7S-01

サービス指向アーキテクチャHAMANAにおける サービス毎にパケット処理するゲートウェイの設計と実装

清水 倫人 † Jesu Peter Maglutac† 大竹 淳 †† 寺岡 文男 †† 金子 晋丈 ††
† 慶應義塾大学大学院理工学研究科
†† 慶應義塾大学理工学部

1 はじめに

ネットワークサービスを利用するユーザ (以下，サービスユーザ) はサービスの提供・享受の仕方に多様な要望を持つが，既存技術の組み合わせではサービスユーザの多様な要望を統一的に満たすことができない．要望の例として，サービスの品質を保証するネットワークの利用 (通信への要望)，サービスユーザが保持する端末に加えてネットワーク上でのデータ処理 (データ処理への要望) 等が存在する．時には希望するネットワークの利用かつ通信途中でデータを処理等の，要望が共存する場合も存在する．以上よりサービスユーザの要望を通信とデータ処理の観点から統一的に実現することが求められている．

ネットワーク仮想化や NFV (Network Function Virtualization)，エッジコンピューティング技術はサービスユーザの要望を部分的にしか満たすことができない．

筆者らは通信とその過程でのデータ処理を統一的に，サービスユーザの要望に応じて行うサービス指向ネットワーキングアーキテクチャ“HAMANA” を考案している [1]．HAMANA はサービスユーザの要望を反映するためのアプリケーション層ゲートウェイとコントローラを配置する．

本稿は HAMANA におけるアプリケーション層ゲートウェイの設計と実装について述べる．

2 関連研究

サービスユーザの通信への要望を満たす要素を持つネットワーク仮想化技術と，データ処理への要望を満たす要素を持つネットワーク上でのデータ処理技術について述べる．

物理ネットワークを仮想化し、複数の仮想ネットワーク (スライス) を構築するネットワーク仮想化技術が研究されている．近年ではパケット処理粒度でのネットワーク仮想化技術が出現しており，多様な特性を持つスライスを作り出すことを可能となっている．[2] スライスに接続する手法として，トンネルや VLAN の ID 毎に接続するスライスを決定するゲートウェイ AGW を利用する．しかしながら，この手法ではあるエンド端末が単一のネットワークアドレスでネットワークサービス毎に異なるスライスへアクセスすることができない．

NFV やエッジコンピューティング，アクティブネットワーキング等，ネットワーク上でパケットやデータを処理する技術が普及しつつある．しかしながら，ネットワーク上にデータ処理機能を追加する研究はサービスを享受するユーザ主体ではない，あるいは機能の開発に制約を伴う問題を抱えている [3][4].

3 HAMANA におけるゲートウェイの設計

3.1 HAMANA

図 1 に HAMANA の概要を示す．HAMANA はサービスユーザ，コントローラ，アプリケーション層ゲートウェイ (以下，ゲートウェイ) の 3 つで構成される．以下に HAMANA の動作概要を示す．エッジネットワークに存在するサービスユーザはサービスを提供・享受する．利用するスライスや通信途中でのデータ処理の導入に対する要望を REST API と XML を利用してコントローラに通達する．コントローラはサービスユーザからの要望を受信後，サービスを利用させるために必要な情報を収集 (サービス解決) する．通信相手のゲートウェイ及びコントローラを探索し，サービスユーザが希望するスライスを介してサービスを提供できるか，ゲートウェイで使われているアドレスやポートを問い合わせる．コントローラはサービス解決後にサービスユーザの要望を反映できるようにゲートウェイを設定する (図 2)．具体的にはサービスユーザが要望するデータ処理をプロセスとしてゲートウェイに組み込み，スライスと接続する．ゲートウェイは設定に基づき，サービスユーザから届いたパケットをデータ処理プロセスあるいはスライスに転送する．

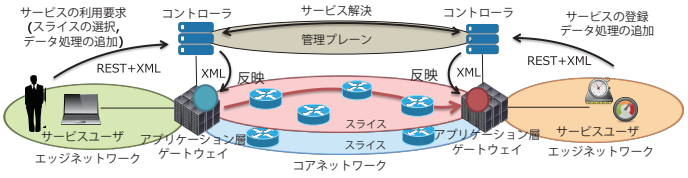

図 1: HAMANA の概要

3.2 アプリケーション層ゲートウェイ

アプリケーション層ゲートウェイはコントローラからの指示に基づき，(1) データ処理プロセスの組み込み，(2) データ処理プロセスの起動，(3) サービス毎のパケット処理，(4) スライスへのパケット転送の 4 つの処理を必要とする．

