「マルチメディア，分散，協調とモバイル
(DICOMO2014)シンポジウム」 平成26年7月

エッジエリアにおける携帯端末のネットワークリソース制御

西原康介† 船橋一訓† 竹中崇†

高度化するＩＣＴ社会の増え続ける情報量に対し、様々な取り組みがなされている。膨大な情報やサービスにアクセスする多彩なネットワーク、及び、多種多様なアプリケーションを、既存アーキテクチャで扱うには難しくなってきている。そこで、フレキシブルなプログラム性をネットワーク内部に導入し、ネットワーク機能の創造や設計、活用に大きな自由度を与え、これから出現する高度なサービスに備えネットワーク基盤を整えることが本研究の大きな目的である。本稿は、その中でもエッジエリアにおけるネットワークリソース制御のプログラマビリティについて述べる。本稿で述べるネットワーク制御機構は、仮想化基盤上に構築され、エッジエリアの情報を選択的に収集する情報収集モジュール、収集した情報を基にルールベースで判断し制御指示を発行する判断モジュール、制御指示に従って制御を行うセキュアに行うネットワーク制御モジュールから構成される。このプログラマビリティにより、様々なアプリケーションに共通したネットワークリソース制御機能を与えることができ、エッジエリアでしか取得できない情報を基にしたサービスを容易に構築可能にする。コンテンツ配信を行うサービスを例に動作検証を行い、エッジエリア情報を用いた制御が容易に記述できることを確かめ、アクセスネットワーク制御によりサービス性能が向上することを確認した。

Network Resource Control for Mobile Device in Edge Area

KOSUKE NISHIHARA† KAZUNORI FUNAHASHI†
TAKASHI TAKENAKA†

1. はじめに

ＩＣＴ社会の高度な発展により流通する情報量は爆発し、その情報を活用したサービスやそのサービスを利用するスマートフォンなどの携帯端末が高機能化､多様化している。インターネットの普及に伴い、やり取りされる情報量は毎年増え続けている。SNS や動画配信などの情報の配信や共有を行うクラウド型のサービスが多数展開され、これらのサービスを携帯端末から利用することが主流になっている。それらのサービスへのアクセスも多様化し､3G 回線やWiFi など様々な NW を経由してアクセスしている。携帯端末の性能向上も著しく、一昔前の PC と同等の処理が実現できるようになっている。

しかし、膨大な情報やサービスを構成する多彩な NW アクセス手段、多種多様なアプリケーションを、現在のインターネットのようなアーキテクチャで扱うには難しくなってきている。複雑化に合わせて機能を高度化、細分化を行ってきたが､その結果容易に扱うことが難しくなっている。また、情報を集中的に処理する従来のクラウドでは処理能力や NW 帯域に限界があり、大量の情報を取り扱うことが難しく､エッジエリアにも知的な処理が必要になっている。日々高度化する攻撃に対して、セキュリティや堅牢性にも不安要素がある。

そこで、フレキシブルなプログラム性をネットワーク内部に導入し、ネットワーク機能の創造や設計、活用に大きな自由度を与え、これから出現するサービスに備えネットワーク基盤を整えることが、本研究の大きな目的である。このため、NW 機能の仮想化や、異なるサービスで使用する資源の独立分離、動的な資源の確保や解放、プログラマブルな資源の活用、セキュアな管理の実現を課題としている。本研究は、独立行政法人情報通信研究機構(NICT)の委託研究「新世代ネットワークを支えるネットワーク仮想化基盤技術の研究開発」(課題イ：サービス合成可能なネットワーク･プラットフォームの研究開発)の一部として行ってきた。

本稿は、その中でもエッジエリアにおけるネットワークリソース制御のプログラマビリティについて述べる。サービス開発者がネットワーク基盤の提供する制御機能をサービスに組み込もうとする場合、NW 制御の専門的な知識が必要、基盤の仕様変更や別基盤への移植に対応しにくい、ユーザ情報やネットワーク基盤情報を取得しにくい、などの課題があり、なかなか思い通りに性能向上が図れないことがあった。このため、サービスとサービスが利用するネットワーク基盤との間に、エッジエリアの情報を有効活用し NW 制御に活かす、アプリ層及び基盤層と独立した抽象化されたプログラマブルな仕組みを導入する。このプログラマビリティは、複数端末からの情報を統合して判断することによる全体制御と各々の情報を基にした個の制御の連携を、基盤に対し要求することで容易に実現する。複数端末から集められたエッジエリアの情報を基に、コア NW のトポロジーの変更要求やサービス構成の変更などの全体の性能に関わる NW リソースの制御を行うと同時に、携帯端

† NEC グリーンプラットフォーム研究所
Green Platform Research Laboratories, NEC Corporation

末がサービスにアクセスするアクセス NW の切り換えや端末保有のセンサデバイスの共有などのエッジエリアの個の性能向上にかかわる NW リソースを制御も行うことができる。

本稿で述べる NW 制御機構は、仮想化基盤上に構築され、エッジエリアの情報を基にエッジ/コア NW のリソース制御をセキュアに行えることに特徴がある。本基盤では、アクセス NW を含め NW 全体が仮想化されている[1]。サービス毎に NW が隔離され、また、端末プロセスごとにその NW への通信制御が可能である。この仮想化基盤に参加した端末には、NW サービスへの安全なアクセスが提供される。

