G-19

e スポーツ大会における拠点間遅延の均一化のための動的遅延補正システムの提案

A Proposal for a Dynamic Latency Compensation System to Equalize Location Delays in eSports Tournaments

温井 直輝 †　谷口 義明 ‡ §　井口 信和 ‡ §　矢藤 邦治 ¶
Naoki Nukui　Yoshiaki Taniguchi　Nobukazu Iguchi　Kuniharu Yato

1. はじめに

近年、e スポーツはプロチームの活躍や全国規模の大会開催により、国内外で急速に市場を拡大している[1][2]。市場調査によれば、国内 e スポーツ市場規模は拡大傾向にあり、2025 年度は 200 億円に達すると見込まれている[3]。このような市場拡大に伴い大会規模も拡大し、従来のオフライン会場での開催に加え、地理的に離れた拠点間でのオンライン開催が普及している。例えば、NASEF JAPAN が主催する全日本高校 e スポーツ選手権（NHEC）では予選はオンライン形式で実施され、決勝はオフライン形式で実施するハイブリット形式を採用している[4]。また、STAGE:0（ステージゼロ）全国高校対抗 e スポーツ大会においても、予選はオンライン形式で実施され、グランドファイナルはオフライン形式で実施される[5]。

しかしながら、拠点間でのオンライン開催では、各拠点のネットワーク環境の違いにより遅延差が生じるため、競技の公平性に影響を与える。拠点間に遅延差が生じる要因として、（1）サーバとの地理的距離による物理的遅延、（2）インターネット回線の品質、（3）ネットワーク機器の処理能力、（4）ファイアウォールや Deep Packet Inspection(DPI)による処理遅延の有無が挙げられる。このようなネットワーク遅延の差異は、e スポーツにおいて深刻な問題を引き起こす。特に FPS や格闘ゲームでは、わずかな遅延差が競技結果に影響を与える要因となる[6][7]。既存の遅延対策手法として、専用線の利用や地理的に中央に位置するサーバの選択などが試みられているが、完全な遅延均一化は困難であり、導入コストや運用の複雑さといった課題がある。

そこで本研究では、オンライン開催における拠点間ネットワーク遅延の差異を補正することを目的に、Simple Two-way Active Measurement Protocol（STAMP）を用いた双方向遅延測定と、Data Plane Development Kit（DPDK）による動的遅延補正を組み合わせたネットワーク遅延均一化システムを提案する[8][9]。これにより、全選手が同一条件下での競技が可能となり、競技としての公平性向上が期待できる。

†近畿大学大学院総合理工学研究科，Graduate School of Science and Engineering, Kindai University
‡ 近畿大学情報学部，Faculty of Informatics, Kindai University
§近畿大学情報学研究所，Cyber Informatics Research Institute, Kindai University
¶近畿大学情報学部学生センター，Student Center, Faculty of Informatics, Kinki University

2. 関連研究・システム

NTT が開発する Innovative Optical and Wireless Network（IOWN）構想の一環として、オールフォトニクス・ネットワーク（APN）を活用した遠隔 e スポーツ支援システムが提案されている[10]。APN は光信号による直接伝送により従来の IP 通信と比較して大幅な低遅延化を実現する技術である。APN を用いた支援システムでは、拠点間の遅延差による競技の不公平性を解決するため、動的な遅延補正機能が組み込まれている。このシステムはネットワーク遅延測定機能と遅延補正機能を備えており、1 マイクロ秒精度でネットワーク遅延を測定し、必要に応じて遅延を補正することが可能である。

Riot Games は、League of Legends の専用システムとして「Latency Service」を開発している[11]。このシステムはオフラインで参加するチームとオンラインで参加するチームとのネットワーク遅延差を補正することを目的に開発されたシステムである。クライアント・サーバ間のネットワークスタックに統合された測定・制御機構により、全選手の ping 値を継続的に取得し、既定の目標遅延値を目標として遅延を補正することで、競技の公平性を確保している。

