SDN ガイドライン

第 0.1 版

NTT コミュニケーションズ株式会社

2015 年 3 月

改訂履歴

日付	版数	備考
2015 年 3 月 31 日	第 0.1 版	・ 初版リリース。

目次

第1編 概説

第1章 はじめに

第1節 背景

1. ネットワークの動向と課題

昨今、クラウドサービス利用の拡大、スマートフォンの普及、センサ情報の活用の進展などに伴うネットワーク利活用環境の変化や、これらを活用した情報通信サービスの多様化が進んでいる。ネットワーク上のトラヒック特性が、よりダイナミックに変化するようになったことに伴い、ネットワークへの要求条件も変化している。この変化に対し、従来のネットワーク構築、制御技術ではこれに迅速に対応することが困難な状況が生じつつあり、より柔軟なネットワーク設計、及び制御を実現する必要性が高まっている。

このため、ネットワーク上の多種多量なデータ（ビッグデータ）の流通を柔軟に制御できるようにすると共に、これらのデータを活用した新たなサービスを支える多種多様なネットワークを迅速に設計・構築・運用できるようにするため、広域ネットワークへの「ネットワーク仮想化技術」の導入が急務である。

ネットワーク仮想化技術の実現には、ネットワークアーキテクチャである「SDN(Software-Defined Networking)」を基盤とした実用化が進められている。

しかしながら、現段階では SDN を通信事業者のネットワークに適用する明確な指針が存在しない状況である。

2. SDN に対する期待

昨今のデータ通信トラヒックの増加やアプリケーションサービスの高度化、高機能化に伴い、通信事業者に対しては、高速大容量通信をより高品質で低コストに提供すると共に、多種多様な利用目的、利用形態に対し、より柔軟に素早くネットワークサービスを提供するという、2 つの相反する要求が高まっている。

前者に対しては光伝送技術の発展に期待するところが大きいが、この技術分野においては光デバイスレベルのブレークスルーが必要であり、ソフトウェアというよりはハードウェアの進化が必要と思われる。一方、後者に対しては、近年提案され検討が活性化している SDN/OpenFlow の適用が期待されている（図 1-1）。

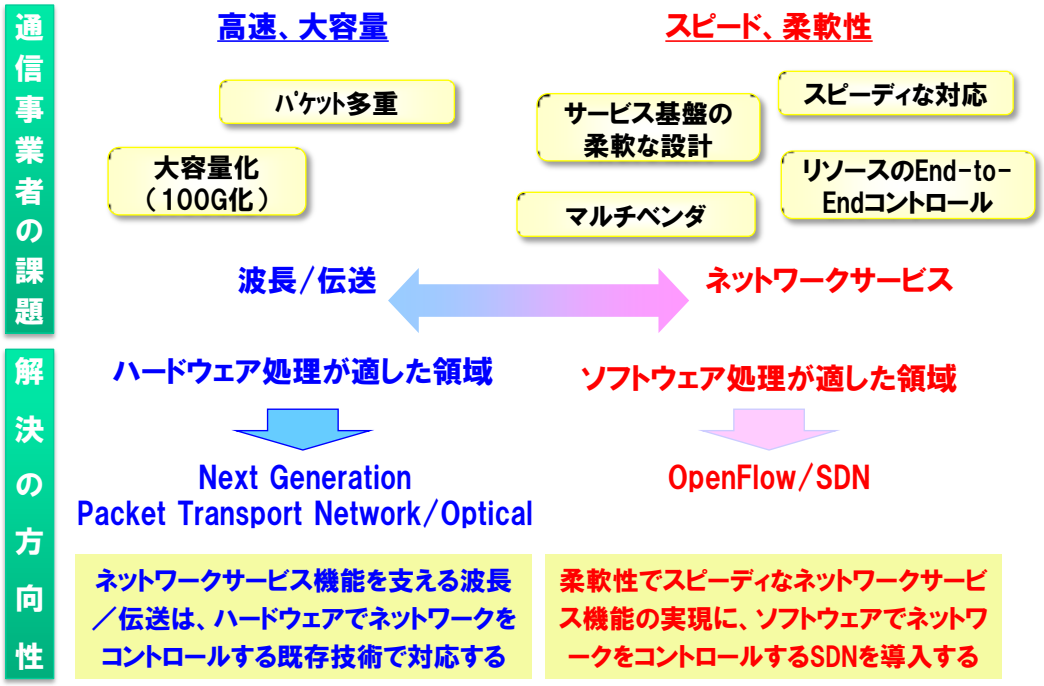

図 1-1　SDN に対する期待

第2節 本ガイドラインの位置付け

1. 概要・目的

本書は、ネットワーク仮想化技術の社会への普及と促進を目指し、SDN を通信事業者のネットワークに適用するための指針や基本的な考え方について、ガイドラインとして体系的にまとめたものである。

- SDN を通信事業者のネットワークに適用するための指針や基本的な考え方を示す。
- 指針や基本的な考え方を解説することを目的とし、技術詳細や具体的な実施手順などを示すものではない。
- 従来のネットワーク技術については他書を参照するものとして最小限の記述に留め、本書では通信事業者のネットワークに SDN を適用するための特有の事項に着目して解説する。

- 想定読者は SDN を用いた通信事業者ネットワークの設計者、構築者、運用者とし、従来ネットワークに関する技術や SDN に関する基礎的な知識を有しているものとする。

2. 目次構成

本ガイドラインの目次構成と概要を表 1-1 に示す。

表 1-1　目次構成と概要

目次構成		概要
第１編	概説	本ガイドラインの全体的な説明や前提条件などを示す。
第１章	はじめに	本ガイドラインの位置付けを明確にする。
第２章	SDN の基礎	SDN を構成する技術について基礎的な考え方を解説する。
第３章	ネットワークモデル	本ガイドラインで対象とするネットワーク構成を示す。
第４章	ネットワーク・ユースケース	SDN を用いたネットワークのユースケースを示す。
第２編	設計・構築フェーズ	業務フェーズにおける設計・構築について示す。
第１章	設計・構築フェーズの基本的な考え方	SDN 適用における設計・構築の基本的な考え方を示す。
第２章	SDN NW の設計・構築	ネットワーク階層における「SDN NW」の設計・構築の考え方について示す。
第３章	仮想 NW の設計・構築	ネットワーク階層における「仮想 NW」の設計・構築の考え方について示す。
第３編	運用・監視フェーズ	今後の課題とする。
第１章	運用・監視フェーズの基本的な考え方	
第２章	SDN NW の運用・監視	
第３章	仮想 NW の運用・監視	
第４編	付属資料	関連する補足情報などを示す。
第１章	用語の定義	本ガイドラインで定義した用語について整理する。
第２章	参照資料の一覧	参照した資料を一覧で示す。

第2章 SDN の基礎

第1節 SDN の概要

1. SDN が出てきた背景

近年、クラウド時代と言われるようになってから、インターネットのトラヒックの流れに変化が生じている。以前は PC(Personal Computer)間においてファイル交換ソフトウェアなどによる Peer-to-Peer トラヒックの増大が課題であったが、最近では情報資源がネットワーク内のクラウドコンピューティング内に蓄積され、スマートフォン、タブレット端末などにより、ネットワークを介して情報資源にアクセスする Cloud-to-End の通信に変わってきている。典型的な例としては「YouTube」などによる映像コンテンツ配信がある。また最近では、企業内の業務システムを仮想サーバ環境で動作させるだけでなく、サービスとして提供されるクラウド上の仮想サーバで動作させる事例も一般的になりつつある。このようにクラウド上の多量な情報資源（ビッグデータ）がネットワークを介して端末に流れることになり、通信事業者としては、トラヒックの大容量化への対応と、変化の速いクラウド時代のトラヒックへの柔軟な対応の両方を同時に実現することが課題となっている。

ネットワーク装置については、従来より装置ベンダが提供するハードウェア／ソフトウェア一体型の市販製品が主流であり、通信事業者が独自にカスタマイズすることは困難であった。そのため通信事業者はベンダが決めた製品仕様に従いサービス展開を行う必要があった。つまり特定ベンダの技術に依存する状態（ベンダロックイン）であり、以下のような課題があった。

- ネットワーク装置で実現できることは、その装置が有している機能のみであるため、新機能が必要であればベンダに要望し機能追加してもらう必要がある。
- 機能追加には時間がかかることが多く、タイムリーなサービス提供ができない。
- 世界中で市販製品であるネットワーク装置を購入できるため、サービスとしての差別化が難しい。
- ネットワーク装置ベンダが異なれば、操作方法も異なり運用が複雑になる。

このような状況の中で、ネットワーク装置における制御機能（Control Plane：C プレーン）と、パケット転送機能（Data Plane：D プレーン）を分離し、管理

や制御機能をソフトウェアで実現することで、新しい機能などをカスタマイズし易いようにするという考えが出てきた。C プレーンと D プレーンの両プレーン間を OpenFlow のような標準的なインタフェースで接続するという考えが広まってきており、現在ではデータセンタ、企業ネットワークなどを中心に SDN 導入の検討が進んでいる。

2. SDN とは

SDN(Software-Defined Networking)とは、「ネットワークの構成、機能、性能などの制御をソフトウェアで動的に設定、変更するためのコンセプト（概念)」である。特徴は、ネットワークのトポロジ管理やルーティング、経路制御などを行うコントロールプレーン（C プレーン）機能と、パケットフォワーディング（パケット転送）処理を行うデータプレーン（D プレーン）機能を分離することにより、ネットワークサービスの迅速な提供と高度化をソフトウェアによるアプリケーションによって対応することである（図 1-2）。SDN は当初 OpenFlow を中核とするネットワーク技術を表現するために使われ始めたが、現在では拡大解釈され、多様な意味合いで用いられている。

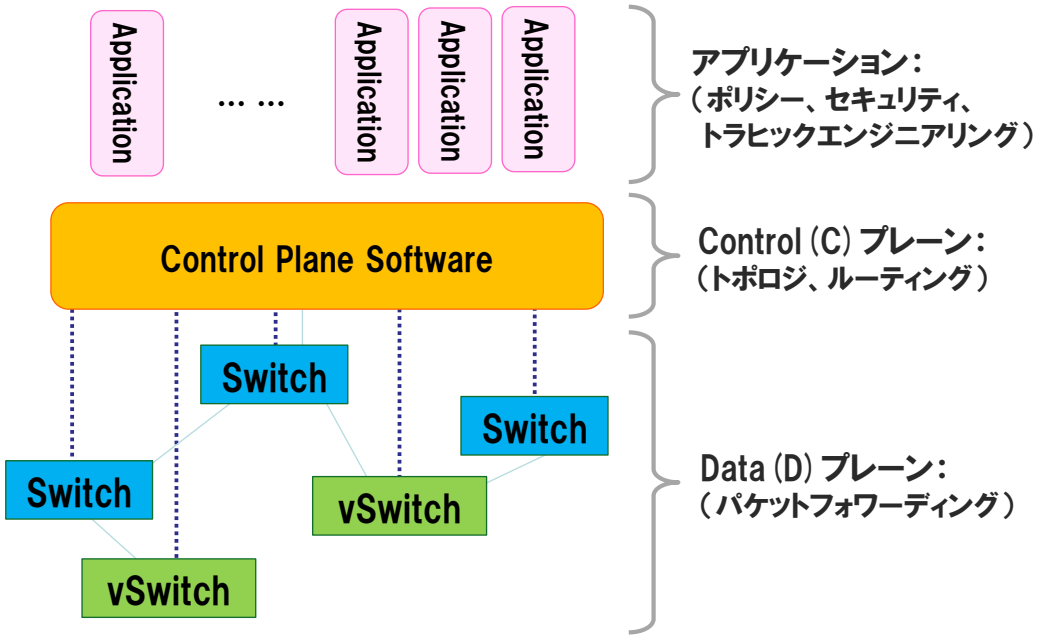

図 1-2　SDN の概念

3. OpenFlow

3.1. OpenFlow とは

OpenFlow は、SDN を実現する技術の 1 つであり、米国スタンフォード大学の研究からスタートし、現在は業界団体である ONF(Open Networking Foundation)が標準化を進めているネットワーク制御技術である。

OpenFlow では、OpenFlow コントローラ（以降は「OFC」と略す場合あり）と OpenFlow スイッチ（以降は「OFS」と略す場合あり）によりネットワークが構成され、OpenFlow コントローラは複数の OpenFlow スイッチを一元管理し、経路計算や受信したパケットの振る舞いの指示などを行う。この OpenFlow コントローラと OpenFlow スイッチの間で情報をやり取りするためのプロトコルが OpenFlow となる。

OpenFlow スイッチは、受信したパケットをどう処理するかを定義した「フローテーブル」に基づき処理を行う。フローテーブルには、「Match Fields に適合したパケットを Instructions に従い処理する」という内容が記述されている（フローエントリ）。このフローテーブルは、OpenFlow コントローラからの指示で情報の追加、削除、変更が可能である（図 1-3）。

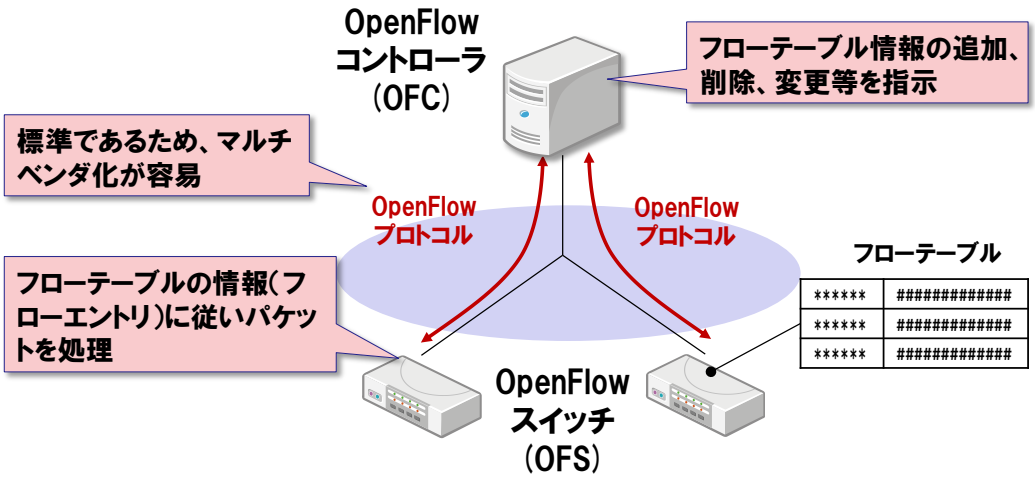

図 1-3　OpenFlow コントローラと OpenFlow スイッチ

フローエントリには、「Match Fields」など 6 種類の情報が含まれており、これらの情報を組み合わせて、受信したパケットの処理方法を決定する。OpenFlow スイッチはパケットを受信した際に、ヘッダ情報を確認してどのフローエントリの「Match Fields」に一致するかを判断する（以降では Match Fields で規定された条件を「Match 条件」と略す)。一致した場合に、そのフローエントリの「Instructions」が指定する処理を実行する。フローテーブルを構成するフローエントリの内容を表 1-2 に示す。

表 1-2　フローテーブルを構成するフローエントリの内容

要素	意味
条件 (Match Fields)	パケット受信時にフローテーブルを検索するためのキー。
優先度 (Priority)	Match Fields に合致するフローエントリが複数存在した場合の優先順位。値が大きい方を優先する。
統計情報 (Counters)	同じフローエントリに一致したパケット数、バイト数、フローエントリが書き込まれてからの時間等。
処理 (Instructions)	Match Fields に合致した場合の処理方法を指定する。
有効時間 (Timeouts)	フローエントリが消滅するまでの時間。
クッキー (Cookie)	コントローラがフローエントリを管理するための値。

3.2. パケット処理方法

パケットの処理方法に着目すると、OpenFlow は「プロアクティブ型」と「リアクティブ型」の方式に分類される。

プロアクティブ型は、OpenFlow コントローラは処理すべきパケットのフローエントリを OpenFlow スイッチのフローテーブルに事前に書き込んでおく方式である。OpenFlow スイッチはフローテーブルに該当するパケットを受信した場合にフローエントリに従いパケットの処理を行い、フローテーブルに該当しないパケットを受信した場合にはパケットをそのまま廃棄する（図 1-4)。

本方式では、事前に OpenFlow スイッチにフローエントリを書き込むため、パケット受信の都度 OpenFlow コントローラに処理を問い合わせる必要がなく、問い合わせによるオーバヘッドの削減というメリットがある。一方で必要なフローエントリを全て登録しておく方式のため、例えば大量のフローエントリを設定する必要がある場合には、フローテーブル数が足りなくなるという課題もある。

リアクティブ型は、フローエントリを事前に登録せず、パケット受信時に OpenFlow コントローラにパケットを転送（Packet-In : パケットイン）して処理

方法を問い合わせる方式である。問い合わせを受けた OpenFlow コントローラは、処理すべきパケットであれば該当するフローエントリを OpenFlow スイッチに書き込む。OpenFlow スイッチにフローエントリが書き込まれた後は、同様のパケットを受信した際にはフローエントリに従い処理を行う。また処理すべきパケットでない場合は、OpenFlow コントローラへパケットインされた際にそのまま廃棄される（図 1-5、図 1-6）。

本方式では、OpenFlow コントローラに対する問い合わせが頻繁に発生することになるが、フローエントリを一定時間で消去（エージング）することで、フローテーブルを有効活用できるというメリットがある。