本機構は、エッジエリアの情報を選択的に収集するモジュール、収集した情報を基に判断し制御指示を発行するモジュール、制御指示に従って制御を行うモジュールから構成される。収集モジュールは端末上で動作するもので、NW 情報やユーザ情報など、どのアプリケーションからも利用される共通するユーザ情報を収集する機能を持つ。判断モジュールがプログラマビリティを与える。サービス開発者がルールを書き、収集した情報を基にルールベースで制御を行う。制御モジュールも端末上で動作し、制御指示に基づきエッジエリアの NW リソースを制御する機能を持つ。

このプログラマビリティにより、様々なアプリケーションに共通したネットワークリソース制御機能を与えることができ、エッジエリアでしか取得できない情報を基にしたサービスを容易に構築可能にする。端末がリッチになりユーザ情報を集めるのが容易になってきたが、これらの情報収集の機能はアプリケーション自体のアルゴリズムとは分離しており、共通化し再利用することでサービス自体の開発に注力することができる。また、仮想化基盤上に展開されたサービスからは得にくい情報もあり、本機能により収集が可能である。これらの情報を基に NW リソースを制御することで、サービス最適化やカスタマイズなどでアプリを高度化できる。

膨大になる情報を扱う取り組みとして、NTT のエッジコンピューティング[2]や、シスコのフォグコンピューティング[3]などがある。また、ユーザそれぞれに応じてカスタマイズ可能なサービスが近年注目されている[4,5]。複数端末間でリソース共有制御することで高負荷処理が必要なサービスを低機能端末で実行する技術や、高負荷処理の一部をサーバや他端末で実行する技術も検討されてきた[6,7]。しかし、これらは、エッジとしてルータや基地局を想定しているものやサーバを対象とするものであり、本稿で対象としている携帯端末を含めたエッジエリアのプログラマブルな NW リソース制御ではない。

本稿は以下のように構成される。第 2 章ではエッジ情報を用いたリソース制御について説明する。第 3 章ではリソース制御の効果を、例を用いて評価する。

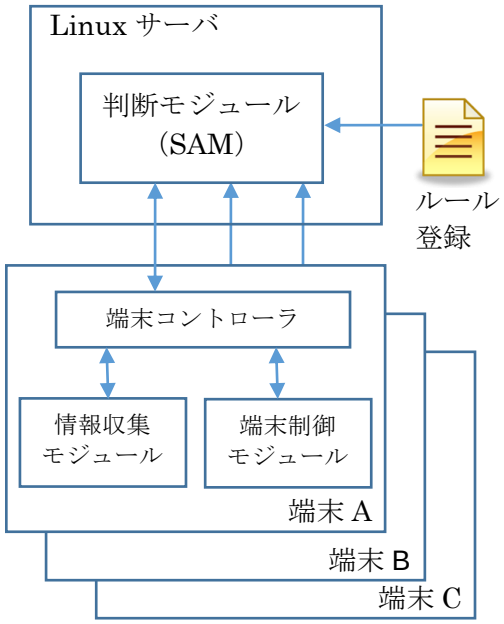

図 1　構成図

2. エッジ情報を用いたリソース制御

本機構は、エッジエリアの情報を収集するモジュール、収集した情報を基に判断し制御指示を発行するモジュール、制御指示に従って制御を行うモジュールから構成される(図 1)。我々はこれまでもリソース制御のプログラマビリティについて検討を行ってきた[8]。本稿は、エッジエリア情報を基にしたエッジ NW のプログラマブルな制御にフォーカスする。

収集モジュールは端末上で動作するもので、NW 情報やユーザ情報など、どのアプリケーションからも利用される共通するユーザ情報を選択的に収集する機能を持つ。情報収集モジュールは、アクセス NW の状況の監視やユーザの振る舞いの監視、端末プロセスの監視などのモジュールからなる。本モジュールの要件は、必要な情報のみを収集できる選択性があることや、監視対象のプロセスをできるだけ改造しないことである。指定する NW だけを監視し余計な情報を収集しないことや、端末プロセスの出力する特定のタグをもつログ情報だけを集めることができる。また、サービスに必要なモジュールを、独自に開発し組み込むことも可能である。収集する端末情報の一例を表 1 に示す。収集する情報は端末 ID、IP アドレス、サービス ID、タイムスタンプのフィールドを持ち、端末情報を（名前、値）の組で指定する。

表 1　サービス・アクセス制御装置が収集する情報例

```
<quality>
  <terminal_id>TM1234</terminal_id>
  <ip_address>1.2.3.4</ip_address>
  <service_id>Service001</service_id>
  <sampling_at>
    2013-01-01T01:02:03
  </sampling_at>
  <param_list>
    <param>
      <key>video_quality</key>
      <value>60</value>
    </param>
    <param>
      <key>fps</key>
      <value>30</value>
    </param>
  </param_list>
</quality>
```

