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

北辻佳憲

クラウド技術を活用した柔軟性の高いサービスシステムの実現が可能になったことに触発されて、仮想ネットワークを実現する研究開発、ならびに仮想ネットワークを用いて柔軟性の高いネットワークサービスを実現する研究開発が活発に行われている。本稿では、この柔軟性の高いネットワークサービスの構築を可能にするサービス合成プラットフォームの研究開発について紹介する。

1　まえがき

インターネットは社会活動や経済活動を支えるインフラストラクチャに成長したが、新しい機能の導入が難しくなっており、アーキテクチャを根本から設計し直すことが期待されている。これに対して、我が国においては新世代ネットワーク[1]、米国では NSF の FIND（Future INternet Design）[2] イニシアティブにおいて、アーキテクチャの見直しが進められ、これまでにも CCN（Content Centric Networking）[3] や ID/ロケータ分離[4] 等の多数の斬新なアーキテクチャが提案されている。これらのアーキテクチャの評価には、大規模な実証実験が必須であり、JGN-X[5] や GENI（Global Engineering for Network Innovation）[6] 等の多数のテストベッドが日米欧で構築されている。

テストベッドでは、多数のアーキテクチャの実験が同時に実行されることが想定されるため、回線ならびにルータの CPU 等の、ネットワーク資源ならびに計算資源が各実験に独立して割り当てられて、実験同士が互いに干渉しないように保証することが必須である。ネットワーク仮想化技術は独立な資源割当てを可能とするため、多数のテストベッドで採用されている。このようなテストベッドでは、仮想的なルータと回線から構成される仮想的なネットワークであるスライスが実験ごとに割り当てられるのに加えて、仮想的なルータ上に実験者（以降、ユーザと呼ぶ）が独自のプログラムをプログラミングできるため、様々なアーキテクチャをスライス上に実装することが可能になる。

しかしながら、従来のネットワーク仮想化技術に基づくテストベッドでは、資源の割当てや解放などのネットワーク管理機能は提供されているが、十分なプログラミング環境は提供されていない。そこで本研究開発では、ネットワークに関する知識が十分でないユーザでも、従来開発されたプログラムモジュールを、トイブロックのように組み合わせてプログラミングできるような、柔軟なプログラミング環境（以降、プラットフォーム）を実現した。本稿では、ネットワーク仮想化を概観した後、本プラットフォームの設計と実装について述べる。

2　ネットワーク仮想化の概要

大規模なスケールでの実証実験を可能とする技術として、ネットワーク仮想化が期待されている。ネットワーク仮想化では、仮想的なルータ（以降、仮想ノードと呼ぶ）と回線（以降、仮想回線と呼ぶ）から構成されるスライスと呼ぶ仮想ネットワークを提供する。スライスは他のスライスと物理的なノード（仮想化ノードと呼ぶ）と回線を共有するが、各スライスに割り当てられた資源を仮想化基盤（スライスを提供する仮想化ノード・回線で構成された基盤ネットワーク）が保証する。

ネットワーク仮想化の実現には、資源の分離・スケーラビリティ・性能・拡張性・プログラム性・セキュリティと管理機能等の要件を満足することが重要である[7]。資源の分離は最重要な要件で、スライスに割り当てた仮想回線や仮想ノードの CPU の占有が保証される。仮想ノードで実行されるプログラムをユーザがプログラミングできるプログラム性は、仮想ネットワークの脆弱性を招くため、セキュリティの向上が必須となる。多数の実験を同時に実行可能なスケーラビリティも重要である。具体的には多数のスライスが同時に実行できることが重要で、複数のスライスを物理回線に多重化できるように、OpenFlow が利用されることがある[8]。

仮想ノードで実行されるプログラムは、パケットのフローだけでなくパケットを処理することが可能であり、前段落で述べた高い性能を求められる。さらに、ユーザのプログラムは多数の仮想化ノードに配置して

実行する必要があるため、どのように仮想化ノードに配置するかも重要である。

本稿で紹介するプラットフォームは、上述の要件の内、スケーラビリティ、プログラム性、ネットワークサービスを構成する機能管理に焦点を当て、スライス上で実行されるプログラム開発と配置をサポートし、更に複数の仮想基盤上でスライスを構成可能とする特徴を持つ。

3　サービス設計環境

3.1　概要

本研究開発によるプラットフォームは、仮想化基盤[9]が提供する仮想ノードで実行するプログラムの開発及び実証実験をサポートすることを目的としている。ユーザが新アーキテクチャをゼロから開発することは効率的でないため、x-kernel[10]やClickモジュラールータ[11]のように、プロトコル処理をモジュール化し、モジュールの再利用によりプログラム開発の生産性を高める。具体的には、仮想ノードで実行可能な最小の単位をブロックと呼び、これらを、トイブロックを組み合わせるようなプログラミング開発を提供する。

