ネットワークとアプリケーション協調を実現するプラットフォームの設計と実装

岸田崇志　野田真 鳥居裕二 根本直樹　野坂正昭

ネットワンシステムズ株式会社 ネットワークテクノロジー本部 研究開発部

〒140-0002 東京都品川区東品川 3-32-42 I.S.ビル

E-mail: {t-kishida, mk-noda, y-torii, n-naoki, nosaka}@netone.co.jp

あらまし　アクセスラインの広帯域化などにより，様々なサービスが誕生し利用されユーザの注目を集めている．そして，SaaS 型サービスと呼ばれる新たな形態で提供されるサービスも増加している．SaaS 型サービスにおいては，ネットワーク品質もアプリケーション品質の一部と見なされるため，サービスとネットワークはより柔軟に連携することが求められる．また，現在は，垂直統合型のサービスモデルから水平分離型サービスモデルへの移行期であり，海外のキャリアにおいても迅速かつ合理的なコストで魅力的なサービスを提供したいという要求が高まっている．そこで，本研究では，IPSphere Forum に参画し，マルチキャリアサービスを実現する SSS Platform の設計と試作を行った．

キーワード　SOA，ネットワーク管理/最適化，Web サービス，ネットワークサービス連携

Design and Implementation of the Platform to Harmonize an Application with a Network

Takashi KISHIDA　Makoto NODA　Yuji TORII　Naoki NEMOTO　Masaaki NOSAKA

Research and Development Division,　Network Technology Operations,　NetOneSystems,

IS BLDG.,　3-32-42 Higashi Shinagawa,　Shinagawa-ku,　Tokyo,　140-0002,　Japan

E-mail:　{t-kishida,　mk-noda,　y-torii,　n-naoki,　nosaka}@netone.co.jp

Abstract　Various services have been created and getting many attractions due to wider bandwidth of access line.　And also the services, which are called "SaaS", are now increasing tendency as new types of the service supplied.　In response to recent trend, the network and the service are expected to harmonize well so that it would regard the quality of the network as the quality of the services. The service models are facing with the period of transition from "vertical integration" to "structural separation".　Many carriers all over the world are looking for the mechanisms which are able to provide appealing network services by reasonable cost.　In this paper, we have tried to design and implement the prototype called "SSS Platform" which would achieve "multi-carrier network service" optimally.

Keyword　SOA,　Network Administration/Optimization,　Web Service,　Harmonized Network Service

1. はじめに

アクセスラインの広帯域化などネットワーク環境が高速化されたことにより，動画配信サービスなどのリッチなコンテンツを含む様々なサービスが誕生しユーザの注目を集めている．これらのサービスは，インターネット越しに提供されるものが多く，SaaS(Software as a Service)型サービスと呼ばれるネットワークを介してソフトウェア機能を提供するサービス形態も増加している．SaaS 型サービスにおいては，ネットワークがアプリケーションに与える影響はより大きくなってきている．

経済産業省は SaaS 向け SLA(Service Level Aggrement)ガイドライン[1]を，総務省は ASP・SaaS における情報セキュリティ対策ガイドライン[2]を公表している．このように，SaaS 型サービスにおいても SLA の担保やセキュリティの担保が従来のアプリケーションと同じ水準で必要になるため，End-to-End の SLA 確保が必要となる．そのため，コンテンツプロバ