制御モジュールも端末上で動作し、制御指示に基づきエッジエリアの NW リソースを制御する機能を持つ。 制御モジュールは、NW を制御するモジュールや端末に備わるセンサを外部から共有可能にするモジュールなどを想定するが、情報収集モジュールと同様に独自に開発し組み込むことも可能である。本モジュールは、アプリ層との独立性を保つため、サービス側の改造をできるだけ行わないように、独立して動作することを要件とする。例えば、アクセス NW を 3G か WiFi かに切り替える NW 制御モジュールは、アプリケーションに対し共通の NW インタフェースを与え、アプリケーションとは独立に切り換えが可能である。この NW の隠蔽や切り替えは仮想化基盤の機能により実現されており、サービス開発者がその詳細を知る必要はない。

情報を収集するモジュール及び、制御を行うモジュールは、端末制御コントローラを通して外部から配置や起動、停止、動作変更などの指示が可能になっている。つまり、モジュール自体の動作も動的に変更することができる。後述の判断モジュールと連携して、必要に応じてもモジュールの配置や取得する情報の変更が可能である。

サービス・アクセス制御装置（Service Access Manager: SAM）と呼ばれる収集した情報を基に判断し制御指示を発行する判断モジュールが、プログラマビリティを提供する。どのような情報を基にどう判断するかは、XML 形式で定義され、収集した情報を基にルールベースで実行される。このルールは、サービス開発者が記述する。

XML で記述されるルールは、Appendix A.1 で示すような構造を持ち、ターゲットとする情報、マッチする条件、アクションとしての制御で定義される。ターゲットでサービスや対象端末を絞り込むことができる。マッチ条件では、単純な比較のための演算ではなく、外部スクリプトを呼ぶことができ高度な演算も可能である。アクションは、端末やサーバにリクエストを出すことや、別のルールを呼び出すことができる。

サービス開発者は、このルールを記述し、NW 制御を独自のアプリケーションのアルゴリズムにとりこむことができる。変化するアクセス NW やユーザの振る舞いに起因する情報を収集し、サービス変更のトリガとして利用でき動的にサービス変更が可能である。サービス開発時には、条件を変えながら容易にトライアンドエラーを行いながら開発もできる。サービス開始後には、状況に応じたサービス変更に加え、想定していた環境が大きく変わった場合でもルールの修正で素早く対応が可能である。

3. リソース制御のユースケースを用いた評価

コンテンツ配信を行うサービスを例に、エッジエリア情報を用いたリソース制御の評価を行う。まず、制御を行う２つのシナリオについて説明し、エッジエリア情報を用いた制御が容易に実現できることを確認する。次に、その１つのシナリオにおいて基礎評価を行い、NW 制御によりサービス性能が向上することを確認する。

ここで用いるコンテンツ配信アプリケーションは Breadcrumb(BC)と呼ばれるもので、仮想化基盤を活用する応用として本 NICT 委託研究で検討されているアプリケーションの一つである[9]。BC は、ルータがキャッシュを持つキャッシュネットワークを前提としており、コンテンツのダウンロード軌跡情報を記録し、同一コンテンツへのクエリを BC ルータに沿って転送することで、キャッシュへと誘導する。この BC ルータは、端末を含む仮想化基盤の仮想化ノード上で機能している。つまり、コンテンツサーバからクライアントまでの間の BC ルータにはコンテンツがキャッシュされ、同じコンテンツへの取得要求があった場合、近くの最近使われたキャッシュされている BC ルータから配信される。

3.1 シナリオ１

１つ目のシナリオとして、アクセス NW 状況やユーザのコンテンツ要求情報を基にして、アクセス NW を切り替えるものを考える。エッジエリアの NW は仮想化基盤により仮想化されており、WiFi 及び 3G の違いが隠蔽されている。つまりアプリケーションからはどの NW につながっているかは意識しなくてもよい。これは仮想化の利点であるが、一方、アクセス NW の接続情報をアプリケーションから取得しにくいという一面も持ち、エッジエリアでしか取得できない情報である。コンテンツ要求情報は、ユーザが要求するコンテンツは何か、どの端末にあるかなどの情報である。これは、端末で動作するアプリケーションプロセスのログを監視し取得する。

NW の切替えの判断はサービス開発者が決定し、ルールとして記述する。NW にはそれぞれの特性があり、どの NW に繋がっているかで性能が異なることがありうる。ただし、それぞれ優劣を持ち合わせ、どれにつなげば良いか一概には言えない。どの NW に繋ぐかをサービス開発者がサービスと独立して決めることで、サービス機能を補完できる。