本プラットフォームは、図1に示すように、サービス設計環境、サービス配置環境、スライスエクスチェンジポイントから構成される。従来のClick等が1つのホストやルータで実行されるブロック群を対象としているのに対して、本プラットフォームは複数の仮想ノード上でのブロック群を対象とする。以下では、1つのアーキテクチャを実行するこれらのノード群をまとめてサービスと呼ぶ。なお、スライスエクスチェンジポイントは世界規模での実証実験を可能とするため、我が国以外で開発した仮想化基盤との相互接続を提供し、複数のネットワーク仮想化基盤上にスライスを構築することが可能である。

3.2　ブロック

チェックサム計算、パケット再送などのパケット処理を扱えるように、Clickモジュラールータのモジュールを、以下の通り、ブロックとして提供する。

(1) ユーザレベルClickブロック
Clickに従って開発されたブロックであり、ユーザ空間で実行される。

(2) カーネルレベルClickブロック
Clickに従って開発されたブロックであり、カーネルレベルClickドライバ上でのみ実行される。

(3) ユーザプロセスブロック
Clickのモジュールに加えて、ユーザが開発したプログラムもブロックとして提供可能とする。

低位レイヤのパケット処理は、Clickに従ってユーザレベルあるいはカーネルレベルのClickブロックとして開発することを想定しており、一方、高位レイヤのプロトコル処理は、任意のプログラミング言語を用いて開発したプログラムを、ユーザ空間でブロックとして実行することを可能にする。

ブロック間の入出力インタフェースはポートとして定義する。ポートは名前とその役割などの属性値の集合として定義し、ブロック同士は名前と全ての属性値が整合するポートのみが接続可能である（ポート間の接続をコネクションと呼ぶ）。ポートは、同一の仮想ノード内のブロック同士だけでなく、異なる仮想ノー