イダからユーザ間までを含めた仕組みづくりも要求されている．これを実現するために，サービスとネットワークはより柔軟に連携することが求められている．

現在は，垂直統合型のサービスモデルから水平分離型サービスモデルへの移行期であるといえる．キャリア[1]も迅速にかつ合理的なコストで魅力的なサービスを提供したいという要望が高まっている．ARPU(Average Revenue Per User)の低下等，収益が下がってきている一方，インターネットトラフィックは年率 140%で増加[3]しており，設備投資のインパクトが大きくなりつつある．このような状況から，キャリアとしても迅速，かつ合理的なコストでサービスを展開することが望まれている．そのために，キャリアがコンテンツプロバイダのサービスを利用し，自サービスとして活用することや，キャリアが他のキャリアネットワーク設備を利用することが実現できると，大きな設備投資なくサービス展開が実現できるため高い ROI(Return on Investment)を実現できる可能性がある．また，キャリア間連携が実現すると地域の強み，キャリアの特性を生かしたサービスが実現できる．これを実現するために海外では積極的にキャリア間連携の方策が議論されており TMF (TeleManagement Forum)[4]や IPSF(IPShpere Forum)[5]等で通信事業者の業務に関わるオペレーションを柔軟かつ効率的にするための仕組み作りが進められている．日本では，総務省の通信プラットフォーム研究会において，通信プラットフォーム連携について議論されているところである[6]．このような各コンテンツプロバイダやキャリアがお互いの資源を利用しながらサービスを利用するモデルを本稿ではマルチキャリアサービスと呼ぶ．これを実現するためには，前述のようにアプリケーション連携とキャリア間連携の 2 つが重要となる．

そこで本研究では，ネットワークサービスを統合しキャリア間連携を促進する SSS Platform(SOA based Service Structuring Platform) の設計と開発を行った．

本稿では，マルチキャリアにおけるポリシ適用を対象とし，キャリア間連携のための必要要件とデータモデル設計及び，実装を行った．2章では，マルチキャリアサービスを実現する上での問題点と必要要件について述べ，SSS Platform の概要について示す． 3章では，システムの設計について述べ，4章でシステムに関する評価を行う．5 章でシステムの有用性について述べ，最後に6章でまとめと今後の課題を述べる．

[1]本稿では，通信事業者並びにインターネット接続サービスを提供するプロバイダのことをキャリアと呼ぶ．

2. SSS Platform

2.1. マルチキャリアサービスを実現するための問題点と必要要件

マルチキャリアサービスを実現するための問題点を述べる．

まず，アプリケーション連携という観点では，様々なサービスをキャリアサービスとして扱うために，コンテンツプロバイダ等が提供するサービスと連携する仕組みが必要である．そのために，ネットワーク制御をアプリケーション駆動型で行う必要がある．しかし，現状ではネットワークとアプリケーションは明確に切り離されている．コンテンツプロバイダからは SaaS ガイドラインに則すため End-to-End でネットワーク品質を保証したい要求も生じてきた．また，多様なサービスをサポートするためには，Web アプリケーションなどの SIP(Session Initiation Protocol)系以外の様々なアプリケーションとも連携する必要がある．

また，アプリケーションによって要求される SLA 条件も異なり，コンテンツプロバイダからエンドユーザまでの経路も異なる．そのため，どの程度の帯域が必要でどの程度の遅延に抑えたいといった要求がある場合，ネットワーク全体のサービスレベルを把握し管理する機構が必要である．

キャリア間連携という観点では，キャリア間でメッセージをやり取りするため，必要とされる情報を洗い出し，交換メッセージについて合意しておかなければ，お互いのメッセージが理解できない．つまり，キャリア間で仕様の取り決めをしなければならない．現在，様々な分野でグローバル連携を視野に入れた取り組みが進んでいる．例えば，医療の分野では HL7(Health Level Seven)[7]により医療情報のメッセージ規定がなされている．また，金融の分野では，XML Web サービスを利用してシームレスに連動するシステムを効率的に活用することを目的として ISO20022(UNIFI: UNIversal Financial Industry message scheme)[8]が規定されている．日本においても 2008 年 4 月に上場企業の財務データのやり取りに XBRL[9]が義務付けられた．しかし，Tele-Communication の分野ではフレームワークの作成は進んでいるもののまだ実装や運用レベルまでの実証が進んでおらず，キャリア間の相互接続実験も進んでいる状況とはいえない．