3.2.1 データ処理プロセスの組み込み

本稿で対象とするデータ処理プロセスの内容はサービスユーザ任意で決定できる．各データ処理プロセスの実行には多様な実行環境を必要とするため，本提案ではデータ処理プロセス単体を組み込むのでは無く，実行環境と両方を組み込む．筆者らは多くのサービスユーザからデータ処理プロセスを組み込むことを考え，実行環境の仮想化技術であるコンテナを利用する．コンテナによる仮想化は一部カーネル機能を使うことができない欠点を持つ一方で，非常に軽量である利点を持つ．

サービスユーザはデータ処理プロセスを含むコンテナを自身で開発あるいはサービス提供者からアプリケーションと共に取得する．そしてコンテナを拡張性の高い REST API と XML を利用してコントローラへ送信する．コントローラは受信したコンテナをデータベースに保存し，サービスユーザが通信の開始を要求する際にゲートウェイに組み込む．

3.2.2 データ処理プロセスの起動

サービスユーザが要望するデータ処理プロセスを組み込んだ後，ゲートウェイはそのデータ処理プロセスを起動する．サービスユーザが要求するデータ処理プロセスが単一でなく複数存在する可能性を考慮し，ゲートウェイは柔軟なデータ交換を可能とするデータ処理プロセス間ネットワークを構築する．そして，データ処理プロセス間ネットワークを制御することで異なるサービスユーザからの不正なアクセスを防止する．データ処理プロセスを含むコンテナにネットワークアドレスを付与し，コンテナ間でネットワークを形成した上でデータ処理プロセスを起動する．ゲートウェイはデータ処理プロセスを起動する上で，どのパス名のプログラムを起動するか，どのようにデータ処理プロセス間ネットワークを構築するか等の情報をコントローラから XML 形式で取得する．

3.2.3 サービスユーザ毎のパケット処理

データ処理プロセスの起動完了後，ゲートウェイは到着パケットをサービスユーザの要望に沿うように処理する．ゲートウェイに仮想スイッチを配置し，サービスユーザの要望に

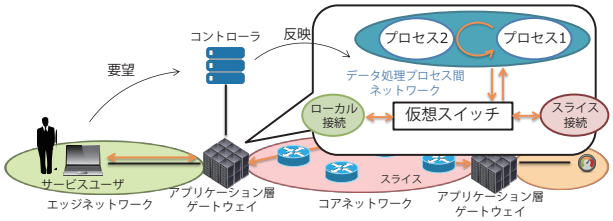

図 2: アプリケーション層ゲートウェイの概要

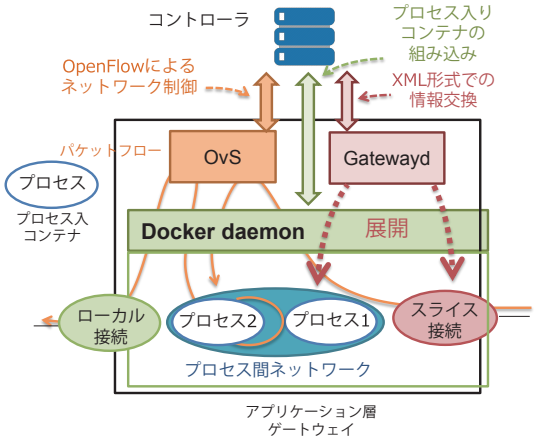

図 3: 実装モジュール

応じて到着パケットがデータ処理プロセス，スライス，ローカルネットワークへ向かうよう OpenFlow で制御する．

3.2.4 スライスへのパケット転送

スライスで利用されるプロトコルはスライス毎に異なり，時にはローカルネットワークと異なる場合がある．そこでパケットをスライスあるいはローカルネットワークへ送信する際にはプロトコルを変換しなければならない．本ゲートウェイは各ネットワークに接続するためにプロトコルを変換するプロセス (接続プロセス) を一つのデータ処理プロセスとして配置する．スライスへの接続プロセスはスライス提供者から提供される前提とする．

4 実装

3 章で述べた設計に基づき，ゲートウェイのプロトタイプを試作した．図 3 は実装モジュール図を示す．本ゲートウェイはモジュールとして Gatewayd，Open vSwitch，Docker を持つ．Gatewayd は XML を利用してコントローラと起動すべきデータ処理プロセスに関する情報を交換し，Docker を介してコンテナ及びデータ処理プロセスを起動する．Open vSwitch は OpenFlow により制御され，サービスユーザから届いたパケットをデータ処理プロセスあるいはスライスへ転送する．