Zander らは、自動レイテンシーバランシングによるマルチプレイヤーネットワークゲームの公平性を確保するシステムを提案している[12]。この研究では、ゲームサーバとクライアント間に配置される Self-Adjusting Game Lagging Utility（SAGLU）を開発し、各選手の遅延差を動的に補正する手法を実証した。SAGLU は各選手の遅延をゲームサーバのリモートコンソール（rcon）インターフェースから継続的に測定し、最高遅延値を目標として遅延を補正することで均一化する。実験では、クライアントサイドボット（GoodBot）を用いた評価手法を導入し、Quake 2 環境において遅延差による不公平性の発生と SAGLU による補正効果を統計的に実証している。

これらの既存手法は遅延補正において成果を示しているが、それぞれ固有の限界を有している。APN は専用の光ネットワークインフラを必要とし、導入コストと運用の複雑性が課題となる。Riot Games の手法は特定のゲームタイトルに最適化されており、汎用性に制約がある。Zander らの手法は遅延均一化の有効性を実証した研究であるが、FreeBSD の dummynet の性能に依存するため、現代の e スポーツ環境への適用には課題が残されている。

本研究では、STAMP によるリアルタイムかつ継続的な双方向遅延測定を基盤に、DPDK を活用した動的遅延補正を組み合わせることで、高精度かつ汎用性の高いネットワーク遅延均一化システムの構築を目指す。

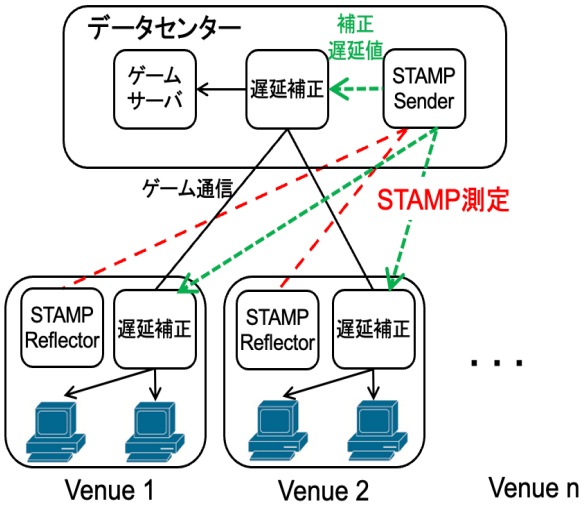

図 1 システム構成図面

3. 提案システム

本章では、まず本システムのシステム構成について述べ、次に STAMP による双方向遅延測定、測定結果に基づく補正遅延の算出および遅延補正について説明する。

3.1 システム構成

本システムの構成を図 1 に示す。本システムは、データセンターと複数の拠点に分かれて動作する分散型アーキテクチャで構成される。データセンター側には、ゲーム状態を管理するゲームサーバ、ゲームサーバへのパケットの送信タイミングを制御する遅延補正モジュール、および測定パケットを各拠点に送信する STAMP Sender モジュールが動作する。

一方、各会場側には、選手が操作するゲーミング PC、ゲーミング PC へのパケットの送信タイミングを制御する遅延補正モジュール、および STAMP Sender から受信した測定パケットを返送する STAMP Reflector モジュールが動作する。

本システムでは、データセンターの STAMP Sender と各拠点の STAMP Reflector の間で双方向の遅延測定を実施し、その測定結果をもとに遅延補正モジュールで動的な遅延補正を行う。このとき、データセンターから拠点方向（下り）および拠点からデータセンター方向（上り）の双方向で遅延を測定することで、ネットワーク遅延の非対称性に対応する。この測定結果に基づいて、上り・下りの双方向に適用する補正遅延値を算出する。上りの補正遅延値はデータセンター側の遅延補正モジュールに送信され、拠点からデータセンターへ送信されるパケットを補正遅延値に基づいてキューに保持し、送信タイミングを制御する。同様に、下りの補正遅延値は各拠点側の遅延補正モジュールに送信され、データセンターから拠点へ送信されるパケットの送信タイミングを制御する。

3.2 STAMP による双方向遅延測定

STAMP は、従来の Two-Way Active Measurement Protocol（TWAMP）からセッション管理機能を排除し、双方向遅延測定に必要な最小限の機能のみで構成されたプロトコルである[13]。TWAMP では、制御セッションの確立・管理の処理が必要となり、実装が複雑になる。一方、STAMP は制御セッションが不要であり、シンプルな設計となっている。本研究では、遅延測定方式に STAMP を採用する。