そこで，上記の問題点に対する必要要件を以下にあげる．

(i) ネットワークが様々なアプリケーションと連携可能なこと

(ii) アプリケーションが要求に応じて柔軟にネットワークをコントロールできること

(iii) キャリア間でやりとりされるメッセージが共通化されていること．また，それを推進する団体が存在すること．

2.2. SSS Platform による解決

前節であげた必要要件を満たすため SSS Platform は各項目に対応した以下の 3 つの特徴をもつ．

(i) Application Oriented Architecture

アプリケーションやワークフローとの親和性を高めるためメッセージ交換には SOAP[10]を用いた．SOAP では WSDL(Web Services Description Language)によりインタフェースを規定することで，API(Application Programming Interface)を公開することができ，ライブラリを使用することなく HTTP 通信を用い様々なアプリケーションと連携可能である．また，ロングタームトランザクションに対応するために WS-Addressing[11]などが標準化されており，ワークフローへの適応が容易である．

(ii) SOA Architecture Model

SOA(Service-Oriented Architecture)は，ビジネスプロセスの構成単位に合わせて構築・整理されたソフトウェア部品や機能を，ネットワーク上に公開し，これらを相互に連携させることにより，ビジネスプロセス実行システムを構築するシステムアーキテクチャである．

SSS Platform では，この概念を適用してキャリア内のネットワーク資源を抽象化し，API を統一することにポリシの違いを吸収する．その情報は組み替えが容易な形とし，ワークフローに応じた組み合わせを可能とする．ワークフローの記述言語としては BPEL(Business Process Execution Language)[12]などがある．それらによりアプリケーションからの要求とネットワーク資源の組み合わせを制御する．これによりアプリケーションの要求や，更には業務形態に則した要求を満たすことができる．

(iii) Global Standard Based Network

Tele-Communication の分野では TMF や IPSF などにおいてグローバル連携技術の議論がなされている．TMF では NGOSS(Next Generation Operations Systems and Software)フレームワークが策定されており，IPSF では SMS(Service Management System)と呼ばれるフレームワークが策定されている．IPSF では，フレームワークを実装レベルまで議論し現在相互接続試験が進められている．そこで，SSS Platform は，IPSF で定義されているメッセージ規定に基づいて設計を行った．

次に，SSS Platform のシステムモデルを図 1 に示す．図 1の(i)～(iii)は，それぞれ，前述の特徴に該当する．SSS Platform は，Service Abstraction 層(以下，SA 層)，Message Exchange 層（以下，ME 層），Service Orchestration 層(以下，SO 層)の 3 層に分かれている．

SA 層では，Service Mediator がキャリア内のネットワーク資源を抽象化する．ME 層では，抽象化された情報を Service Orchestrator との間で交換が行われる．SO 層では，収集した情報をアプリケーションの要求に応じて組み替えワークフローの作成が行われる．また，アプリケーションからのリクエストにより実行されるワークフローを選択する．

このモデルのうち SSS Platform では，SA 層と SO 層に IPSphere Technical specification R1.0[13]を採用し実装を行った．

2.3. IPSphere Forum

IPSF は，Infranet Initiative Council（IIC）を母体として，2005 年 06 月に設立された次世代のキャリアサービスをテクノロジー・ビジネスの両側面から検討する団体である．

世界中よりネットワーク機器ベンダやソフトウェアベンダ，キャリアが加盟しており，08 年 07 月時点にて会員数は約 30 社である．

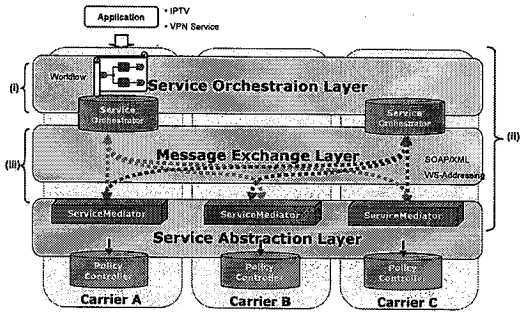

図 1. SSS Platform モデル

3. システム構成

図 2に SSS Platform の構成図を示す．

IPSF では SSS Platform での Service Orchestrator は SMS Parent と呼ばれ，Service Mediator は SMS Child と呼ばれる．また，エンドユーザへのサービス提供元となるコンポーネントを AO(Administrative Owner)と呼び，各キャリアのリソースをエレメントテンプレート化し登録するコンポーネントを EO(Element Owner)と呼ぶ．EO は，SMS Child を 1 つ以上持ち，AO は 1 つの SMS Parent を持つ．図 2では，Carrier A は，AO であり，かつ EO である．Carrier B は EO である．

