社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

エッジコンピューティング環境におけるサービス機能の配置がユーザの通信品質に与える効果の評価

金田　純一†　荒川　伸一†　村田　正幸†

† 大阪大学 大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1–5
E-mail: †{j-kaneda,arakawa,murata}@ist.osaka-u.ac.jp

あらまし　近年、モバイル端末に付属するカメラやセンサーが取得した情報を、別拠点で処理し、結果をエンド端末へ提示するサービスが考えられている。しかし、データセンタ集中型のサービス提供形態では、地理的要因や負荷の集中により利用者が享受する遅延が増大する。このような問題に対応すべく、エッジコンピューティングの導入が進められている。エッジコンピューティングでは、サービス機能を仮想化しネットワーク外縁部に配置することで、地理的な遅延の解消および負荷の分散によるサービスに対する応答性向上が期待されている。しかし、エッジコンピューティングによって応答性向上が期待される一方で、仮想化環境でのソフトウェア動作による処理速度の低下が懸念される。そこで、本稿では、実機を用いたエッジコンピューティング環境を構築し、サービスを柔軟に提供した際に生じるユーザの通信品質に与える効果を明らかにした。その結果、遠隔地から利用者に近い拠点にサービス機能を配置することにより、利用者が享受する遅延が最大でおよそ 30%低減され、ユーザの通信品質が改善されることが明らかとなった。

キーワード　エッジコンピューティング、ネットワーク機能仮想化 (NFV)、OpenStack、通信品質

On the Effect of Service Function Placement in Edge Computing on Communication Quality of Users

Junichi KANEDA†, Shin'ichi ARAKAWA†, and Masayuki MURATA†

† Graduate School of Information Science and Technology, Osaka University
Yamadaoka 1–5, Suita, Osaka, 565–0871 Japan
E-mail: †{j-kaneda,arakawa,murata}@ist.osaka-u.ac.jp

Abstract　Modern mobile devices are equipped with many sensors and cameras, and new service applications that use the information from mobile devices are being developed. In such applications, the information is first transferred to the data center and is processed at the data center. Then, the results are send back to the mobile devices if necessary. The processing at data center will lead to a penalty on application delay, i.e., delay experienced by users, due to geographical factors and load concentration. To relax an increase of the delay, a concept of Edge computing is being introduced. Edge computing virtualizes the service functions and places them at the edge of networks. The service application uses the functions at the edge of networks rather than the functions at the data center. It is expected that a response to the service is improved by eliminating geographical delay and relaxing the load concentration. However, there is another concern that the processing delay at the edge may be increased due to the software operation in the virtualized environment. In this paper, we investigate delay experienced by users and communication delay occurred at the networking devices in an Edge computing environment. For this purpose, we construct an Edge computing environment using PCs and OpenStack and clarified the effect on the communication quality of users by changing the location of service functions. The results of our experiments reveal that the application delay is reduced by about 30% at the maximum, and the communication quality of the user is improved by reallocating the service functions from the remote place to the edge close to the user.

Key words　Edge Computing, Network Functions Virtualization (NFV), OpenStack, Communication quality

1. はじめに

端末技術の発展により、通信機能を持ったエンド端末に多くのセンサーが搭載されるようになった。このようなカメラやセンサーなどを搭載したエンド端末において取得された実世界の情報を、データセンタに集め処理する新たなアプリケーションやサービスが考えられている [1]。さらに、データセンタで処理した結果を、再びエンド端末へ転送しユーザに提示するアプリケーションが考えられている [2]。しかし、データセンタ集中型のサービス提供形態では、情報を発信する端末と情報を処理するデータセンタが地理的に離れるため遅延が発生する。さらに、多数のエンド端末から発生する膨大なデータが集中し、データセンタの処理量が増大し、アプリケーションやサービスが享受する遅延が増大する課題が指摘されている [3, 4]。