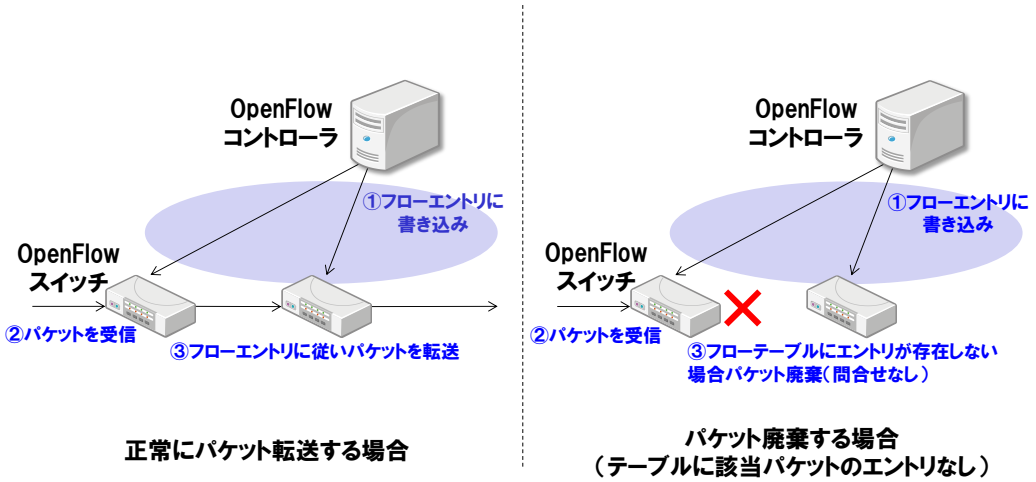

図 1-4　プロアクティブ型動作

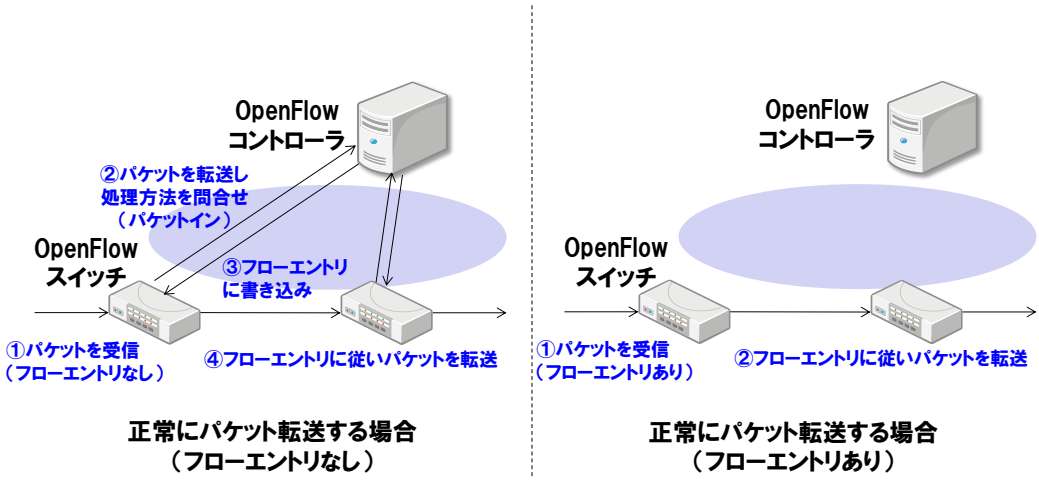

図 1-5　リアクティブ型動作（正常転送）

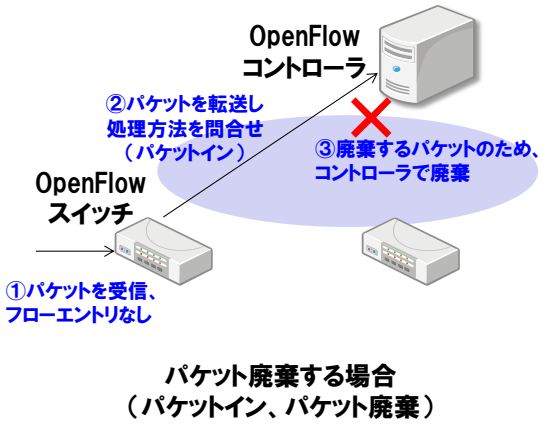

図 1-6　リアクティブ型動作（パケット廃棄）

3.3. パケットの流れる経路

パケットの流れる経路に着目してみると、OpenFlow の動作により、4 つのパタン（A～D）に整理できる（図 1-7）。従来のパケット転送と比べてパケットの流れが複雑になるため、通信事業者にとっては運用の工夫が必要になるが、従来には無いような様々な新しいサービスの可能性が考えられる。

A) Output（指定されたポートからパケットを送出する）によりフローテーブルに指定された経路を流れる

B) Packet-In（パケットイン：コントローラにパケットを送信し処理を問合せる）や Packet-Out（パケットアウト：指定ポートからパケット送出を指示する）によりコントローラを介して流れる

C) Drop（パケットを廃棄する）により廃棄される

D) Group テーブル（グループ毎に Actions を指定した情報）により分岐する

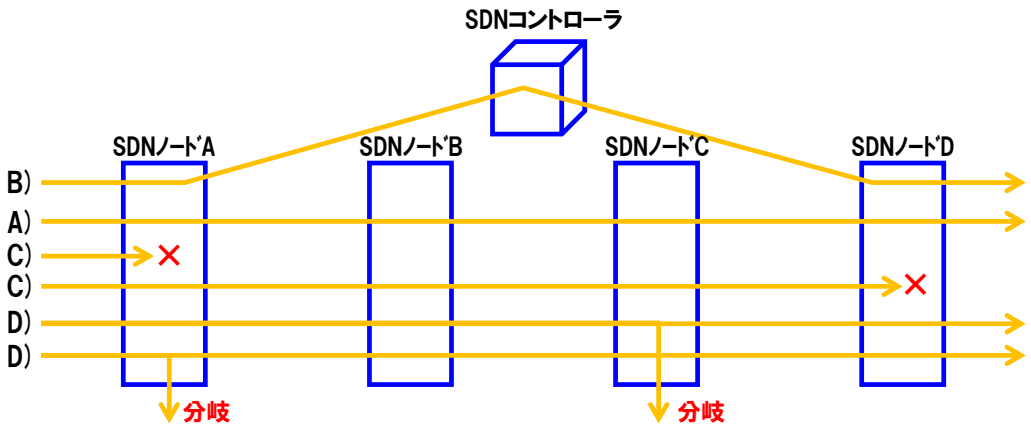

図 1-7　パケットが流れる経路

4. SDN の構成要素

SDN を構成する要素は、大きく 3 つに分類できる（図 1-8）。

1 つめは、パケット転送を担う「SDN ノード」である。この SDN ノードは、専用ハードウェアで構成される物理スイッチやソフトウェアで動作する仮想スイッチがある。従来のパケット転送装置との違いは、SDN コントローラからの制御によって動作する点である。2 つめは、SDN ノードの制御や管理を行う「SDN コントローラ」である。3 つめは、SDN コントローラの上位で、ポリシー制御やトラヒック制御など、各種機能を実現する「ネットワークアプリケーション」である。

また、SDN の構成要素の上位には、SDN ネットワークやクラウドネットワーク、既存ネットワークなど、複数のコントローラの管理や制御を行うオーケストレータがあり、各コントローラと連携を行う。

各構成要素の詳細については次節で述べる。

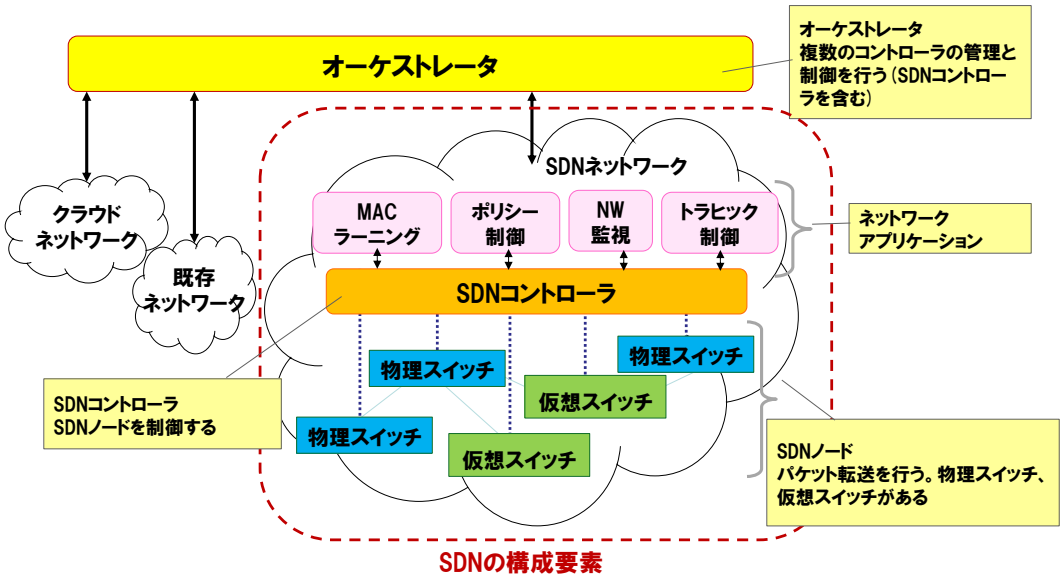

図 1-8　SDN の構成要素

従来ネットワークと SDN を用いたネットワークの運用面での違いは、従来ネットワークでは、ネットワーク技術者がネットワーク全体を見通した上で設計し、それに応じて各装置に個別の設定を行うことで、ネットワーク全体の整合

性を保っていた。一方 SDN を用いた場合では、ネットワーク全体を制御するソフトウェアを用意し、そのソフトウェアにより全ての装置を制御するという考え方である。ソフトウェアを利用してパケット転送を制御し、各装置を集めたネットワーク全体を一つの単位として、一括で制御するという考え方である。

第2節 SDN の構成要素

1. SDN ノード

SDN ノードは、実際のパケット転送を担う装置である。この SDN ノードは、専用ハードウェアで構成される物理スイッチと汎用ハードウェア等の上のソフトウェアで動作する仮想スイッチがある。仮想スイッチはサーバ仮想化ソフトウェアの中に含まれていることが多い。この SDN ノードの制御には、OpenFlow やベンダ独自の API(Application Programming Interface)が利用される。

物理スイッチは、NEC などの多数のベンダにより提供されている。

仮想スイッチは OpenFlow 等に対応した仮想スイッチソフトウェアであり、装置構成やカスタマイズに柔軟性がある。この仮想スイッチには Open vSwitch や O3 プロジェクト※の Lagopus(ラゴパス)などがある。Lagopus は、広域ネットワークでの利用を目指し、OpenFlow の最新の安定版仕様である OpenFlow バージョン 1.3.4 に幅広く準拠した高性能な SDN ソフトウェアスイッチであり、オープンソースソフトウェアとして 2014 年 7 月に公開されている。特徴はマルチコア CPU のような近年のサーバアーキテクチャの特徴を効率的に利用するパケット処理の実装と、I/O 性能を高速化する Intel DPDK の技術を利用することで、広域ネットワークで要求される大規模・広帯域な通信処理を可能とする、100 万フロー制御ルールのサポートや 10Gbps の通信性能を実現していることである。

※：世界初の広域 SDN(Software-Defined Networking)実現を目指す研究開発プロジェクト。NEC、日本電信電話株式会社、NTT コミュニケーションズ、富士通、日立製作所が参画。

また、最近ではホワイトボックススイッチを利用するケースも増えてきている。従来ネットワーク装置ベンダはハードウェア（筐体や基板）とソフトウェア（OS 等）を一体化した製品を販売してきた。ホワイトボックススイッチとは、スイッチのハードウェア部分のみの製品を指し、スイッチに搭載する OS やアプリケーションといったソフトウェア部分は、スイッチを使うユーザが自ら選んで導入する（Open Network Linux 等）、もしくは自社で開発したものを利用することができる（図 1-9）。

このホワイトボックススイッチの利用例としてはクラウドサービス事業者がある。大規模クラウドサービス事業者は、大量のネットワーク装置を必要としており、装置導入コストは膨大なため、これを抑制する必要がある。そこでODM(Original Design Manufacturing)※ベンダに自社で利用するホワイトボックススイッチの製造を依頼し、そこに OSS(Open Source Software)等のソフトウェアを搭載して利用することで、既存スイッチに比べて導入コストを下げつつ、サービスに必要なネットワーク機能を追加して運用を行っている。ホワイトボックススイッチの特徴を以下に挙げる。

- コストが安い
- ベンダロックインの回避が可能
- 迅速なサービス提供
- カスタマイズの柔軟性

※：ODM：汎用部品を組み合わせて設計・製造、販売を行うこと。

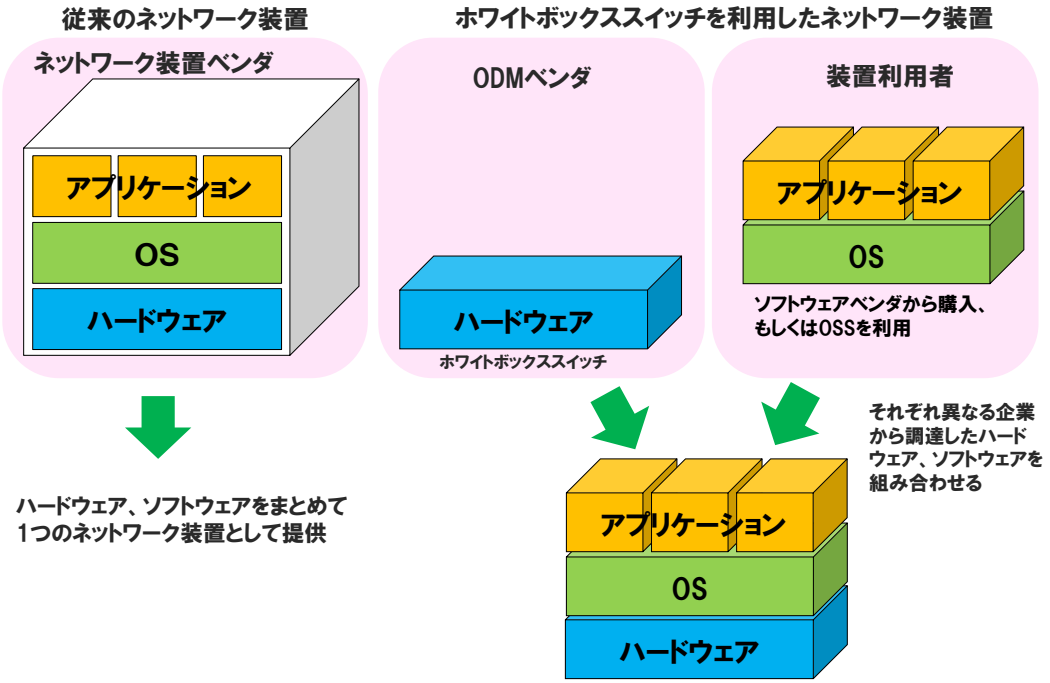

図 1-9　ホワイトボックススイッチを利用したネットワーク装置

2. SDN コントローラ

SDN コントローラは、パケット転送を担う SDN ノードであるスイッチ群が、どのようにパケットを処理するかを管理･制御するソフトウェアである。SDN コントローラは、ベンダや企業が開発を行う商用版と OSS コミュニティ等で開発されるものがある。

SDN コントローラからスイッチ群を制御するための API は、サウスバンド API とも呼ばれており、OpenFlow やベンダ独自の API が使われている。

SDN コントローラには、NTT 研究所が開発したオープンソースコントローラである「Ryu」や、米 Linux ファウンデーションのオープンソースコントローラである「Open Daylight」等がある。

Ryu(Ryu SDN Framework)とは、NTT 研究所開発のオープンソースコントローラであり、OpenStack の開発と同じプログラム言語である Python で書かれている。また、OpenFlow バージョン 1.0、1.2、1.3、1.4 に対応しており、OpenStack との連携が可能である（図 1-10)。

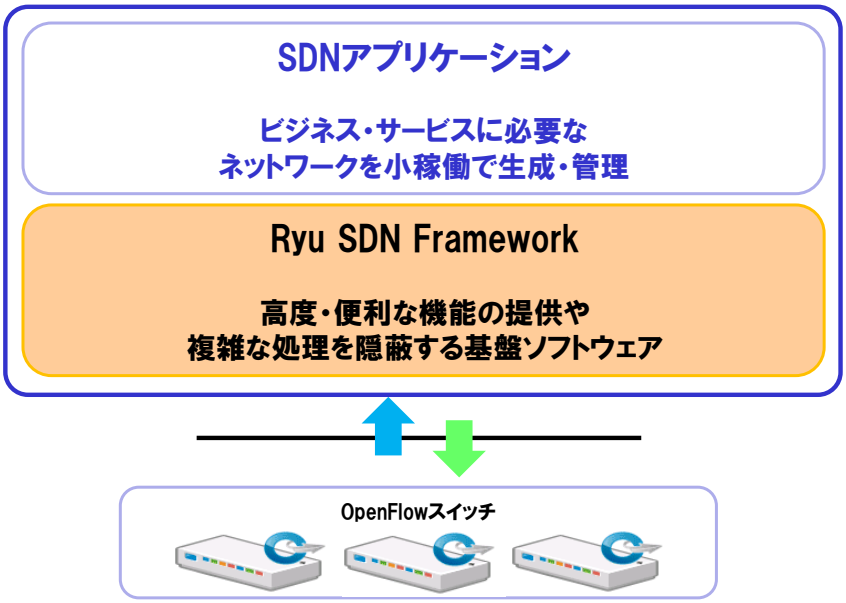

図 1-10　SDN コントローラ「Ryu SDN Framework」

3. ネットワークアプリケーション

ネットワークアプリケーションは、SDN コントローラの上位で動作し、経路計算やネットワーク監視、トラヒック制御などの機能を実現するアプリケーションである。SDN では各種機能を実現するための専用のハードウェアを導入することなく、ソフトウェアのプログラミングによってネットワークアプリケーションとして導入することで、各種制御機能や付加機能を柔軟に実現できる。その反面、利用者が自らソフトウェアを準備する必要があり、このソフトウェアが無い場合には、MAC アドレスラーニングやパケット経路計算のような基本機能についても必要に応じてソフトウェアとして準備することとなる。

第3節 オーケストレータ

現在の IT システムは、ネットワークだけでなく、ストレージやサーバなど複数の装置で構成される大規模なシステムとなりつつある。そのため制御対象を分割し、ネットワークを制御する「ネットワークコントローラ」、クラウドを制御する「クラウドコントローラ」など、大規模 IT システムにおいて制御の範囲

毎に複数のコントローラが存在することが多い。また冗長化構成のため同一種類で複数のコントローラが存在することもある。そのため、これら複数のコントローラを含むシステム全体を管理し制御するソフトウェアが必要となり、これは「オーケストレータ」と呼ばれている（図 1-11）。

オーケストレータとネットワークアプリケーション、オーケストレータと SDN コントローラとのインタフェースはノースバウンド API とも呼ばれ、現在は各ベンダ等が独自の API を提供していることが多い。

オーケストレータには、仮想サーバと仮想ネットワークを管理・制御することができるオープンソースソフトウェアである OpenStack のオーケストレーション機能である Heat や CloudStack などがあり、他にもデータセンタ等を運営する事業者が開発するケースもある。