3.1. 構成要素

3.1.1. SMS Child

配下のネットワーク機器が持つ帯域やフィルタ条

件等の設定を SA 層でエレメントテンプレートとして抽象化して管理する役割を持つ．それぞれのエレメントテンプレートに対し Unique ID(UID)を付与し，ME 層でエレメントテンプレートをやり取りし，SMS Parent へ登録する．また，サービスを起動する際は，SMS Parent から予め登録されたエレメントテンプレートの起動命令を受け，その命令を配下のポリシーコントローラ特有の命令に変換し送信する．各メーカのポリシーコントローラのインタフェースの差分を吸収する役割も持つ．

3.1.2. SMS Parent

SMS Child から登録されたエレメントテンプレートを管理する役割を持つ．また，エンドユーザがサービスを起動するための API を持ち，API へのリクエストを契機に，予め SMS Child から登録されたエレメントテンプレートの一覧から，SO 層で必要なものを組み合わせてサービステンプレートと呼ばれるワークフローの作成を行う．その後，サービステンプレートにリストされたエレメントテンプレートの所有者である SMS Child に対し，サービス起動や変更，停止を行いサービステンプレートの状態を管理する．SMS Parent は SMS Child と 1:N の関係を持つ．

サービステンプレートを用いユーザが利用したいアプリケーションに合わせて各エレメントテンプレートを組み合わせることで，アプリケーションの要求に合わせたリクエストを生成できる．

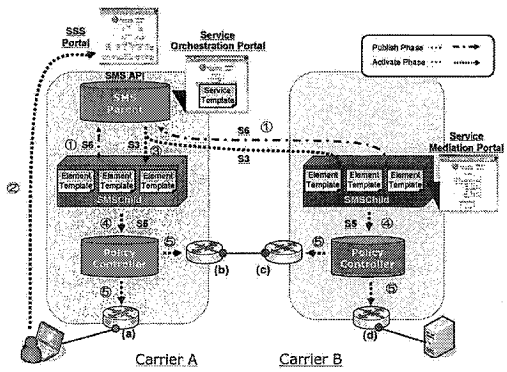

図 2. システム構成図

3.1.3. ポリシーコントローラ

各キャリア内のルータのポリシを制御する役割を持つ．SMS Child から送られてきた命令を，配下のネットワーク機器が処理可能な命令に変換し，ネットワーク機器へ設定を投入し実行する．SSS Platform では Juniper Networks 社の製品である SRC(Session and Resource Control)[14]を用いた．SRC は，主にポリシ管理や加入者の管理と AAA(Authentication, Authorization, Accounting)，およびネットワークリソースの管理を行う．SOAP インタフェースを持っており外部アプリケーションとの柔軟な連携が可能である．

3.2. インタフェース

SSS Platform のインタフェースは WSDL により定義されており，SOAP/XML でやり取りされる．次項より，図 2 の各インタフェースについて説明する．

3.2.1. SMS API

エンドユーザまたはコンテンツプロバイダに対し，SMS サービスを提供する為に用いられるインタフェース．ポータルサービスを作成するための API 群である．主な提供機能としては，起動可能なサービステンプレート一覧の取得や起動，停止，そして起動中のサービステンプレートの一覧の取得などである．

3.2.2. S3 インタフェース

SMS Child によって提供され，SMS Parent が SMS Child との通信に用いるインタフェースである．表 1 のメッセージをエレメントテンプレートの UID に基づいてやり取りし，それらを契機にユーザが各エレメントテンプレートと紐付く実際のネットワークリソース（抽象化されたサービス）について，サービスの実行を SMS Child に促す．

表 1. S3 インタフェースでのメッセージ

メッセージ	概要
Setup Start/Complete	サービスに必要なリソース確認を行う
Execute Start/Complete	サービスの実行/変更を行う
Assure Start/Complete	サービス品質の保障開始/終了を行う

