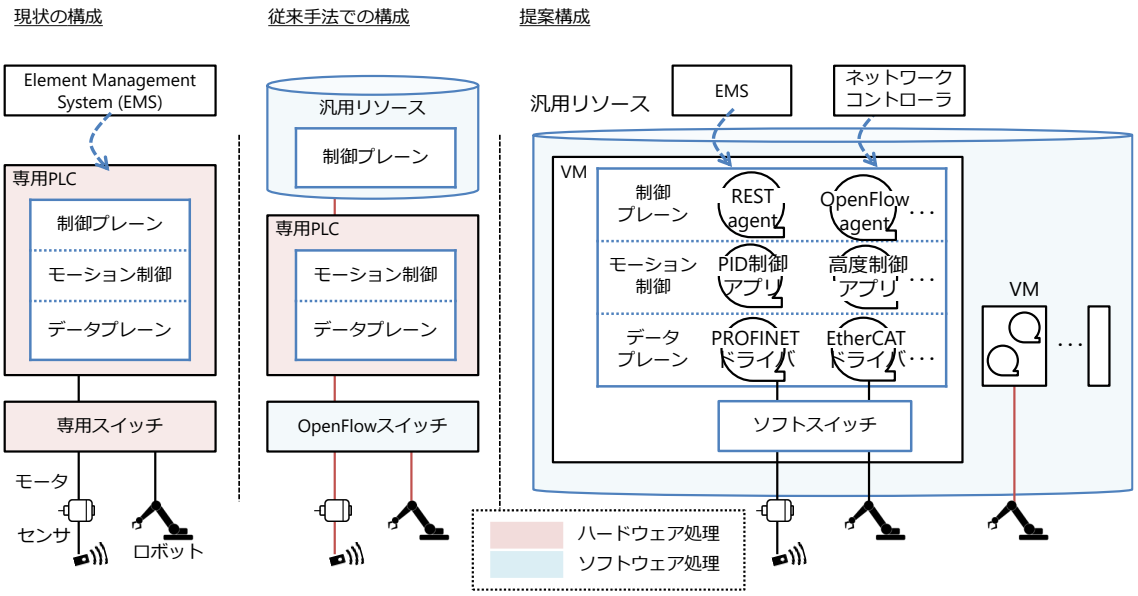


図 7-1　工場でのフィールドネットワーク構成

7-2-2　　産業用ネットワークプロトコル機能のソフト化

プロトコル機能をソフト化することで，専用ハードウェアで行われていたパケット処理をソフトウェアで実現する．図 7-2 は提案アーキテクチャの詳細を示し，Algorithm 1 は処理フローを示す．IoT 端末から送られてきたセンサパケットはソフトウェアスイッチにより，プロトコルごとに振り分けられ，対応するドライバに受け渡される．(1) ドライバはセンサパケットを受信し，(2) センサデータ $x_0'(t)$ を取り出して，(3) 共通のデータ形式 $x_0(t)$ に変換してモーション制御アプリケーションに受け渡す．(4) モーション制御アプリケーションはセンサデータ $x_0(t)$ と目標値 $r(t)$ をもとに，任意の制御アルゴリズムで制御値 $u(t)$ を算出，ドライバに受け渡す．(5) ドライバは制御値を各プロトコルのデータ記述形式 $u'(t)$ に変換し，(6) プロトコルのデータ部に格納，宛先や VLAN といった UDP/IP 情報やブートタイムやシーケンス番号といったプロトコル固有の情報を付け加えて，制御パケットを作成する．(7) 各プロトコルの制御パケットはソフトウェアスイッチを通じて IoT 端末に送信される．この操作を繰り返すことで，汎用サーバからのアプリケーションの制御が可能になる．それぞれの機能は分けられ，対象のアプリケーションに沿うように個別に書き換えたり置き換えたりできる．

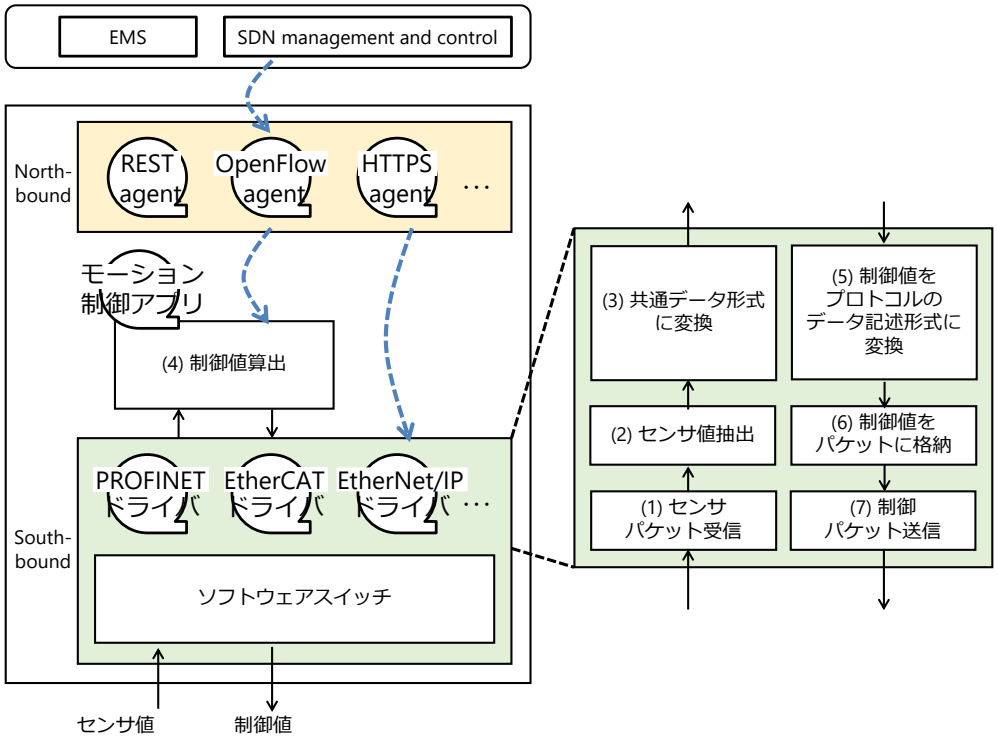


図 7-2　提案アーキテクチャ

Algorithm 1 Algorithm for proposed architecture

Input: Sensor packet, Reference data $r(t)$
Output: Control packet
1: **while** $TRUE$ **do**

Driver

2: Receive sensor packet
3: Extract sensor data $x'_0(t)$ from sensor packet
4: Convert sensor data $x'_0(t)$ to common data format $x_0(t)$
5: Pass $x_0(t)$ to motion control application

Motion control application

6: Get $x_0(t)$ from driver
7: Control command $u(t)$ is calculated based on $r(t), x_0(t)$ by any control algorithm
8: Pass $u(t)$ to driver

Driver

9: Get $u(t)$ from motion control application
10: Change $u(t)$ to data description format $u'(t)$
11: Store $u'(t)$ in control packet
12: Send control packet
13: **end while**

スイッチ機能は DPDK を用いたソフトウェアスイッチを用いて実装される．ソフトウェアスイッチは一般に専用のスイッチより遅延性能が劣るが，DPDK を用いる事で専用機器と同程度の遅延性能を達成することができる．DPDK はカーネルをバイパスして，システムコール無しでイーサネットのハードウェアを直接制御することで，パケット処理性能とスループットを向上させるよう設計されたパイプラインアーキテクチャである．

7-2-3 モーション制御機能のソフト化

モーション制御機能は C 言語のような高速で処理できる汎用言語で実装される．これらのアプリケーションはこれまで専用コンポーネントを用いてしか実現できなかった制御手法を簡単に実装することが可能である．加えて，AI やビッグデータといった技術や，ロバスト制御，現代制御といった様々な複雑な制御と組み合わせることで，従来は困難だったより高度な制御が実現可能である．ドライバとモーション制御アプリケーションはメモリ共有やその他の高速な方法でデータを交換する．