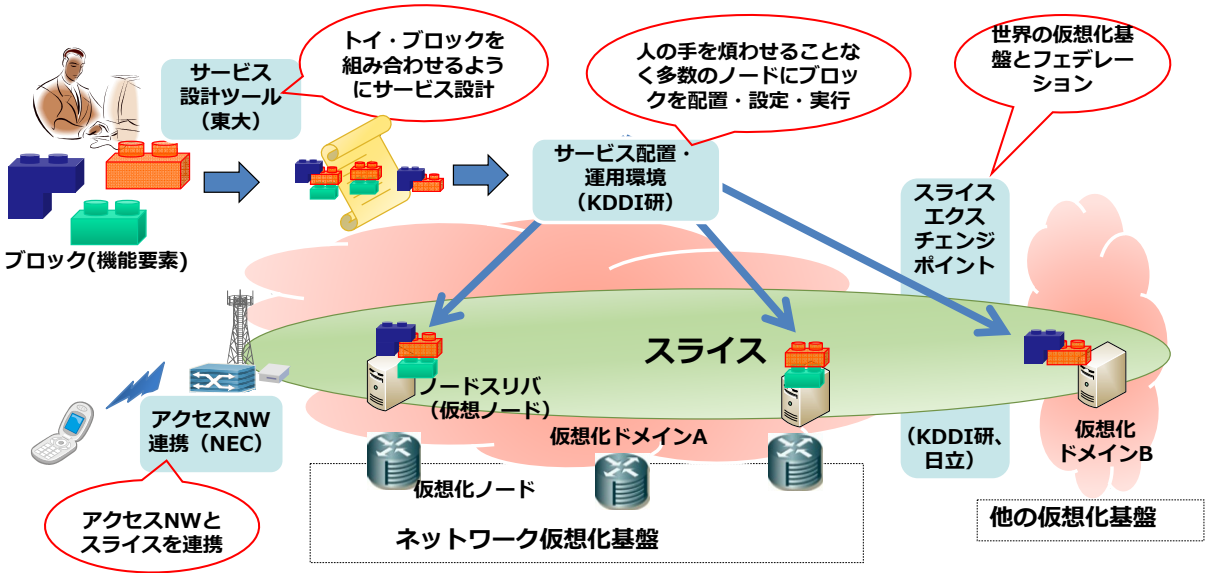

図1　プラットフォームの概要

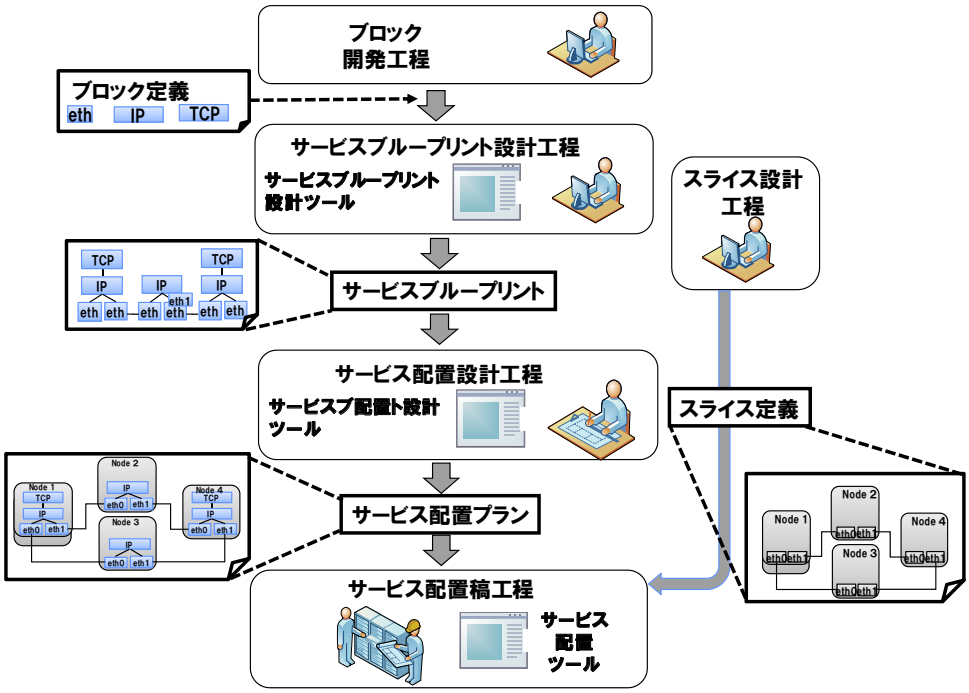

図 2　サービス開発・配置工程

ドで動作するブロック間の接続にも使用される。例えば、HTTP プロトコルのクライアントとサーバをブロックとして実装する場合、異なる仮想ノードで実行するクライアントとサーバを実行するブロックのポートは、それぞれ HTTP（クライアント、・・・）、HTTP（サーバ、・・・）のように定義されることになる。ポートは XML で記述されるが、詳細は割愛する。

なお、仮想ノードは OS（Operating System）として Linux を採用しているため、現状、上記のブロックは Linux 上で実行可能なプログラムを前提としている。

3.3　サービスブループリント設計ツール

サービスを、図 2 の工程に従って開発する前にまずユーザは、個々のブロックを開発するか、あるいは他者が開発済みのプログラムを探し、これらのブロックを本プラットファームが扱えるように、ブロック定義を準備する。ブロック定義は、ブロックの名前と、必要な個数のポート定義から構成される。

次に準備したブロック群を用いて、サービスブループリントを設計する。サービスブループリントは、サービスのリファレンスと呼ぶべきもので、特定の仮想ノードを想定することなく、サービスを構成する最小限のブロック群を組み合わせたものである（本プラットフォームでサービス開発は、サービスブループリントを作成することに相当する）。

図 3 に TCP/IP サービスを定義するサービスブループリント例を示す。IP ルータと送受信のホストを実現する 3 台の仮想ノードが接続されており、それぞれ

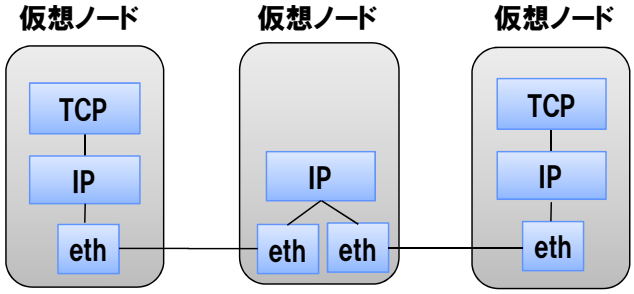

図 3　サービスブループリントの例

IP とイーサネットのブロック、TCP、IP ならびにイーサネットのブロックが実行される。サービスブループリントは XML ファイルとして記述することも可能であるが、GUI（Graphical User Interface）を用いて作成することを可能とするサービスブループリント設計ツールを用いて作成することも可能である。

3.4　サービス配置設計ツール

サービスブループリントは、ブロックがどの仮想ノードで実行され、どの仮想ノード同士を接続するかなどの個々の実装に関する詳細は規定していない。このため、ユーザはまず、仮想化基盤の規約に従って、仮想ノードと仮想回線から構成されるスライスを定義する。次に、サービス配置設計ツールを用いて、サービスブループリント上の仮想ノードを、スライス定義のどの仮想ノードで実行するかを指定することにより、サービス配置プランを作成する。サービス配置プランは、ブロック群を仮想ノード群に配置するものであり、

本プランに従ってブロック群を仮想ノードにインストールすることにより、サービスの実行が可能となる。

4　サービス設計環境

4.1　物理的な構成

サービス配置環境は、サービス配置プランに従って、ブロック群を仮想ノードに配置し、サービスを開始・停止するための機能の総称である。本環境は図 4 に示すように、仮想化ノード上で実行する仮想化ノードコントローラと、ブロック群の配置・実行・停止を司るサービス制御ノードから構成される。サービス制御ノードは、仮想ノードとは異なる Linux ベースのマシンとして実現されており、仮想ノードとは仮想化基盤とは独立な物理ネットワーク（制御プレーン）で接続されている。

サービス制御ノードは仮想化基盤に対して 1 つだけ存在し、仮想化基盤で作成される全てのスライスで実行するサービス、すなわちサービス用のブロック群の配置・実行・停止を司る。

4.2　物理的な構成

サービス制御ノードと仮想ノードにはそれぞれ、サービスコントローラと仮想化ノードコントローラと呼ぶプログラムが実行される。制御プレーン上で TCP/IP 通信により接続され、サービスの配置･実行･停止などの機能を提供する。