ネットワークを管理するオーケストレータの例としては、O3 プロジェクト等で NEC が提供する SDN プラットフォームである OdenOS(Object-Defined Network OS)が挙げられる（図 1-12）。この OdenOS は、既存のネットワークをトポロジー（ノード、ポート、リンク）とフロー（End-to-End の通信）に抽象化したモデルで表現するネットワークを管理する「ネットワークオーケストレータ」であり、パケットトランスポートノードや光トランスポートノードを抽象化して制御することが可能である。また SDN コントローラ機能も有している。（OdenOS ダウンロードサイト：
http://www.o3project.org/ja/download/index.html）

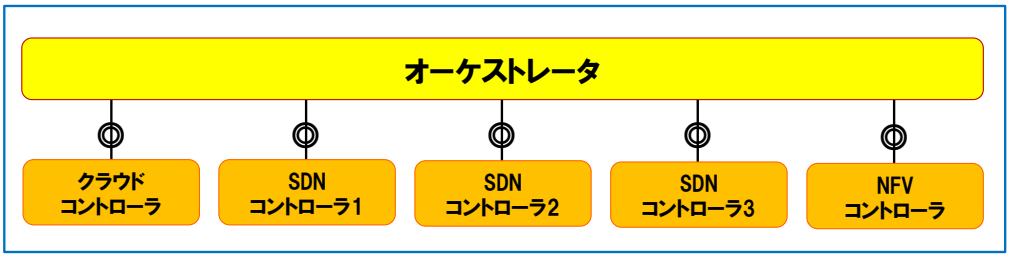

図 1-11　オーケストレータのイメージ

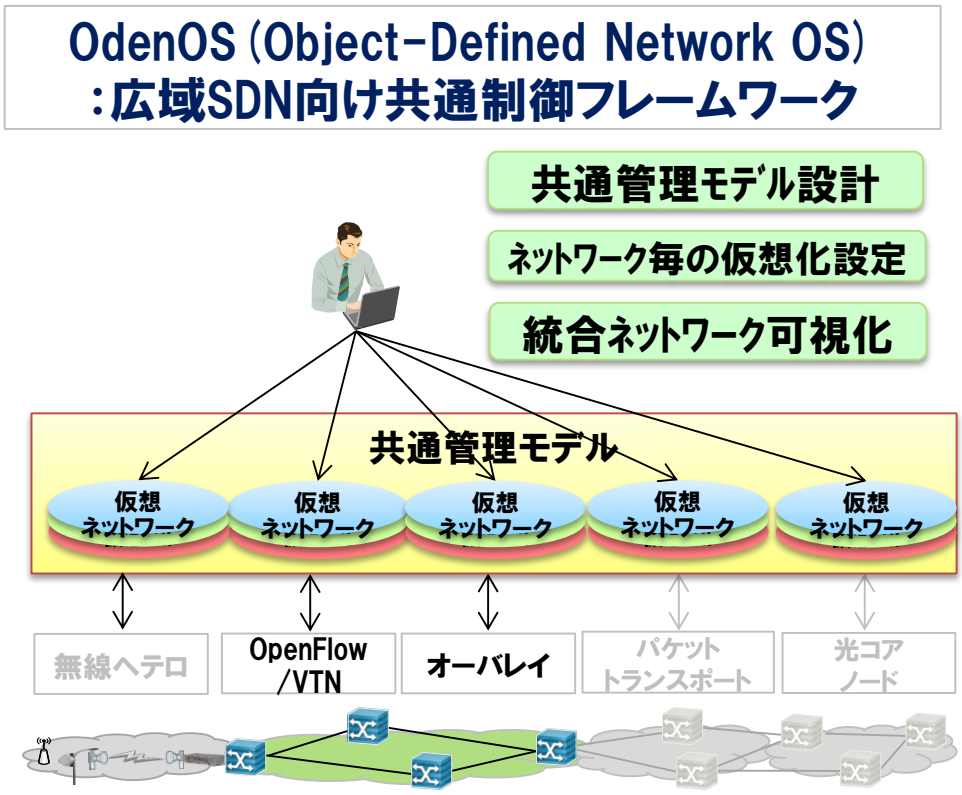

図 1-12　OdenOS イメージ

第4節 SDN の適用方式

1. OpenFlow の適用方式

OpenFlow を前提として SDN を適用する場合には、「ホップ・バイ・ホップ（Hop by Hop）方式」と「オーバレイ（Overlay）方式（トンネル方式)」と呼ばれる二つの方式がある。

2. ホップ・バイ・ホップ（Hop by Hop）方式

この方式は、ネットワーク上の全てのスイッチを OpenFlow 対応とし、コントローラが全てのネットワーク機器を OpenFlow で管理する方式である。
OpenFlow コントローラが経路を計算し、OpenFlow プロトコルを利用して各スイッチに経路情報を配布し制御を行う（図 1-13）。

全てのネットワーク機器が OpenFlow に対応している必要があるため、費用の観点から導入のハードルは高くなるが、ネットワーク機器毎に OpenFlow によってきめ細かく経路を制御できる利点がある。

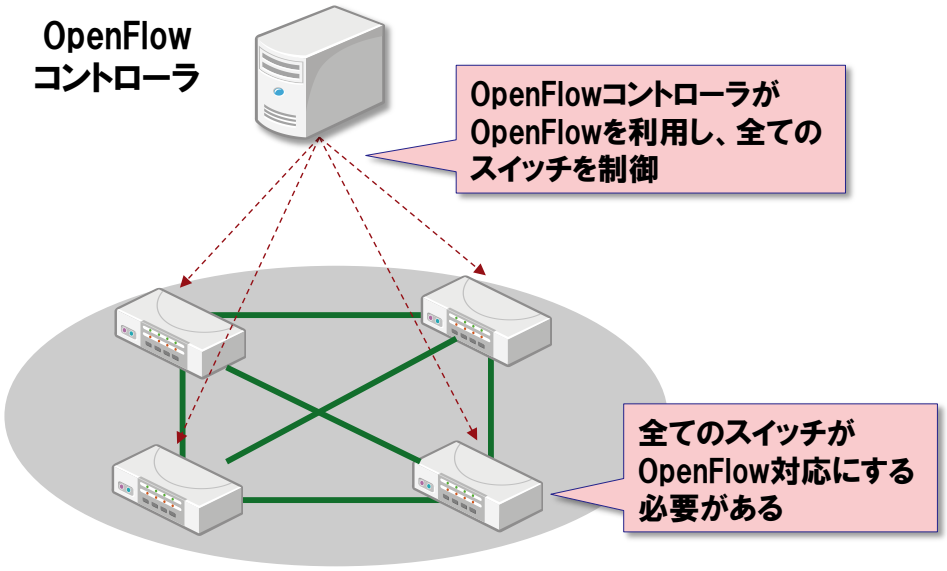

図 1-13　ホップ・バイ・ホップ方式

3. オーバレイ（Overlay）方式（トンネル方式）

この方式は、全ての機器を OpenFlow 対応スイッチにするのではなく、ネットワークのエッジに配置されたスイッチ（エッジスイッチ、主に仮想スイッチ）を OpenFlow コントローラが OpenFlow で制御し、エッジスイッチ間の通信はトンネリングプロトコル※を利用して仮想的に直接接続する（図 1-14）。

既存のネットワーク機器を活かすことができる利点があるが、物理スイッチ側からはトンネル内部のフレームを関知できないため、End-to-End での負荷分散や帯域制御、監視は難しいという課題もある。

※：GRE(Generic Routing Encapsulation)、VXLAN(Virtual Extensible LAN)、NVGRE(Network Virtualization using GRE)等がある

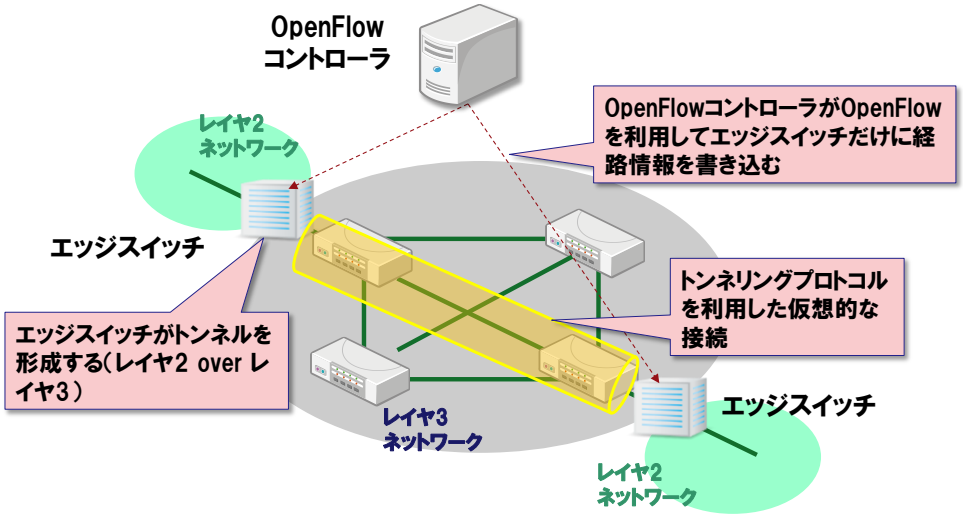

図 1-14　オーバレイ方式

4. 適用方式の比較

ホップ・バイ・ホップ方式とオーバレイ方式の比較を表 1-3 に示す。

表 1-3　ホップ・バイ・ホップ方式とオーバレイ方式の比較

方式	既存スイッチの活用	コントローラによる管理	初期コスト
ホップ・バイ・ホップ	既存スイッチ（OpenFlow 未対応）は使用不可のため、全てを OpenFlow に対応させる必要がある。	・ 全ての OpenFlow スイッチを管理する必要がある。 ・ ネットワーク上で粒度の細かい多様な制御が可能。	大規模ネットワークの場合、全てのスイッチを OpenFlow 対応にする必要があるため、初期コストがかかる場合がある。
オーバレイ	既存ネットワーク装置を利用し、エッジスイッチ（仮想マシン直近スイッチ）のみ OpenFlow に対応させればよい。	仮想スイッチ（エッジスイッチ）と仮想ネットワークの対応関係のみを管理すれば良い。	既存設備を活かし、エッジスイッチのみを OpenFlow 対応とすることで実現可能。

第3章 ネットワークモデル

第1節 通信事業者の一般的なネットワーク

1. 従来のネットワークモデル

通信事業者の一般的なネットワーク構成をモデルとして図 1-15 に示す。

図 1-15 通信事業者の一般的なネットワークモデル

ネットワークの全体構成を区間と機能などで整理し、構内／イントラネット、アクセスネットワーク、中継ネットワーク、データセンタネットワーク（以降、DC ネットワークと略す）で構成されたモデルで表す。通信事業者として直接管理を行うアクセスネットワークと中継ネットワークに加え、ユーザにネットワークサービスを提供するという観点から、構内／イントラネットと DC ネットワークを含めて考える。なお、通信事業者のアクセスネットワーク、中継ネットワークは、各々、他社の異なるネットワーク（異なるドメイン）にて構成される場合もある（通信事業者間接続）。

2. 構内／イントラネット

構内／イントラネットはユーザ宅内のネットワークである。基本的には各ユーザにて管理され、通信事業者と契約して広域ネットワークと接続することで、ネットワークサービスの提供を受ける。

3. アクセスネットワーク

アクセスネットワークは中継ネットワークと構内／イントラネットを接続するネットワークである。主な特徴として以下が挙げられる。

- アクセスポイントからユーザ宅内までが該当し、通信事業者のネットワークへユーザを収容する。
- ユーザの利用する各種ネットワークサービスを多重する。
- 有線、無線、インターネットなどの多様な通信手段との相互接続を行う。
- アクセスサーバやルータなどの多様な設備で構築される。

4. 中継ネットワーク

中継ネットワークは通信事業者間や ISP(Internet Service Provider)を接続する通信事業者の基幹ネットワークである。主な特徴として以下が挙げられる。

- アクセスネットワークの回線を収容・集線してユーザを多重する。
- 各種ネットワークサービスを多重し、またサービス別のネットワークに振分けを行う。
- 長距離・大容量のパスでトラヒックを集約して高速大容量伝送を行う。
- 地理的に広がった構成をとる。
- 経路冗長や切り替え機能を有して高信頼のネットワークを実現する。

5. DC ネットワーク

昨今の DC ネットワークは拡大を続け、サーバ／ストレージの仮想化技術を活用したクラウドデータセンタが主流になり、主にクラウドサービス事業者により管理される。主な特徴として以下が挙げられる。

- 1 つのサーバ上に多数の仮想サーバが生成できるため、ネットワーク規模が膨大である。

- 仮想サーバのインスタンスが動的にネットワークを移動するため、ネットワークの柔軟な運用が必要である。
- CMS(Cloud Management System)と連携して、各種クラウドサービスを提供する。

第2節 SDN を適用した通信事業者ネットワーク

1. SDN を用いたネットワークモデル

本ガイドラインで想定する SDN を用いた通信事業者のネットワークのモデルを図 1-16 に示す。前提としては既設の L2/L3 ネットワークの存在に捉われず、SDN 技術を適用したネットワークを新規に構築する立場で考えるものとし、SDN の適用方式としてはホップ・バイ・ホップ型と呼ばれる構成に分類される。

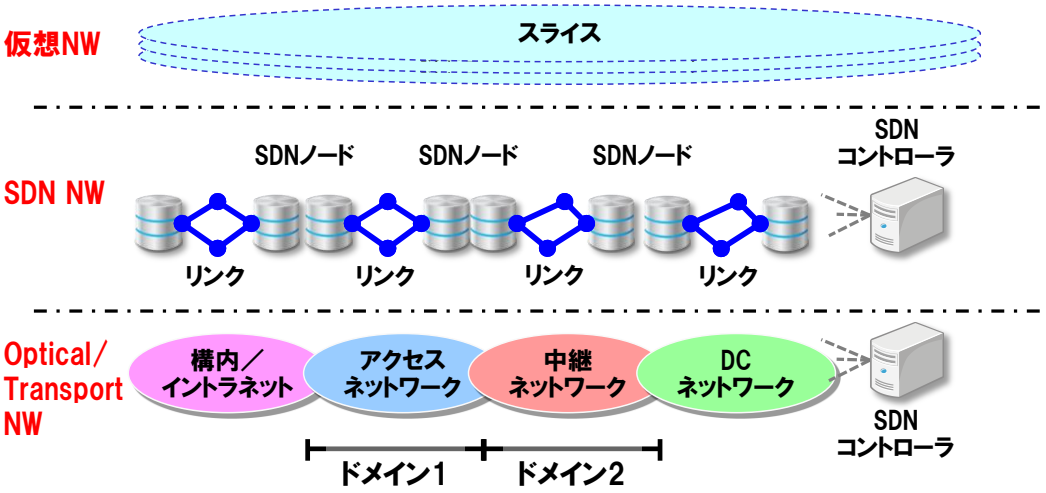

図 1-16　SDN を用いたネットワークモデル

伝送レイヤと SDN ノード、及び仮想ネットワークの 3 階層からなる構成で考え、各々を「Optical/Transport NW」、「SDN NW」、及び「仮想 NW」と呼ぶネットワーク階層として定義する（表 1-4）。

表 1-4　ネットワーク階層の概要

ネットワーク階層	概要
仮想 NW	SDN NW 上に構成される L2 以上の論理ネットワーク。
SDN NW	SDN ノードと、SDN ノード間のリンクで構成され、SDN コントローラから制御されるネットワーク。
Optical/Transport NW	SDN ノード間にリンクを提供するネットワーク。

「Optical/Transport NW」は、SDN ノード間にリンクを提供するネットワークである。SDN の仕組みを取り入れ、「Optical/Transport NW」リソースを抽象化して SDN コントローラからの制御を可能とした場合を「トランスポート SDN」と呼ぶ。

「SDN NW」は、SDN ノードと、SDN ノード間のリンク（論理リンク）で構成され、SDN コントローラから制御されるネットワークである。

「仮想 NW」は、SDN NW 上に構成される L2 以上の論理ネットワークであり、論理ネットワークで用いるリソースの単位を「スライス」と呼ぶ。

1.1. ネットワークの階層構造

図 1-16 に示す SDN を用いたネットワークモデルについて、ネットワークの階層構造に着目して詳細に示す。

「Optical/Transport NW」は Optical と Packet Transport により伝送レイヤを構成する。Optical は光ノードと物理リンクから構成され、Packet Transport に対して波長パスを提供する。Packet Transport は PTN(Packet Transport Network)ノードと論理リンクから構成され、「SDN NW」に対してパスを提供する。

「SDN NW」は SDN ノードと論理リンクから構成され、論理リンクは「Optical/Transport NW」から提供されるパス／波長パスを用いて（ホップ・バイ・ホップ方式の場合)、「仮想 NW」を構成する（図 1-17)。

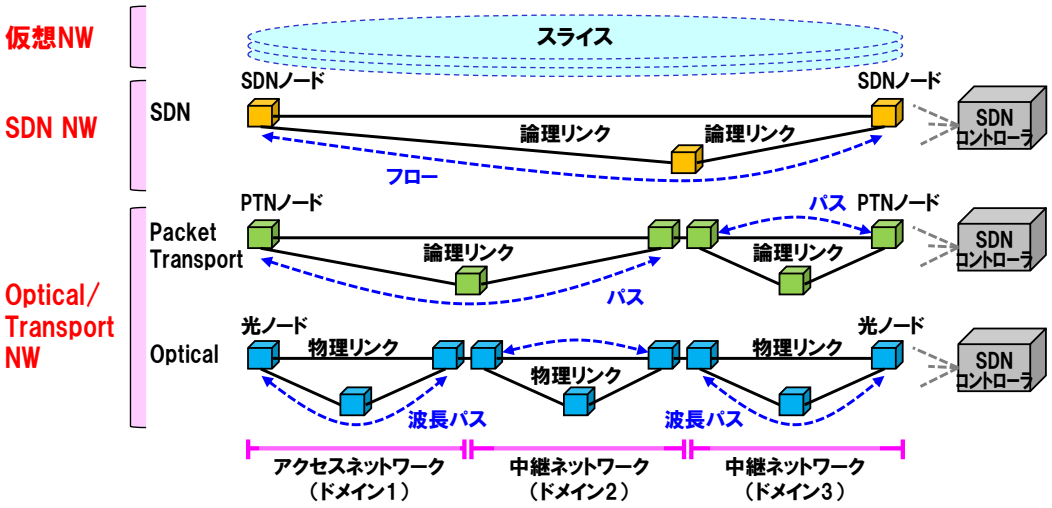

図 1-17　ネットワークの階層構造

リンクの構造に着目すると、SDN ノード-SDN ノード間の SDN 用のリンクは、パケットトランスポート用のリンク、WDM(Wavelength Division Multiplex)用のリンクの入れ子構造で構成される（図 1-18）。