このような問題に対応すべく、データセンタ集中型のサービス提供形態に加え、エッジコンピューティングの導入が進められている。エッジコンピューティングでは、エンド端末が必要とする一連の情報処理および情報連携の一部を、よりエンド端末に近いネットワーク外縁部に配置したエッジサーバで行う [5]。文献 [5] の想定アプリケーションとして、図 1 に示すような、AR (Augmented Reality) コンテンツの配信サービスが考えられている。AR は、実世界の映像に情報を付加しユーザに表示するため、高いリアルタイム性が求められる。そこで、エッジサーバに利用頻度の高い AR コンテンツをキャッシュし、エンド端末に配信することで、往復遅延時間 (RTT: Round-Trip Time) の低減、高い帯域幅の確保が期待できる。AR は、すでに利用可能なサービスが多数存在し、将来は屋内外問わず様々な場所で活用できる。

また、エッジコンピューティングでは、展開するアプリケーションやサービスの機能拡張や規模の拡大を柔軟に行うため、ネットワーク外縁部に配置するエッジサーバの仮想化にネットワーク機能仮想化 (NFV: Network Functions Virtualization) を応用することが考えられている [5, 6]。NFV は、ルータやファイアウォール、ロードバランサなど、従来、専用機器で運用されてきたネットワーク機能をソフトウェアで実現し、汎用サーバ上の仮想マシンで運用する考え方である [7]。NFV では、仮想化されたネットワーク機能を VNF (Virtual Network Function) と呼び、VNF を実際に動作させるために必要な汎用サーバや仮想化ソフトウェアを NFVI (NFV Infrastructure) と呼ぶ。さらに、VNF および NFVI を含めた NFV 全体の管理・調整を行うソフトウェアを NFV-MANO (NFV Management and Orchestration) と呼ぶ [8]。NFV では、ネットワーク機能をソフトウェア処理し、ハードウェア資源と分離することで、それらを柔軟かつ動的に展開し動作させることが可能となり、ネットワークの設計・運用、サービスそのものに柔軟性が生まれる [9]。

エッジコンピューティングが、地理的な遅延の解消および負荷の分散によるサービスに対する応答性向上を期待する一方で、サービス機能が展開する仮想マシンは、ハイパーバイザやエミュレータなどの仮想化ソフトウェアの上に成り立っており、

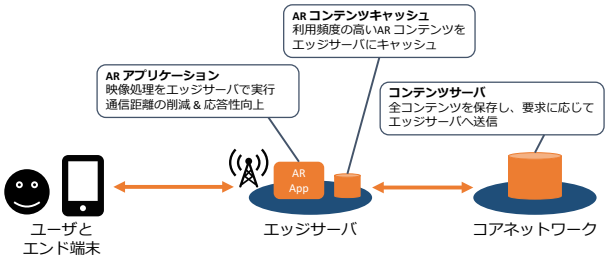

図 1　エッジコンピューティングによる AR コンテンツの配信

ソフトウェア動作による処理速度の低下が懸念される。そのため、利用するアプリケーションやサービスに対する応答性への影響が懸念される。

そこで本稿では、実機を用いて、エッジコンピューティング環境で生じるアプリケーション遅延を測定する。エッジコンピューティング環境を構築するにあたり、仮想化環境構築ソフトウェアである OpenStack [10] を用いる。OpenStack を用いて構築されるネットワーク仮想化環境において、ネットワーク機能ではなく、サービスアプリケーションを仮想化マシン上で動作させ、エッジコンピューティング環境を構築する。また、買い物代行サービスなどのエッジコンピューティングのアプリケーションを想定し、ユーザと遠隔地にあるロボットが映像配信を行い連携するシステムを作成した。本システムにおいて、映像加工を行う拠点を変更しつつエンド間のアプリケーション遅延を測定する。

本稿の構成は以下の通りである。まず、想定するサービス内容と構築したエッジコンピューティング環境を 2 章で説明する。次に、3 章でアプリケーション遅延の測定実験の結果とその評価を行い、サービス機能を遠隔地からユーザに近い拠点に配置することによりユーザの通信品質が改善されることを確認する。最後に、本稿のまとめと今後の課題を 4 章で述べる。