3.2.3. S5 インタフェース

SMS Child とポリシーコントローラ間の通信に用いられるインタフェース．各ポリシーコントローラの仕様に依存する．SSS Platform では，SRC の SOAP インタフェースである DSA(Dynamic Service Activator)機能を用い，SMS Child とポリシーコントローラの間でやり取りする．送信される代表的な項目は，起動するポリシ名とオプション，ポリシーコントローラのユーザ名，パスワード等である．

3.2.4. S6 インタフェース

サービスの提供元となる SMS Parent に対し，各キャリアの SMS Child からエレメントテンプレートの登録を行う際に用いるインタフェース．パブリッシャインタフェースとも呼ぶ．送信する代表的な項目は，エレメントテンプレート UID や SMS Child のアドレスと SLA 条件，そしてそれを起動する際の課金情報等が含まれる．

3.3. 動作概要

SSS Platform の動作概要について示す．

動作は大きく分けて Publish Phase と Activate Phase に分けられる．Publish Phase は，SMS Child のエレメントテンプレートを SMS Parent に登録するフェーズであり，Activate Phase はユーザからのリクエストを元にサービステンプレートを起動するフェーズである．

3.3.1. Publish Phase

(1) 各キャリアがエレメントテンプレートを Service Mediator Portal(図 3)に登録し，SMS Parent に対して S6 インタフェースを使用し送信する(図 2 ①)．送信される項目は，ネットワーク機器固有の情報を抽象化したものであり，帯域情報などの SLA 条件や SMS Child のアドレス等である．

(2) Service Orchestration Portal(図 4)で利用できるサービスとしてエレメントテンプレートを組み合わせてサービステンプレートを作成する．サービステンプレートは，どの SMS Child をどのように起動するかを記述したものである．現在は，Web 画面から手動でサービステンプレートを作成するが，将来的には BPEL 等を用いアプリケーション単位で動的に作成する予定である．

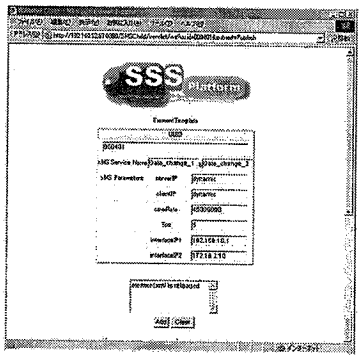

図 3. Service Mediation Portal 画面

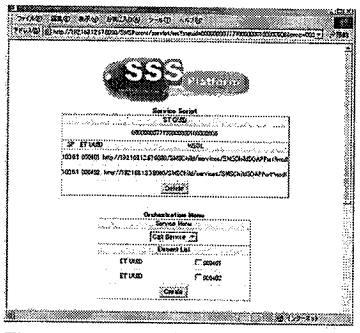

図 4. Service Orchestration Portal 画面

3.3.2. Activate Phase

(1) エンドユーザが SSS Portal(図 5)へアクセスし，サービスメニューリスト内の希望のサービスを起動する(図 2 ②)．Start ボタンを押下することで Web アプリケーションに紐付けられたサービステンプレートが起動される．

(2) サービステンプレート内にリストされたエレメントテンプレートを参照し，その内容に基づき各 Element Owner の SMS Child に S3 インタフェースを通じて命令を送信する(図 2 ③)．

(3) 各 SMS Child はエレメントテンプレートに記載された抽象化した内容から実行可能な命令に変換しポリシーコントローラに対して S5 インタフェースを使用し命令を送信する(図 2 ④)．

(4) ポリシーコントローラは，図 2 の各ルータの(a)～(d)で示したインタフェースに対してポリシを適用する(図 2 ⑤)．

(5) サービス起動後，図 6のように S3 インタフェースの各メッセージステータスと起動後のセッション情報が表示され，エンドユーザからコンテンツサーバまで要求する帯域が確保される．