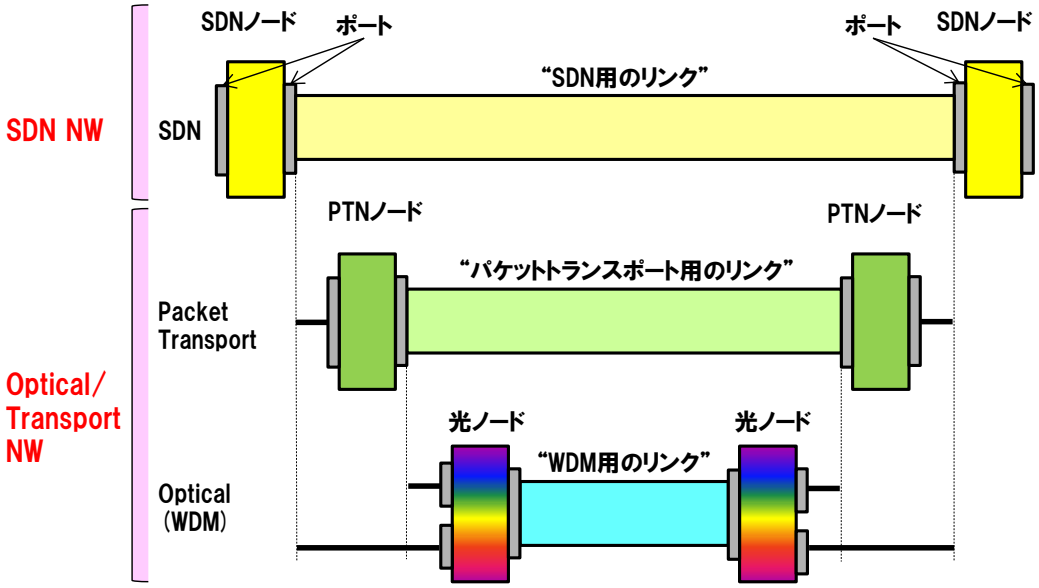

図 1-18　リンクの階層構造

1.2. コントロールプレーンとデータプレーン

「SDN NW」は、ネットワーク制御を担うコントロールプレーンと、パケット転送を担うデータプレーンが分離したアーキテクチャをとる(詳細は第 2 章 SDN の基礎)。コントロールプレーンは SDN コントローラと監視制御網で構成され、データプレーンは SDN ノードと論理リンクで構成される（図 1-19)。

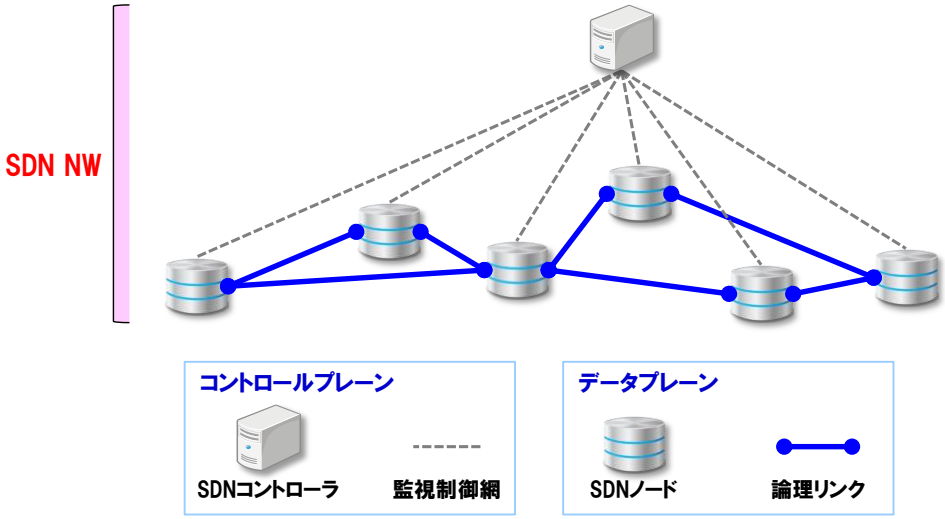

図 1-19　コントロールプレーンとデータプレーン

1.3. スライスの概要

「仮想 NW」を構成するスライスは、物理ネットワーク（SDN NW、Optical/Transport NW）のリソースを分割したものであり、各種ネットワークサービスを提供するためのフローは、スライスのリソースを使用して設定される（図 1-20)。スライスは提供するネットワークサービス単位やユーザ単位など、目的や用途、規模などに応じて設定するものである。

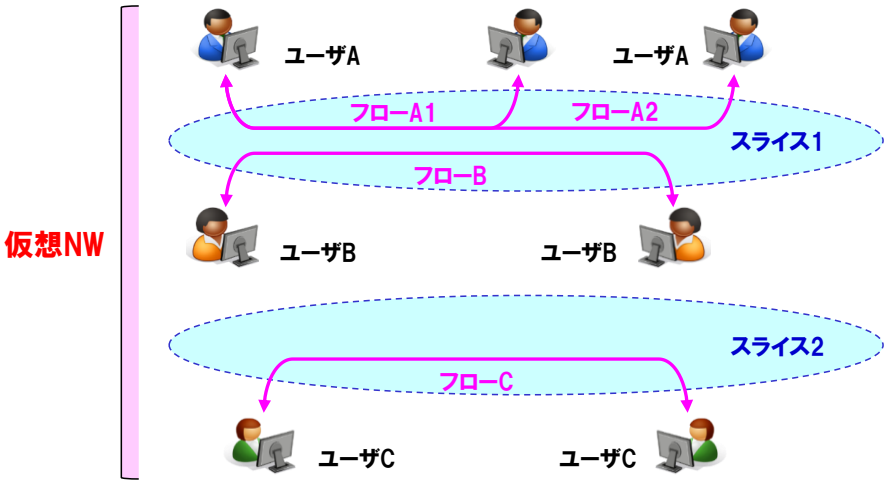

図 1-20　スライスのイメージ

2. ネットワーク管理モデル

SDN を用いた通信事業者のネットワークでは、マルチレイヤ、マルチドメインの多様な構成が想定される。このようなネットワークを制御し、さらにクラウド技術や NFV(Network Functions Virtualization)と組み合わせて新しいサービスを機動的に提供するための管理システムの構成例を図 1-21 に示す。この例では階層毎、ドメイン毎に SDN コントローラを配備し、分散的な制御を行っている。これにより各階層／ドメインで要求される機能、性能を個々のコントローラで個別に開発、維持管理が可能となる。各コントローラを束ねるオーケストレータをその上位に配備することで、ネットワークの End-to-End の制御が可能となる。オーケストレータはクラウド資源や NFV で用いられる網機能資源を提供するアプリケーションを制御するコントローラも束ねて管理している。このような構成では、例えばクラウド資源とネットワーク資源を組み合わせてネットワークサービスを提供する場合、それを実現するアプリケーションはオーケストレータの API(Application Programming Interface)を利用するアプリケーションを実装することになる。

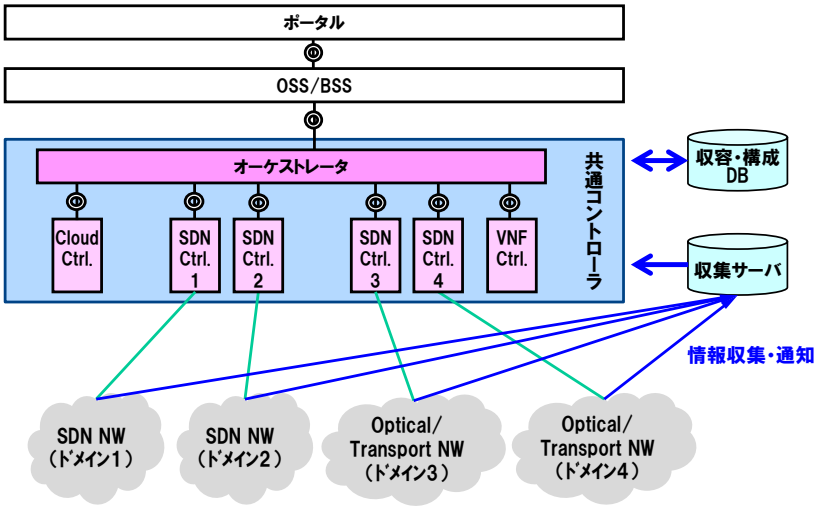

図 1-21　ネットワーク管理モデルの例

この例ではオーケストレータは複数の SDN コントローラを管理することとなり、プログラムが複雑化することが懸念される。それを回避する策として、図 1-22 に示すようなネットワークコントローラを中間に配備し、ネットワーク資源の管理、抽象化、さらには制御用の API 機能を配備することが有効と考えられる。O3 プロジェクトで開発を進めているネットワークコントローラとして「OdenOS」がある。

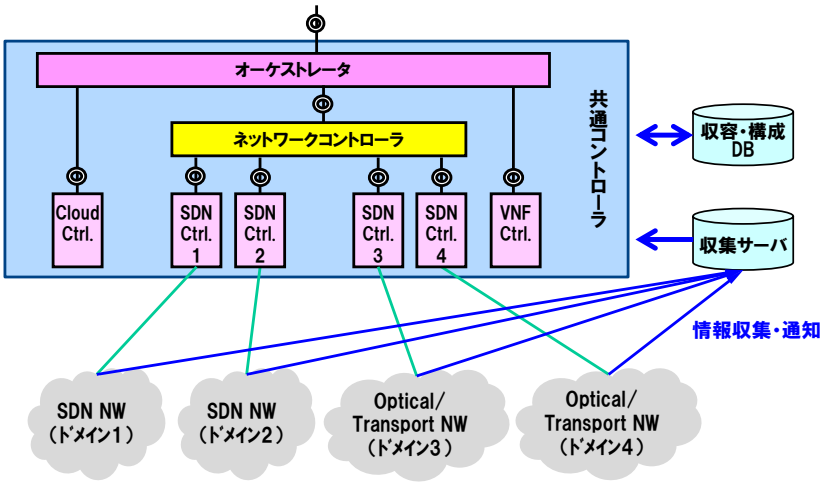

図 1-22　ネットワークコントローラの例

第3節 ネットワークモデルと本ガイドラインの対象

本ガイドラインで対象としている SDN を用いたネットワークについて、図 1-16 のモデルを用いて示す。

本ガイドラインでは通信事業者が管理する中継ネットワーク、アクセスネットワークに関して、「仮想 NW」、「SDN NW」、及び「Optical/Transport NW」を制御している SDN コントローラを想定して解説している（図 1-23）。

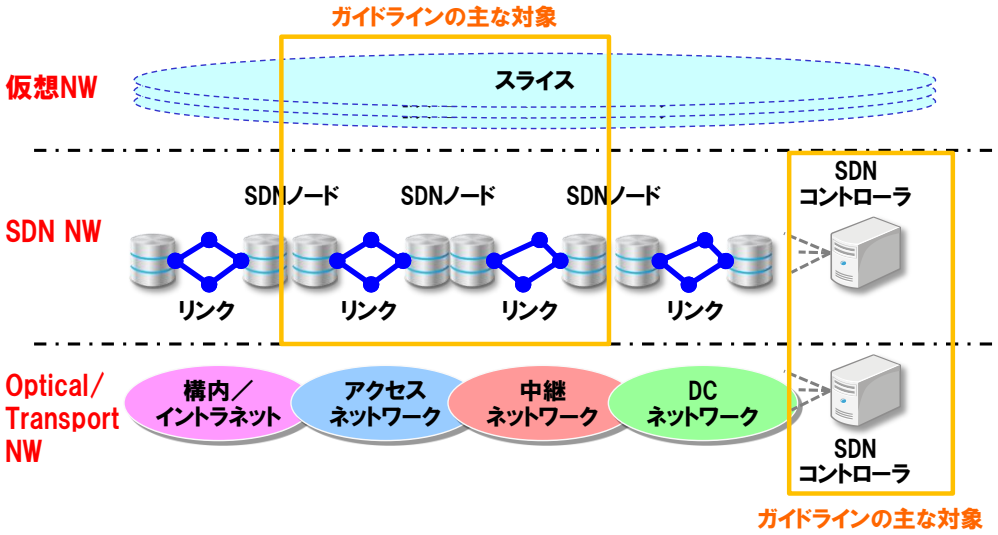

図 1-23　ネットワークモデルと本ガイドラインの対象

第4章 ネットワーク・ユースケース

第1節 ユースケースの概要

本ガイドラインで示すユースケースでは、Point-to-Point の回線サービスを想定する。通信事業者はユーザからネットワークサービスの利用申込を受け、ユーザの 2 ヶ所の拠点間を通信事業者のネットワークで結び、End-to-End の回線接続を提供する。対象とするネットワークは、通信事業者が管理する中継ネットワークとアクセスネットワーク、そしてユーザが管理する構内／イントラネットで構成される（図 1-24）。

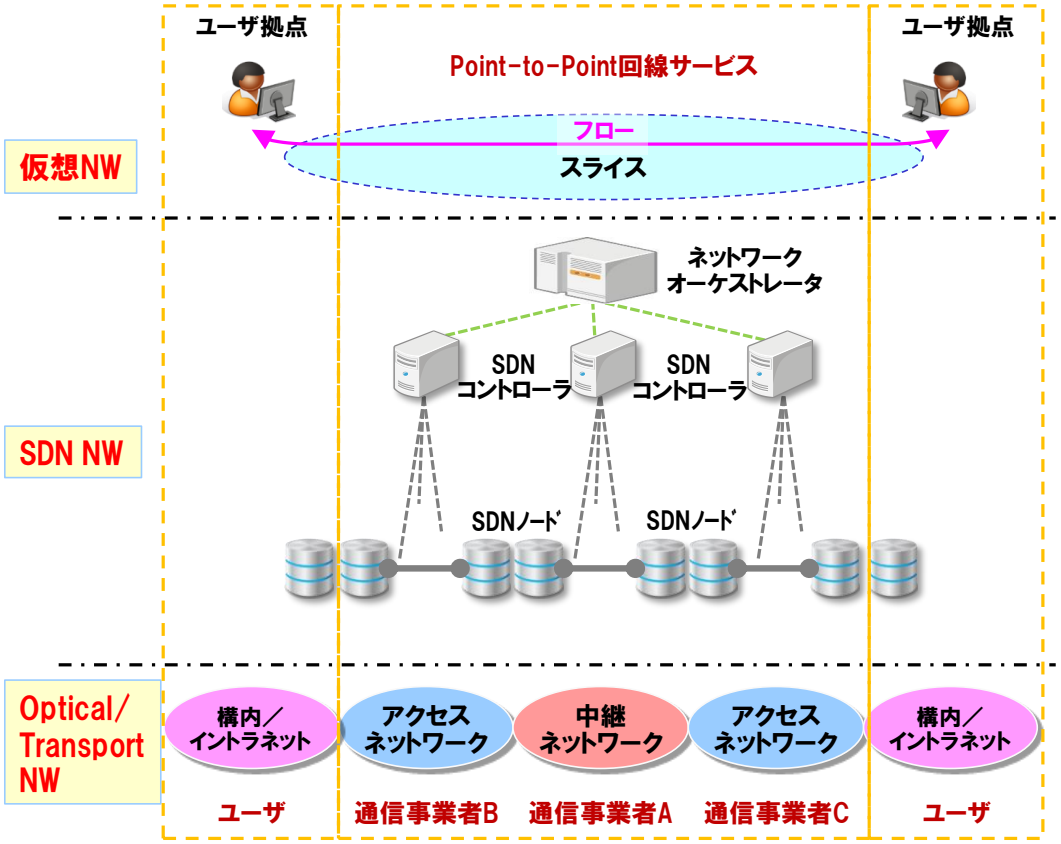

図 1-24　ユースケースの概要

回線サービスを提供する通信事業者 A は、中継ネットワークを管理しているものとする。ユーザの網内／イントラネットを接続するアクセスネットワークは一

方を通信事業者 B、もう一方を通信事業者 C とし、異なる通信事業者（異なるドメイン）によって管理されている。

各通信事業者が管理する中継ネットワーク、アクセスネットワークでは、SDN ノードからなる「SDN NW」が構築され、各通信事業者の SDN コントローラによる制御が可能である。また各通信事業者の SDN コントローラを統合的に制御するネットワークオーケストレータが配備され、通信事業者 A はネットワークオーケストレータを介して、ネットワークの End-to-End の制御が可能な構成とする。

「仮想 NW」では回線サービスを提供するための、ユーザ拠点間を結ぶフローが設定される。

第2節 SDN NW の設計・構築

「SDN NW」の設計・構築について示す。通信事業者 A は、通信事業者 B,C、及びユーザ拠点に SDN ノードを設置し、ホップ・バイ・ホップ方式にて「SDN NW」を構成するものとする（図 1-25）。

- 通信事業者 A は､アクセスネットワークを提供する通信事業者 B,C のコロケーションサービスを利用して SDN ノードを設置し、パスを調達する。
- 各ドメインの SDN ノードに対応する SDN コントローラ(コントローラ A,B,C)、及びネットワークオーケストレータより、ネットワークの制御を行う。
- ユーザ拠点に SDN ノードを設置する。