A）スライス作成
ユーザはサービス制御ノードを介してスライス定義を仮想化基盤に送り、スライスを作成する。この結果、作成された仮想ノードには Secure Shell サーバプロセス（sshd）が起動される（本機能には、サービスコントローラは直接関与せず、仮想化基盤が実行する）。

B）サービス配置機能
サービスコントローラは、ユーザから受け取ったサービス配置プランおよびスライス定義に従って、ブロックを配置する仮想ノードを決定し、制御プレーンでの仮想ノードの IP アドレスと TCP ポート番号を仮想化基盤から取得する。その後、sshd を介して、仮想ノード上の仮想化ノードコントローラに、ブロック群をまとめて 1 つにした実行ファイル、具体的には Linux の Debian パッケージを送信する。その後、仮想化ノードコントローラは受信した実行ファイルを、仮想ノードにインストールする。

C）サービス起動・停止
サービスに必要な実行ファイルのインストール後、サービスコントローラはユーザの指定に従って、仮想化ノードにインストールした実行ファイルの起動・停止を仮想化ノードコントローラに指示することが可能となる。仮想化ノードコントローラは、1 つの仮想ノードに対応するブロック群、すなわち実行ファイルの起動・停止を行う。例えば、仮想ノードで実行するブロック群が全てユーザレベル Click ブロックの場合は、ユーザレベル Click ドライバを起動･停止する。あるいは、ユーザプロセルブロックとユーザレベル Click ブロックが混在する場合は、そのプロセスとユーザレベル Click ドライバの双方を起動・停止する。

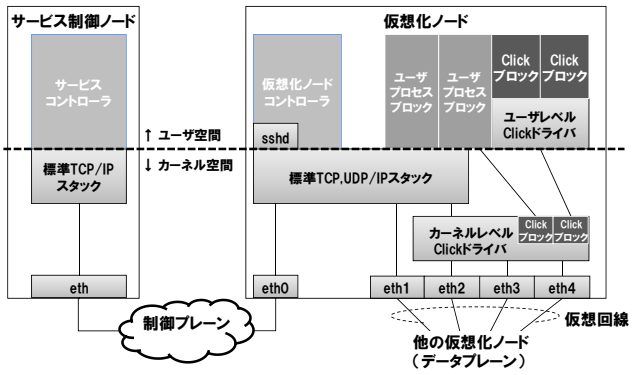

図 4　サービス配置環境の実装

5　サービス例

開発したサービス配置環境の検証を目的として、ユーザプロセスブロックを用いた高位レーヤプロトコルとして、動画配信サービスを開発した。

動画配信サービスは別途、Linux 上に実装したプログラム[12] を図 5 に示すように、ユーザプロセスブロックとしてモジュール化することにより実現した。サービスは、ロードバランスブロック（BL）、コンテンツレポジトリブロック（BC）、トランスコードブロック（BT）を TCP ブロック、IP ブロック、イーサネットブロックで構成する、動画再生端末はスライス外であるため、仮想化基盤が提供する AGW（アクセスゲートウェイ）を介して収容する。なお TCP 以下のブロックは仮想ノードのカーネル（Linux）内の標準 TCP/IP スタックを用いている。表 1 にこれらのブロックのポート定義を示す。

本サービスでは、まず配信要求を受けたコンテンツレポジトリが、動画再生端末へ収容されるアクセスネットワークの帯域に適した解像度のコンテンツを送信する。配信要求とコンテンツ送信は、標準 TCP/IP スタックと接続するポート（socket_stream）を介して行う（ここで c と s はクライアントとサーバの役割を

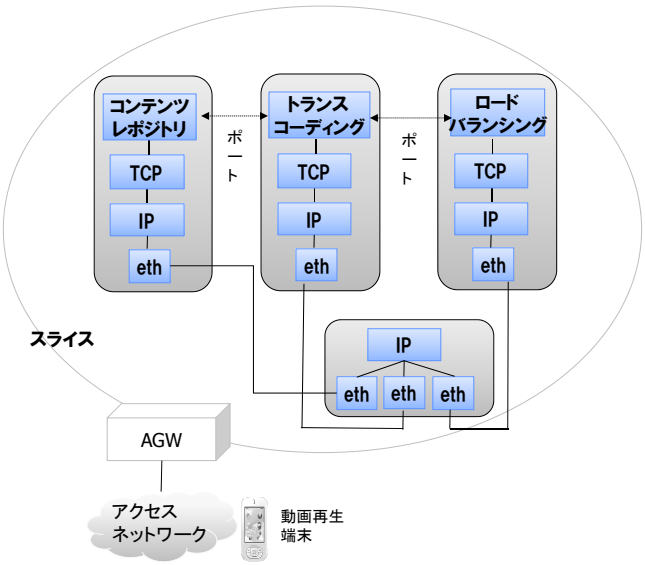

図５　動画配信サービスのブロック構成

示している)。アクセスネットワークに輻輳が発生した場合、動画再生端末は、コンテンツの解像度を下げる(トランスコーディングする)ことをロードバランスブロックへ要求し、同ブロックはトランスコーディングブロックにポート HTTP_sig_conntent_notify を介して、トランスコーディングの開始を要求する。一方、トランスコーディングブロックがコンテンツレポジトリブロックに、HTTP_sig_reuest_media を介してコンテンツの宛先を動画再生端末から同ブロックに変更する。この結果、コンテンツレポジトリブロックから送信されたコンテンツはトランスコーディングブロックでトランスコーディングされてから動画再生端末に送信されることになる。