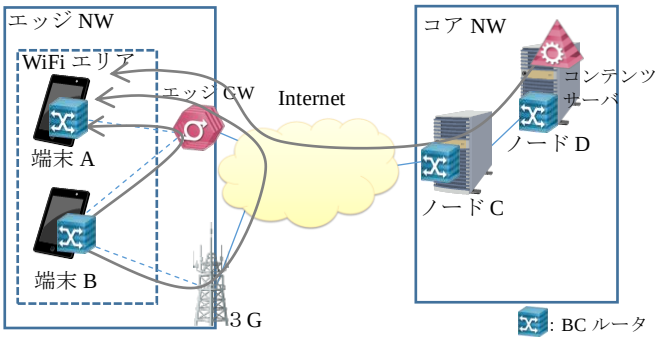

図 2　NW 構成図

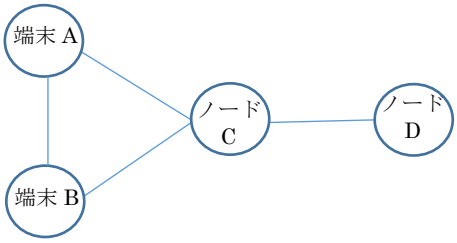

図 3　Breadcrumb NW トポロジー

NW の実行スループットなどを基に決定できるが、ここでは簡単のため WiFi が 3G に比べ高速であるという点にのみ注目する。遅延が 3G に比べ少なく、想定する NW では物理的な距離も近いためである。ここで図 2 に示される NW を想定し、二つの端末が連続して同じコンテンツを取得することを想定する。図に示すように、コア NW 及び端末が属するエッジ NW から成り、エッジ NW へは Internet を介しエッジ GW で接続している。

オリジナルのコンテンツはノード D の BC ルータが保有しているとする。初めはノード D から取得するが、コンテンツを取得した端末（端末 B）には当該コンテンツがキャッシュされる。後でコンテンツ要求した端末（端末 A）は、端末 B からコンテンツを取得でき、コア NW 上のノード D から取得するより高速にダウンロードできる。

仮想化された NW として、BC からは図 3 のような構成として認識される。各端末（A、B）はノード C につながっているように構成される。アクセス NW は仮想化されているが物理的には、端末 A、B 間には 3G 及び WiFi など複数経路が存在する。端末間の通信は、WiFi 経由ではエッジエリア内部で閉じているが、3G はカバー範囲が広いもののいったん外部に出て Internet 経由で通信することになる。

ここで複数あるアクセス NW を最適に選択できれば、端末間でのコンテンツ共有の性能向上を図ることができる。端末 A が要求するコンテンツが端末 B に存在するとき、NW 状況に応じアクセス NW を選択することで高速に取得ができる。今の場合、両端末が同じドメインの WiFi エリアに所属し互いに当該 NW に接続可能なら、その NW に接続し WiFi 経由でコンテンツ取得する方が高速である。

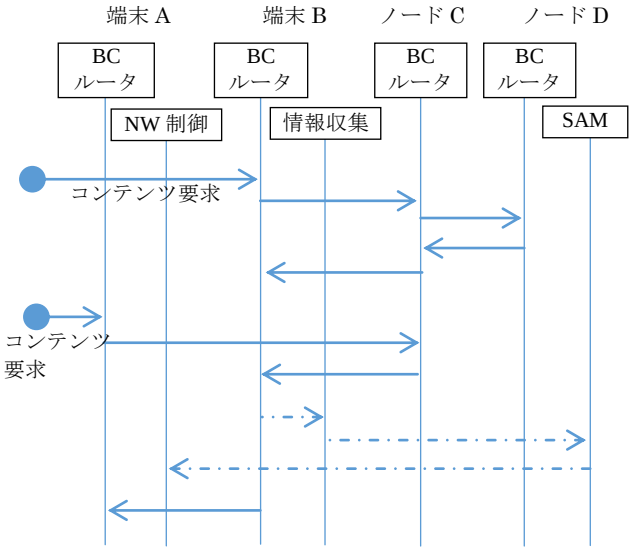

図 4　シーケンス図

以下図 4 をもって説明するように、本機能を使うことで上記制御が容易に実現できる。基本的な動作は以下である。まず、端末 B からコンテンツ要求を行う。初回なのでノード C を経てノード D からコンテンツを取得する。次に、A からコンテンツ要求を行う。今度は端末 B にコンテンツが存在するため B に送信要求が行われ、B から A にコンテンツが送信される。端末 B に対してコンテンツの送信要求が行われた場合には、端末 B の BC ルータはコンテンツ要求元の端末アドレスやコンテンツ名等の情報をログ出力する。

制御の流れは以下である。初期状態として端末 A が WiFi、端末 B が 3G につながっているとする。端末 B には、ログを監視する情報収集モジュール、端末 A には、アクセス NW の情報を取得する NW 情報収集モジュール及び NW 切り替えを行う NW 制御モジュールが動作している。NW 情報収集モジュールはアクセス NW 状況を監視し、変更がある場合に SAM に送信する。ログ監視モジュールは端末 B に対するコンテンツ送信要求のログを取得し、SAM に送信する。SAM では、コンテンツ送信要求情報取得時に、要求元及び要求先の端末が同一の WiFi に接続可能か判断し、可能の場合端末 A に WiFi に接続するように要求を行う。

判断のためのルールを Appendix A.2 に示す。本シナリオは 3 つのルールから構成される。1 つ目では、送られた情報がコンテンツ送信要求か判断する。コンテンツ送信要求である場合は、2 つ目のルールを呼び出す。2 つ目では、コンテンツ要求に含まれる要求元の IP アドレスを基に、当該端末のアクセス NW 情報を取得し、3 つ目のルールを呼び出す。3 つ目は、要求元が要求先と同じ WiFi に接続可能か判断し、可能な場合は要求元にアクセス NW 切替要求を行う。このようにシナリオは単純なルールの組み合わせで構成される。本シナリオは、計 1 2 1 行のルールでアクセス NW の切り換えを行うことができ、個の性能向上に貢献する。