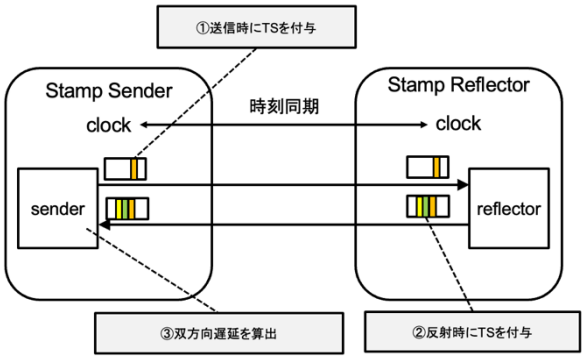

図 2 STAMP の動作概念

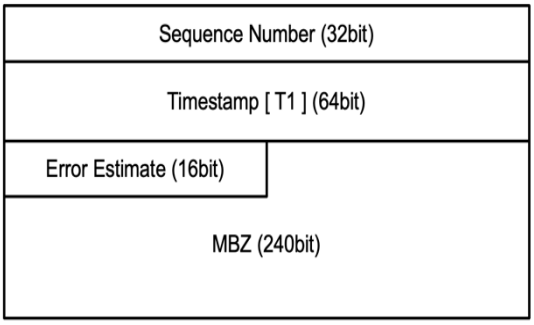

図 3 STAMP 要求パケットの構造

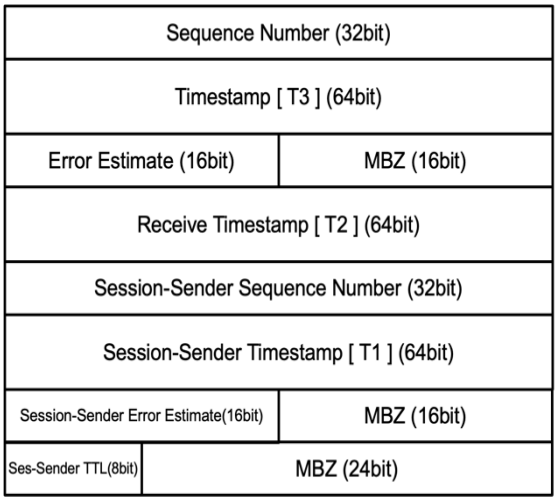

図 4 STAMP 返送パケットの構造

STAMP の動作概念を図 2 に示す。STAMP は、Sender と Reflector の 2 つのモジュールで構成され、Sender が送信する要求パケットに対して Reflector が返信パケットを返すシンプルなフローに基づいて動作する。

Sender は、図 3 に示す要求パケットを生成し、パケット内の Timestamp[T_1]フィールドに送信時刻 T_1を記録して Reflector に送信する。Reflector はこのパケットを受信すると、T_1を Session-Sender Timestamp[T_1]フィールドとして保持しつつ、受信時刻 T_2を取得する。その後、図 4 に示す返信パケットを生成し、パケット内の Receive Timestamp[T_2]フィールドに受信時刻 T_2を記録し、Timestamp[T_3]フィールドに送信時刻 T_3を記録して Sender に送信する。Sender は返信パケットを受信した時に受信時刻 T_4を保持し、T_1～T_4の

タイムスタンプに基づいて以下のように遅延を計算する。

下り方向遅延（DC→会場）：$\delta_f = T_2 - T_1$
上り方向遅延（会場→DC）：$\delta_r = T_4 - T_3$

また、本システムでは、高精度な遅延測定を行うため、Precision Time Protocol（PTP）を用いて STAMP Sender と STAMP Reflector でタイムスタンプの同期を行う。NTP では、時刻同期に数ミリ秒レベルの誤差が発生する可能性がある。PTP を活用することで、マイクロ秒レベルの同期精度でのタイムスタンプ比較を可能とする。