先に開発したコンテンツレポジトリ等のプログラムは、ブロック定義を記述するだけで本プラットフォームで扱うことが可能であり、短期間に本サービスを仮想化基盤上で実行させることが可能であった。

6　仮想化基盤間統合管理機構

6.1　概要

本プラットフォームでは、異なるアーキテクチャによる仮想化基盤を利用してサービス実行のエリア拡大を可能にするため、異なる仮想化基盤間とのスライス間連携を実現した。このような連携をフェデレーションと呼ぶ。フェデレーションは、異なるアーキテクチャの仮想化基盤間で、一方の仮想化基盤から他方の仮想化基盤上の資源を利用するスライス構築、監視等の機能を実現する。そのために、スライス間の制御プレーン、データプレーンを連携させる SEP (Slice Exchange Point)[13] アーキテクチャ及び仮想化基盤間

表１　各ブロックのポート定義

ブロック	ポート定義	接続可能ブロック
B_L	socket_stream (c)	標準 TCP/IP
	HTTP_sig_content_notify (c)	B_T
B_C	socket_stream (c)	標準 TCP/IP
	HTTP_sig_request_media (s)	B_T
B_T	socket_stream (c)	標準 TCP/IP
	HTTP_sig_content_notify (s)	B_L
	HTTP_sig_request_media (c)	B_C

において、サービス構築機能、監視機能を提供する共通 API を開発した。

6.2　Slice Exchange Point アーキテクチャ

図６に SEP アーキテクチャの概要を示す。SEP アーキテクチャは集中配置された管理機能 (SEP コア) と、各仮想化基盤 (図中では「ドメイン」と記載) と SEP コアの間を取り持つ機能により構成される。SEP の目的は以下の２つに大別される。まず、ユーザが、普段から使用している仮想化基盤の制御機能を使って複数の仮想化基盤上に展開するスライス全体を制御・管理できるようにする。さらに、他仮想化基盤が持つ特長的な機能を制限無く活用できるようにする。

この２つを実現するために、SEP は以下４つの特長を持つことが望ましい。

I. 仮想化基盤からの中立性
　制御・管理機構が特定の仮想化基盤に依存しないアーキテクチャ。
II. 単一インタフェース
　各仮想化基盤と SEP が単一インタフェースで連携できる。
III. 抽象化
　複数の仮想化基盤のリソース･機能の共通の能力･特徴を抽出して SEP における制御として扱える。
IV. 機能の拡張性
　仮想化基盤の個別の機能拡張に応じて、同仮想化基盤と SEP のインターワークが拡張できる。

Gatekeeper ならびに Gateway は以下の機能を持つ。

- Gatekeeper (GK)
　制御プレーンの変換を行う。具体的には、各仮想化基盤独自のコマンド／リソース定義と SEP コアのコマンド／リソース定義を相互に変換する。
- Gateway (GW)
　データプレーンを中継する。仮想化基盤と SEP ネットワークの間でパケット形式 (プロトコル、ネットワークパラメータ等) を変換する。なお、GW は GK により制御される。