7-2-4　　ネットワークコントローラ

データプレーンはネットワークコントローラによって制御され，ネットワークコントローラからドライバやソフトウェアスイッチへの設定の送信を仲介するために，REST や OpenFlow，HTTPS などのエージェントが起動する．これにより，流れるパケットの分析や制御関連のパケットにのみ優先制御を適用するなど，柔軟なネットワーク構成を外部から簡単かつ高速に構築することが可能になる．

7-3　提案手法の実機検証

7-3-1　　実験構成

提案アーキテクチャの有効性を検証するため，実機実験を行った．PROFINET プロトコルのソフトウェアでのモーション制御を実装した．送受信プログラムは Go 言語で記述し，モーション制御プログラムは C++言語で記述した．プログラム間のデータ転送はループバックを用いた．

サーバスペックは Intel Xeon E3-1270 (3.80 GHz, 4 cores) CPU の 32 GB メモリである．OS は Ubuntu 16.04 である．PROFINT PLC は 1510SP-1 PN を用い，リモート I/O の死活監視のためのみに用いた．PROFINET リモート I/O は ET200SP を用いた．モータは Maxon EC-max 40 を用い，モータドライバは EPOS2 24/5 を用いた．モータドライバとリモート I/O は RS232 ケーブルで接続した．計算された制御値は電流値に変換され，電流制御が行われる．ソフトウェアスイッチとして Open vSwitch (OvS) を用い，ネットワークコントローラとして ONOS を用いた．

既存の産業用ネットワークの SDN での実装を検証するために，以下の 3 つの条件で実験を行った．

- 実験 1：PROFINET 専用スイッチとソフトウェアスイッチの遅延性能比較
- 実験 2：遅延補償制御の有無によるモーション制御アプリケーションの比較
- 実験 3：ONOS を用いたネットワークコントロールの有無による比較

工場内で広く用いられている古典制御を採用し，PI 制御のゲインはパラメータチューニングによって決定した．

制御周期を 0.5 s とし，整梯時間を目標値の±10%に収まったときの時間とした．

7-3-2　　実験結果

実験 1

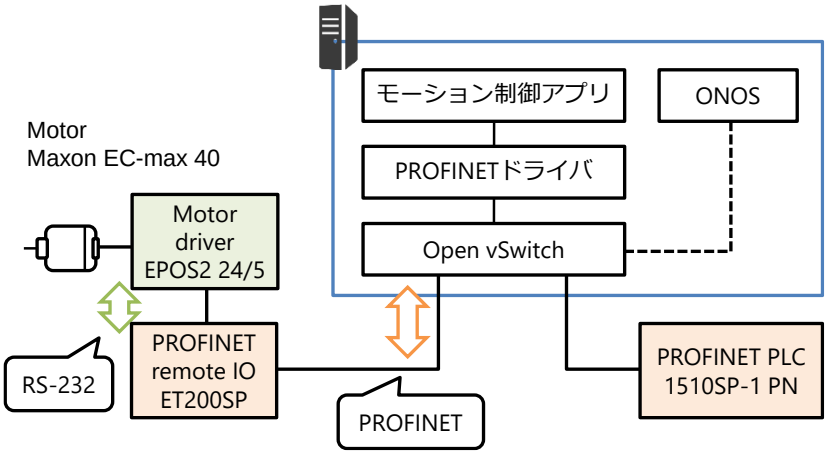


図 7-3　実験 1, 2 の提案手法の実験構成

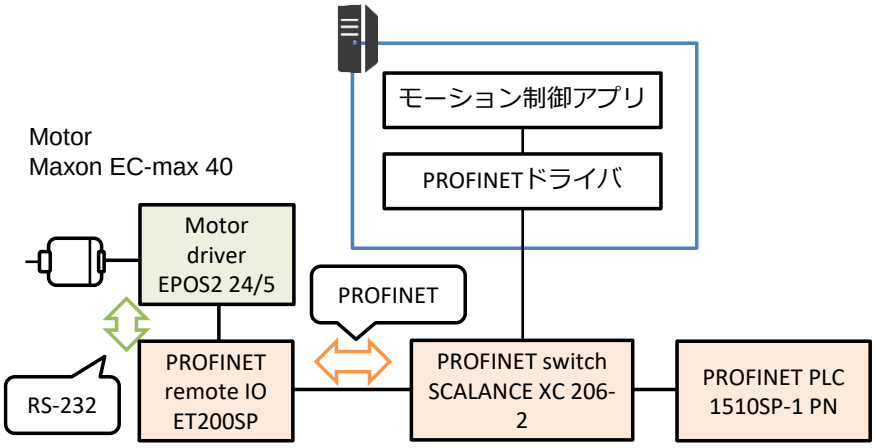


図 7-4　比較手法の実験構成

まず，PROFINET パケットがドライバで正しく作成されているか，モーション制御のためのリモート I/O との通信が正常に行われているかを検証する．提案手法の実験構成を図 7-3 に，比較手法を図 7-4 に示す．

No.	Time (s)	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.0.1	192.168.0.2	PNIO-CM	220	Write request, IODWr
2	0.013124485	192.168.0.2	192.168.0.1	PNIO-CM	206	Write response, OK,
3	0.093877576	192.168.0.1	192.168.0.2	PNIO-CM	206	Read request, IODRea
4	0.104188631	192.168.0.2	192.168.0.1	PNIO-CM	213	Read response, OK, I
5	0.527956159	192.168.0.1	192.168.0.2	PNIO-CM	216	Write request, IODWr
6	0.539275370	192.168.0.2	192.168.0.1	PNIO-CM	206	Write response, OK,
7	0.618382938	192.168.0.1	192.168.0.2	PNIO-CM	206	Read request, IODRea
8	0.629272665	192.168.0.2	192.168.0.1	PNIO-CM	213	Read response, OK, I
9	0.663798781	192.168.0.1	192.168.0.2	PNIO-CM	209	Write request, IODWr
10	0.675119769	192.168.0.2	192.168.0.1	PNIO-CM	206	Write response, OK,
11	0.755165792	192.168.0.1	192.168.0.2	PNIO-CM	206	Read request, IODRea
12	0.767120427	192.168.0.2	192.168.0.1	PNIO-CM	221	Read response, OK, I

図 7-5　パケットの記録データ

図 7-5 は制御値が送信され，センサ情報をモータから受信したときに交換されたパケットのデータである．パケット番号 1-4 はひとつの制御値の送信用で，5-12 はひとつのセンサ情報の受信用である．ドライバで作成された送信元 192.168.0.1 からの要求入力は，送信元 192.168.0.2 のリモート I/O によって OK と応答された．作成したパケットは PROFINET パケットとして認識され，PNIO-CM (PROFINET Input Output Control Management) プロトコルとして表示されている．この結果から汎用サーバと PROFINET 実験系の間で通信が成功し，センサ値を正しく受信できたことが示された．