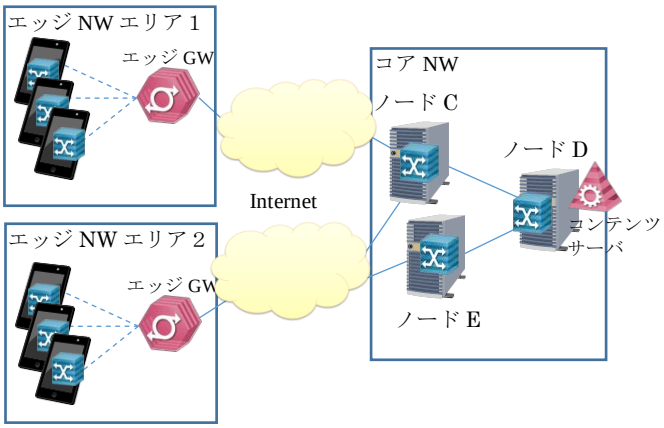

図 5　NW 構成図

3.2　シナリオ 2

２つ目のシナリオは、端末のサービス参加状況に応じ、サービス構成を変更するものである。アプリケーションはある程度の前提条件の下で性能を上げるアルゴリズムを持つが、その前提条件が大きく変わると動作モードやサービス構成を大きく変更する必要がある。この前提条件の変化の認識がアプリケーションのアルゴリズムとは独立しており、複数アプリケーションに共通するものなら、基盤側で対応することで開発を容易にできる。ここでは、エッジエリアでの携帯端末の NW 接続状況を基にしたサービス参加動向により、制御を行うシナリオを考える。

ここで図５に示される NW を想定する。図に示すように、ユーザは二つの物理的に離れた場所からサービスに参加可能とする。参加の分布に大きな変動がない場合は問題にはならないが、大きく偏りが起こる場合はサービス構成を変更することで性能改善につながる場合がある。例えば、一つのエリアからの接続がほとんどない場合と、二つのエリア均等に接続がある場合では、キャッシュの使われ方や NW 負荷が大きく異なることが予想される。

ユーザ情報に応じて処理負荷の軽いノードを停止することができればリソースの節約を図ることができる。BC から認識される仮想化された NW を図６に示す。状態１は、エリア２への端末参加が少なく、ノード E の機能を停止した場合に該当する。エリア２への端末接続が増加すると、ノード C へのアクセスが増加しキャッシュが溢れるなどの負荷が増加する。例えば、ある条件を超えるとノード E の機能を回復することでサービスのトポロジー変更し、全体性能を調整することができる。

本機能を使うことで上記制御を容易に実現する。基本的な動作は以下である。まず、エリア２からの接続はなく、エリア１からのコンテンツ要求がある。各端末の BC ルータはコンテンツ取得時に取得時間や要求コンテンツ名などの情報をログ出力する。次に、エリア２からのコンテンツ要求が増える。

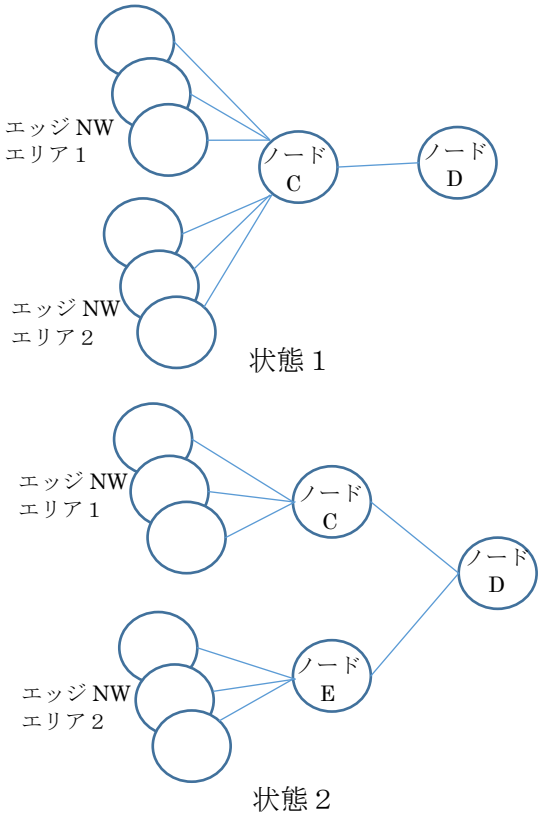

図 6　BreadCrumb NW トポロジー

制御の流れは以下である。初期状態は、図６状態１の NW 構成とする。各端末には、ログを監視する情報収集モジュールが動作しており、コンテンツ取得時間等のログを取得し、SAM に送信する。SAM では、当該情報を基にサービストポロジーの変更が必要か判断し、必要な場合はサービス構成変更要求を行う。サービス構成の変更は、基盤の機能として実現されている。