5 評価

本ゲートウェイにサービスユーザの要望を実現するための機能を実装した上で，設定に要する時間を評価した．コントローラとゲートウェイには，Intel Xeon E5-2680 CPU を 2 つ，64GB メモリ，OS として Ubuntu 14.04 を搭載したマシンを利用しており，マシン間は 10Gbps のリンクで直接接続されている．

5.1 評価項目

評価項目は以下のとおりである．

1. ゲートウェイへのコンテナの組み込み時間
   コントローラからコンテナをゲートウェイに送り，ゲートウェイでコンテナ及びプロセスを起動可能な状態にするまでの時間を測定する．ただし，コンテナ全てを送るのではなく，ベースコンテナからの差分 (データ処理プロセスや実行に必要な環境) のみを送信し，予めベースコンテナを持つゲートウェイでコンテナを再構築する．

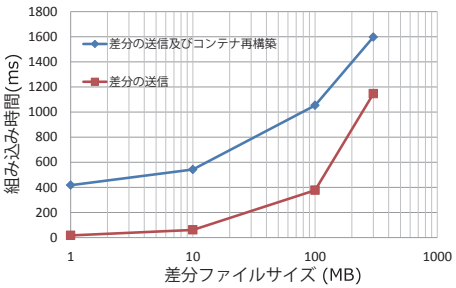

図 4: ゲートウェイへのコンテナの組み込み時間

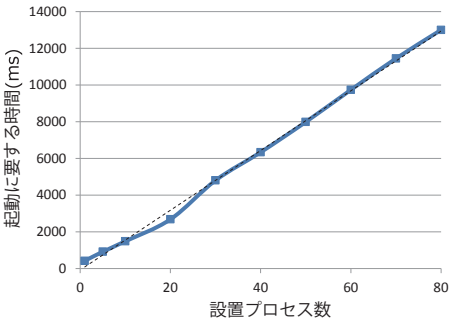

図 5: プロセスの起動に要する時間

2. データ処理プロセスの起動に要する時間
   コンテナに IP アドレスを付与し，データ処理プロセス間ネットワークを形成した上でプロセスを起動完了するまでに必要とする時間を測定する．

5.2 評価結果

図 4 は差分ファイルサイズを変更した際のコントローラからゲートウェイへのコンテナの組み込み時間を示している．この結果から約 100MB の差分 (データ処理プロセス及び実行に必要な環境) を組み込むのに約 1 秒かかった．図 5 は複数個のコンテナ及びデータ処理プロセスの起動に要する時間を示している．6 個のデータ処理プロセスを起動するのに約 1 秒要した．プロセス数が増加してもおおよそ線形に増加しており，近似式 163 × (プロセス数) – 73(ミリ秒) で表すことができる．上述の結果から，サービスユーザがデータ処理付きの通信開始を要求してからゲートウェイ全体の設定が完了するまでにおおよそ数百ミリ秒から数秒以上かかる結果となった．

6 おわりに

本稿では “通信及びその過程でのデータ処理” をサービスユーザの要望に応じて実現する HAMANA 及びそのアプリケーション層ゲートウェイについて述べた．アプリケーション層ゲートウェイはサービスユーザが要望するデータ処理機能を追加し，スライスへパケットを伝送する．評価によってゲートウェイの設定にはおおよそ数百ミリ秒から数秒以上の時間を必要とすることが分かった．今後はサービスユーザとコントローラ，アプリケーション層ゲートウェイで利用する XML を拡張し，より高い機能性の実現を目指す．

参考文献

[1] J. P. Maglutac and R. Shimizu and S. Otake and F. Teraoka and K. Kaneko, “Hamana: An Application-oriented Network Architecture with Service-driven Programmable Gateways,” IEICE-IA2015-68, pp.153–158, 2015.

[2] 山田一久，中尾彰宏，金田　秦，才田好則，雨宮宏一郎，“統合管理型ネットワーク仮想化基盤技術，” 情報通信研究機構研究報告，第 61 巻，pp.31–40，2015.

[3] D. Willis and A. Dasgupta and S. Banerjee, “ParaDrop: A Multi-tenant Platform to Dynamically Install Third Party Services on Wireless Gateways,” MobiArch’14, pp.43–48, 2014.

[4] J. W. Lee and R. Francescangeli and J. Janak and S. Srinivasan and S. A. Baset and H. Schulzrinne and Z. Despotovic and W. Kellerer, “NetServ: Active Networking 2.0,” ICC’11, pp.1–6, 2011.