7S-02 サービス指向アーキテクチャHAMANAにおけるクライアントAPIとテストアプリケーションの設計と実装

大竹 淳† 清水 倫人‡ Jesu Petar Maglutac‡ 寺岡 文男† 金子 晋丈†
†慶應義塾大学理工学部 ‡慶應義塾大学理工学研究科

1 はじめに

データセンタの成熟やIoTの出現はネットワークを利用した情報サービスの多様化を予期させる．しかし、現在のネットワークサービスの多くはユーザが自身の端末を用いてネットワークから直接データを受信，加工し，利用することを前提としており，(1) 送受信するデータの特性に応じた適切な品質のネットワークの選択や (2) データの加工に必要な計算資源の確保と計算, (3) ユーザ端末には最終的にユーザが必要とする情報のみを伝送，という統合的なネットワークサービスの枠組みは存在しない．そこで，筆者らはユーザが主導的にネットワークサービスを構成するサービス指向ネットワークアーキテクチャHAMANA[1]を検討している．HAMANAでは、エッジネットワークとコアネットワークを接続するゲートウェイを拡張，仮想化して上述の機能を実現すると共に，ユーザ端末がエッジネットワークに配置されたコントローラを介してゲートウェイに指示する機構となっている．本稿では，HAMANAにおけるユーザ端末の構成に注目し，ユーザ端末で動作するクライアントアプリケーションが利用するHAMANAのAPIの設計と実装について述べる．

2 サービス指向アーキテクチャHAMANA

2.1 HAMANAの概要とアーキテクチャ

HAMANAのアーキテクチャは，エッジネットワークとコアネットワーク，管理プレーンから構成される．(図1) HAMANAを利用するユーザはサービスユーザといい，エッジネットワークに位置する．サービスユーザはユーザ端末上のアプリケーションを介してネットワークサービスを提供・享受する．クライアントはコントローラに対しサービスユーザの要望を伝え，HAMANAの機能と資源を要求するアプリケーションである．コントローラは，クライアントからの要望を受け取り，統一的に処理する．コントローラは管理プレーンを介してサービス提供元コントローラと通信し，アプリケーション層ゲートウェイを制御する．コアネットワークは仮想ネットワークから構成され，それぞれの通信特性をもつ．ゲートウェイは，サービス毎に仮想ネットワークを選択し，またデータ処理機能を追加する．

2.2 コントローラへの要望

クライアントの要望には，ネットワーク上にデータ処理機能を導入し，計算資源を確保することや，ネットワークを選択することがある．HAMANAではこの要望をXMLに記述し，RESTを利用し，ネットワークサービスのセットアップ時と変更時，終了時にコントローラにそれぞれ送信される．データ処理機能はService Functionと記載され，使用するファンクション名またはプログラムを記述し，自由に順序を決定できる．ネットワークの選択はSlice Preferenceと記載され，クライアントがサービスを利用する際に使用する仮想ネットワークを選択可能である．クライアントはXML形式に従い自由に要求を記述可能である．

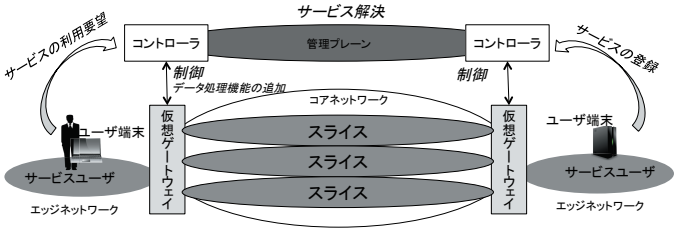

図 1: HAMANA のアーキテクチャ

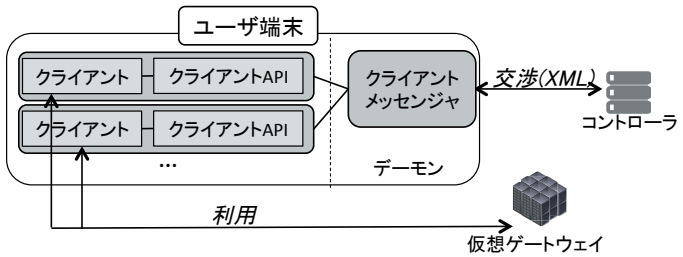

図 2: エンド端末の構成図

3 ユーザ端末の構成

3.1 ユーザ端末の要求事項

サービスユーザは複数のネットワークサービスをユーザ端末上で利用したいと考える．そこでユーザ端末は，複数のクライアントが動作可能であること，それぞれのクライアントの状況を統一的に管理すること，コントローラとネットワーク越しの交渉を統一的に扱うことが要求される．上記の要求事項を満たすユーザの構成を次節で述べる．

3.2 設計

要求事項から考えられる，ユーザ端末の構成は図2のようになる．クライアントはAPIを通じてクライアントメッセンジャ使用し，コントローラと交渉をする．クライアントはコントローラからの応答を取得し，ゲートウェイと直接通信をする．