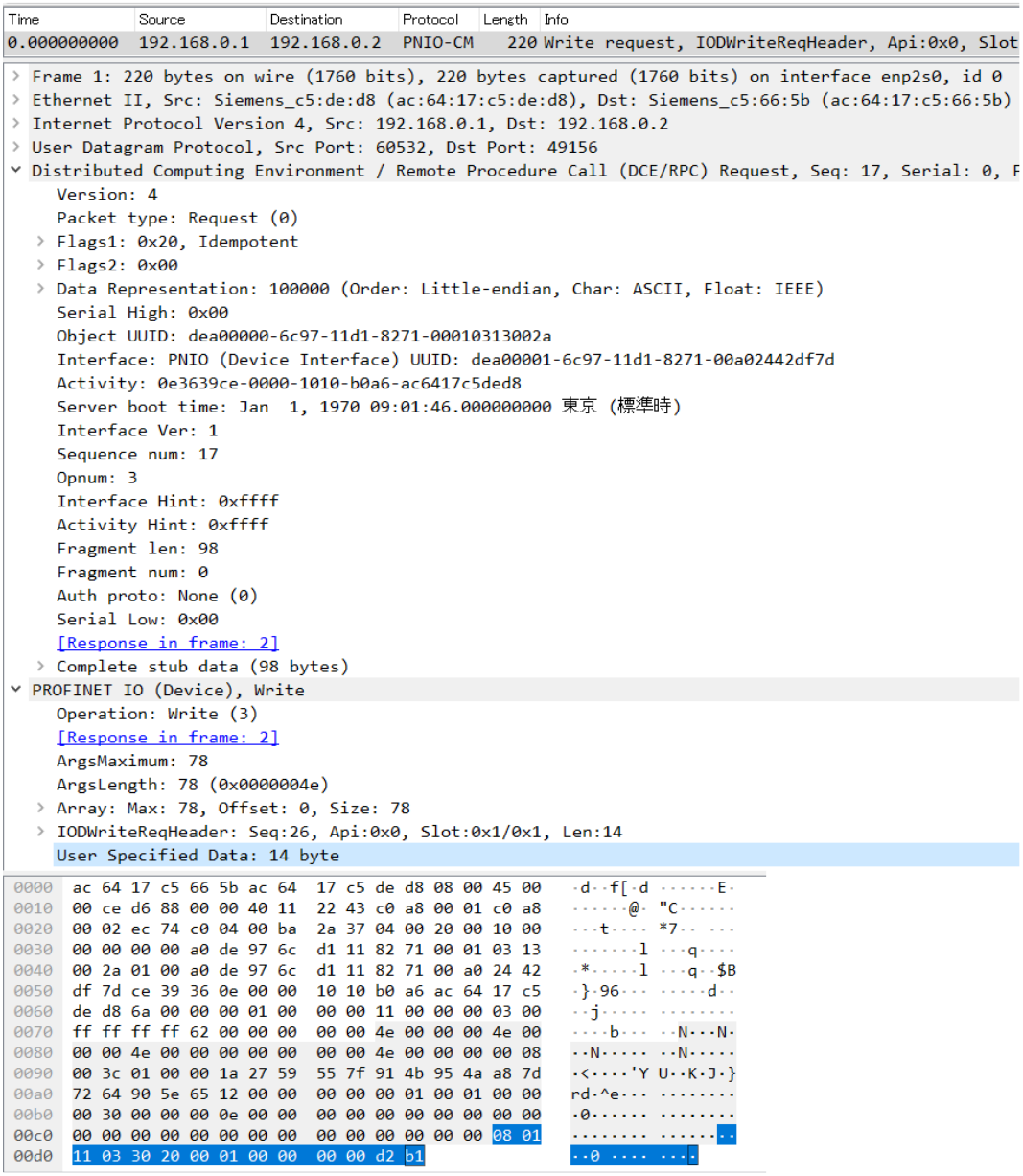
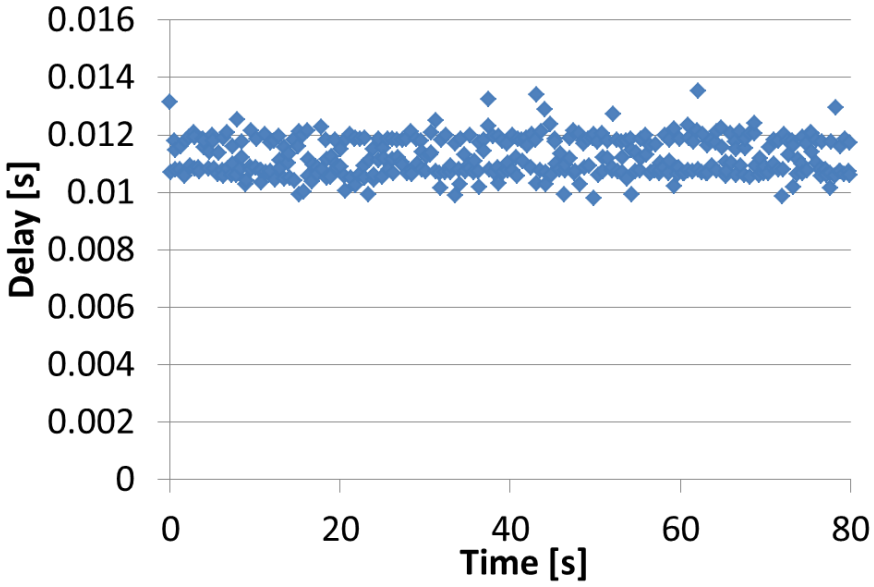
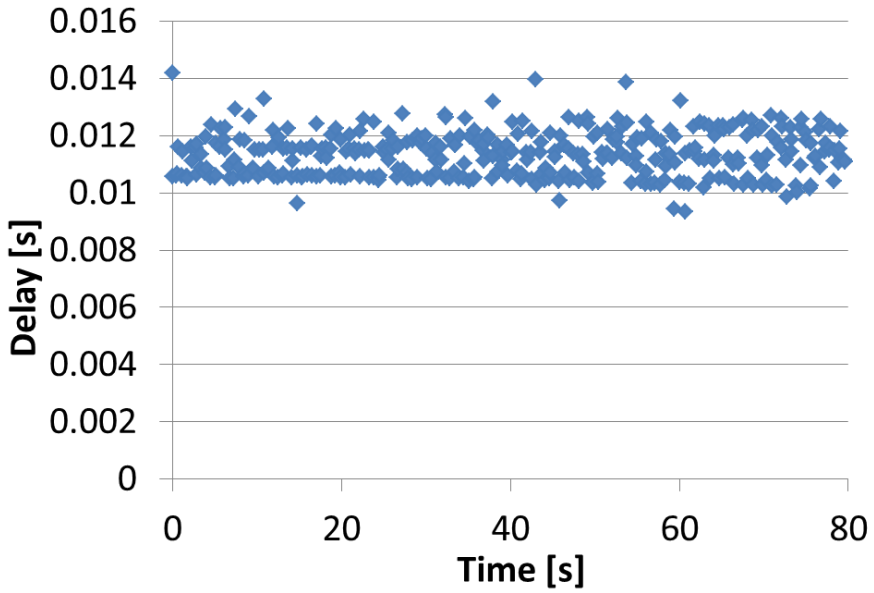


図 7-6　パケットの詳細

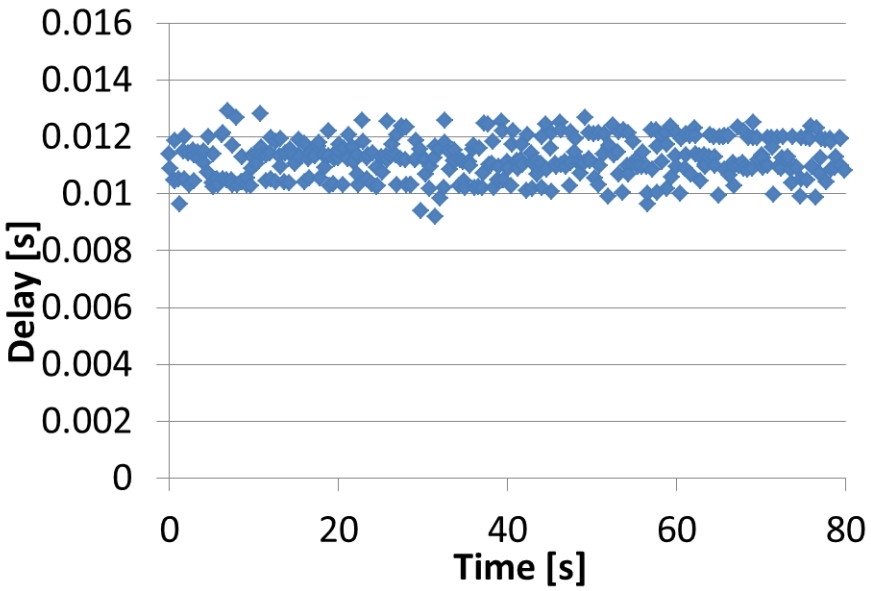
図 7-6 はモータへの入力制御値のパケット情報を解析したものである．PROFINET NRT パケットは UDP/IP フォーマットで，PROFINET 固有のデータはデータ部に格納されている．Object UUID や Activity，Server boot time や Sequence num といった PROFINET NRT プロトコルのデータはセッションに固有の値であり，適切に更新される必要がある．PROFINET のデータ部の末尾にはモータの制御値を格納し，PROFINET でのコマンドの転送を可能にする可変データ格納領域が存在する．これらの値は対象のアプリケーションによって記述形式が変わる部分である．