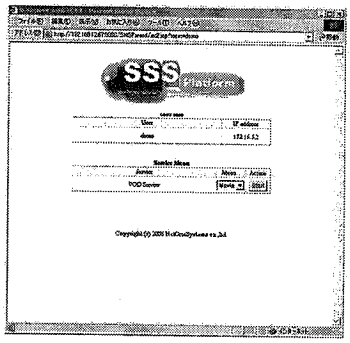

図 5. SSS Portal 画面(サービス起動前)

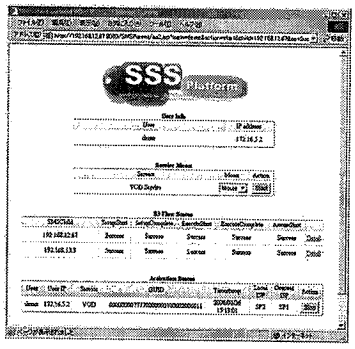

図 6. SSS Portal 画面(サービス起動後)

3.4. 開発環境と動作環境

SSS Platform の開発は表 2の環境で行った．SOAP エンジンには Apache Axis 1.4 を用い Eclpse3.2 上で開発を行った．SMS Child と SMS Parent の 2 つのコンポーネントに分かれており，WAR(Web Application aRchive)形式で提供される．これらのコンポーネントは Apache Tomcat 5.5 上でデプロイされるため，Apache Tomcat 5.5 がインストールされている環境であればOS を問わず動作が可能である．

表 2. 開発環境と動作環境

	種別
OS	Solaris10, CentOS4.4
Application Server	Apache Tomcat 5.5
SOAP エンジン	Apache Axis 1.4
言語	Java 1.5

4. 評価

4.1. ストリーム伝送ツールによる性能評価

SSS Platform を用いて異種ベンダの機器を交えた構成でキャリアを跨いだポリシ適用における評価を行った．実験構成を図 7に示す．

本構成では，ポリシの異なる 2 つのキャリア間で，SSS Platform を用いることで資源の抽象化と API の統一を行い，その API により帯域を確保させることを目的としている．キャリア B 内のユーザがキャリア A 内のコンテンツサーバから映像ストリームを受信する．コア網のルータには Cisco VXR-7200, Juniper M7i を用い，エッジルータには Juniper ERX-310 を用いた．

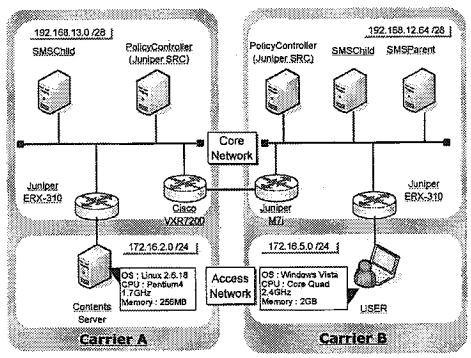

図 7. 実験構成

映像ストリームを想定してコンテンツサーバからエンドユーザに 60 秒間 CBR(Constant Bit Rate)の UDP ストリームを送信する．ストリームの送信には NLANR (National Laboratory for Applied Network Research)で開発されたストリーム伝送ツール iperf[15]を用いた．設定したパラメータは，帯域 768[kbps],ペイロード長 1470[byte]である．計測結果を図 8に示す．計測結果はエンドユーザで取得したものである．起動したサービスはコンテンツサーバからエンドユーザ向きを 45[Mbps]の帯域を確保し，デフォルトは 24[kbps]の帯域制御を行っている．図 8(a)で SSS Portal 画面からサービスを起動，図 8(b)でサービスを停止している．

図 8のスループット推移から，ルータに適切なポリシが適用されていることが確認できた．つまり，サービステンプレートに記述された通りにシステム全体での帯域確保・開放が正しく行われていることが確認できた．また，アプリケーション側から見てそれぞれネットワーク資源の違うキャリアでも SSS Platform を用いることで 1 つの API を起動するだけで帯域を確保することができた．