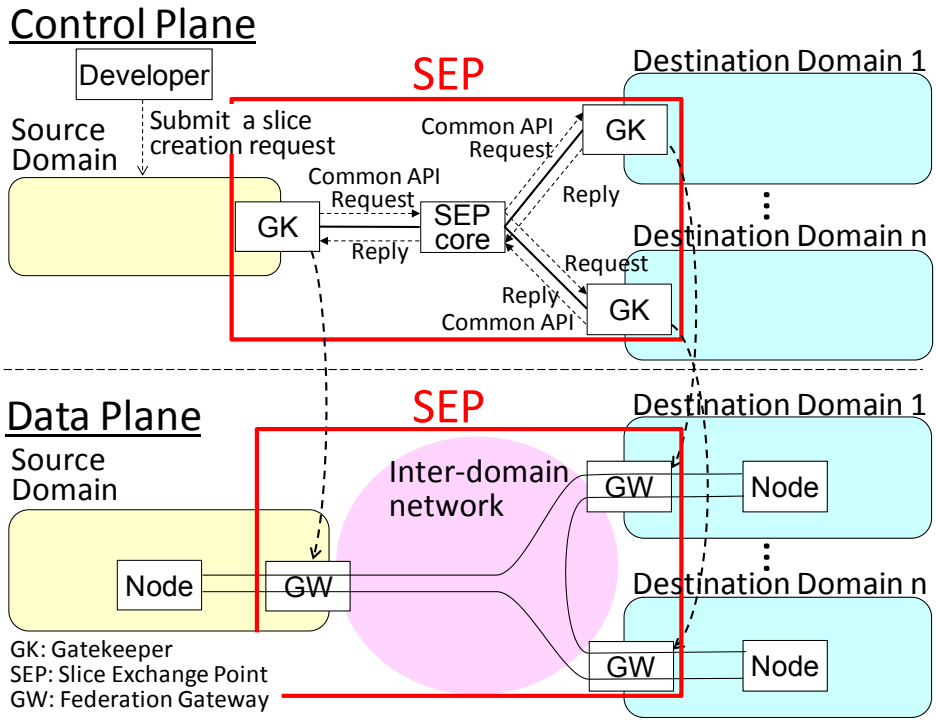

図 6　SEP アーキテクチャの概要

図中の“Source Domain”はスライス制御（作成、アップデート、削除）の契機となる仮想化基盤を示しており、“Destination Domain”は Source Domain から制御を受ける仮想化基盤を示している。

6.3　共通スライス定義

図 7、8 に、SEP では、接続する仮想化基盤に依存しない独自のスライス定義（共通スライス定義）を制御・管理する。SEP へ接続する各仮想化基盤のスライス全体、及び個々の仮想リソース（仮想ノード、仮想回線）の状態遷移を図 7、8 と仮定して、共通スライスの状態遷移を図 7 の動きとする。各仮想化基盤内部の実装として、図 7、8 と異なる状態遷移を採用することが考えられ、その場合は各仮想化基盤の GK が、状態遷移の違いを吸収する。

6.4　データプレーン

異なる仮想化基盤に属する仮想ノード間を結ぶ仮想回線（仮想化基盤間仮想回線）は、仮想化基盤内部の仮想回線、ならびに仮想化基盤を結ぶ回線で構成される（図 9）。これらは GW により接続される。仮想化基盤間仮想回線は論理的に 1 本の回線である。

SEP は、仮想化基盤間仮想回線に用いるパラメータ（リンクの種類、MAC アドレス、VLAN 番号等）を自由に決めることができ、各仮想化基盤の内部ネットワークのパラメータ（実装方式）に依存しない。本プラットフォームでは、下記 2 つの方法によって仮想化基盤間仮想回線に用いるパラメータを決定する。

（ア）GK 間ネゴシエーション

GK が仮想化基盤の回線を管理する。各仮想化基盤の GK が、共通スライスの構築要求とその応答である信号処理を通じてパラメータをネゴシエーションする。例えば、使用可能な VLAN 番号などを決定する。

（イ）SEP コアによる決定

SEP コアが、仮想化基盤間回線を構成するネットワーク（SEP ネットワーク）の回線を管理し、共通スライスに必要なパラメータを設定する。さらに、スライス構成要求を受ける仮想化基盤のパラメータを要求元仮想化基盤へ通知する役割を担う。

7　実証実験

7.1　概要

サービス配置実行ならびにフェデレーションについて、日米欧 3 拠点の実証実験を行ったので、その概要を紹介する。

7.2　日米欧 3 拠点のフェデレーション実証実験

図 10 にフェデレーションシステムの構成を示す。3 つの仮想化基盤は、VNode 仮想化基盤（NICT）、ProtoGENI 仮想化基盤（米国ユタ大学）、Virtual Wall 2 仮想化基盤（ベルギー iMinds）で、独立にスライスを構築する管理機構を備えている。なお、Virtual Wall 2 は、管理機構として、ProtoGENI の

Aggregate Manager-API（AM-API）を採用しているが、SEP を使った ProtoGENI から Virtual Wall 2・VNode 方向のフェデレーションでは、ProtoGENI の AM API のスライス作成、制御コマンドから、一旦 SEP を介した後に、再度 Virtual Wall 2 側の GK が、Virtual Wall 2 向けの AM-API に再変換する動作とした。

制御プレーンは各仮想化基盤に対応する 3 台の GK と、SEP コアにより実現される。SEP コアと VNode・ProtoGENI・Virtual Wall2 の GK は、各々ユタ大学に設置されたサーバ上に VM により実装されている。

SEP コアは 1 つの仮想化基盤から要求される共通スライス設定（構成）コマンドを 2 つの仮想化基盤に送付するとともに、この 2 つの仮想化基盤から返送された応答をマージし、1 つの共通応答として要求元仮想化基盤に返送する。

図 10 の接続環境において、図 11 に示すスライスを構成し、実装方法の異なる仮想化基盤の間のフェデレーションを実現できることを確認した。この SEP アーキテクチャによって、コマンドとスライス定義の変換機能による仮想化基盤からの中立性と単一開発者インタフェースを実現し、開発者がフェデレーションにより作成されたスライス全体を、開発者が普段使い慣れている仮想化基盤の制御機能を介して管理（作成、制御）を可能にした。これにより、1 つの仮想化基盤のリソース（仮想ノード・仮想回線）の制限を超えるスライスの構成が可能であることを実証した。

8　まとめ

本稿では、ネットワークに関する知識が十分でないユーザでも、従来開発されたプログラムモジュールを、トイブロックのように組み合わせてプログラミングできるような、柔軟なネットワークサービスシステムの構築を可能にするプラットフォームを紹介した。