2. エッジコンピューティング環境の構築

本稿では、実機を用いて、エッジコンピューティング環境で生じるアプリケーション遅延を測定する。エッジコンピューティング環境を構築するにあたり、仮想化環境構築ソフトウェアである OpenStack を用いる。NFV の実装プロジェクトにおける採用例が多いことから OpenStack を採用した。OpenStack を用いて構築されるネットワーク仮想化環境において、ネットワーク機能ではなく、サービスアプリケーションを仮想化マシン上で動作させ、エッジコンピューティング環境を構築する。本章では、まず、エッジコンピューティング環境で提供するサービスと、それを実現するために使用したアプリケーションについて説明する。次に、OpenStack を用いて構築したネットワーク仮想化環境と、エッジコンピューティング環境への応用について説明する。

2.1 サービスとアプリケーション

近年登場した新しいサービスとして、ロボット Pepper を用いた買い物代行サービスを実現することを考えた。買い物代行サービスでは、Pepper が実店舗に赴き、ユーザは自宅にいな

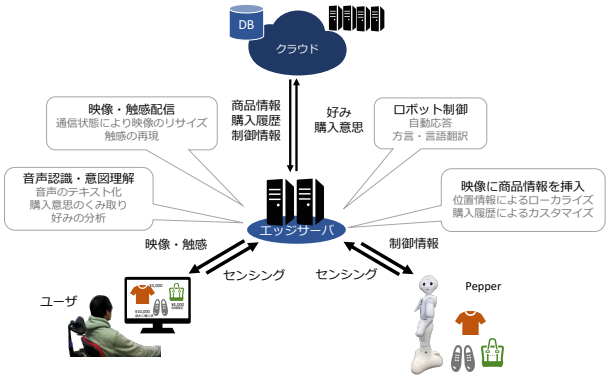

図 2　買い物代行サービスの処理内容

がら買い物を楽しめる。さらに Pepper が取得した実世界映像に、1 章で紹介した AR 技術を用いて、商品情報を重ね合わせ、ユーザに提示する。この処理を、外部のサーバで行い、商品情報はクラウドから取得する。また、センシング技術を活用して、商品の触感やにおいなどをユーザへ提示し、ユーザの意図理解によるロボット制御も行う。買い物代行サービスを実現するアプリケーションの処理内容を図 2 に示す。ただし本稿の想定するサービスでは、Pepper からの映像のライブストリーミングのみを想定する。

なお、Pepper は、ユーザの感情認識が可能な人型ロボットである。SDK (Software Development Kit) が配布されており、搭載するカメラやセンサー、動作モジュールを活用した Pepper 内外で動作するアプリケーションの開発を、API (Application Programming Interface) を利用し高い自由度で行うことができる。本稿では、ネットワーク仮想化環境でアプリケーションやサービスを柔軟に提供した際に生じる遅延を計測するとともに、将来的にはアプリケーションレベルの品質への影響を調査することを目的としている。Pepper を用いることで、搭載するカメラやセンサー、動作モジュールを活用したアプリケーションの構築が可能となり、実機を用いたサービスが実現される。これにより、ユーザ体験にもとづくアプリケーションレベルの品質の測定が可能となる。

そこで本稿では、外部アプリケーションとして主に ffmpeg [11] と Pepper を連携し、想定するサービスを実現する。Pepper の取得した映像のユーザ PC へのライブストリーミング配信には、ffmpeg と fserver、ffplay を使用する。ffmpeg は、Pepper と外部サーバにおいて動作する。Pepper の OS である NAOqi (Gentoo Linux ベース) 上で動作する Pepper 内部の ffmpeg は、カメラが取得した映像を mpeg 形式に圧縮し、外部サーバの ffmpeg へ送信する。外部サーバの ffmpeg は、映像の加工 (文字列挿入) と ffserver への中継役として利用する。ストリーミングサーバアプリケーションである ffserver はユーザ PC へ映像を配信し、ユーザ PC 上のメディアプレーヤ ffplay によって映像がディスプレイへ表示される。なお、ffserver はトランスポートプロトコルとして、映像の受信に UDP を、配信に TCP を利用する。