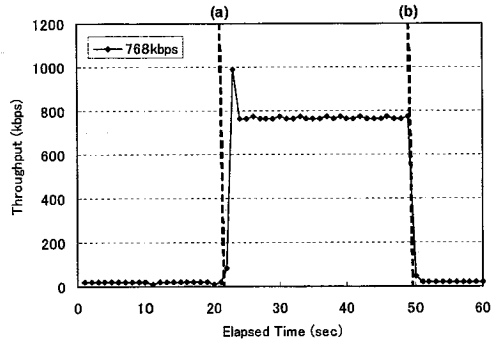

図 8. ポリシ適用によるスループットの変化

4.2. 対外接続実験

SSS Platform では，ME 層に IPSphere Technical specification R1.0 を採用している．同様に RedZinc 社でも，IPSphere Technical specification R1.0 に基づいたソフトウェアを開発している．そこで，ME 層の動作を検証するために RedZinc 社のソフトウェアと SSS Platform とのインタオペラビリティーテストを行った．接続実験構成図を図 9に示す．Juniper 社のシドニー支社で提供される PCT(Pre-Commercial Testbed)ゲートウェイに相互に接続して実験を行った．S6 インタフェースに関して，相互に SMS Parent にエレメントテンプレートが登録できることを確認できた．S3 インタフェースに関しても接続実験を継続中である．

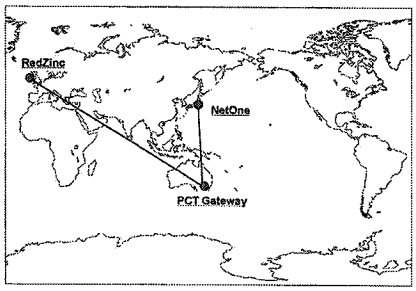

図 9. 対外接続実験構成図

5. システムの有効性

5.1. スケーラビリティーに関しての考察

5.1.1. サービスのステート数について

起動できるサービス数は，コンテンツプロバイダからエンドユーザ間の帯域に依存する．コンテンツサーバからエンドユーザまでのリンク数を n とすると各リンクの帯域はそれぞれ BWn と表すことができる．その中で最小のリンクがボトルネックとなるためサービスを起動できる最大帯域は(1)式で表せる．

$$MaxBW = \min\{BW_1, BW_2, \cdots, BW_n\} \quad \cdots(1)$$

End-to-End の帯域に依存するが，コンテンツサーバに着目すると，ボトルネックとなるのはコンテンツプロバイダとキャリア間のリンクである場合が多いと想定されるため，$MaxBW$ をコンテンツプロバイダとキャリア間の帯域と仮定する．その帯域はコンテンツプロバイダ向きインタフェースの帯域の総和となるため，以下のように最大セッション数を(2)式に定義する．

$$MaxSession = \frac{\sum_{i=0}^{n} EdgeIF_i}{MaxServiceBW} \quad \cdots(2)$$

$EdgeIF$：エッジルータのコンテンツプロバイダ向きインタフェースの帯域

$MaxServiceBW$：サービスの利用する最大帯域

そこで以下，コンテンツプロバイダ向きインタフェースが 10 Gb Ethernet のインタフェース数が 2 つであると想定し，サービスを H.264 のコンテンツサービスを提供するとした場合について考察する．

最も高品質なサービスを H.264 の HD 8Mbps(1920x1080,24fps)と想定すると，最小セッション数は，2 x 10Gbps / 8 Mbps = 2500 となり，1 コンテンツプロバイダ当たり 2500 セッションのサービスを起動することができる．

最も低品質なサービスを H.264 のモバイルコンテンツ 50kbps(176x144,10fps)の場合と想定すると，最大セッション数は，2 x 10Gbps / 50kbps = 400000 となる．

そのため，この場合のセッション数は 2500～400000 サービス/コンテンツプロバイダと想定できる．ただし，この値は使用するルータにもよる．

仮に 5 社のコンテンツプロバイダが 1 キャリアに収容されていたとすると 1 キャリアあたりセッションは高々12500～2000000 サービスなので，これが各キャリアの SMS Parent で管轄するステート数となる．HTTP 経由で行われるため WEB サーバの負荷分散技術や DB レプリケーションにより十分管理可能と考えている．