この本プラットフォームでは、仮想化基盤が提供する仮想ノードで実行するプログラムの開発及び実証実験をサポートすることを目的として、ネットワークサービスシステムのプロトコル処理や各種機能をモジュール化し、モジュールの再利用によりプログラム開発の生産性を高めるため、仮想ノードで実行可能な最小の単位（ブロック）を、トイブロックを組み合わせるようなプログラミング開発を提供する。更に複数の仮想化基盤のリソースを組み合わせて広大なスライスを構築するフェデレーションを実現するため、SEP アーキテクチャを考案し、その実現性を実証実験により示した。

なお、**7** で紹介した実証実験（日米欧間フェデレーション）については、2015 年 3 月 16・17 日のネットワーク仮想化（NV）研究会（東京、小金井）、ならびに 2015 年 3 月 23～26 日に米国 Washington D.C. で開催された GEC22 においてライブデモによる実証実験を実施した。また、本研究開発は、NICT 委託研究「新世代ネットワークを支えるネットワーク

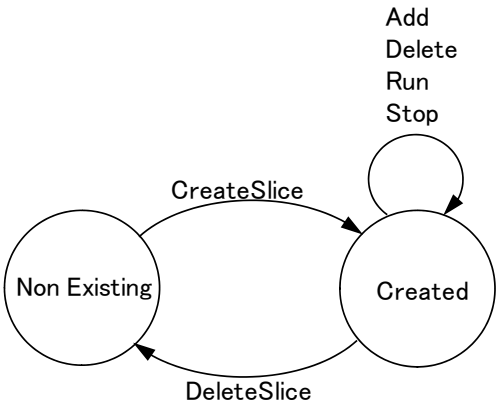

図 7　スライスの状態遷移

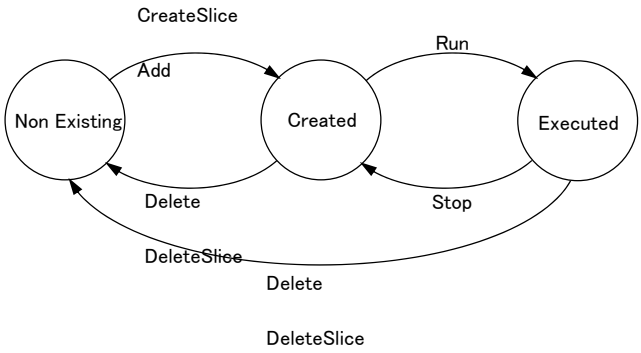

図 8　仮想リソース（仮想ノード／リンク）の状態遷移

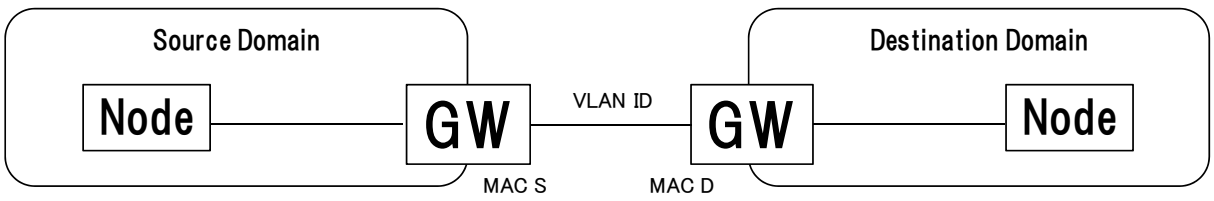

図 9　仮想リソース（仮想ノード／リンク）の状態遷移

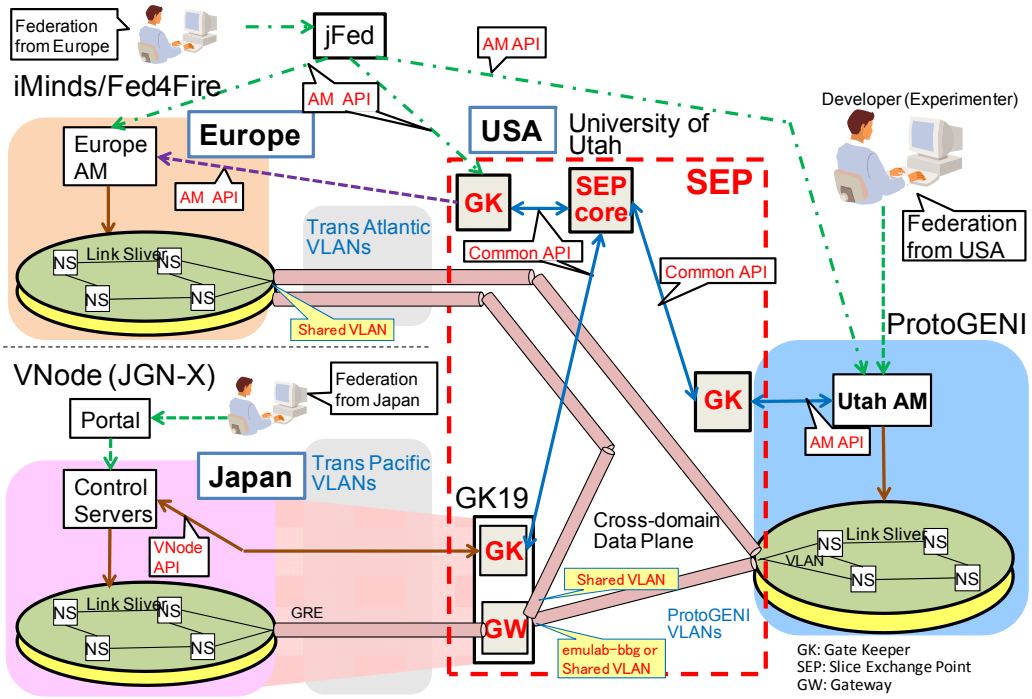

図 10　日米欧 3 拠点間フェデレーション実験システム全体構成図

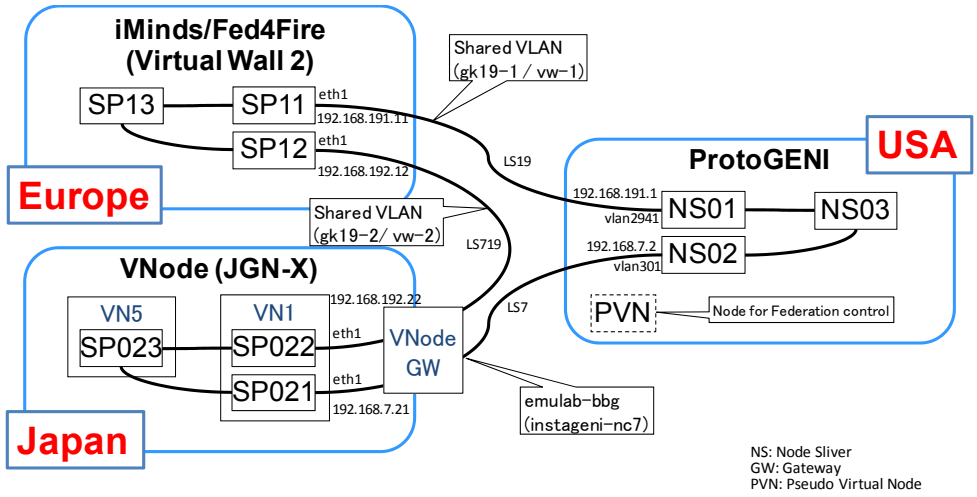

図 11　日米欧 3 拠点間に構成したスライス

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」において、東京大学情報学環・日本電気株式会社・株式会社日立製作所・株式会社 KDDI 研究所が共同で実施した。

【参考文献】

1　新世代ネットワーク推進フォーラム，http://forum.nwgn.jp/
2　FIND Initiative, http://www.nets-find.net/
3　V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard,. "Networking Named Content," Proceeding of ACM CoNEXT'09, pp.1–12, Dec. 2009.