(a) PROFINET 専用スイッチを用いた場合



(b) OvS を用いた場合



(c) OvS に DPDK を併用した場合

図 7-7　遅延の時刻歴

表 7-1　遅延の比較結果

	Average delay [ms]	Standard deviation [ms]
PROFINET dedicated switch	11.2	0.66
OVS	11.3	0.78
OVS with DPDK	11.2	0.69

PROFINET 専用スイッチを OvS で置き換えた場合の遅延性能を，フォーマットやデータ量，測定時間などを同じパラメータとして公平な条件下で比較した．具体的には，図 7-7 と表 7-1 に示すように，それぞれのスイッチを使用した場合の PROFINET NRT パケットのネットワーク遅延の変化量を測定した．このときのネットワーク遅延は制御アプリケーションが制御値を送信し，返答を受信するまでの時間とした．図 7-7 (a) は PROFINET 専用スイッチである SCALANCE XC 206-2 を用いたときの遅延の時刻歴を示す．遅延は 11.2 ms で，標準偏差は 0.66 ms であった．図 7-7 (b) に OvS を用いたときの遅延の時刻歴を示す．遅延は 11.3 ms で，標準偏差は 0.78 ms であった．図 7-7 (c) に OvS と DPDK を併用したときの遅延の時刻歴を示す．遅延は 11.2 ms で，標準偏差は 0.69 ms であった．これらの結果から，ソフトウェアスイッチでは専用スイッチに比べて遅延やジッタがより発生するが，DPDK を用いることでパケット送受信処理を高速化することができ，専用スイッチと近しい遅延とジッタの値になることが示された．

実験 2

続いて，モーション制御アプリケーションにスミス予測器を用いることで制御性能を向上するか確認した．PI 制御のみを用いた場合とスミス予測器を用いた PI 制御を比較する．モータモデルは以下の式で表される．このとき，数値はモータのスペックシートから決定した．

$$\hat{G}_P(s) = \frac{195000}{s^2 + 114s + 150} \tag{7-1}$$

角度制御を行い，目標値を 2.0×10^6 ps とした．実験系を図に示す．

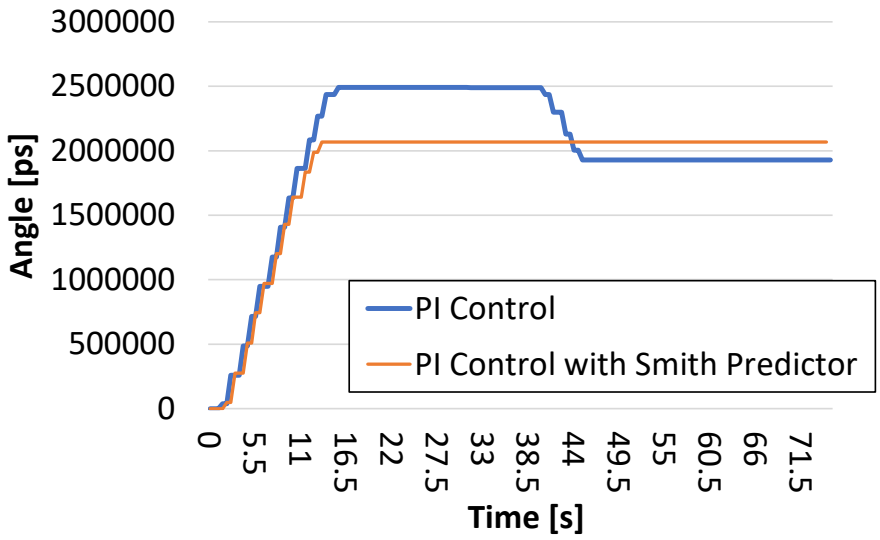


図 7-8　モータ制御結果

図 7-8 はモータの制御結果を示す．PI 制御の整定時間は 43.0 s であったのに対し，スミス予測器を用いた PI 制御は 11.5 s であった．遅延補償制御といった制御は既存の PLC では実装することが難しいが，汎用言語でソフト実装することで可能となる．

モータドライバはセンサ情報を得るために 4 回のデータ交換を必要とし，PROFINET を介したこれらの交換はデータ衝突を避けるためのスリープ時間も含めて，650 ms かかった．それゆえ，より高速なアプリケーションに変更することでより短い整定時間を達成できる可能性がある．全体として，これらの結果は SDN によってスミス予測制御といった高度な制御方式の導入が可能であることを示した．

実験 3

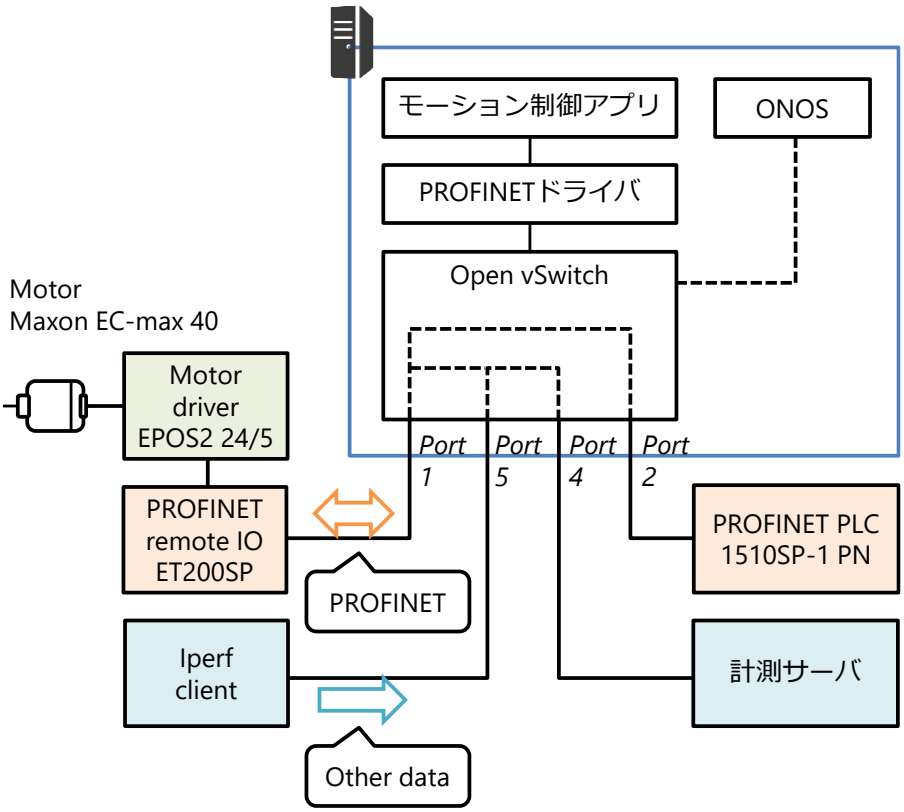


図 7-9　実験 3 の実験構成

最後に，制御プレーンからの工場ネットワークのデータプレーンの制御を検証するために，ONOS を用いた優先制御を行った．実験系を図 7-9 に示す．

PROFINET リモート I/O と iperf からのパケットは OvS を介して，測定サーバに伝送される．測定サーバでは Wireshark を用いて各ソースからのパケットの帯域幅を測定する．なお，すべてのラインレートは 1 Gbps である．OvS が ONOS を用いて PROFINET パケットの優先度を上げる場合と，優先度を設定しない場合で比較した．