2.2　OpenStack によるネットワーク仮想化環境の構築

OpenStack は、IaaS 型のクラウド環境を構築するオープンソースソフトウェア群であるが、近年 NFV においてインフラストラクチャ部分である NFVI やそれを管理する VIM (Virtualized Infrastructure Manager) としての活用されることが考えられている [12]。これは、NFV ではクラウドコンピューティングとともに発展したサーバ仮想化技術を応用してネットワーク機能の仮想化を実現しようとしているためである。Linux Foundation の下で NFV のフレームワーク全体を既存のオープンソースコンポーネントの統合によって実装することを目指している OPNFV (Open Platform for NFV) [13] は、NFVI と VIM の主要構成要素として OpenStack を採用している。また、様々な NFV-MANO のオープンソース実装においても、NFVI および VIM として OpenStack の利用を前提としているものが多い [14]。本稿の対象となる研究における環境構築の最終的な目標は、完全な NFV 環境を構築しエッジコンピューティング環境に応用することであるが、現時点では OpenStack による NFVI とそれを管理する VIM の構築にとどめる。なぜならば、NFV においてシステム全体の管理および調整を行う最も重要な構成要素 NFV-MANO のオープンソース実装が、サービス機能の配置がユーザの通信品質に与える効果を調べる上で重要となる動的な資源管理の実現に至っていないためである [14]。OpenStack は、半年ごとにリリースが行われることとなっており、各リリースには、アルファベット順に地名に由来するコードネームがつけられている。本稿においては、2017 年 2 月にリリースされ執筆時点の最新版である Ocata を用いる。なお、OpenStack に関する説明では、OpenStack のドキュメントに準じ、仮想マシンをインスタンスと呼ぶ。

OpenStack は、モジュラーアーキテクチャを採用しており、提供するサービスごとにモジュール化されたソフトウェアが API を通じて連携し動作する。これらのソフトウェアモジュールを OpenStack ではプロジェクトと呼ぶ。全てのソフトウェアモジュールは、OpenStack 環境を構成する 3 種類の物理ノード上で、さらに細分化されたエージェントとして展開する。3 種類の物理ノードとは、コントローラノード、ネットワークノード、コンピュートノードである。コントローラノードは、OpenStack 環境全体を管理するエージェントが展開するノードであり、コントローラノードが果たす役割は NFV における VIM に相当する。ネットワークノードは、特にネットワークに関するエージェントが動作するノードである。ネットワークノード上に、OpenStack 管理下の仮想ネットワークと外部ネットワークを接続する仮想ルータやファイアウォールとして動作するエージェントが展開する。コントローラノードとネットワークノードは同一の計算機上で動作可能であるため、本稿における構成では 1 台のサーバで実現する。また、コンピュートノードは、NFV における NFVI の物理資源に相当し、実際にインスタンスが実行されるノードである。インスタンスの管理やネットワーキング機能の提供に関わるエージェントが展開する。本稿における構成では、コンピュートノードは 2 台用意し、ハイパーバイザとして KVM と QEMU を用いる。以上の 3 種