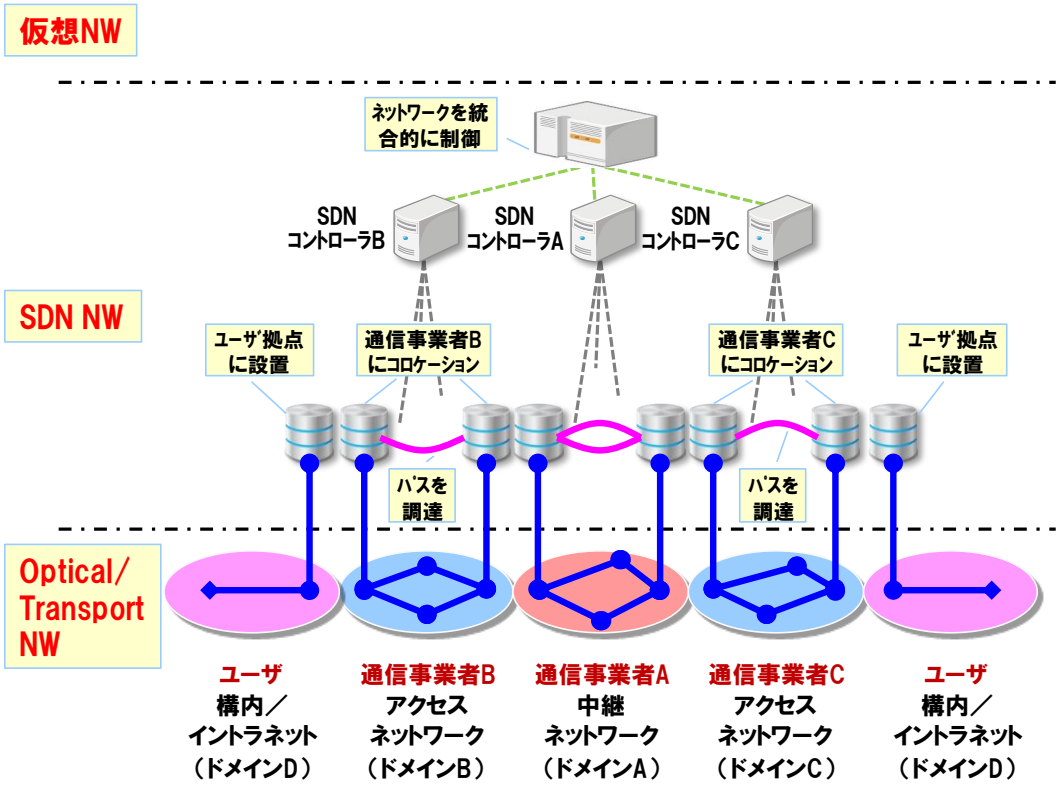

図 1-25　SDN NW の設計・構築

第3節 仮想 NW の設計・構築

「仮想 NW」の設計・構築について示す。通信事業者 A は、ユーザからの回線サービス申込みを受け、ユーザ拠点を結ぶフローを設定して回線サービスを開通する（図 1-26）。

- 通信事業者 A は回線サービスを提供するため、ネットワークリソースを確保する。
- ユーザからのサービス利用申込みを受け、各種条件を確認する。
  - ユーザの拠点
  - 品質条件
  - 信頼性条件
  - 帯域　　など
- ユーザの各種条件に基づきフローを設定して、回線サービスを開通する。

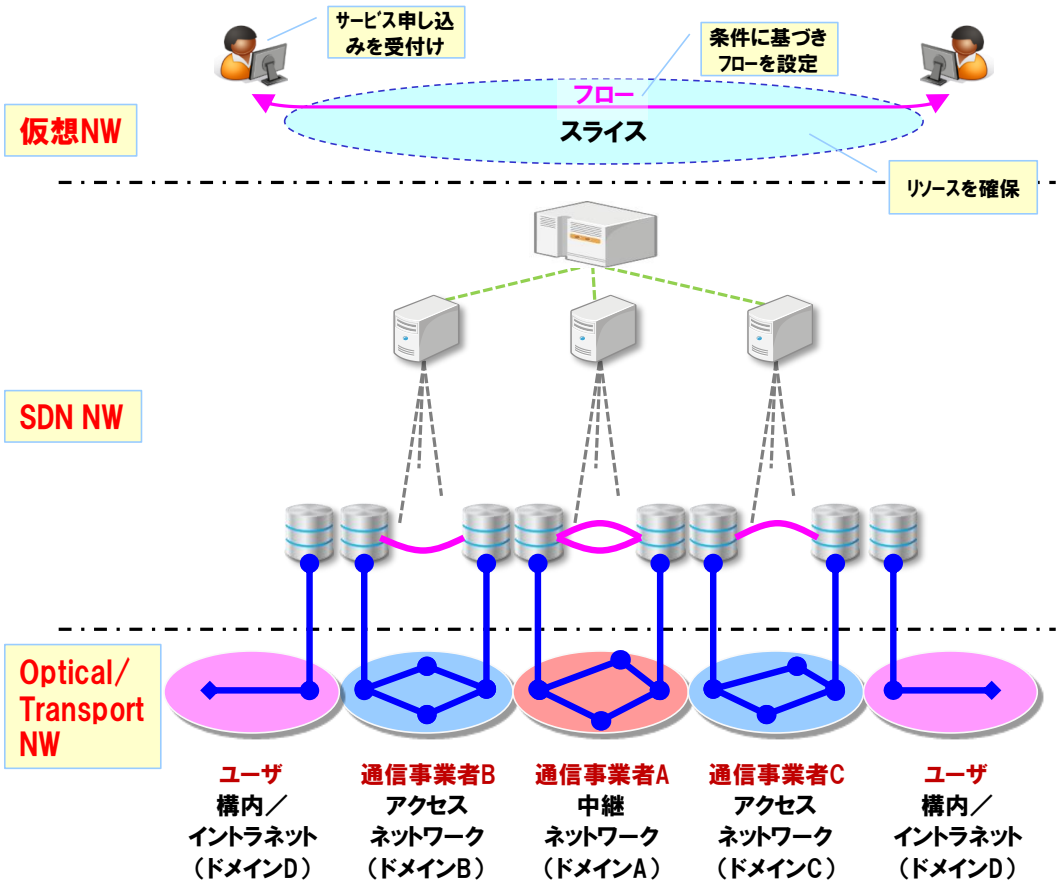

図 1-26　仮想 NW の設計・構築

第4節 通信事業者による運用・監視

通信事業者による運用・監視について示す。通信事業者 A はネットワークオーケストレータを介して、サービス提供状況の確認や、ネットワーク設定情報の変更を行うものとする。またネットワークの状態を常時監視し、異常発生時には、その検出と対処を行う（図 1-27）。

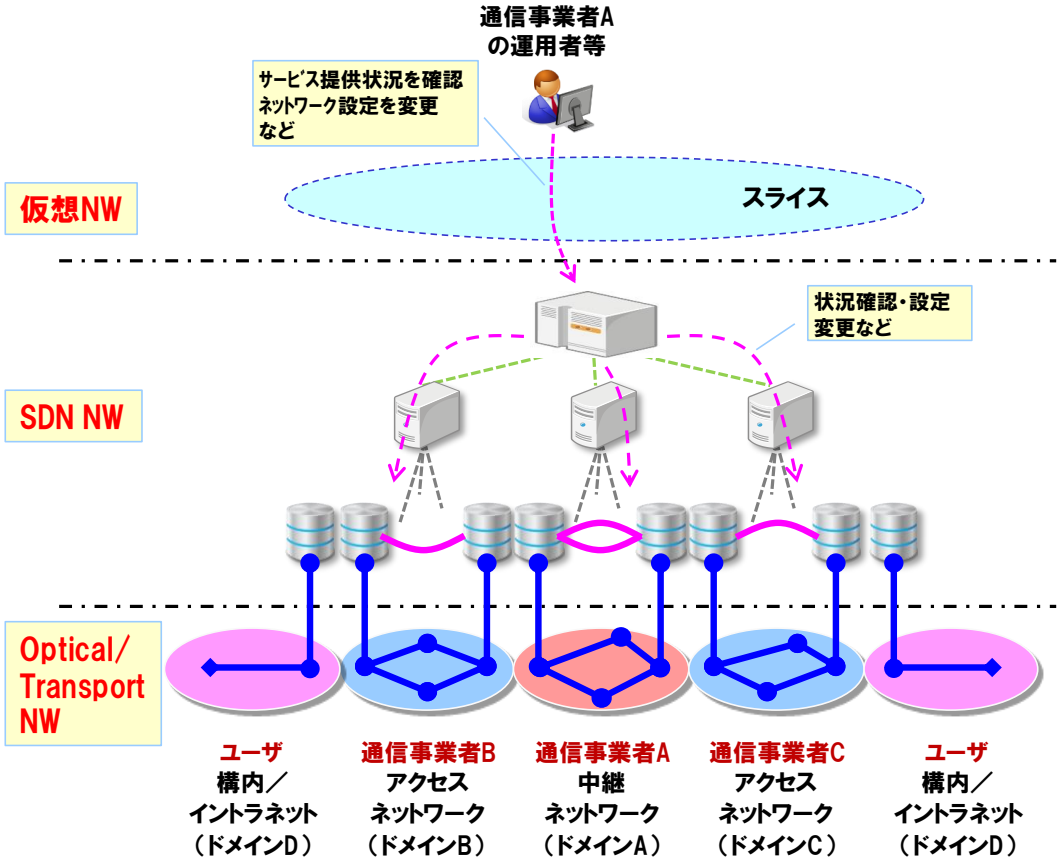

図 1-27　通信事業者による運用・監視

第5節 ユーザによる設定変更

ユーザによる設定変更について示す。回線サービスを利用するユーザは限られた条件においてネットワークの設定変更を行えるものとする。例えば回線の帯域や経路について、利用状況に応じて随時、ユーザに解放されたネットワーク制御用の API を介して変更を行う。また通信事業者においては充分なネットワークのセキュリティ対策（アクセス制限、アカウント管理等）を講じておく必要がある（図 1-28）。

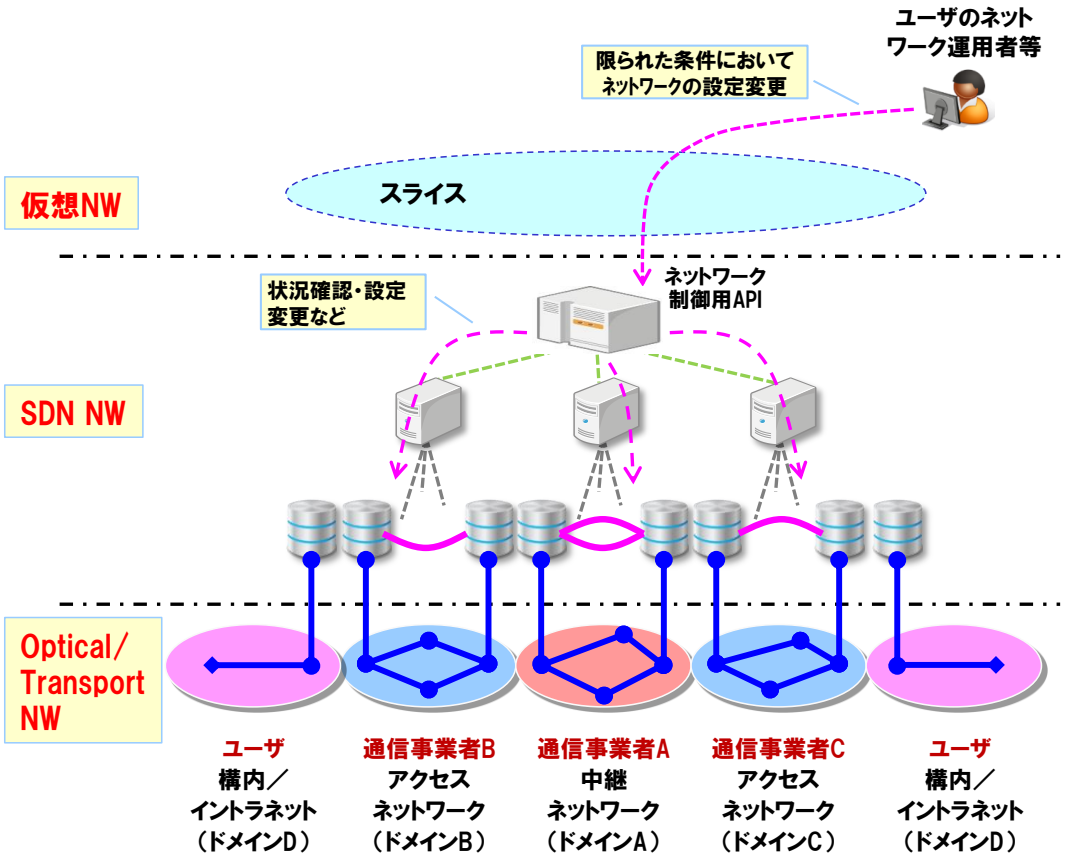

図 1-28　ユーザによる設定変更

第2編 設計・構築フェーズ

通信事業者がネットワークサービスを提供するにあたって、一般的に設計、構築、運用、監視といった業務フェーズがある。本ガイドラインは一般的な業務フェーズに則すものとし、第 2 編において設計・構築について、通信事業者のネットワークに SDN を用いるための基本的な考え方について示す。なお、第 3 編では運用・監視について示す。

- 第 2 編 設計・構築フェーズ
- 第 3 編 運用・監視フェーズ

第1章 設計・構築フェーズの基本的な考え方

第1節 設計・構築フェーズの位置付け

設計・構築フェーズでは、提供するネットワークサービスの要求条件を満たすためのネットワーク構成を定め、定めたネットワーク構成に従い「SDN NW」を構成する SDN ノードや SDN コントローラを設置・設定する。また「仮想 NW」としてスライスやフローの設定を行い、ユーザに対してネットワークサービスを開始する。

第 2 編では、第 2 章で「SDN NW」の設計・構築について、第 3 章で「仮想 NW」の設計・構築について示す。

- 第 2 章 SDN NW の設計・構築
- 第 3 章 仮想 NW の設計・構築

第2節 設計・構築のサイクル

第 2 章において「SDN NW」、第 3 章において「仮想 NW」の設計・構築について示すのに先立ち、ネットワークの階層によって設計・構築を行うサイクルが異なることについて、「Optical/Transport NW」も含めて補足する。

「Optical/Transport NW」はネットワークの基盤を構成する階層であり、光ノードや PTN(Packet Transport Network)ノードの設計・構築を行う。また各種機器を設置するビルやビル間を繋ぐ光ケーブルといったレイヤ 0 と呼ばれる設備を設計・構築する場合は、膨大な期間や費用を要すこともあり頻繁な変更は困難である。これより、数ヶ月から数年先を見据えたサービスの需要予測に基づき設備を計画的に見直す必要があり、「Optical/Transport NW」の設計・構築サイクルは比較的に長いことが想定される。

「SDN NW」は「Optical/Transport NW」の上位ネットワークとして構成する階層であり、SDN ノードや SDN コントローラの設計・構築を行う。ネットワークサービスの利用状況や機器の使用率などを監視しつつ、必要に応じて機器の見直しを行うことが想定される。但し、ソフトウェアで機能する機器はソフトウェア更新が頻繁に発生する可能性がある。

「仮想 NW」は「SDN NW」の上位に論理的に構成する階層であり、スライスやフローの設定を行う。ユーザからの日々のサービスオーダ（利用申し込み）を受けて「SDN NW」にフローを設定してユーザに提供することとなり、サービスオーダに応じて随時、フローの設計・構築を行う。またネットワークサービス戦略の見直しや新規ネットワークサービスの提供、ユーザからの要望などに応じて随時、スライスやフローを変更・追加していくことが想定される。

ネットワークの各階層の設計・構築サイクルのイメージを図 2-1、及び表 2-1 に示す。

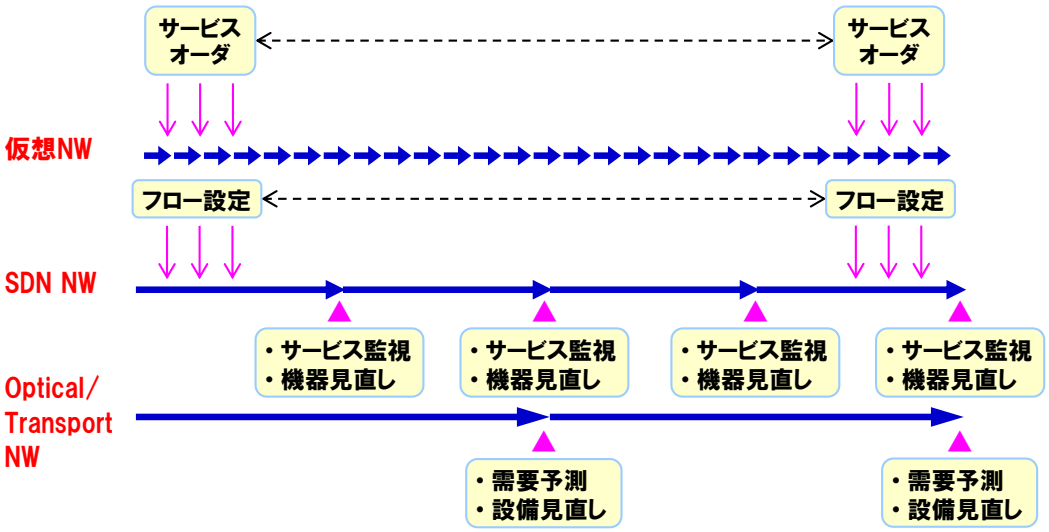

図 2-1　ネットワーク階層と設計・構築サイクル

表 2-1　ネットワーク階層と設計・構築サイクル

ネットワーク階層	設計・構築サイクル	サイクルの目安
仮想 NW	サービスオーダに応じて随時フローやスライスを設定	数時間～数日
SDN NW	サービスの利用状況や機器の使用率に応じて機器を見直し	数ヶ月
Optical/Transport NW	サービスの需要予測に基づき計画的に設備を見直し	数ヶ月～数年

第3節 OpenFlow を用いた設計・構築

SDN を実現する主要な技術として OpenFlow がある。OpenFlow はネットワーク機器のデータプレーンとコントロールプレーンを分離し、データプレーンの機能を OpenFlow スイッチに、コントロールプレーンの機能を OpenFlow コントローラに持たせ、ネットワークの集中制御を行うアーキテクチャを採る（詳細は第 1 編第 2 章 SDN の基礎）。

本ガイドラインでは通信事業者のネットワークにおいて、SDN を実現する技術として OpenFlow を用いることを想定している。そのため SDN ノードは OpenFlow スイッチ、SDN コントローラは OpenFlow コントローラを前提として解説する。

- SDN ノード：OpenFlow スイッチを想定
- SDN コントローラ：OpenFlow コントローラを想定

第2章 SDN NW の設計・構築

『第 2 章 SDN NW の設計・構築』では、第 1 節においてコントロールプレーンについて、第 2 節においてデータプレーンについて基本的な考え方を示す。

- 第 1 節 コントロールプレーン
- 第 2 節 データプレーン

コントロールプレーンの設計・構築では、OpenFlow コントローラの種類や配備の考え方について、データプレーンの設計・構築では、OpenFlow スイッチの種類や配備の考え方について示す。

「SDN NW」の設計・構築の考え方は、基本的には従来ネットワークの設計・構築の考え方を踏襲するものである。本ガイドラインでは OpenFlow の特徴を考慮し、OpenFlow を用いる際に特有の事項に着目して解説するものとし、従来ネットワーク技術の考え方に従う事項については最小限の記述に留める。