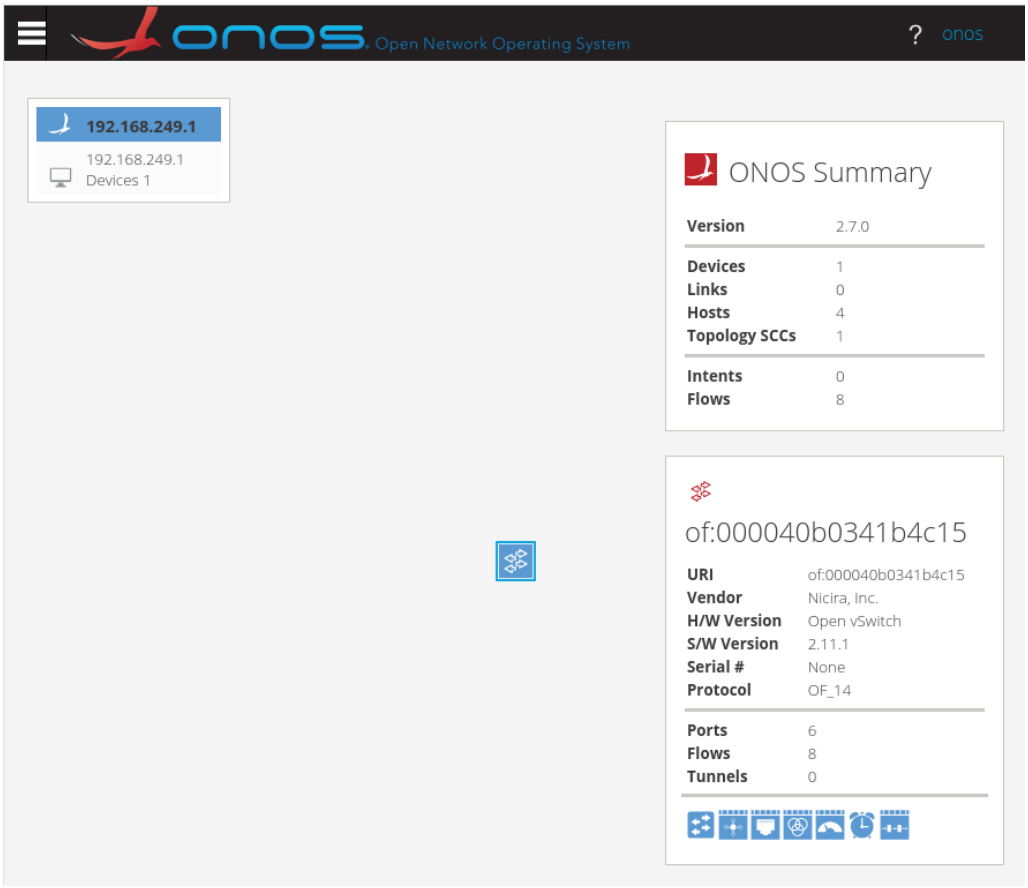
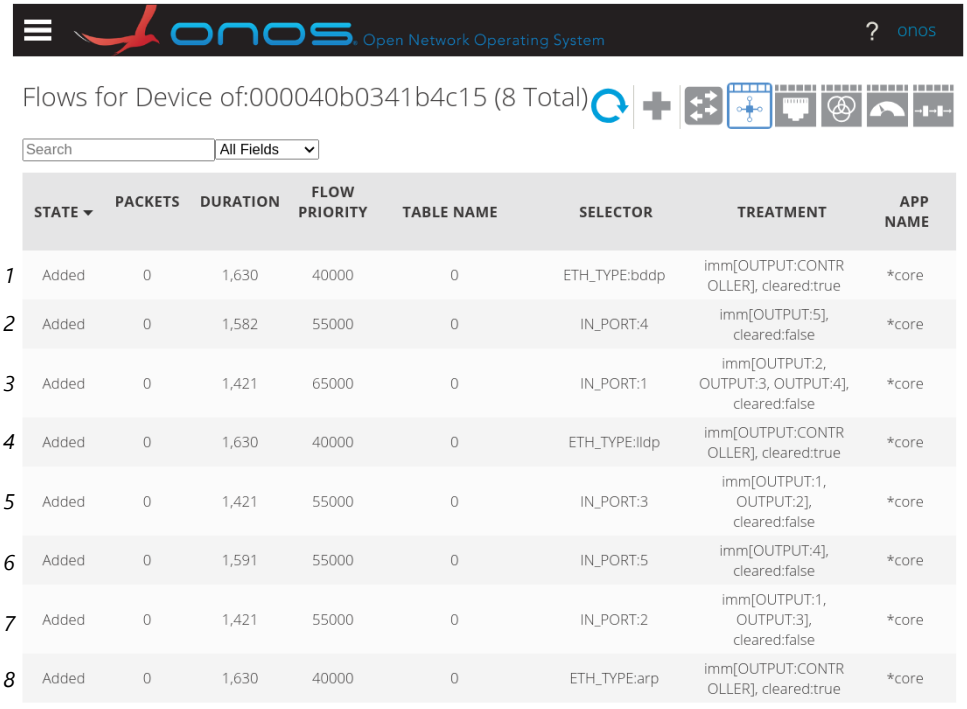


図 7-10　ONOS サマリー



	STATE	PACKETS	DURATION	FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT	APP NAME
Flow 1	Added	0	1,630	40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Flow 2	Added	0	1,582	55000	0	IN_PORT:4	imm[OUTPUT:5], cleared:false	*core
Flow 3	Added	0	1,421	65000	0	IN_PORT:1	imm[OUTPUT:2, OUTPUT:3, OUTPUT:4], cleared:false	*core
Flow 4	Added	0	1,630	40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true	*core
Flow 5	Added	0	1,421	55000	0	IN_PORT:3	imm[OUTPUT:1, OUTPUT:2], cleared:false	*core
Flow 6	Added	0	1,591	55000	0	IN_PORT:5	imm[OUTPUT:4], cleared:false	*core
Flow 7	Added	0	1,421	55000	0	IN_PORT:2	imm[OUTPUT:1, OUTPUT:3], cleared:false	*core
Flow 8	Added	0	1,630	40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true	*core

図 7-11　ONOS フロー

図 7-10, 7-11 は ONOS の画面である．OvS が OpenFlow スイッチとして認識され，フローの優先度が設定されていることが確認できることから，設定が正しく投入されていることが確認できる．図 7-10 より，OvS は ONOS 上で OpenFlow スイッチとして認識できている．図 7-11 は，OvS に投入されたフローを示しており，SELECTOR と TREATMENT で示されるポートが入力と出力を表し，Flow Priority がフローの優先度を示す．この実験では，Flow 3 で表されるリモート I/O からの PROFINET パケットと，Flow 6 で表される iperf からのパケットの間で優先度をつけ，優先度が同じ場合と，PROFINET パケットの優先度を上げた場合を比較した．その他のフローは OpenFlow ネットワークの通常の転送ルールが適用される．これらのフローは正しく投入された．

表 7-2　iperf 実験結果

	0 Gbps traffic by iperf	1 Gbps traffic by iperf
No priority control	256 kbps	230 kbps
Priority control to PROFINET packets by ONOS	256 kbps	256 kbps