本シナリオのルールも前述と同様に３つのルールから構成される。１つ目では、送られた情報がコンテンツ取得時間情報か判断する。コンテンツ取得時間情報である場合は、２つ目のルールを呼び出す。２つ目では、コンテンツ取得時間の履歴を参照しエリアごとの取得時間の時間平均と要求コンテンツ数を算出し、３つ目のルールを呼び出す。３つ目は、平均時間が閾値を超えたかどうか、各エリアの要求コンテンツ数がキャッシュサイズ以下かを判断し、該当する場合はサービス変更要求を行う。本シナリオのルール記述は前述と構造が同様のため記載は省略するが、計１３７行でサービス構成制御を行うことができ、サービス全体の性能向上に貢献する。

3.3　アクセス NW 切替の基礎評価

シナリオ１における、アクセス NW の切り替えによるコンテンツ取得時間の影響及び端末情報の転送サイズの基礎評価のための測定を行った。端末通信環境を表２に示す。

ファイルサイズを変更しながら端末 B から端末 A へのコンテンツ配信における時間を測定した。端末 A が 3G 接続

表 2 通信環境

携帯端末	Nexus 4
3G	FOMA データ通信 最大 14Mbps
WiFi	IEEE802.11g 最大 54Mbps

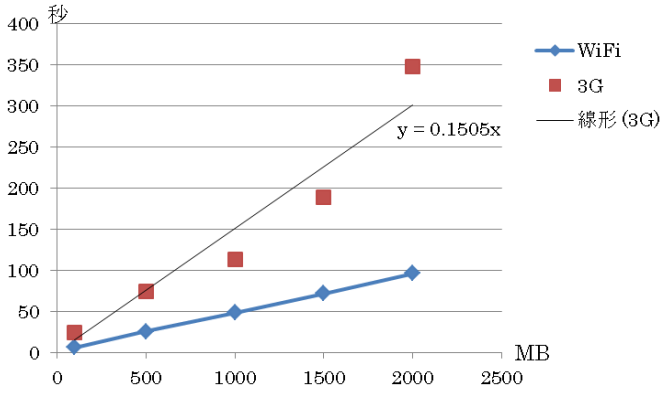

図 7 コンテンツ取得時間

の場合と WiFi 接続の場合の結果を図 7 に示す。3G 接続はバラつきがあるため線形近似をとっているが、WiFi の方が同一サイズで概ね 3 倍高速に取得できることが確認できる。実利用では、切り替え時間の影響も考慮する必要がある。

SAM への情報通知は端末 B へのコンテンツ送信要求時に行われるが、通信頻度も少なく転送量も数 MB 程度であり NW への負荷は少ない。

3.4 ルール規模評価

シナリオ 1 、 2 でコンテンツ配信における異なるユースケースを見た。どちらもエッジアリアの情報を基に制御することで、サービスの性能を向上させるものであった。表 3 に示すように、どちらも 100 行程度の平易な記述を行うことで性能拡張ができ、サービスで独自に対応するのに比べエッジエリア情報を用いた NW 制御が容易に実現できると言える。

表 3 ルール規模

	行数
シナリオ1	121行
シナリオ2	137行

シナリオ 1 に関する評価において、アクセス NW の制御を行うことで、コンテンツの取得時間を約 1/3 に削減できることを確認した。これは、エッジエリア情報を用いることで、個の性能向上を図るリソース制御が効果的に行えることを示す。

4. おわりに

エッジエリアの情報に基づいたプログラマビリティを導入することで、サービスに容易に NW リソース制御などの拡張機能を付加できる手法を述べた。サービス開発者が、アプリケーションと独立にエッジエリア情報を基にリソース制御するルールを記述し、エッジエリアでしか取得できない情報を基にしたサービスを容易に構築可能にすることができることを示した。コンテンツ配信を行うサービスを例に動作検証を行い、エッジエリア情報を用いた制御が容易に記述できることを確かめ、アクセス NW 制御によりサービス性能が向上することを確認した。WiFi や 3 G などのアクセス NW の切り換えやユーザのサービス参加状況によるサービス構成の変更などは、当初のコンテンツ配信サービスの機能として想定しているものではなかったが、本機能によりエッジエリア独自の情報を基に本来のサービスとは独立に専門的な知識も必要なく対応が可能となった。

情報爆発の時代、増え続ける情報量をうまく取扱い、必要な情報を必要な人に届けることが重要になってくる。コンテンツ配信のユースケースで本手法の有効性を見たが、実際の現場において有効であるか客観的に評価する必要があり、今後の課題とする。今後は、現状に則した様々なユースケースで本基盤の実用性を確認していきたい。

謝辞

本研究の一部は、独立行政法人情報通信研究機構(NICT)の委託研究「新世代ネットワークを支えるネットワーク仮想化基盤技術の研究開発」(課題イ：サービス合成可能なネットワーク・プラットフォームの研究開発)によるものである。