4　Hiroaki Harai, Kenji Fujikawa, Ved P. Kafle, Takaya Miyazawa, Masayuki Murata, Masaaki Ohnishi, Masataka Ohta, and Takeshi Umezawa, "Design Guidelines for New Generation Network Architecture," on Communications, Vol.E93-B, No.3, pp.462–465, March 2010.
5　http://www.jgn.nict.go.jp/english/index.html
6　GENI (Global Environment for Network Innovations) , http://www.geni.net/
7　A. Nakao, "Network Virtualization as Foundation for Enabling New Network Architectures and Applications," IEICE Transactions on Communications, Vol.E93B, Issue 3, pp.454–457, March 2010.
8　N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks" ACM SIGCOMM Computer Communication Review, Vol.38, Issue 2, pp.69–74, April 2008.
9　山田，" ネットワーク仮想化基盤における仮想ネットワーク管理制御技術，" 電子情報通信学会ネットワーク仮想化時限研究会，第 4 回研究会資料，2012 年 7 月 .
10　http://www.ieice.org/~nv/nv201207-05-yamada.pdf
11　N. Hutchinson and L. Peterson, "The x-kernel: An architecture for implementing network protocols," IEEE Transactions on Software Engineering, Vol.17, Issue 1, pp.64–76, January 1991.

12 小森田 , 伊藤 , 横田英俊 , Makaya, Falchunk, " ユーザと連携したオープンサービスネットワークにおけるサービス構成手法の提案 ," 電子情報通信学会モバイルマルチメディア研究会報 MoMuC2011-9, pp.87-92, 2011 年 9 月.
13 岡本，松本，黒木，宮本，大垣，林 , " フェデレーションならびに新世代サービスに対応する新たなネットワーク資源管理技術 ," 電子情報通信学会 IN 研技術報告 , 2012 年 10 月 (to appear) .

北辻佳憲　（きたつじ　よしのり）
株式会社 KDDI 研究所モバイルネットワークグループリーダ
博士 (情報工学)
ネットワークアーキテクチャ、モバイルネットワーク