表 7-2 に iperf クライアントから UDP パケットを送信したときの結果を示す．0 Gbps のとき，PROFINET パケットはどちらも 256 kbps で流れている．1 Gbps のとき，優先制御を行わなかった PROFINET トラフィックの帯域は 230 kbps に低下したが，優先制御を行った場合では，256 kbps を維持した．これらの結果から優先制御では輻輳が起こる状況でも，制御パケットを落とすことなく，継続してモーション制御が実現できることを示した．一方で，PROFINET パケットは 256 kbps と狭い帯域幅であったため，1 Gbps のフルラインレートを使用しない限り輻輳は発生せず，PROFINET パケットの帯域は低下しなかった．

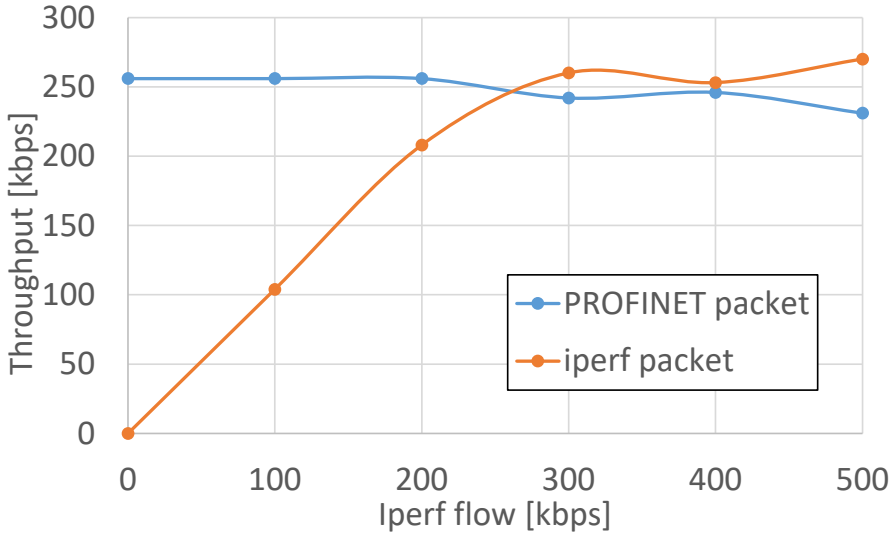


図 7-12　ネットワークコントローラがない場合の結果

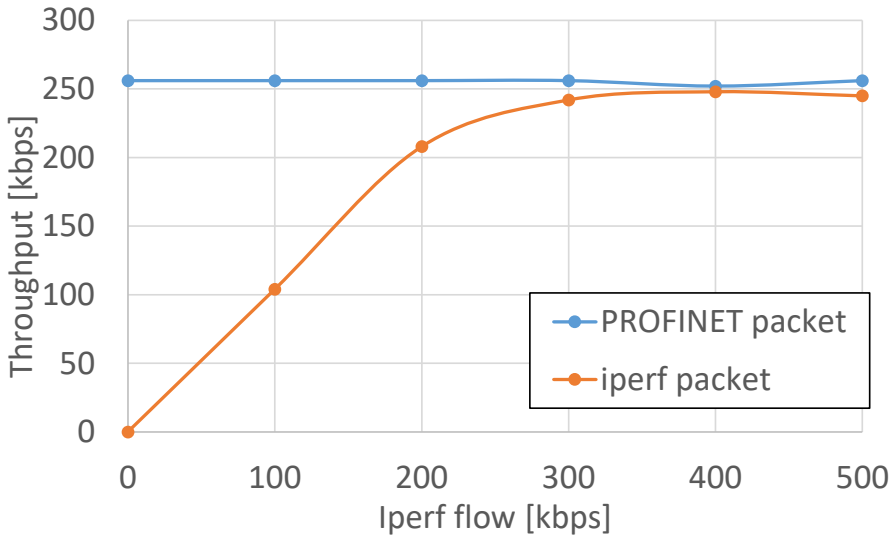


図 7-13　ネットワークコントローラがある場合の結果

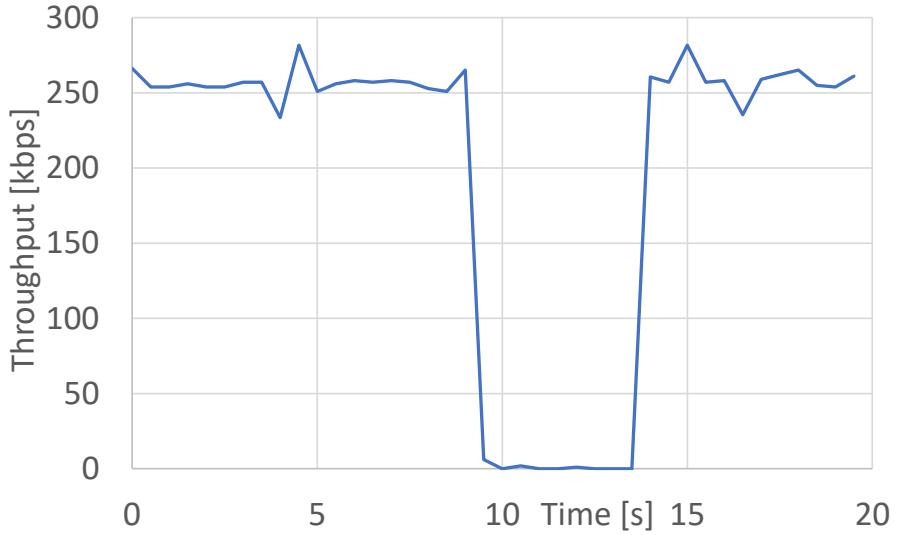


図 7-14　パケット導通が止まったときの時刻歴

図 7-12, 7-13 はラインレートが小さい場合を想定し，ポリシング設定によりラインレートを 500 kbps に絞ったときの結果を示している．優先制御がない場合では， iperf クライアントからのパケットの帯域が 300 kbps を超えると，PROFINET パケットの帯域が低下している．一方，優先制御を行った場合では，256 kbps を維持している．図 7-14 のように，優先制御を行わないとき，パケットロスが引き起こすシーケンス番号の齟齬によって，PROFINET システムが再起動し，パケットの導通が止まることを確認した．このような状態は制御システムの停止を引き起こすため，上位レベルでの優先制御が必要であることが示された．

議論

これらの実験では次の 3 つのポイントに注目した．

- まず，PROFINET NRT パケットが送受信可能なドライバを開発した．また，DPDK を用いたソフトウェアスイッチの遅延が 11.2 ms で専用スイッチと同等のパケット処理性能を持つことを確認した
- 汎用言語を用いた制御機能のソフト化により，これまでの PLC では難しかった複雑な制御を簡単に実現できることを検証するため，スミス予測制御を用いた遅延補償制御を行い，制御性能の向上により，整定時間を 43.0 s から 11.5 s に低減できることを確認した．
- 最後に，ネットワークコントローラとして ONOS を用い，制御関連のパケットのスループット低下を防ぐため，PROFINET RT で規定されている高い優先度の割当により優先制御を行い，優先制御なしの場合では 230 kbps に低下する場合でも，優先制御によって 256 kbps を維持できることを確認し，有効性を確かめた．

これらの実験により，提案アーキテクチャによって，汎用サーバ上で既存の専用装置の機能を実装し，ソフトウェア定義の産業用ネットワークを構築できることが実証された．

実際の産業用ネットワークへの適用という観点では，セキュリティや冗長性などの課題への対応も必要である．現在の工場ネットワークはクローズドな環境であることでセキュリティを担保しているが，外部ネットワークと接続されるとセキュリティに重大な影響を及ぼす可能性がある．この問題に対し，攻撃検知手法の利用といった研究が行われており，セキュリティに強いネットワークの構築にはこれらの手法を用いる必要がある [11].