参考文献

1) 飯星貴裕, 才田好則, 渡邊義和, 森田弦, 狩野秀一, “仮想化ノードにおける容易なネットワークサービス実験のためのスマートデバイスへのスライス延伸,”電子情報通信学会ネットワーク仮想化時限研究会, No. NV2014-8, Apr.2014.
2) NTT, “高レスポンスやビッグデータ処理が要求される新たなアプリケーションの開拓を推進する「エッジコンピューティング構想」を策定”, NTT 持株会社ニュースリリース, 2014.1.23, http://www.ntt.co.jp/news2014/1401/140123a.html
3) Cisco, “IoT 時代の新たなイノベーションを加速させるフォグコンピューティングのプラットフォーム Cisco IOx を発表,” 日本版ニュースリリース, 2014.2.25, http://www.cisco.com/web/JP/news/pr/2014/012.html
4) Adrian K. Clear, “A situation-oriented framework for programming context-aware applications,” in *Proceedings of the 10th ACM international conference on Ubiquitous computing (Ubicomp '08) Adjunct Programs*, Sep. 2008, pp.87-91.
5) Brian Y. Lim and Anind K. Dey, “Toolkit to support intelligibility in context-aware applications,” in *Proceedings of the 12th ACM international conference on Ubiquitous computing (Ubicomp '10)*, Sep. 2010, pp.13-22.
6) 高部優一郎, 山際基, 上原稔, “センサーネットワークとクラウドの効率的な連携方法の提案,” 信学技報, vol. 112, no. 2, CPSY2012-8, pp. 43-48, Apr. 2012.
7) 森雅生, 中藤哲也, 廣川佐千男, “マッシュアップの軽量実装のための提案,” 電子情報通信学会 第 18 回データ工学ワークショップ(DEWS2007), C7-152, 2007.03.
8) 船橋一訓, 西原康介, 酒井淳嗣, 竹中崇, “端末クラウド協調サービスのための柔軟な制御記述,” 電子情報通信学会総合大会講演論文集, 2013 年(基礎・境界), 288, Mar.2013.
9) 柳生智彦, “コンテンツ指向ネットワークにおけるインネットワーク誘導拡張方式の相互作用に関する評価,” 信学技報, vol. 112, no. 463, NS2012-234, pp. 403-408, Apr.2013

Appendix

A.1 情報判断ルール XML の構造

rule			effect アトリビュート: ルール判定処理の実行される条件で、指定できるのは ALWAYS と REVIEW_ONLY の２通り。ALWAYS (default)では常に実行される。REVIEW_ONLY では後述する review 指定で呼ばれたときのみ判定される。
	rule_id		ルールを一意づける ID。ルール登録の際に自動採番する。評価はルールの登録順に行う。
	rule_name		ルールを一意づける名前。指定は任意で、指定する場合は名前がユニークである必要がある。 ルールを登録する際に名前をつけておくことで、アクションでルールを指定できる。
	target		絞り込み対象。 ・expire アトリビュート: リソース有効期限。ルール判定処理の実行される時刻から指定された期間のリソースを評価対象とする。
		subjects	対象とする端末の範囲を指定する。any_terminal, all_terminal, terminal_id の３通りのいずれかを選択する
		any_terminal	通知が行われた端末一つを指定。
		all_terminal	品質情報があるすべての端末を指定。
		terminal_id	特定の端末を指定。
		resources	
		resource	評価対象のリソースを指定する。指定可能なのは service_id, terminal_id, パラメータの key 名。 後述する pos 指定を行う場合、順序を確定するため指定する。
	condition		条件群
		match	条件 ・times アトリビュートで、繰返し回数を指定可能（オプション)。最新データから比較して、連続して繰返し回数満たすときに条件成立とみなす。 ・pos アトリビュートで、データの参照位置を示す。key の値について、最新値（pos=0）からさかのぼって pos 番目の値を参照する。 （注） pos パラメータを指定する場合は、リソース指定(resource)による絞込みが必要である。指定をしないと品質時刻順に並べて評価する際に対象外のリソースの ID を参照する可能性がある。
		key	リソース名 ・key_type アトリビュートで、リソースの種別（QUALITY_KEY, STATE_KEY）を指定する。 ・terminal_id アトリビュート: key の参照先端末 ID を示す。 terminal_id の指定は、key_type の指定と組み合わせて判断を行う。 key_type=QUALITY_KEY(デフォルト)の場合、指定された端末 ID に対する品質データ(quality)の要素名(key)が指定されたものとして、指定端末 ID の最新値に変換して判断を行う。(terminal_id を指定しないときと同じ動作) key_type=STATE_KEY の場合、指定された端末 ID に対するステート値の要素名(key)が指定されたものとして、指定端末 ID のステート値に変換して判断を行う。
		value	比較対象値。参照先は自端末のみのため、value では terminal_id は指定できない。値種別を value_type アトリビュートで指定する。 LITERAL (デフォルト) value は固定値として判断。value に記述した値がそのまま文字列もしくは数値として扱う。 QUALITY_KEY value は品質データの要素の値として判断。品質データ(quality)の要素名(key)が指定されたものとして、その値に変換して扱う。 STATE_KEY value はステート値として判断。ステート値の要素名(key)が指定されたものとして、その値に変換して扱う。