Design and Implementation of Client API and Test Application
†Sunao Otake ‡Rinto Shimizu ‡Jesu Petar Maglutac †Fumio Teraoka †Kunitake Kaneko
†Faculty of Science and Technology, Keio University
‡Graduate School of Science and Technology, Keio University

3.2.1 クライアント API

クライアント API は 3 種類の関数を用意している．それぞれのプロトタイプを表 1 にまとめた．hSetup() 関数はアプリケーションを HAMANA 上で使用するための，はじめに必要な交渉であり，メッセンジャの Setup メソッドを使用する．クライアントは文字列配列を用意し，一番目に要望が記述された XML を指定し，2 番目はコントローラから応答された XML をセットするため空とする．引数に上記の文字列配列と int 型の変数を与えることにより，コントローラからの交渉に成功した際，応答された XML とコネクション ID を取得する．またその際に，以後の関数を使用する際に必要なコネクション ID も int 型として引数の変数にセットされる．次に hChange() 関数はアプリケーション使用中に，HAMANA での構成をアプリケーションが使用したくなった際に必要な交渉であり，メッセンジャの Change メソッドを使用する．引数には hSetuo() 関数で得たコネクション ID と変更の要望を記述した XML を指定すれば，文字列配列に応答の XML がセットされる．最後に，hEnd() 関数は HAMANA での利用を終わらせるための交渉であり，メッセンジャの End メソッドを使用する．コネクション ID を引数に取れば，コントローラにサービス終了を通知する．全ての関数の返り値はエラー番号である．

3.2.2 クライアントメッセンジャ

クライアントメッセンジャはクライアント API からの要求をデーモンで受け付け，要求に応じたメソッドを使用する．コントローラへの要求には REST API を利用する．以下は各メソッドの仕様を簡潔に示す．Setup メソッドは，サービス開始要求をコントローラに POST し，応答からコネクション ID とセッションキーをデータベースに格納し，API に応答する．応答情報が不十分であった場合，コントローラに対してポーリングする．Change メソッドは PUT を利用し，コントローラへサービス変更を送信する．交渉に必要なセッションキーをレコードから取り出す．その他の動きは Setup メソッドと同様である．End メソッドは DELETE を使用し，サービスを終了させる．データベースから当該の情報を削除し，API に終了が成功したことを通知する．

3.3 テストアプリケーション

今回のテストアプリケーションに，要望を反映するファイル転送アプリケーションを用意した．(図 3) このアプリケーションは，仮想ゲートウェイにファイルサイズを確認し，大きさによって任意にネットワークを選択する機能を要望する．ネットワークは容量が大きいネットワークと，容量が少ない仮想ネットワークを使用する．

4 実装と評価

4.1 実装

マルチプラットフォームで動作するように java と REST API を用いて実装した．どの環境でも簡単に用意できるデータベースである SQLITE を用いた．

表 1: クライアント API のプロトタイプ

プロトタイプ	機能
int hSetup(int[] connectionid，String[] xml)	サービス開始
int hChange(int connectionid，String[] xml)	サービス変更
int hEnd(int connectionid)	サービス終了

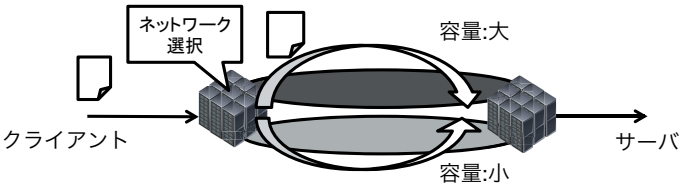

図 3: テストアプリケーション

4.2 評価項目と評価環境

テストアプリケーションから XML を用意し，API を使用しコントローラへ要求する．今回はクライアント API の評価に絞るため，用意された XML を単純に返信する擬似コントローラを作成し，それぞれのメソッドを動作させた際の実行時間を評価した．クライアントを動作させる CPU は 2.6 GHz Intel Core i5 で，メモリ 8GB を使用し，擬似コントローラの CPU に Intel(R) Xeon(R) CPU X5690@3.47GHz で，メモリ 24GB で，Cent OS を使用した．

4.3 評価結果

hSetup() 関数で 65ms 前後，hChange() 関数で 24ms 前後，hDelete() 関数で 21ms 前後所要した．

5 おわりに

本稿は HAMANA におけるユーザ端末側の構成を示した．ユーザ端末は HAMANA の使用の要望をするクライアントと要望を伝えるメッセンジャとメッセンジャのための API が存在する．クライアントが API を使用する実行速度はいずれの関数も 10ms から 100ms の範囲を所要する．また今回は HAMANA のユーザ端末側の評価に絞っているため，ゲートウェイでのデータ処理機能の展開や，コントローラ同士のサービス解決の時間を考慮していないので，クライアントはより多くの時間を所要する．

参考文献

[1] Jesu Petar Maglutac, Rinto Shimizu, Sunao Otake, Fumio Teraoka, and Kunitake Kaneko. Hamana: An Application-oriented Network Architecture with Service-driven Programmable Gateways. *IEICE-IA2015-68*, pp. 153–158.