7-4 結論

ソフトウェア定義の産業用ネットワークを構築するためのアーキテクチャを提案した．この目的のため，産業用イーサネットプロトコル機能をソフトウェア化することで制御プレーンを SDN に変換し，モーション制御機能を汎用言語でソフトウェア化して高度化し，データプレーンをネットワークコントローラから制御した．産業用イーサネットプロトコルを用いた実機実験系でモーション制御を行い，提案アーキテクチャの有効性を検証した．既存の産業用プロトコルをソフト化することでシステムやアプリケーションの変更なしに，ソフト定義の産業用ネットワークを構築することができる．実験結果より，提案アーキテクチャでは遅延補償制御を用いることで 11.5 s 以内にモータ制御を実現できることを示した．また，ONOS を用いて，制御関連のパケットの優先付けを行い，高負荷ネットワーク環境下でも制御が実現できることを検証した．

7-5 参考文献

[1] M. Becker, Z. Lu, and D. -J. Chen, “An Adaptive Resource Provisioning Scheme for Industrial SDN Networks,” 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), pp. 877–880, 2019.

[2] L. Moutinho, P. Pedreiras, and L. Almeida, “A Real-Time Software Defined Networking Framework for Next-Generation Industrial Networks,” IEEE Access, vol. 7, pp. 164468–164479, 2019.

[3] M. A. Metaal, R. Guillaume, R. Steinmetz, and A. Rizk, “Integrated Industrial Ethernet Networks: Time-Sensitive Networking over SDN Infrastructure for Mixed Applications,” 2020 IFIP Networking Conference (Networking), pp. 803–808, 2020.

[4] K. Ahmed, J. O. Blech, M. A. Gregory, and H. Schmidt, “Software Defined Networking for Communication and Control of Cyber-Physical Systems,” 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), pp. 803–808, 2015.

[5] F. Ansah et al., “Controller of Controllers Architecture for Management of Heterogeneous Industrial Networks,” 2020 16th IEEE International Conference on Factory Communication Systems (WFCS), pp. 1–8, 2020.

[6] T. Goldschmidt, M. K. Murugaiah, C. Sonntag, B. Schlich, S. Biallas, and P. Weber, “Cloud-Based Control: A Multi-tenant, Horizontally Scalable Soft-PLC,” 2015 IEEE 8th International Conference on Cloud Computing, pp. 909–916, 2015.

[7] T. Cruz, P. Simões, and E. Monteiro, “Virtualizing Programmable Logic Controllers: Toward a Convergent Approach,” IEEE Embedded Systems Letters, vol. 8, no. 4, pp. 69–72, 2016.

[8] A. Badar, D. Z. Lou, U. Graf, C. Barth, and C. Stich, “Intelligent Edge Control with Deterministic-IP based Industrial Communication in Process Automation,” 2019 15th International Conference on Network and Service Management (CNSM), pp. 1–7, 2019.

[9] J. Mellado and F. Núñez, “A Container-based IoT-oriented Programmable Logical Controller,” 2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS), pp. 55–61, 2020.

[10] D. Javier Perez, J. Waltl, L. Prenzel, and S. Steinhorst, “How Real (Time) Are Virtual PLCs?,” 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1–8, 2022.

[11] P. Verma, M. P. De Leon, J. G. Breslin, and D. O’Shea, “FedTIU: Securing Virtualized PLCs Against DDoS Attacks Using a Federated Learning Enabled Threat Intelligence Unit,” 2023 IEEE International Conference on Smart Computing (SMARTCOMP), pp. 233–236, 2023.

第 8 章　結論

本研究の結論を以下にまとめる．

IoT 端末の増加によって，今後 IoT 端末をネットワーク越しにモーション制御する需要が増加することが考えられる．IoT 端末のモーション制御では遅延が制御性能に大きく影響を与えるため，遅延の影響を低減することが求められる．

本研究ではアクセスシステム上の通信局舎やデータセンタ等のエッジサイト上に計算リソースを設置し，エッジコンピューティングにより IoT 端末のモーション制御を行う，アクセスエッジコンセプトを提案した．このコンセプトでは，ユーザと距離が近いアクセス部にリソースを設置することから，通信遅延をクラウドからの制御に比べて抑えることができ，リアルタイムモーション制御が可能になる．また，ローカルからの制御に比べて，リソースの拡張が容易であり，AI やビッグデータ等を活用した高度な制御が実現できるうえ，これまでユーザが管理していたリソースをキャリアやデータセンタ管理者に任せることができ CAPEX, OPEX の削減につながるといったメリットがある．アクセスエッジコンセプトの課題としては，アクセスシステムにもクラウドと比べて小さいながら遅延やジッタが存在することである．そこで本研究では制御性能に悪影響を及ぼす遅延・ジッタの影響を低減し，アクセスエッジコンセプトの技術確立を目標とした．

第 2 章では，アクセスエッジコンセプトの関連技術であるアクセスネットワーク，モーション制御技術，SDN 技術と産業用ネットワークについて述べ，本研究と既存研究との差分を明らかにした．

第 3 章では，光アクセスネットワークの Point-to-Point 構成を対象に，制御端末と制御器の間の経路上に存在する通信機器から直接処理情報を収集することで，経路全体の通信遅延を取得し，遅延補償制御に用いる手法を提案した．提案手法の有効性をモータとドローンを用いた実機実験により検証し，従来手法では制御できない状態でも 2.7 s の整定時間で制御できることを確認した．

第 4 章では，Point-to-Multipoint 構成を対象に，DBA アルゴリズムにより遅延が発生する PON システムにおいて，OLT から現在の設定パラメータを取得し，作成した遅延モデルに入力することで，遅延を予測し，遅延補償制御を行う手法を提案した．モータを用いた実機実験により有効性を検証し，従来手法では遅延変化に対応できず制御不能になる際に，提案手法では整定時間 4.0 s で制御できることを確認した．

第 5 章では，アクセスシステムの仮想化基盤である SEBA を用いることで，IoT 端末の制御結果をネットワーク設定にフィードバックし動的に変更することで，制御に最適なネットワークを構築する手法を提案した．提案手法の有効性をモータを用いた実機実験により検証し，整定時間が 3.0 s 以内になるように DBA 周期を更新し，5 回のフィードバックののち DBA 周期を 9 ms に設定完了することを確認した．

第 6 章では，次世代のネットワーク基盤であるオールフォトニクスネットワークを対象に，ネットワークで発生するジッタを低減するために，通信端末間の遅延を逐次取得・収

集し，End-to-End の通信遅延をリアルタイムに算出，この値を用いてジッタバッファを行うことで遅延を固定値にし，ジッタを抑えたネットワークを構築する手法を提案した．FPGA をもちいて提案手法を実装し，モータを用いた実機実験により，遅延が時変の場合でも，3.0 s で整定できることを確認した．

第 7 章では，これまで提案したアクセスエッジコンセプトをより要求の厳しい産業用ネットワークで実現し，End-to-End でのコンセプト実証を行うために，既存の産業用プロトコル機能をソフトウェアで実装し，汎用の計算リソースより制御する手法を提案した．モータを用いた実機実験により有効性を検証し，11.5 s で整定できることを確認した．

以上のように，本研究では光アクセスネットワークとモーション制御の連携により，遅延の制御に及ぼす影響を低減させる手法を提案し，IoT 端末を対象にした制御性能の検証を行った．第 3 章から第 6 章までの 4 つのアプローチにより，ネットワークの状態をモーション制御に受け渡すことで，ネットワークを考慮した制御を実現するとともに，ネットワークの状態がモーション制御に適切になるように，ネットワーク情報の収集・分析・制御を行い，モーション制御に特化した低遅延・低ジッタなアクセスネットワークを構築した．また，第 7 章ではアクセスエッジコンセプトを実現する具体的なユースケースを想定し，要求の厳しい産業用ネットワークを SDN 化することで，End-to-End でのアクセスエッジコンセプトを検証した．これらにより，光アクセスネットワーク上でのエッジコンピューティングによる End-to-End でのリアルタイムモーション制御を実現するアクセスエッジコンセプトを実証した．