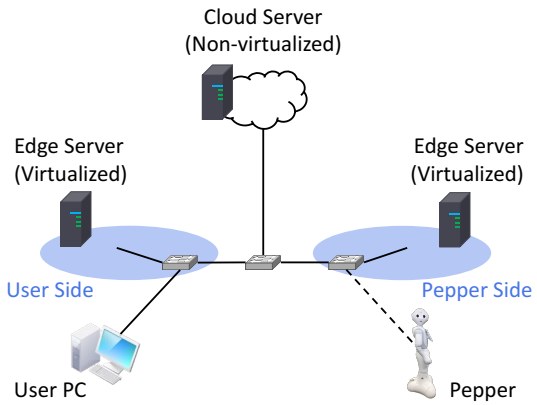

図 3　構築したエッジコンピューティング環境の全体構成

類のノードがネットワークで相互に接続され、それぞれのノードで展開するエージェントがやり取りすることで OpenStack による仮想化環境が動作する。

本稿においては、Nova、Neutron、Keystone、Glance、Horizon、Ceilometer、Heat の 7 つのプロジェクトを導入し仮想化環境の構築を行った。このうち主要な機能を提供するのは、Nova と Neutron である。Nova は、コンピュートサービスを提供する OpenStack の最も重要なソフトウェアモジュールである。各コンピュートノードのハイパーバイザへリクエストを発行し、インスタンスの起動や停止、資源の割り当てを行う。Neutron は、OpenStack 環境でインスタンスにネットワーク機能を提供する。Neutron では、コンピュートノード内のスイッチング機能の提供は、Linux Bridge もしくは Open vSwitch を利用できる。また、SDN コントローラとの連携が可能であり、OpenFlow によってネットワークを制御できる。本稿における環境では、L2/L3 の基本機能に加え分散仮想ルータ (DVR: Distributed Virtual Router) を利用し、スイッチング機能は Open vSwitch を採用する。SDN コントローラとの連携は行なっていない。分散仮想ルータを利用することで、初期設定ではネットワークノードのみで展開する仮想ルータを全てのコンピュートノード上に分散的に配置することが可能となる。また、Keystone は IaaS 型クラウドサービスの管理者が利用者に対しユーザ認証を行うためのソフトウェアモジュールである。本稿における環境では、管理者ユーザのみが存在する。Glance は、インスタンスのイメージ管理を行い、Horizon はダッシュボードと呼ばれるウェブ GUI を提供する。Ceilometer は、各種資源の利用状況の計測を行い、Heat は、ユーザが構築したいクラウドシステムをテンプレートに記述することで、様々なリソースの展開、運用を自動化できる。なお、OpenStack には、Tacker と呼ばれる OpenStack のみで NFV-MANO の全機能実現を目指すプロジェクトがある。現状の OpenStack が果たしている VIM 以外の機能である Orchestrator および VNF Manager を実装中であるが、動作実績が十分でないため導入していない。

2.3　エッジコンピューティング環境への応用

4 台の同性能の計算機、ロボット Pepper、ユーザ PC をスイッチで接続し、実験室内にエッジコンピューティング環境を構築した。全体構成を図 3 に示す。4 台の計算機のうち 1 台は、OpenStack コントローラノードおよびネットワークノードとして動作する。この計算機は図中には示していない。残りの 3 台は、エッジサーバを想定した仮想化された 2 台の OpenStack コンピュートノードと、クラウド型のデータセンタを想定した仮想化されていないサーバとして利用する。図中の Edge Server は、OpenStack を用いて仮想化環境が構築されたコンピュートノードであり、アプリケーションはコンピュートノード上の個別の仮想マシンで実行される。買い物代行サービスでは、Pepper とユーザが地理的に離れるため、それぞれに近い処理拠点として、2 つのエッジサーバを用意する。また、Cloud Server はエッジコンピューティングにおいてネットワーク中心部にあるデータセンタを想定したサーバである。このサーバは仮想化環境が構築されておらず、アプリケーションはサーバ上で直接実行される。ただし、このネットワーク構成では、エンド端末であるユーザ PC および Pepper と、データセンタが地理的に離れていることが再現できていない。そのため、実験結果の評価を行う際は、地理的な要因によって発生する遅延を考慮する必要がある。コンピュートノード上の仮想マシンで動作するアプリケーション、クラウドを想定したサーバ上で実行されるアプリケーション、Pepper、ユーザ PC は、データプレーンネットワークを利用して通信を行う。Pepper は、Wi-Fi アクセスポイントを介して、無線でデータプレーンに接続している。2 台の OpenStack コンピュートノードは内部で仮想ルータおよび仮想スイッチが動作し、仮想マシンのデータプレーンへの接続が可能となっている。OpenStack コントローラノードおよびネットワークノードは、マネジメントネットワークを利用してコンピュートノードとやり取りする。これらは、データプレーンの通信には一切関与しない。

3.　計測実験と評価

エッジコンピューティング環境におけるサービス機能の配置がユーザの通信品質に与える効果を評価するにあたり、ライブストリーミング映像のエンド端末間における遅延に着目し、構築したエッジコンピューティング環境において、ライブストリーミング映像の遅延時間を計測する実験を行なった。加えて、各サーバー (Edge Server と Cloud Server) の処理時間を計測する実験を行なった。本章では、シナリオの設定、計測の方法とその結果、評価を述べる。