3.3 補正遅延値の算出

補正遅延値は、STAMP 測定によって得られた結果をもとに算出される。これは、各遅延補正モジュールが適用すべき待機時間として用いられる。

本システムでは、ネットワーク遅延の時間的変動を考慮し、一定周期ごとに STAMP による測定および補正遅延値の再計算を行う。測定時刻を $t_k(k = 1,2,\dots)$とし、各周期において、遅延推定値および目標遅延値をもとに補正遅延値を動的に更新する。これにより、時間帯やトラフィック状況によって変動するネットワーク特性に対しても、常に最適な遅延補正を維持することが可能となる。

まず、各会場 i において、データセンターからの下り方向遅延 $\delta_{f,i}$および会場からの上り方向遅延$\delta_{r,i}$に対して、周期 t_kにおける複数回の測定値から、それぞれの中央値を算出し、遅延推定値$\tilde{\delta}$とする。

$$\tilde{\delta}_{f,i}(t_k) = median\{\, \delta_{f,i}^{(1)}(t_k), \delta_{f,i}^{(2)}(t_k), \cdots, \delta_{f,i}^{(n)}(t_k) \,\}$$
$$\tilde{\delta}_{r,i}(t_k) = median\{\, \delta_{r,i}^{(1)}(t_k), \delta_{r,i}^{(2)}(t_k), \cdots, \delta_{r,i}^{(n)}(t_k) \,\}$$

次に、すべての拠点における上下方向の遅延推定値を比較し、それぞれの最大値を求め、目標遅延値$\Delta(t_k)$とする。

$$\Delta_f(t_k) = max\{\, \tilde{\delta}_{f,i}(t_k) \,\}$$
$$\Delta_r(t_k) = max\{\, \tilde{\delta}_{r,i}(t_k) \,\}$$

最後に、各会場における遅延推定値と目標遅延値との差分を計算し、補正遅延値 $D(t_k)$を算出する。

$$D_{f,i}(t_k) = \Delta_f(t_k) - \tilde{\delta}_{f,i}(t_k)$$
$$D_{r,i}(t_k) = \Delta_r(t_k) - \tilde{\delta}_{r,i}(t_k)$$

算出後、下り方向の補正遅延値$D_{f,i}(t_k)$ は、拠点側の遅延補正モジュールに送信され、ゲームサーバからのパケットをキューに保持する時間として使用される。一方、上り方向の補正遅延値$D_{r,i}(t_k)$ はデータセンター側の遅延補正モジュールに送信され、各拠点からのパケットをキューに保持する時間として使用される。

3.4 遅延補正

遅延補正モジュールでの遅延補正に DPDK を使用する。DPDK は、ユーザ空間で動作する高速パケット処理フレームワークであり、カーネルバイパスによって Network Interface Card（NIC）との直接的な通信を実現するライブラリ群を提供する。本システムでは、DPDK の高速パケット処理機能を活用して、遅延補正を行う。

設計概要を図 5 に示す。本設計は、NIC からのパケット受信（RX スレッド）、遅延待機・時刻判定（WORKER スレッド）、パケット送信（TX スレッド）の 3 つのスレッドで構成される。パケットは、RX スレッド、WORKER スレッド、TX スレッドの順で処理される。各スレッド間は、Queue0 と Queue1 によって連携される。