謝辞

本論文を作成するにあたり，懇切なる御指導，および御鞭撻を賜りました北海道大学大学院情報科学研究科　吉田智暁　客員教授（日本電信電話株式会社　NTT アクセスサービスシステム研究所　光アクセス基盤プロジェクト　プロジェクトマネージャ）に謹んで深謝の意を表します．また，有益な御指導，および御助言を賜りました北海道大学大学院情報科学院　大鐘武雄　教授，齊藤晋聖　教授，土橋宜典　教授に謹んで深謝の意を表します．

本論文は，筆者が NTT アクセスサービスシステム研究所において，2017 年から取り組んでいる光アクセスネットワーク上でのリアルタイムモーション制御のためのエッジコンピューティング技術に関する研究成果をまとめたものです．論文化する機会を与えていただきました NTT 先端集積デバイス研究所／NTT デバイスイノベーションセンタ　寺田純　企画部長（元　光アクセス基盤プロジェクト　プロジェクトマネージャ）に深謝の意を表します．可児淳一　上席特別研究員（光アクセス基盤プロジェクト　光アクセス基盤 SE グループ　グループリーダ），ならびに島田達也　主席研究員（光アクセス基盤プロジェクト　光アクセス基盤高度化グループ　グループリーダ）には，本研究の機会を与えていただくとともに，研究開始当初から多くの御指導，および御激励をいただきました．ここに謹んで感謝の意を表します．

公立千歳科学技術大学　電子光工学科　山田崇史　教授（元　光アクセス基盤プロジェクト　光アクセス基盤高度化グループ　主幹研究員），NTT アクセスサービスシステム研究所 光アクセス基盤プロジェクト 光アクセス基盤高度化グループ　秦野智也　主任研究員，NTT アクセスサービスシステム研究所 光アクセス基盤プロジェクト 光アクセス基盤 SE グループ　鈴木貴大　特別研究員には入社以来，上司，および指導者として，アクセスネットワークの基礎知識から実験評価方法，技術文書の執筆，研究計画の策定法等，多岐にわたって懇切丁寧に御指導いただきました．心より感謝申し上げます．

最後に，いつも研究生活を支えてくれている妻　もえ，両親，義両親，友人に深く感謝の意を表し，本論文の謝辞といたします．

発表論文

筆頭著書

【論文】

[1] Y. Koyasako, T. Suzuki, S. Kim, J. Kani and J. Terada, “Motion Control System With Time-Varying Delay Compensation for Access Edge Computing,” IEEE Access, vol. 9, pp. 90669-90676, 2021.

[2] Y. Koyasako, T. Suzuki, T. Yamada, T. Shimada and T. Yoshida, “Demonstration of Real-Time Motion Control Method for Access Edge Computing in PONs,” IEEE Access, vol. 10, pp. 168-175, 2022.

[3] Y. Koyasako, T. Suzuki, T. Hatano, T. Shimada and T. Yoshida, “Demonstration of real-time motion control on a 10G-EPON edge computing platform with SDN enabled broadband access,” Journal of Optical Communications and Networking, vol. 14, no. 12, pp. 951-959, 2022.

[4] Y. Koyasako, T. Suzuki, T. Hatano, T. Shimada and T. Yoshida, “Demonstration of real-time jitter-buffered architecture that collects network information for deterministic optical communication,” Journal of Optical Communications and Networking, vol. 16, no. 2, pp. 171-179, 2024.

[5] Y. Koyasako, T. Suzuki, T. Hatano, T. Shimada and T. Yoshida, “Demonstration of Industrial Ethernet Protocol Softwarization and Advanced Motion Control for Full Software-Defined Factory Network,” IEEE Access, vol. 12, pp. 104020-104030, 2024.

【査読付き国際会議】

[1] Y. Koyasako, T. Suzuki, S. Kim, J. Kani and J. Terada, “Real-Time Motion Control Method Using Measured Delay Information on Access Edge Computing,” 2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC), pp. 1-4, 2020.

[2] Y. Koyasako, T. Hatano, T. Yamada, T. Shimada and T. Yoshida, “Demonstration of Real-Time Jitter Buffered Architecture for Motion Control in All-Photonics Network,” GLOBECOM 2022 - 2022 IEEE Global Communications Conference, pp. 4292-4297, 2022.

[3] Y. Koyasako, T. Suzuki, T. Hatano, T. Shimada and T. Yoshida, “Full Software-Defined Factory Networks by Industrial Ethernet Protocol Softwarization,” 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), pp. 885-886, 2023.

[4] Y. Koyasako, T. Suzuki, T. Hatano, T. Shimada and T. Yoshida, “Real-Time Software Motion Control Method of Massive Devices for Industrial Networks,” 2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC), 2025.

【国内講演】

[1] 小屋迫優士, 鈴木貴大, 金相燁, 可児淳一, 大高明浩, “光アクセス NW を介したリアルタイム制御に向けた遅延の影響評価,” 電子情報通信学会ソサイエティ大会, A-15-11, 2018.

[2] 小屋迫優士, 鈴木貴大, 金相燁, 可児淳一, 寺田純, “アクセスエッジにおける遅延情報を用いたリアルタイム遠隔制御,” 電子情報通信学会総合大会, A-17-5, 2020.

[3] 小屋迫優士, 鈴木貴大, 秦野智也, 島田達也, 吉田智暁, 山田崇史, “産業用イーサネット機能の SDN 化によるマルチプロトコルの実現,” 電子情報通信学会総合大会, B-8-17, 2023.

[4] 小屋迫優士, 原田拓弥, 宮本健司, 秦野智也, 島田達也, 吉田智暁, “高度な遠隔操作実現に向けたベンチマークモデルおよび映像転送性能に関する一検討,” コミュニケーションシステム研究会, CS2023-46, 2023.

連名著書

【論文】

[1] T. Suzuki, S. Kim, Y. Koyasako, J. Kani and J. Terada, “Demonstration of Adaptive Image Transmission That Meets Various Application Requirements in 10G-EPON,” IEEE Access, vol. 8, pp. 186433-186440, 2020.

[2] T. Suzuki, Y. Koyasako, K. Nishimoto, K. Asaka, T. Yamada, J. Kani, T. Shimada, T. Yoshida, “Zero touch provisioning compliant with authentications of IEEE PON packages A and B for SDN-enabled broadband access,” Journal of Optical Communications and Networking, vol. 13, no. 11, pp. 244-252, 2021.

[3] T. Suzuki, Y. Koyasako, K. Nishimoto, K. Asaka, T. Yamada, J. Kani, T. Shimada, T. Yoshida, “Demonstration of IEEE PON Abstraction for SDN Enabled Broadband Access (SEBA),” Journal of Lightwave Technology, vol. 39, no. 20, pp. 6434-6442, 2021.

[4] T. Suzuki, Y. Koyasako, S. Kim, J. Kani and T. Yoshida, “Real-time demonstration of industrial protocol applications by PHY softwarization for fully virtualized access networks,” Journal of Optical Communications and Networking, vol. 15, no. 7, pp. 449-456, 2023.

【査読付き国際会議】

[1] K. Nishimoto, Y. Koyasako, T. Yamada, J. Kani and A. Otaka, “Software module-based 10G Optical Line Terminal for enhancing flexibility of access networks,” 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), pp. 500-505, 2018.

[2] T. Suzuki, S. Kim, Y. Koyasako, J. Kani and J. Terada, “Application-Oriented Optical Transmission Control for Computationally Efficient Edge Computing,” 2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC), pp. 1-2, 2020.

[3] T. Suzuki, Y. Koyasako, S. Kim, J. Kani and T. Yoshida, “Demonstration of Industrial Network Applications by PHY Softwarization for Fully Virtualized Access Networks,” 2023 Optical Fiber Communications Conference and Exhibition (OFC), pp. 1-3, 2023.

[4] S. Otaki, T. Yamada, Y. Koyasako, T. Hatano, T. Shimada and T. Yoshida, “Implementing EtherCAT Protocol for the Virtualization of Industrial Networks,” 2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC), 2025.