3.1　シナリオ

エッジコンピューティング環境において、サービス機能を配置する拠点の違いによるアプリケーション遅延の発生要因や発生量を調べるため、仮想化環境の有無や TCP 通信の距離など、サービスの提供携帯とそれに伴うサービス機能の配置を変えた 4 つのシナリオを用意した。シナリオ名に Edge のつくシナリオは、エッジコンピューティング環境を想定しており、シナリオ Cloud と Direct は比較シナリオとして設定している。各シナリオにおけるアプリケーション配置と通信経路を図 4 に示し、以下で詳細を説明する。

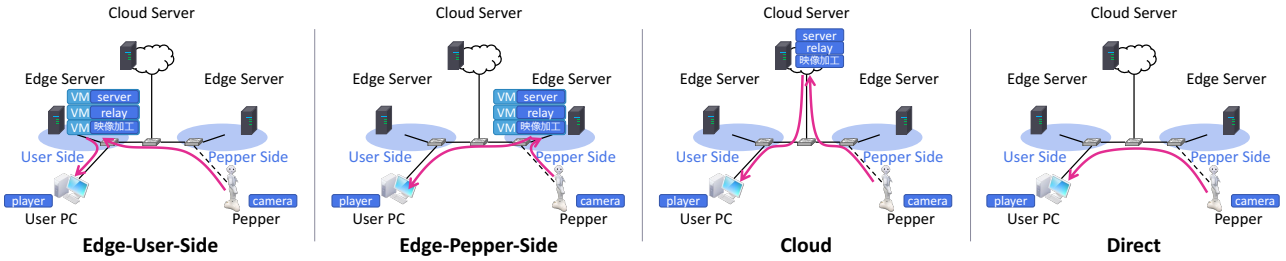

図 4　各シナリオにおけるサービス機能の配置と通信経路

- **Edge-User-Side:** このシナリオでは、2 つのエッジサーバのうち、ユーザ側で展開する拠点に映像加工と配信処理を行うアプリケーションを配置しサービスを提供する。アプリケーションは仮想マシン上で実行される。このサーバにサービス機能を配置することで、TCP 通信の通信距離を小さくしている。

- **Edge-Pepper-Side:** このシナリオでは、2 つのエッジサーバのうち、Pepper 側で展開する拠点に映像加工と配信処理を行うアプリケーションを配置しサービスを提供する。アプリケーションは仮想マシン上で実行される。このサーバにサービス機を配置することで、TCP 通信の通信距離を大きくしている。

- **Cloud:** このシナリオでは、エッジコンピューティングにおいてネットワーク中心部にあるデータセンタを想定したサーバに、映像加工と配信処理を行うアプリケーションを配置しサービスを提供する。アプリケーションはこのサーバ上で仮想化を行うことなく実行される。TCP 通信の通信距離は Edge-User-Side より大きく、Edge-Pepper-Side より小さい。

- **Direct:** 映像加工と配信処理を行うアプリケーションを配置せず、Pepper の ffmpeg からユーザ PC のメディアプレーヤ ffplay に直接映像を送信する。エンド端末の処理時間を計測することを目的とする。通信はすべて UDP 通信で行われる。

3.2　測 定 結 果

構築したエッジコンピューティング環境において、ライブストリーミング映像の遅延時間を求める方法として、まず、Pepper の前でミリ秒単位で表示されるデジタル時計を表示し、その映像をライブストリーミングし、ユーザ PC に表示する。ここで Pepper に見せるデジタル時計は時刻同期の観点からユーザ PC の画面に表示する。次に、このデジタル時計と Pepper からのライブストリーミング映像をデスクトップ上に並べ、スクリーンショットを撮影する。これを、1 秒おきに 100 回繰り返す。最後に、100 枚のスクリーンショットすべてについて、撮影された 2 つの時刻の差を求め、平均遅延時間を算出する。この方法で、ライブストリーミング映像の遅延時間を、4 つのシナリオでそれぞれ 2 回測定した結果とその平均および、遅延の発生要因を表 1 にまとめる。なお、TCP 経路長は各シナリオの TCP 通信経路を構成する端末とスイッチ間、スイッチ同士を結ぶリンクの数を表す。