第1節 コントロールプレーン

1. OpenFlow コントローラ

1.1. OpenFlow コントローラの種類

SDN コントローラには、商用の OpenFlow 対応のコントローラ製品がいくつかある。コントローラ製品を使用する場合は、製品によって機能や性能条件などが異なるため、適用するネットワークの性質や規模などに応じて適切な機器を選定する必要がある。但し、現時点では提供される製品種が少なく、また高価である場合が多い。

また、コントローラ製品を使用せずに、自主開発を行う方法もある。自主開発を行う場合は、OpenFlow コントローラのフレームワークを用いて、適用するネットワークに応じた機能や性能条件を備えた OpenFlow コントローラを実現できるが、開発のための技術検討やコストを要する。

1.2. OpenFlow コントローラフレームワークの特徴

OpenFlow コントローラフレームワークには、OSS(Open Source Software)と非 OSS（ソースコードを公開していないソフトウェア（プロプライエタリソフトウェア））がある。OSS を利用することで、効率的な開発が可能となりカスタマイズも容易となる。

OSS については、既に主要な言語に対応したフレームワークが存在しており、開発言語に対する要求条件に応じた選択の自由度が確保されている。例えば、実行効率よりも短期開発という生産性を重視する場合は、少ないコードで実装可能なスクリプト言語（Ruby、Python 等）に対応したフレームワークを選択したり、開発要員や既存のソフトウェア資産の活用、連携などを重視する場合は、C や Java に対応したフレームワークを選択することが考えられる。いくつかのフレームワークには、開発やデバックに便利なツールが提供されている。

OSS のライセンス形態としては、GNU General Public License(GPL v2/GPL v3)、Apache License(Apache 2.0)、Eclipse Public License(EPL v1)などがある。GPL v2/GPL v3 は、再配布、二次的著作物についても同じライセンスを適用する必要があるため、商用利用の際は注意が必要となる。

主要な OSS の OpenFlow コントローラのフレームワークを表 2-2 に示す。

表 2-2　主要な OpenFlow コントローラフレームワーク

名称	開発言語	開発元	ライセンス	特徴
Trema	Ruby、C	NEC	GPL v2	・ 日本国内では、非常に良く使われているコントローラ。 ・ 実行速度より開発効率に重点をおいている。 ・ ツール類も豊富。
NOX	C++	スタンフォード大学	GPL v3	・ OpenFlow の登場直後から開発している。 ・ OpenFlow 仕様を作った研究者が関係している。 ・ 歴史が古いため、Web での情報が集めやすい。
POX	Python	UC バークレイ	GPL v3	・ NOX から派生。 ・ Linux/Mac/Windows と OS を問わず動作が可能。 ・ アプリケーションの数は少ない。
Floodlight	Java	Big Switch Networks	Apache 2.0	・ プログラム人口の多い Java を使用。 ・ POX と同じく OS を問わず動作できる。

OpenDaylight	Java	OpenDaylight プロジェクト	EPL v1	・ 主要ベンダが共同で開発。 ・ SDN コントローラおよび Northbound /Southbound API 等の標準化を目指している。
Ryu	Python	NTT 研究所	Apache 2.0	・ OpenFlow バージョンが更新されると即座に最新版が提供される。 ・ 仮想スイッチの Open vSwitch を Ryu で制御可能。

2. SDN コントローラの冗長化

OpenFlow コントローラは監視制御網の OpenFlow チャネルを通して OpenFlow スイッチと接続して制御を実施する。通信事業者の提供する大規模なネットワークでは制御対象となる OpenFlow スイッチが多数となるため、OpenFlow コントローラに異常が発生した場合の影響が大きい。そのため信頼性の確保のために OpenFlow コントローラの冗長化を考慮する必要がある。

OpenFlow では、1 つの OpenFlow スイッチが複数の OpenFlow コントローラへ接続するための仕様が規定されている。冗長構成における OpenFlow コントローラの動作は「Role」と呼ばれる表 2-3 の 3 種類のいずれかの属性の割り当てにより決定される。

この仕組みを利用して OpenFlow コントローラの役割を決定することが可能である。例えば、OpenFlow コントローラ同士が Ping にて死活監視を行い、SLAVE が MASTER の異常を検知した場合、自身の役割を MASTER に変更する方法で冗長性を確保することが可能となる。

表 2-3　OpenFlow コントローラの冗長化

属性	内容
EQUAL	・ OpenFlow スイッチに対してフルアクセスを許可している。 ・ Packet-In 等の全ての非同期メッセージを受け取る。
SLAVE	・ OpenFlow スイッチに対し Read-Only のアクセスが可能。 ・ Packet-In 等の全ての非同期メッセージを受け取らない。
MASTER	・ EQUAL と同様の権限を持つが、OpenFlow スイッチは MASTER を 1 つのみ持つ。

また、OpenFlow コントローラを冗長構成とした場合には、コントローラ間でのデータの同期の方法を定め、故障等でコントローラが切り替わった際には、待機系のコントローラが管理・制御を引き継げるような仕組みを用意しておく必要がある。

3. 同一ドメイン内の SDN コントローラの配備

OpenFlow コントローラは、対象とするドメイン内の OpenFlow スイッチを制御する。コントローラはドメイン内でスイッチの集中管理を行うが、処理性能や拡張性などが課題となる場合には、コントローラの集中型と分散型の使い分けを考慮する必要がある（図 2-2）。

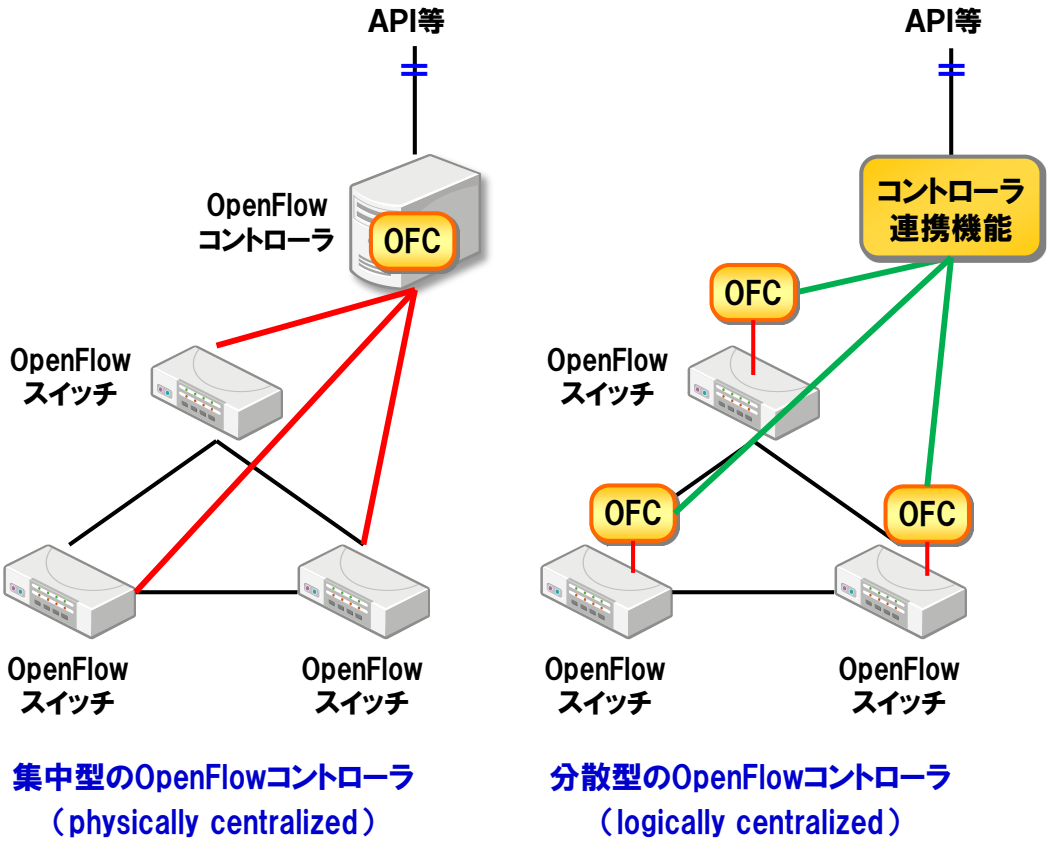

図 2-2　集中型と分散型の OpenFlow コントローラ

3.1. 集中型コントローラ

一つの OpenFlow コントローラで集中的に管理を行う方式である。マクロな視点でのトラヒックエンジニアリングにより、多様なニーズに対する経路探索への対応が可能である。分散型と比較して運用・管理が容易である一方、処理能力や拡張性などがボトルネックとなる場合がある。

3.2. 分散型コントローラ

OpenFlow スイッチに OpenFlow コントローラの機能を分散配備する方法である。また分散配備した機能をコントローラ連携機能などにより階層化する方法も考えられる。

フロー切り替え等の、比較的単純であるが短時間に多数の処理を要求される場合など、処理性能の条件が厳しい制御に有効である。

実現例としては、スイッチ内の OS に OpenFlow コントローラをインストールする方法が考えられる。この場合、OpenFlow プロトコルはスイッチ内でやり取りされることになる。但し、外部のコントローラとの制御情報のやり取りの規定は OpenFlow にはないため、通信方式を検討して OpenFlow コントローラに実装する必要がある。

3.3. 集中型と分散型の使い分け

集中型と分散型の両方式のコントローラの協調制御も考えられる。例えば、隣接ノードの発見などの単純な制御を分散型コントローラで行い、集中型コントローラがその情報を集約して、全体トポロジの作成・管理を担当する方法である。

このように集中型コントローラと分散型コントローラの特性を踏まえ、適用するネットワークの規模や想定される処理負荷、将来の拡張性などを考慮して、設計の段階で適切な方式を選択することが望ましい。

またコントローラの試験などにより実際の処理性能を測定して、要求される条件を満たすかどうかを判断することも重要である。

4. 異なるドメイン間の SDN コントローラの連携

複数の SDN を用いたネットワーク（異なるドメインや OpenFlow 以外の SDN ネットワーク）を接続してサービスを提供する場合、接続先のネットワークを含めた状態管理や制御を行うために、各ネットワークを管理する SDN コントローラ同士の連携が必要となる。

4.1. SDN コントローラ間の連携

各 SDN コントローラが管理するネットワーク構成情報を、SDN コントローラ間で通知することで、相互のネットワーク状態の管理を行う（図 2-3）。但し、この方法は連携のための SDN コントローラの処理が増大する懸念がある。また SDN コントローラ間での通信方式が標準化されておらず、通信方式を検討して SDN コントローラに実装する必要がある。

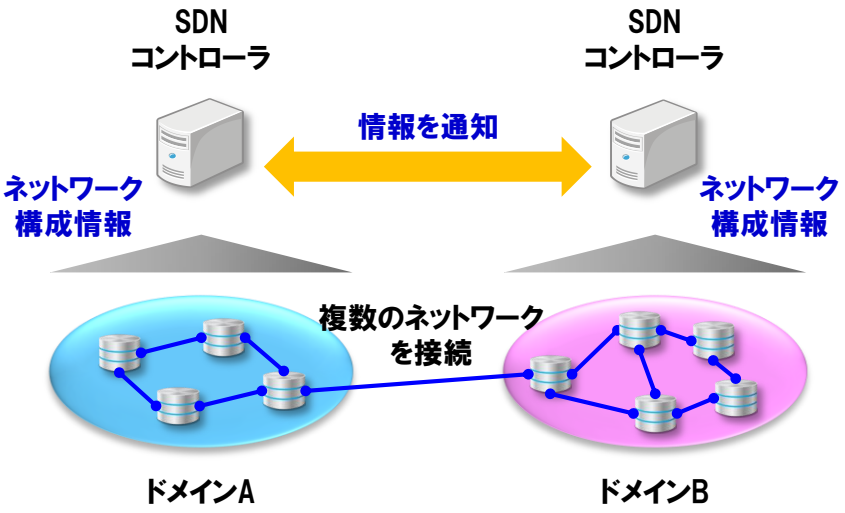

図 2-3　SDN コントローラ間の連携

4.2. オーケストレータを介した連携

各 SDN コントローラが管理するネットワーク構成情報をオーケストレータに集約し、統合管理を行う方法も考えられる（図 2-4）。コントローラ間で連携を行う場合は独自に通信方式を定める必要があるが、オーケストレータを介した連携は SDN コントローラの API を介すことで早期の実現が期待できる。また SDN コントローラの処理の一部をオーケストレータに分散させることにより、SDN コントローラの処理負荷軽減や機能分散の効果も期待できる。

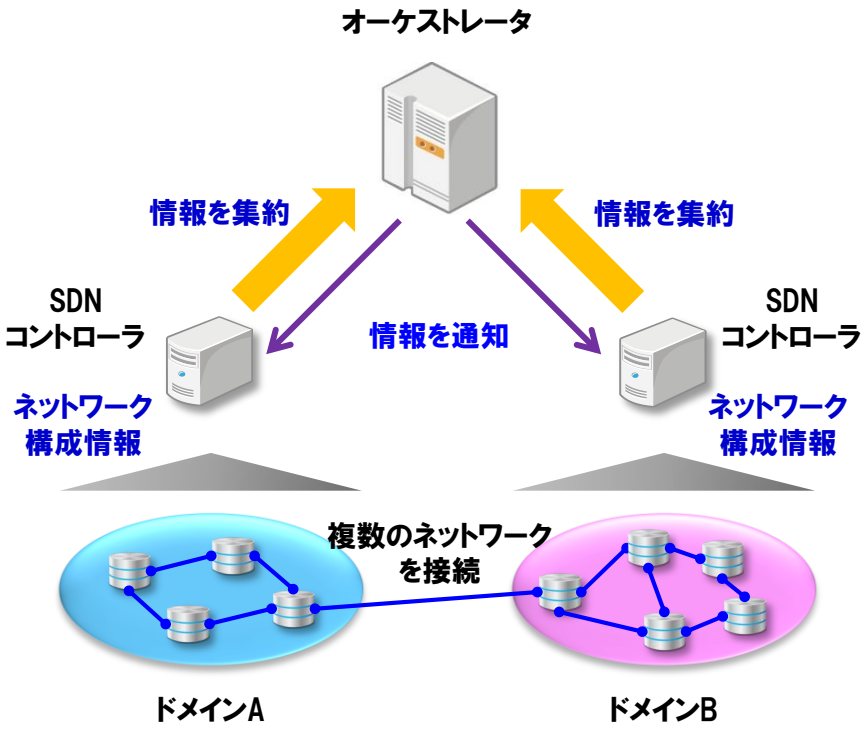

図 2-4　オーケストレータを介した連携

第2節 データプレーン

1. OpenFlow スイッチ

1.1. OpenFlow スイッチの種類

OpenFlow スイッチは、OpenFlow 対応のハードウェアスイッチと、汎用サーバに仮想スイッチのソフトウェアをインストールして使用するソフトウェアスイッチがある。

1.1.1. ハードウェアスイッチ

ハードウェアスイッチは、OpenFlow スイッチとしてのみ動作する機器と、従来の L2/L3 スイッチとしても使用できるハイブリッド型とがある。従来のスイッチの OS やファームウェアをアップグレードすることにより OpenFlow 対応スイッチとするアプローチが主流になってきており、スイッチの選択肢が増えてきている。

ハードウェアスイッチはパケット転送能力に優れ、大規模ネットワークへの適用が期待できるが、一方でフローテーブルのエントリ数の上限が少ないなど機能に制限があったり、チップベンダのリリースタイミングに影響され、新機能や新プロトコルへの対応が遅れるなどの懸念もある。

一般的なハードウェアスイッチの主な特徴を以下に整理する。

- パケット転送能力に優れる。
- 大規模ネットワークへの適用に期待。
- 物理ポート数が多い。
- フローテーブルの上限が少ない。
- 新機能や新プロトコルへの対応が遅れる場合がある。

1.1.2. ソフトウェアスイッチ

ソフトウェアスイッチは汎用サーバを用いて安価に、かつ、要求条件や性能要件に応じて幅広い用途で使用することが期待される。ハードウェアスイッチと比べると実装が容易であることから新サービスや新プロトコルへ早期に対応しや

すく、機能追加に容易に対応できるため柔軟性に優れている。一方でフローエントリ数の増加や Match 条件の複雑化、またショートパケットが多い場合などにパフォーマンスが低下するなど、パケット転送能力に制約が生じる場合がある。

ソフトウェアスイッチの例としては NTT が開発した「Lagopus（ラゴパス）」があり、オープンソースソフトウェア（OSS）として提供されている。

一般的なソフトウェアスイッチの主な特徴を以下に整理する。

- 汎用サーバを用いて実装が容易。
- 機能追加などの柔軟性に優れ適応領域が広い。
- パケット転送能力に制約が生じる場合がある。

1.1.3. 動作条件の考慮事項

OpenFlow には複雑な仕様や数々の Optional があり、OpenFlow に準拠したスイッチであっても対応している動作（Match Fields や Actions など）に制限がある場合がある。OpenFlow に準拠したスイッチに対して、OpenFlow 仕様の実装状況をテストして結果を公開する [Ryu Certification]があり、スイッチ選定の際には活用することが望ましい。

1.2. ソフトウェアスイッチの利用形態

ソフトウェアスイッチはハードウェアスイッチのように専用機器を購入する必要がなく、様々な用途で使用可能であり導入が容易である。また機能が充実して適用範囲が広いといえる。

ソフトウェアスイッチの利用形態には、物理ハードウェアの汎用 OS 上に実装する方法と仮想環境に実装する方法がある。仮想環境への実装はハイパーバイザと呼ばれる仮想化ソフトウェアを利用する。ソフトウェアスイッチの代表的な利用形態を図 2-5 に示す。

A) 物理ハードウェア上にソフトウェアスイッチを実装（ベアメタル）

B) ハイパーバイザ内にソフトウェアスイッチを実装

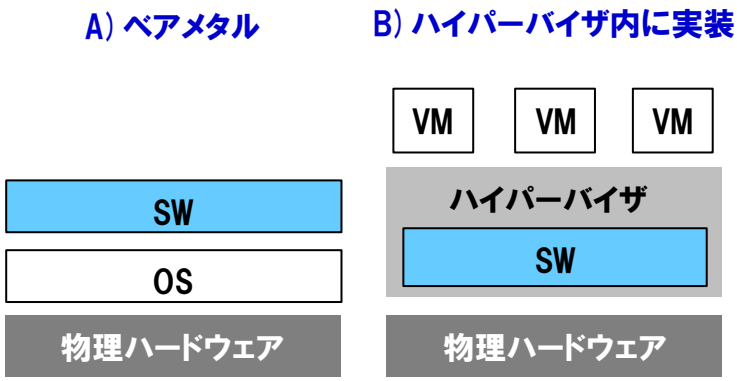

図 2-5　ソフトウェアスイッチの代表的な利用形態

物理ハードウェア上に（ハイパーバイザを介さずに）ソフトウェアスイッチを実装するベアメタル(A)は、物理ハードウェアのリソースを占有できるため高い処理性能が期待できる。一方、多重化ができないため、多数の OpenFlow スイッチを必要とする場合には適さない。(B)は仮想化を実現するためのハイパーバイザにソフトウェアスイッチを組み込む構成であり、仮想化環境において主流となる形態である。