			operator	演算子。記述や実装簡易のため、LESS, MORE, EQUAL, NOT_EQUAL, BEFORE, AFTER の 6 つ。
		match/function		ユーザ定義関数条件
			key	
			value	
			operator	
			function alias	
	action			アクション: condition 条件群のすべてを満たしたときに処理を行う
		terminal_request		端末リクエスト
			terminal_id	固定値の他、${<キー名>} を指定することで参照可能。※複数端末の指定不可
			method	POST / PUT / GET / DELETE
			port	
			path	
			header	送信するメッセージのヘッダ内容
			body	送信するメッセージのボディ内容
			state	アクションの結果で設定する指定値 ・status_code アトリビュート: HTTP リクエストや端末リクエストの結果のステータスコードによりステート値を設定する。 比較は 3 桁の前方一致で評価する。例えば、<state status_code=“2”> はステータスコードが 2xx (200 や 201 など)とマッチする。state の条件はファーストマッチとする。 接続失敗など「応答がない時」を指定する場合 NO_CONNECT を指定する。どれにも該当しないものを指定する場合 DEFAULT を指定する。 ・save アトリビュート: false (デフォルト)アクションは成功とみなし、保存しない。 true アクションは失敗とみなし、HTTP リクエストおよび端末リクエストを実行した内容を保存する。
				key
				value
		http_request		HTTP リクエスト
			method	POST / PUT / GET / DELETE
			uri	送信先
			header	送信するメッセージのヘッダ内容
			body	送信するメッセージのボディ内容
			state	アクションの結果で設定する指定値
				key
				value
		state		ステート値設定
			key	
			value	value_type アトリビュート: value_type に指定する値の種類を指定する。指定可能な種類は LITERAL, QUALITY_KEY, STATE_KEY, FUNCTION_ALIAS の４つである。 LITERAL (デフォルト): 固定値として、value の値がそのまま文字列もしくは数値として扱う。 QUALITY_KEY: 品質データ(quality)の要素名(key)が指定されたものとして、値に変換して扱う。 STATE_KEY: ステート値の要素名が指定されたものとして、値に変換して扱う。 FUNCTION_ALIAS: 関数の戻り値を指定されたとして、関数の名前(name)を指定する。
		review		再評価指定ルール名を指定。 通常は情報が送られてきたタイミングで登録されたルールで評価を行う。しかし、STATE 値を更新した場合など、すぐに再評価したいときに再評価指定を行う。

A.2 シナリオ 1 のルール記述

```
<?XML version="1.0" encoding="UTF-8"?>
<rule
XMLns:xsi="http://www.w3.org/2001/XMLSchema-instanc
e"
xsi:noNamespaceSchemaLocation="../../schema/sam.xsd">
<rule_id>1001</rule_id>
 <rule_name>rule1</rule_name>
 <target>
  <subjects>
   <any_terminal />
  </subjects>
  <resources>
   <resource>terminal_id</resource>
   <resource>service_id</resource>
  </resources>
 </target>
 <condition>
  <match>
   <key>package</key>
   <value>com.nec.jp.cdpf</value>
   <operator>EQUAL</operator>
  </match>
  <match>
   <key>message</key>
   <value>cache_hit</value>
   <operator>EQUAL</operator>
  </match>
  <match>
   <key>reqIPADDR</key>
   <value>FALSE</value>
   <operator>NOT_EQUAL</operator>
   <function alias="ret_ipaddr">
    /opt/sam/function/getReqIPAddr.sh
   </function>
  </match>
 </condition>
 <action>
  <state>
   <key>req_ipaddr</key>
   <value value_type="FUNCTION_ALIAS">
    ret_ipaddr
   </value>
  </state>
  <review>rule2</review>
 </action>
</rule>
```

```
<?XML version="1.0" encoding="UTF-8"?>
<rule effect="REVIEW_ONLY"
XMLns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../schema/sam.xsd">
 <rule_id>1002</rule_id>
 <rule_name>rule2</rule_name>
 <target>
  <subjects>
   <all_terminal />
  </subjects>
  <resources>
   <resource>terminal_id</resource>
   <resource>service_id</resource>
  </resources>
 </target>
 <condition>
  <match times="1">
   <key>terminal_id</key>
   <key>ip_address</key>
   <key key_type="STATE_KEY">req_ipaddr</key>
   <value>FALSE</value>
   <operator>NOT_EQUAL</operator>
   <function alias="ret_tid">
    /opt/sam/function/getReqTID.sh
   </function>
  </match>
 </condition>
 <action>
  <state>
   <key>req_tid</key>
   <value value_type="FUNCTION_ALIAS">
    ret_tid
   </value>
  </state>
  <review>rule3</review>
 </action>
</rule>
```

```
<?XML version="1.0" encoding="UTF-8"?>
<rule effect="REVIEW_ONLY"
XMLns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../schema/sam.xsd">
 <rule_id>1003</rule_id>
 <rule_name>rule3</rule_name>
 <target>
  <subjects>
   <any_terminal />
  </subjects>
  <resources>
   <resource>terminal_id</resource>
   <resource>service_id</resource>
  </resources>
 </target>
 <condition>
  <match>
   <key key_type="STATE_KEY">req_tid</key>
   <value>termB</value>
   <operator>EQUAL</operator>
  </match>
  <match>
   <key>connectedLS</key>
   <value>WIFI</value>
   <operator>EQUAL</operator>
  </match>
  <match>
   <key terminal_id="${req_tid}">connectedLS</key>
   <value>MOBILE_3G</value>
   <operator>EQUAL</operator>
  </match>
 </condition>
 <action>
  <http_request>
   <method>POST</method>
   <uri>http://localhost:1234</uri>
   <header></header>
   <body>Hello</body>
  </http_request>
 </action>
</rule>
```