さらに、シナリオ Edge-User-Side、Edge-Pepper-Side、Cloud において、各アプリケーションのサーバ処理時間を計測する実験を行った。サーバ処理時間を直接計測することは困難であるため、各シナリオにおいて、サーバのネットワークインターフェースでパケットキャプチャを行い、出入りするパケットの時差を求めた。ただし、サーバへ到達したパケットは、処理を経て別のパケットとしてサーバから送信されるため、同じデータを運ぶパケットを双方向で特定することは難しい。そのため、ライブストリーミングの開始・停止を繰り返し、開始時に流れる最初のパケットと停止時に流れる最後のパケットについて、パケットの通過時刻の差を調べた。サーバ処理時間の計測結果を表 2 に示す。

3.3　評価と考察

各シナリオのライブストリーミング映像のエンド間の遅延とサーバ処理時間の結果から次のことが言える。（1）エンド処理にかかる時間は 425.19 [ms] である。（2）TCP 経路長 1 あたりの遅延時間は 6.2 [ms] と算出される。（3）映像加工と配信処理にかかる時間は 28.85 [ms] と算出される。そのうち、サーバ処理時間は 7.60 [ms] であり、残りの 21.25 [ms] はプロトコルオーバーヘッドである。（4）仮想化による遅延増大は 13.04 [ms] と算出される。そのうち、サーバ処理時間の増大は 4.0 [ms] であり、残りの 9.04 [ms] はプロトコルオーバーヘッドの増大である。

また、文献 [15] によれば、一般的にエンド端末からクラウドを提供するデータセンタまでの距離によって生じる平均往復遅延は、国内のデータセンタで 100 [ms] 以下、米国のデータセンターで約 100 [ms]、欧州のデータセンターで約 200 [ms] であるとされ、いずれの場合でも百ミリ秒単位で遅延が発生する。そのため、シナリオ Cloud のライブストリーミング映像のエンド間の遅延 472.64 [ms] は、最大でおよそ 670 [ms] になると考えられる。距離による平均往復遅延と比較し、仮想化による遅延の増大 13 [ms] は十分に小さく、クラウドを展開する遠方のデータセンタからネットワーク外縁部のエッジサーバにサービス機能を配置することでユーザの通信品質を向上させることができると言える。

以上により、ネットワーク仮想化によるソフトウェア動作から生じる処理速度低下は、地理的な要因によって生じる遅延に比べ十分小さく、遠方のデータセンタからエッジサーバにサー

表 1　ライブストリーミング映像の遅延時間と遅延の発生要因

シナリオ	計測 1 [ms]	計測 2 [ms]	平均 [ms]	遅延の発生要因
Edge-User-Side	476.19	482.76	479.48	エンド処理、映像加工と配信処理、仮想化、TCP 経路長 2
Edge-Pepper-Side	482.76	500.99	491.88	エンド処理、映像加工と配信処理、仮想化、TCP 経路長 4
Cloud	467.35	477.93	472.64	エンド処理、映像加工と配信処理、TCP 経路長 3
Direct	419.54	430.84	425.19	エンド処理

表 2　サーバ処理時間

シナリオ	サーバ	処理時間 [ms]
Edge-User-Side	Edge Server (User Side)	11.348
Edge-Pepper-Side	Edge Server (Pepper Side)	11.899
Cloud	Cloud Server	7.595

ビス機能を配置することで、エンド間のアプリケーション遅延を 30%程度低減できることを示した。したがって、遠隔地からユーザに近い拠点にサービス機能を配置することにより、ユーザの通信品質を改善できることを実機を用いて示した。

なお、エンド処理にかかる時間については、映像の圧縮処理をエッジサーバで行えばある程度の削減が可能であると考えている。また、Pepper に搭載されている Intel Atom Processor E3845 は現在汎用的に使用されている Intel Core i7 と比較し、浮動小数点演算処理能力 (FLOPS) が 10 分の 1 程度である。そのため、将来的にこのような CPU が Pepper に搭載された場合の処理時間は約 40 [ms] にまで削減されるとみられ、ネットワークで発生している遅延の割合が増大すると考えられる。