仮想化環境でソフトウェアスイッチのパフォーマンスを安定的に使用するためには、スイッチに割り当てるリソースを固定化する設定を行うことが望ましい。但しその場合には制御を行う VM(Virtual Machine)数などに制限が生じる場合がある。また利用する仮想化ソフトウェアによって機能に違いが生じたり、CPU やボードによって性能が異なることがあるので、事前に十分な検証を行うことが重要である。

いずれの利用形態においても物理ハードウェアのスペックについて考慮が必要である。特に仮想環境にソフトウェアスイッチを実装する場合、1 台の物理ハードウェアで複数の VM のリソースをシェアするため、CPU やメモリ、及びディスク容量には充分な余裕があることが望ましい。

2. コントロールプレーンとの接続

コントロールプレーンとデータプレーンの分離に起因する考慮事項について示す。コントロールプレーンとデータプレーン間に通信断が発生した場合（監視制御網に通信断が発生した場合）にはネットワークの制御情報のやり取りができなくなり、データプレーンのパケットのフォワーディング動作に影響を与えることがある。

OpenFlow のプロトコルでは、OpenFlow スイッチと OpenFlow コントローラとの間に通信断が発生した場合の、OpenFlow スイッチのフォワーディング動作を規定している（表 2-4）。通信断時のモードは OpenFlow スイッチの初期設定で定められ、パケット処理方式（プロアクティブ型／リアクティブ型）と合わせて考える必要がある。

表 2-4　OpenFlow プルトコルでのコントローラとの通信断時の動作の規定

モード	動作概要
Emergency Flow Cache	エマージェンシーフローエントリを適用して、フォワーディング動作を継続。
Fail Secure Mode	既に登録されているフローエントリを使用して、フォワーディング動作を継続。
Fail Standalone Mode	通常の（OpenFlow ではない）スイッチまたはルータとして機能。

「Emergency Flow Cache」モードでは、OpenFlow コントローラとの接続が絶たれた場合に、OpenFlow スイッチはエマージェンシーモードに移行する。通常時のフローエントリを削除し、スイッチの初期設定で定められたエマージェンシーフローエントリを適用して、フォワーディング動作を継続する。但し、OpenFlow 1.1 以降では「Emergency Flow Cache」は削除された。

「Fail Secure Mode」では、OpenFlow コントローラとの接続が絶たれた場合に、OpenFlow スイッチは既に登録されているフローエントリを使用してフォワーディング動作を継続する。フローエントリ登録時に指定した Time Out 値に達してフローエントリが削除されるまでフォワーディング動作が継続されるため、Time Out 値をどの程度に設定するかが重要である。

パケット処理方式がプロアクティブ型の場合はOpenFlowスイッチにフローエントリが予め登録されているため、「Fail Secure Mode」と組み合わせることでOpenFlowコントローラとの接続が絶たれた場合でもフォワーディング動作を継続できる。一方、リアクティブ型の場合はOpenFlowスイッチが宛先不明のパケットを受信した後にフローエントリを登録する方式であり、宛先不明なパケットに対する学習が不可能となりフォワーディング動作が停止する。

「Fail Standalone Mode」では、OpenFlowコントローラとの接続が絶たれた場合に、OpenFlowスイッチはフローエントリによるフォワーディングは行わずに、通常の（OpenFlowではない）スイッチまたはルータとして機能する（ハイブリッド型のスイッチの場合）。但し、宛先不明のパケットをマルチキャストするモードに移行し、ブロードキャストストームを起こしてバーストするケースがあるため注意が必要である。

3. 冗長化方式

従来ネットワークの信頼性の考え方として、例えばノードではVRRP(Virtual Router Redundancy Protocol)などのプロトコルがあり、各装置はプロトコルに従い自立制御で冗長化を実現している。またリンクではアクティブ／スタンバイなどによる代替リンクを各装置に設定して、冗長化を実現している。

OpenFlowでは、OpenFlowスイッチの冗長化を制御するのはOpenFlowコントローラの役目である。スイッチの異常をコントローラで検出してソフトウェア制御で切り替えを実施する必要があり、そのための機能をネットワークアプルケーションとしてコントローラに実装しておく必要がある。ノードやリンクについては必要に応じて冗長化構成を取れるよう設計をしておく必要がある。

切り替えにはノード／リンク毎での切り替えや，End-to-Endのパスで切り替えるなど様々な方法がある（切り替え方式の詳細については第3編の運用・監視フェーズで述べるものとし、今後の課題とする。）。

異常検出の観点では、OpenFlowコントローラからOpenFlowスイッチに対して「Echo Request/Reply」メッセージを定期的に送信することで、スイッチの死活監視が可能である。

第3章 仮想 NW の設計・構築

『第 3 章 仮想 NW の設計・構築』では、第 1 節においてフローと回線について、また第 2 節ではネットワークサービスを提供するにあたり必要となる情報の管理について基本的な考え方を示す。

- 第 1 節 フローと回線
- 第 2 節 情報の管理

第1節 フローと回線

1. 回線サービスの提供

「仮想 NW」は「SDN NW」の上位の論理ネットワークとして位置付けられ、SDN を用いたネットワークにおいてネットワークサービスを構築して提供する。通信事業者が SDN を用いて提供するネットワークサービスは各種が想定されるが、本ガイドラインでは、各種ネットワークサービスの基本となる Point-to-Point でユーザ拠点間を接続する回線サービスを例として取り上げ、ネットワークサービスの実現方法や考慮事項などについて示す（図 2-6）。

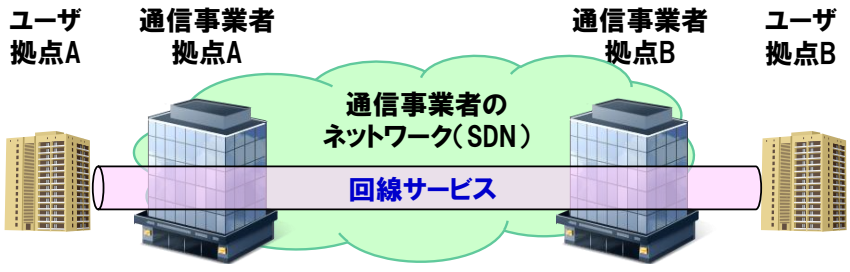

図 2-6 回線サービスの提供

2. フローと回線の定義

フローは、リンク（SDN 用のリンク）及び SDN ノードを通過するパケットの通り道であり、End-to-End のポート間に片方向に設定されるものと定義する。OpenFlow では、OpenFlow コントローラ（SDN コントローラ）から OpenFlow スイッチ（SDN ノード）のフローテーブルに対し、フローエントリとして設定される。

回線とは、1 つ以上のフローで構成され、ユーザに対してネットワークサービスを提供する単位として定義する。End-to-End に Point-to-Point で設定され、ユーザの拠点間を接続する。

また回線は SDN 以外のドメインを組み合わせてネットワークサービス提供を行う場合がある（表 2-5、 図 2-7）。

表 2-5　フローと回線の定義

	定義
フロー	・ リンク及び SDN ノードを通過するパケットの通り道。 ・ End-to-End のポート間に片方向で設定。
回線	・ ユーザに対してネットワークサービスを提供する単位。 （SDN 以外のドメインを組み合わせてサービス提供を行う場合あり） ・ 1 つ以上のフローで構成。

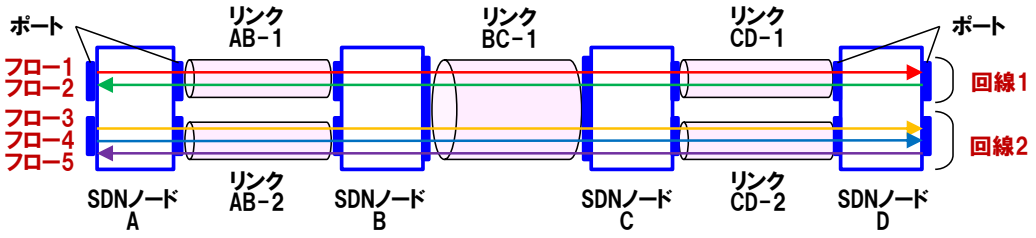

図 2-7　フローと回線の定義

3. プロアクティブとリアクティブ

フローはパケットの処理方法から、プロアクティブとリアクティブに分類される。プロアクティブとリアクティブのフローの特徴を以降に示す。

なお、フローで構成される回線についても同様に、プロアクティブとリアクティブの回線が存在して同様の特徴をもつ。

3.1. プロアクティブのフロー・回線の特徴

プロアクティブの場合は、SDN ノードのフローテーブルにフローエントリが予め設定済みであり、静的な回線設定となる。SDN ノードに到着したパケットはフローエントリに従い転送される。一度設定したフローは回線の廃止オーダまで消えることはない。従来の伝送パスの設定と同じ考え方といえる。

SDN ノードでは必要なフロー数に見合うフローテーブルのサイズが必要となる。

3.2. リアクティブのフロー・回線の特徴

リアクティブは OpenFlow 特有の制御方式である。回線を構成するフローは、ユーザからのサービスオーダ（利用申し込み）を受け、フロー設計の段階で OpenFlow コントローラが経路情報を把握しておくが、その時点では OpenFlow スイッチのフローテーブルへの設定は行わない。

フローテーブルへの設定は、最初のパケットが OpenFlow スイッチに到着した時点で行われる。到着したパケットは「table miss（該当するフローエントリが見つからない)」となり OpenFlow コントローラに対するパケットインが発生する。コントローラは予め保持するフロー情報と照らし合わせ、該当するフローエントリを各スイッチに対して設定する。

また指定された時間を越えてパケットの到着がない場合は、当該のフローエントリは OpenFlow スイッチにより消去される。

リアクティブではフローテーブルの使い回しができるので、フローテーブルのサイズが小さい場合に有効である。

4. フローと回線の種類

4.1. 基本回線と回線オプション

OpenFlow が具備する機能を利用することで複雑なパケット制御が可能となり、従来には無いような様々なネットワークサービスが提供される可能性がある。

本ガイドラインでは OpenFlow で実現可能なパケットの流れる経路に着目し、通信事業者として考えられる回線サービスを整理する。提供する回線サービスを“基本回線”と“回線オプション”として定義し、回線の種類の分類とサービスイメージを示す。

4.1.1. 基本回線

基本回線を以下の通りに定義する（図 2-8、図 2-9）。

- 回線はサービス提供の単位である。
- 回線は 1 つ以上のフローから構成される。
- 回線は 2 つの端点となるポート間で設定される。
- 同じポートに異なる回線が設定できる（多重アクセス回線）。
- 回線の種別として両方向、片方向がある。
- 両方向の回線の場合、両方向の 2 つのフローは同じ経路を通る。

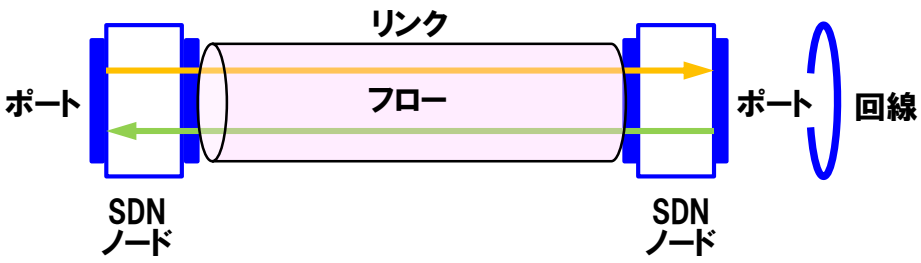

図 2-8　基本回線：両方向通信

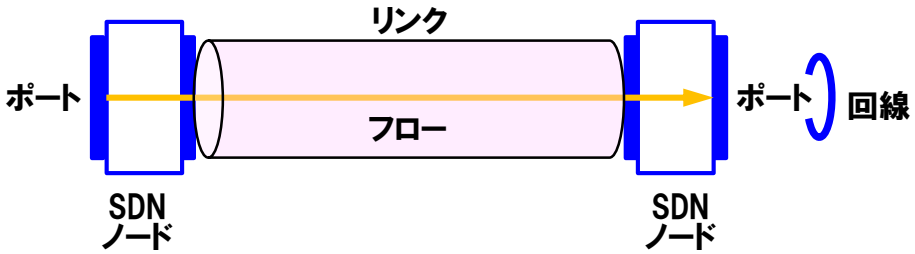

図 2-9　基本回線：片方向通信

4.1.2. 回線オプション

回線オプションを以下の通りに定義する。

- 基本回線に対して機能を追加するもの。
- 基本回線は、複数の回線オプションを選択できる。
  - オプションなし
  - 破棄（パケットは「null」ポートに到達して「Drop」されると考える）
  - 分岐
  - VLAN 付与　　など
- 両方向の回線では、両方向の 2 つのフローは異なる経路の場合がある。

4.2. 回線オプションの種類とサービスイメージ

回線オプションは基本的な回線サービス（基本回線）に対して機能を付加するものである。回線オプションの考え方とサービスイメージを示す。

4.2.1. VLAN を付与

ある SDN ノードにおいて、特定のパケットを Match 条件により識別し、VLAN タグを付与する。あるいは、VLAN タグが付与されたパケットを Match 条件により識別し、VLAN タグを除去する（図 2-10）。

図 2-10　VLAN を付与

4.2.2. パケットの経路を変更

ある SDN ノードにおいて、特定のパケットを Match 条件により識別し、フローが通過する経路を変更する。例えば利用用途による要求遅延を考慮し、遅延要求の高い通信は最短経路となるようパケットを振り分けるといったサービスが可能となる（図 2-11）。

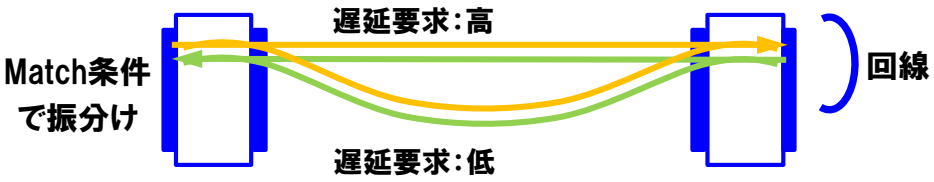

図 2-11　経路を変更

4.2.3. 指定通信のみ許可

ある SDN ノードにおいて、特定のパケットを Match 条件により識別し、条件に合うパケット（または合わないパケット）は破棄することにより、ACL（Access Control List）のようなサービスが提供可能となる。

OpenFlow の具体的な動作としては、OpenFlow スイッチで受信したパケットが「null」ポートで Drop されると考える。サービス構築のパタンとしては、エッジの SDN ノードで Drop する方法と（図 2-12）、経路の途中に Drop 用の SDN ノードを設ける方法が考えられる（図 2-13）。

図 2-12　特定通信のみ許可（1）

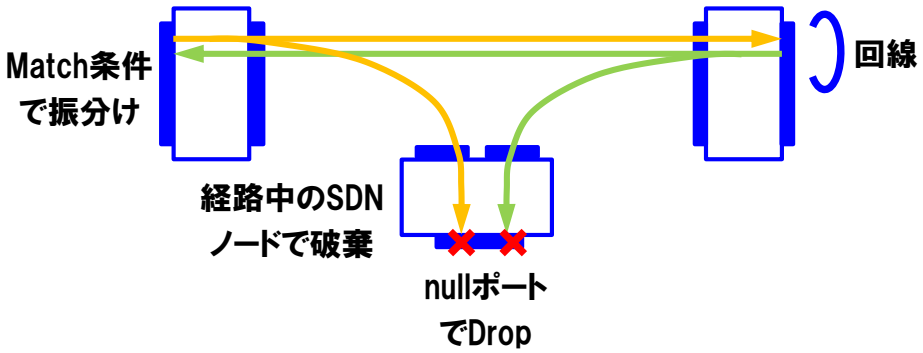

図 2-13　特定通信のみ許可（2）

4.2.4. パケットの宛先を変更

ある SDN ノードにおいて、特定のパケットを Match 条件により識別し、条件に合うパケット（または合わないパケット）の経路を分岐させる（図 2-14）。特定のパケットの宛先を変更することによりロードバランサのようなサービスが提供可能となる（図 2-15）。

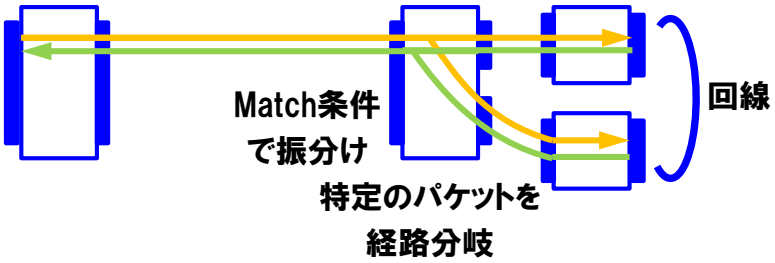

図 2-14　パケットの宛先を変更

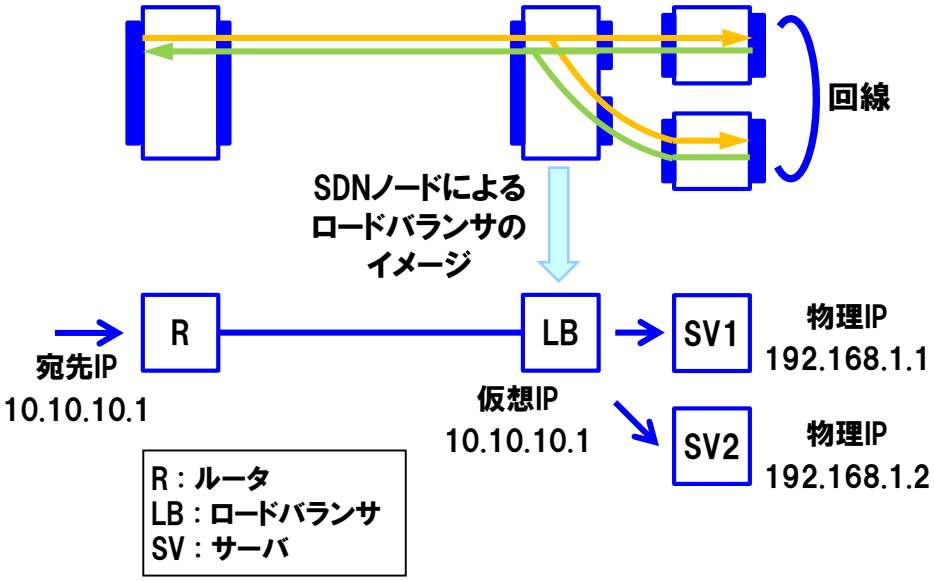

図 2-15　SDN ノードによるロードバランサのイメージ

OpenFlow の具体的な動作としては、OpenFlow スイッチの複数のポートを束ねる「Group」という機能を利用する。グループの処理方法（Type）として「Select」を指定することにより、グループ内からどれか１つのポートを選んで転送が可能となり、同じフローエントリのパケットを異なる経路に転送可能となる。