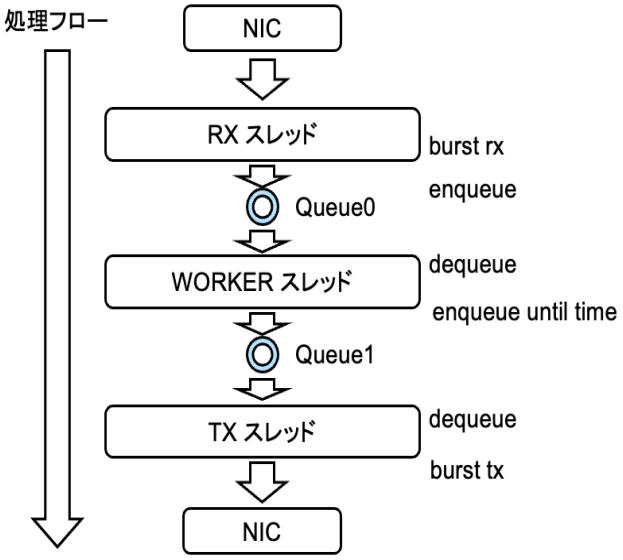

図 5 遅延補正モジュールの設計概要

3.4.1 RX スレッド

RX スレッドは、NIC からパケットを受信し、送信予定時刻を Time Stamp Counter（TSC）で記録する役割を持つ。具体的には、DPDK の rte_eth_rx_burst() 関数を用いて各パケットを取得し、各パケットの到着時刻に補正遅延値Dを加算することで、送信予定時刻を算出する。この送信予定時刻は、DPDK の rte_mbuf のユーザ定義のメタデータ領域に記録する。その後、各パケットは送信予定時刻を保持した状態で Queue0 に投入される。Queue0 へ各パケットを投入後は、遅延処理を担当する WORKER スレッドに処理を引き継ぐ。また、本スレッドは専用の CPU コア上で常時動作し、スレッド内で他の処理を行わないことで、RX 処理の安定性とスループットを確保する。

3.4.2 WORKER スレッド

WORKER スレッドは、Queue0 から各パケットを取得し、送信タイミングの制御を行う役割を持つ。パケットには、送信予定時刻が、メタデータ領域に記録されている。本スレッドでは、現在の TSC 値とパケットに記録された送信予定時刻を比較し、送信予定時刻に達していないパケットは再度 Queue0 に戻す。一方、送信予定時刻を超過したパケットは Queue1 に投入される。Queue1 へ各パケットを投入後は、送信処理を担当する TX スレッドに処理を引き継ぐ。この処理により、各パケットはあらかじめ設定された送信予定時刻を超過時にのみ送信処理へ進む構造となり、送信タイミングの制御を可能にする。

さらに本設計では、Queue0 にパケットが格納されている状態で補正遅延値Dが動的に変更された場合、すでに Queue0 内にあるパケットに対しても新しい補正遅延値 Dを反映し、送信予定時刻を再計算する。これにより、補正遅延の変更がパケットの送信順序やタイミングに与える影響を最小限に抑える。また、本スレッドも同様に専用の CPU

コア上で常時動作する。

3.4.3 TX スレッド

TX スレッドは、Queue1 からパケットを取得し、NIC へ送信する役割を持つ。具体的には、DPDK が提供する rte_eth_tx_burst() 関数を用いて、各パケットを NIC の送信リングバッファに登録することで、パケットを送信する。また、他のスレッドと同様に、本スレッドも同様に専用の CPU コア上で常時動作する。

4. 実験

本章では、今後予定している実験について説明する。実験として、本システムが正常に動作していることを確かめるための STAMP による遅延測定の動作確認および遅延補正モジュールの動作確認を予定している。

4.1 STAMP による遅延測定の動作確認

STAMP の動作確認として、意図的に遅延を導入した仮想ネットワーク環境を構築し、STAMP による測定結果が遅延設定値と一致するかを評価する。具体的には、仮想ネットワーク環境上において、遅延を Linux の tc コマンドを用いて意図的に付与し、STAMP で測定された下り・上り遅延がそれぞれの設定値とどの程度一致しているかを確認する。

4.2 遅延補正モジュールの動作確認

本実験では、提案システムにおける DPDK ベースの遅延補正モジュールが、補正遅延値に基づき正しくパケットの送信タイミングを制御できているかを確認する。また、補正遅延値が動的に変更された場合に、すでにキューに格納されているパケットに対して新たな補正遅延値が適用され、送信予定時刻が再計算されているかを検証する。

そのために、以下のような複数拠点をクラウド環境上で再現し、実験する。

パターン 1：DC→拠点 1＝10ms、拠点 1→DC＝12ms
パターン 2：DC→拠点 2＝25ms、拠点 2→DC＝30ms
パターン 3：DC→拠点 3＝5ms、拠点 3→DC＝8ms
⋮
パターン n：DC→拠点 n＝30ms、拠点 n→DC＝25ms