4. おわりに

エッジコンピューティングの標準化が進み、導入や展開が現実味を帯びてきている状況において、エッジコンピューティング環境における遅延の発生要因や発生量を理解し、サービス機能の配置がもたらすユーザの通信品質に与える効果を調査することは、今後のネットワーク設計、サービス設計の観点から重要である。そこで、本稿では、その第一段階としてユーザの通信品質に与える効果を調査すべく、サービスのエンド端末間におけるアプリケーション遅延に着目し計測を行った。その結果、仮想化が行われていないサーバでサービスアプリケーションを実行する場合に比べ、仮想マシン上で実行する場合の方が、ライブストリーミング映像の遅延が大きくなることが確認された。さらに TCP 通信の距離を小さくするようにサービス機能を配置すると、遅延が抑えられることが確認された。加えて、各サーバの処理時間を計測する実験を行なった結果、仮想化されたサーバの処理時間の方が仮想化が行われていないサーバの処理時間に比べ大きいことが確認された。

今後は、Pepper とユーザの双方向の通信における遅延の計測や、仮想化環境上でハードウェア資源の負荷やネットワーク負荷が高まった際に、仮想マシンのライブマイグレーションによるサービス機能の動的な再配置がもたらす、通信品質への影響を調査する予定である。また、ユーザの通信品質のみならずユーザの体験品質に対する効果を明らかにすることを目指す。

謝辞

本研究の一部は NICT 委託研究「未来を創る新たなネットワーク基盤技術に関する研究開発」によるものである。ここに記して謝意を表す。

文　献

[1] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, “The rise of “big data” on cloud computing: Review and open research issues,” *Information Systems*, vol. 47, pp. 98–115, Jan. 2015.

[2] Z. Huang, W. Li, P. Hui, and C. Peylo, “CloudRidAR: A Cloud-based Architecture for Mobile Augmented Reality,” in *Proceedings of ACM workshop on Mobile augmented reality and robotic technology-based systems*, pp. 29–34, June 2014.

[3] N. Fernando, S. W. Loke, and W. Rahayu, “Mobile cloud computing: A survey,” *Future generation computer systems*, vol. 29, pp. 84–106, Jan. 2013.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog Computing and Its Role in the Internet of Things,” in *Proceedings of ACM SIGCOMM 2012*, pp. 13–16, Aug. 2012.

[5] “Mobile-Edge Computing - Introductory Technical White Paper.” ETSI, Sept. 2014.

[6] “Mobile-Edge Computing (MEC); Technical Requirements.” ETSI GS MEC 002 V1.1.1, Mar. 2016.

[7] R. Mijumbi, J. Serrat, J.-l. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, “Network Function Virtualization: State-of-the-Art and Research Challenges,” *IEEE Communications Surveys & Tutorials*, vol. 18, pp. 236–262, Sept. 2015.

[8] “Network Functions Virtualisation (NFV); Architectural Framework.” ETSI GS NFV 002 V1.1.1, Oct. 2013.

[9] “Network Functions Virtualisation - Update White Paper.” ETSI, Oct. 2013.

[10] OpenStack.org, “Home » OpenStack Open Source Cloud Computing Software.” [Online]. Available: `https://www.openstack.org/`.

[11] FFmpeg, “FFmpeg.” [Online]. Available: `https://ffmpeg.org/`.

[12] “Network Functions Virtualisation - White Paper #3.” ETSI, Oct. 2014.

[13] OPNFV, “Home | Open Platform for NFV (OPNFV).” [Online]. Available: `https://www.opnfv.org/`.

[14] R. Mijumbi, J. Serrat, J.-l. Gorricho, S. Latré, M. Charalambides, and D. Lopez, “Management and Orchestration Challenges in Network Functions Virtualization,” *IEEE Communications Magazine*, vol. 54, pp. 98–105, Jan. 2016.

[15] 田中 裕之, 高橋 紀之, 川村 龍太郎, “IoT 時代を拓くエッジコンピューティングの研究開発,” *NTT* 技術ジャーナル, pp. 59–63, Aug. 2015.