5.2. アプリケーション連携

SOAP を採用したことにより既存の Web アプリケーションやビジネスアプリケーションとの連携を行える仕組みを作ることができた．また，WSDL を公開することで開発者に対しても API を公開することができ，より多くのアプリケーションとの連携を誘発できる．また，SSS Platform で API を統一することによりアプリケーション側からは個々のネットワーク資源やポリシの違いを意識することを減らすことができる．このようにアプリケーションがネットワークと連携することによって，アプリケーションとネットワークの利用領域を拡大することができ，利用可能性を広げることができる．

インタフェース単位で機能を公開できるため，Parlay Group[16]の Parlay X などの他のフレームワークとも連携することができ，認証機構に OpenID[17] などを活用することも可能である．

5.3. キャリア間連携

キャリア間連携においては，フレームワークの策定だけでなく，実キャリア間で接続実験を行う必要がある．現在，IPSF では，SSS Platform でも採用している IPSphere Technical specification R1.0 を基に France Telecom や Telstra, Telus などキャリアが参加して接続実験を行う予定である．SSS Platform を用いて IPSF での参照実装を行うことで，接続実験において実際に動作させながら各キャリアの要望を汲み取りながらデータモデルを規定することが可能となった．メッセージの規定や進め方はまだ検討の余地はあるが，フォーラムできちんと方向を定め，合意形成されるならば，キャリアを跨いで課金やポリシ，そしてアプリケーションからの要求に則した形でネットワークも制御することも可能となる．今後は，他ベンダにより作成された IPSF フレームワークとも相互実験を進め，より発展的なフレームワークを作成していきたい．

6. おわりに

本稿では，ネットワークとアプリケーション協調を

実現する SSS Platform の設計と開発を行った．本研究では，アプリケーションとの連携部分とポリシのキャリア間連携についての設計と実装を行ったが，今後は，IPSF において各ソフトウェアベンダとのインターオペラビリティ試験やテストベッド環境でのキャリア間連携などの実証実験を行い，さらに認証，課金などを含めたモデルへ発展させていきたい．

謝辞

本研究に際し，Juniper Networks 社 Tod Shimizu 氏と Richard Bayliss 氏，RedZinc 社 Donal Morris 氏と Cameron Ross Dunne 氏に感謝致します．また，IPSphere Forum の皆様に有益なご助言を頂き感謝致します．

文　献

[1] 経済産業省，“SaaS 向け SLA ガイドライン”，*http://www.meti.go.jp/press/20080121004/03_guide_line_set.pdf/.*

[2] 総務省，“ASP・SaaS における 情報セキュリティ対策ガイドライン”，www.mhlw.go.jp/shingi/2008/07/dl/s0730-18o.pdf.

[3] 総務省，“我が国のインターネットにおけるトラヒックの集計・試算”，http://www.soumu.go.jp/s-news/2005/pdf/050125_3_1.pdf.

[4] “TM Forum”，http://www.tmforum.org/

[5] “IPsphere Forum”，http://www.ipsphereforum.org/.

[6] 総務省，“通信プラットフォーム研究会”，http://www.soumu.go.jp/joho_tsusin/policyreports/chousa/platform/pdf/080227_2_si1-2.pdf.

[7] “Health Level 7”，http://www.hl7.org/.

[8] “ISO 20022”，http://www.iso20022.org/.

[9] “XBRL Japan”，http://www.xbrl-jp.org/.

[10] “Simple Object Access Protocol (SOAP) 1.1”，http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[11] “Web Services Addressing (WS-Addressing)”，www.w3.org/Submission/ws-addressing/.

[12] “WS-BPEL”，www.oasis-open.org/committees/wsbpel/.

[13] “IPSphere Technical specification R1.0”，http://www.ipsphereforum.org/library/specifications.html.

[14] Juniper Networks，“Session and Resource Control”，http://juniper.co.jp/products_and_services/session_and_resource_control/.

[15] “iperf”，http://dast.nlanr.net/Projects/Iperf/.

[16] “The Parlay Group”，http://www.parlay.org/.

[17] “OpenID Foundation”，http://openid.net/.