この環境下で、本システムとは独立した STAMP 測定を並行して実施する。これは、補正モジュールの効果を客観的に評価し、補正前後の遅延変化を定量的に把握するためである。

実験では、まず遅延補正機能を無効にした状態で各拠点の初期遅延を測定する。その後、遅延補正を有効化し、同様の測定を再度実施する。補正後の結果から、拠点間の遅延差が適切に補正されているかを確認する。さらに、パケットの順序性の維持についても検証する。送信側ではパケットに連続したシーケンス番号を付与し、受信側でその順序が正しく保たれているかをチェックすることで、補正処理が順序整合性に影響を及ぼしていないかを評価する。

5. まとめ

本研究では、e スポーツ大会のオンライン開催における拠点間ネットワーク遅延の差異という課題に対し、STAMP プロトコルによる双方向遅延測定と DPDK を活用した動的遅延補正を組み合わせたネットワーク遅延均一化システムを提案した。これにより、全選手が同一ネットワーク条件下での競技参加が可能となり、競技環境の公平性向上が期待できる。

今後は、提案システムの各モジュールの実装を行う予定である。また、予定している実験を実施することにより、STAMP による遅延測定の精度検証および遅延補正モジュールの動作確認を行い、提案システムの有効性を評価する。

参考文献

[1] eSports Market Size, Share, and Industry Analysis, By Streaming Type (Live and On-demand),By Revenue Streaming (Media Rights, Advertisement, Sponsorship, Ticket & Merchandise, Game Publisher Fees, and Others), By Gaming Genre (Real-Time Strategy Games, First Person Shooter Games, Fighting Games, Multiplayer Online Battle Arena Games, Mass Multiplayer Online Role-Playing Games, and Others), and Regional Forecast, 2024-2032, 入手先< https://www.fortunebusinessinsights.com/esports-market-106820 >, (参照 2025-7-20)

[2] 日本の e スポーツの発展に向けて～更なる市場成長、社会的意義の観点から～, 入手先< https://jesu.or.jp/wp-content/uploads/2020/03/document_01.pdf >, (参照 2025-7-20)

[3] 日本 e スポーツ白書 2024, 発行元:一般社団法人 日本 e スポーツ連合, 入手先< https://jesu.or.jp/contents/news/news-250331/ >, 2024

[4] NASEF JAPAN 全日本高校 e スポーツ選手権, 入手先< https://www.nasef-nhec.jp/ >, (参照 2025-7-20)

[5] STAGE:0（ステージゼロ）｜全国高校対抗 e スポーツ大会 公式サイト, 入手先< https://stage0.jp/ >, (参照 2025-7-20)

[6] e スポーツイベントに必要な通信環境とは？ 開催のポイントをプロに聞いてみた, 入手先< https://www.nttbp.net/column/blog/2022/12/post-94.html >, (参照 2025-7-20)

[7] Esports latency curse: How to stop lag from ruining matches, 入手先< https://esportsinsider.com/how-to-make-esports-matches-lag-free >, (参照 2025-7-20)

[8] RFC 8762 - Simple Two-Way Active Measurement Protocol, < https://datatracker.ietf.org/doc/html/rfc8762 >, (発行日 2020-3)

[9] DPDK, < https://www.dpdk.org/ >, (参照 2025-7-20)

[10] e スポーツを支える光ネットワーク技術を世界に先駆け実証～複数会場の e スポーツイベントでも単一会場のようなゲーミング UX を実現～, 入手先< https://group.ntt/jp/newsrelease/2021/11/02/211102b.html >, (参照 2025-7-20)

[11] Riot Games Tech Blog:Artificial Latency for Remote Competitors,入手先< https://lolesports.com/en-US/news/riot-games-tech-blog-artificial-latency-for-remote-competitors >, (参照 2025-7-20)

[12] Sebastian Zander, Ian Leeder, Grenville J. Armitage.: “Achieving fairness in multiplayer network games through automated latency balancing”, Proceedings of the International Conference on Advances in Computer Entertainment Technology, 2005

[13] RFC 5357 - A Two-Way Active Measurement Protocol, < https://datatracker.ietf.org/doc/html/rfc5357 >, (発行日 2008-10)