4.2.5. ネットワーク機能の挿入

ある SDN ノードにおいて、特定のパケットを Match 条件により識別し、条件に合うパケット（または合わないパケット）に対してネットワーク機能（DPI: Deep Packet Inspection 等）を付加することで、サービスチェイニングのような利用が可能となる。

サービス構築のパタンとしては、ネットワーク機能を挿入するための SDN ノードをフローの経路の途中に設ける方法と（図 2-16）、エッジの SDN ノードでネットワーク機能を挿入するための SDN ノードに振り分ける方法が考えられる（図 2-17）。

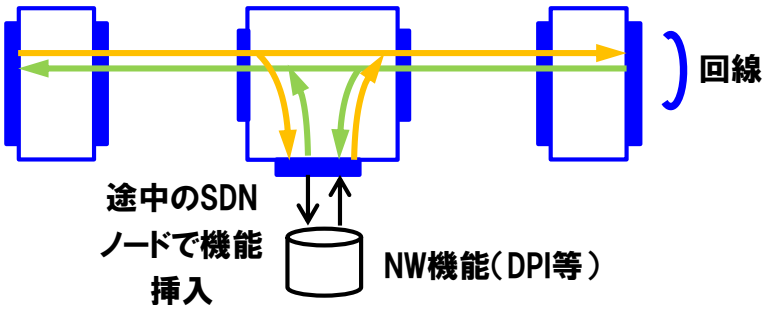

図 2-16　ネットワーク機能の挿入（1）

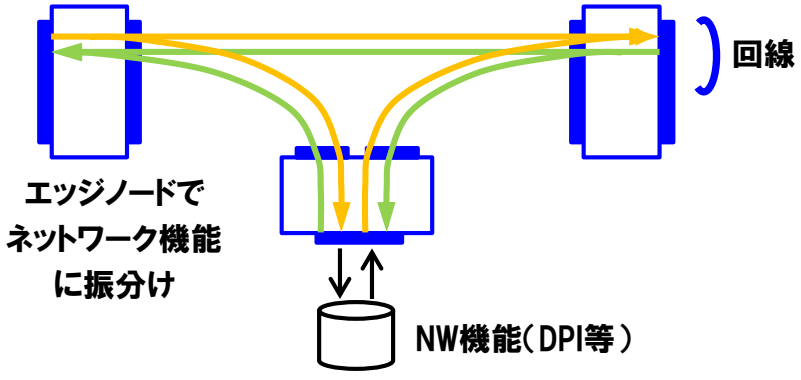

図 2-17　ネットワーク機能の挿入（2）

4.2.6. 冗長あり回線

基本回線に対してオプションとして冗長回線を設定する。以降に冗長回線の例として 3 つの方式を示す。なお、冗長化に関しては双方向で使用するものであるが、図では煩雑になるため片方向のフローのみを示している。

- 冗長あり回線（1:1）
- 冗長あり回線（1+1）
- 冗長あり回線（N:1）

(1) 冗長あり回線（1:1）

平常時に使用する基本回線（Act：アクティブ）に対して、バックアップ用の異経路を通る冗長回線（Sby：スタンバイ）を設定する。平常時は片系のみを使用し、Act の回線に異常が発生した際に、Act から Sby に回線を切り替えてサービスを継続する（図 2-18）。

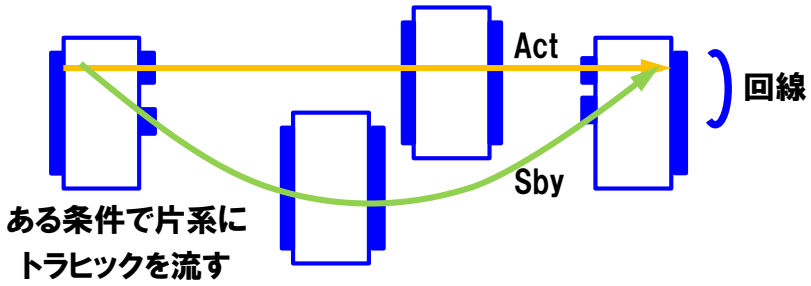

図 2-18　冗長あり回線（1:1）

冗長回線の設定方法としてリアクティブ方式を適用することも可能である（図 2-19）。リアクティブ方式の場合は冗長回線のフローエントリが SDN ノードに事前に設定されておらず、ある条件が発生した後にフローエントリが設定されるため、回線の切替えの際に時間を要する。

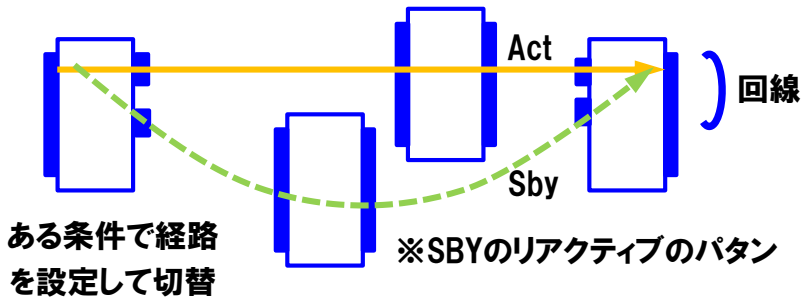

図 2-19　冗長回線がリアクティブのパタン

(2) 冗長あり回線（1+1）

基本回線（Act：アクティブ）に対して、平常時にも使用する異経路を通る冗長回線（Act：アクティブ）を設定する。この場合は平常時から両系の回線を使用するものとし、トラヒックはある条件で双方の回線に分岐する。通常時から両系の回線を使用することで、どちらか一方の回線に異常が発生した場合でもサービスを継続する（図 2-20）。

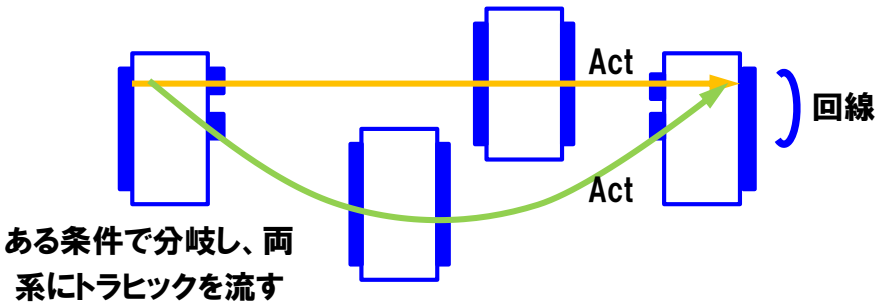

図 2-20　冗長あり回線（1+1）

(3) 冗長あり回線（N:1）

平常時に使用する複数（N 本）の基本回線（Act：アクティブ）に対して、共用となるバックアップ用の異経路を通る冗長回線（Sby：スタンバイ）を設定する。いずれかの回線に異常が発生した際に、Act から Sby に回線を切り替えてサービスを継続する。図 2-21 は N=2 本のアクティブ回線に対して、スタンバイの冗長回線を設定した例である。

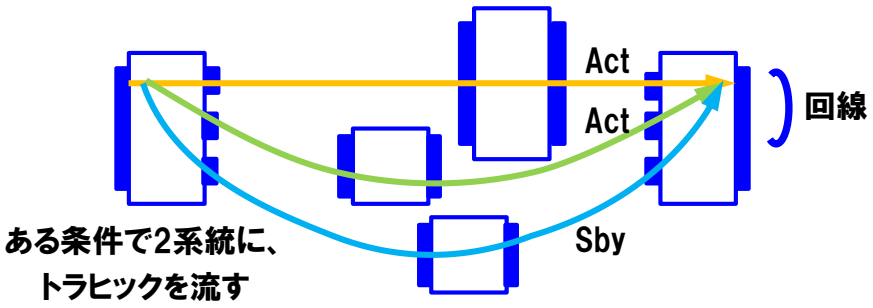

図 2-21　冗長あり回線（N:1）

4.2.7. 3 ポート以上を結ぶ回線

(1) 3 ポート以上を結ぶ回線の基本パタン

これまでの例では 2 ポート（2 拠点）間を Point-to-Point で結ぶ回線の例として解説をしてきた。本項では従来サービスと照らし合わせ、Point-to-Point から Point-to-Multipoint の回線への応用について考える。

3 ポート以上の場合については、2 ポート間の回線の組合せとして同様に考えることができる。具体的には 3 ポートの場合の基本パタンは、6 フロー3 回線の組合せとして考えることができる（図 2-22）。

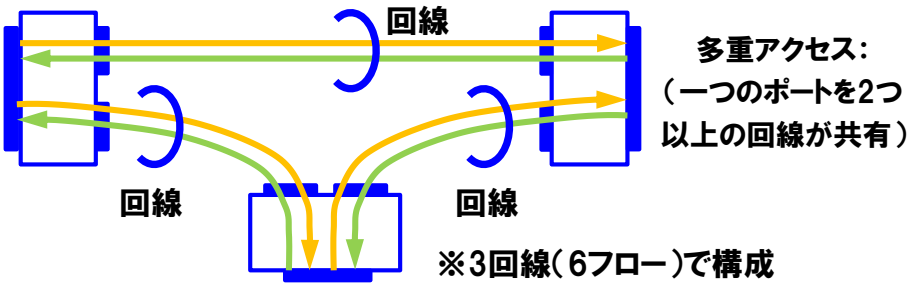

図 2-22　3 ポートを結ぶ 3 つの回線（基本パタン）

(2) 3 ポート以上の場合のマルチキャスト回線の例

3 ポート以上の場合に複数の宛先に対して同じデータを送信するマルチキャスト通信を提供する場合は、片方向の分岐する回線を宛先ポート数分追加することで、特定のパケットをマルチキャストすることができる（図 2-23）。

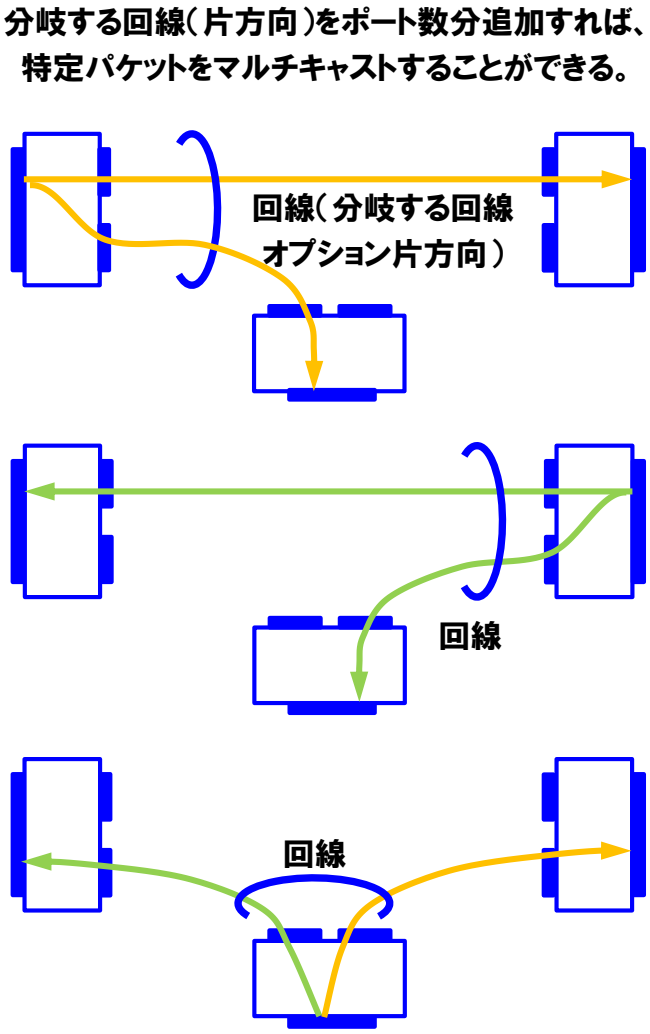

図 2-23　分岐する回線オプションによるマルチキャスト回線の例

5. ユーザフローの分離

5.1. ユーザ多重に関する課題

通信事業者が提供する回線サービスでは、同一のネットワーク上に複数のユーザを多重する利用形態が想定される。ユーザが契約する回線はフローで構成され、OpenFlow では各々のフローは Match 条件で識別されるが、異なるユーザが同じ l2/L3 アドレスや VLAN ID を使用している場合も想定される。異なるユーザが同じ情報を Match 条件として指定した場合や、スイッチのイングレスポート（IN_PORT）のみでフローを識別する場合、それらのフローは同一のパケット転送ルールと見なされ、回線個別のパケット制御が不可となる。またユーザに提供する回線の閉域性という観点からも問題である。

そのため通信事業者のネットワーク内では、Match 条件によるフローの識別の他に、ユーザのフローをネットワーク内でユニークに識別するための仕組みが必要となる。

5.2. ユーザフローの識別

通信事業者のネットワーク内でユーザのフローを識別するための仕組みとして、ユーザのフロー毎にネットワーク内でユニークとなるタグを付与する方法が考えられる。以降ではユーザフローの識別のために付与するタグを「網内タグ」と呼ぶ。

通信事業者のネットワークの入口（エッジノード）においてユーザのパケットを識別・分類し、別途管理するユーザのフローと網内タグの対応関係に従い、エッジノードにてパケットに網内タグの付与を行う。通信事業者のネットワーク内では網内タグに従い転送制御を行い、出口のエッジノードにおいて付与した網内タグの除去を行う。

OpenFlow ではタグの付加／削除は Actions における Push/Pop で制御が可能であり、使用するタグの種類としては、Ethernet(VLAN ID,PBB I-SID)、MPLS(Label)などが考えられる。但し、使用する OpenFlow スイッチによって対応しているタグが異なる場合があるため注意が必要である。

5.3. 網内タグの考え方

使用する網内タグのビット数（アドレス空間）がネットワーク内に設定できるフロー数に影響するため、収容するフロー数を考慮して適切なタグを選定する必要がある。参考として網内タグと設定可能フロー数の関係を表 2-6 に示す。

表 2-6　網内タグと設定可能フロー数（参考）

タグ	ビット数	設定フロー数
VLAN ID	12 ビット	4,096 フロー
Label(MPLS)	20 ビット	1,048,576 フロー
PBB I-SID	24 ビット	16,777,216 フロー

ユーザのフローをユニークに識別するための網内タグの付与の方法はいくつか考えられる。

図 2-24 は管理するネットワーク全体でユニークとなるよう、ユーザのフロー毎に網内タグを設定する方法である。ネットワーク内の 6 つのフローに対してユニークとなるよう、網内タグも 6 つの ID を付与して識別する。

この方法はネットワーク内で網内タグが一意に定まるため網内タグの管理は容易であるが、多数のフローを収容する場合は多数の網内タグを使用することになるため、設定可能フロー数に制約が生じる場合がある。

図 2-25 はリンク内でユニークとなるよう、ユーザのフロー毎に網内タグを設定する方法である。この場合は各 SDN ノードの IN_PORT の情報と合わせて網内タグの ID を付与して識別する。

この方法は各リンクで網内タグの ID の使い回しができるため、タグのアドレス空間を有効に活用することができるが、リンクを通過する毎に各ノードで網内タグの付け替え（Pop/Push）が発生すると共に、各リンク別にフローと網内タグの対応を制御する必要があり管理が複雑となる。

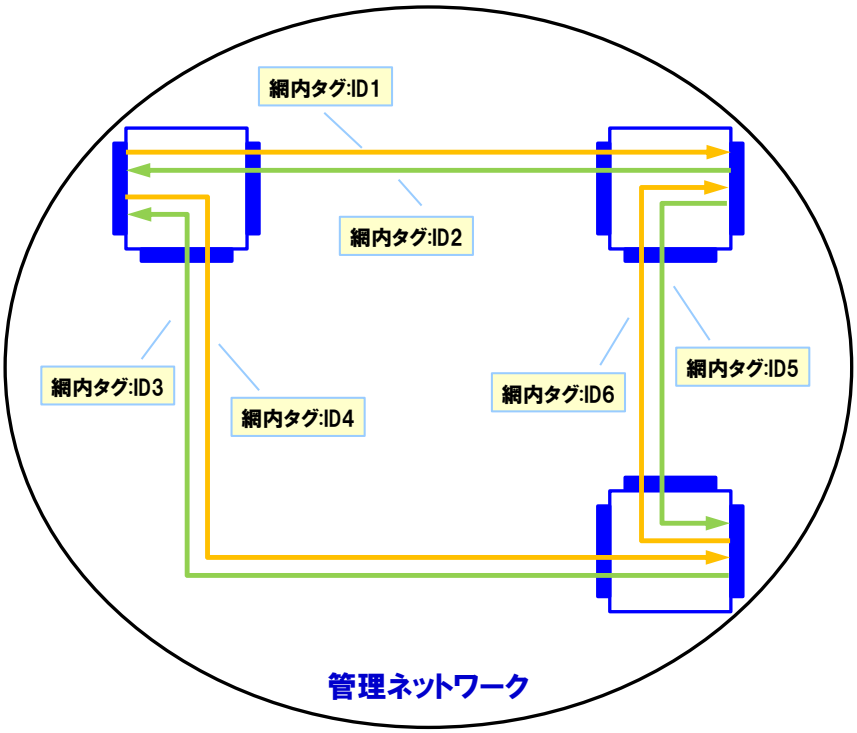

図 2-24　網内タグをネットワーク全体でユニークに付与

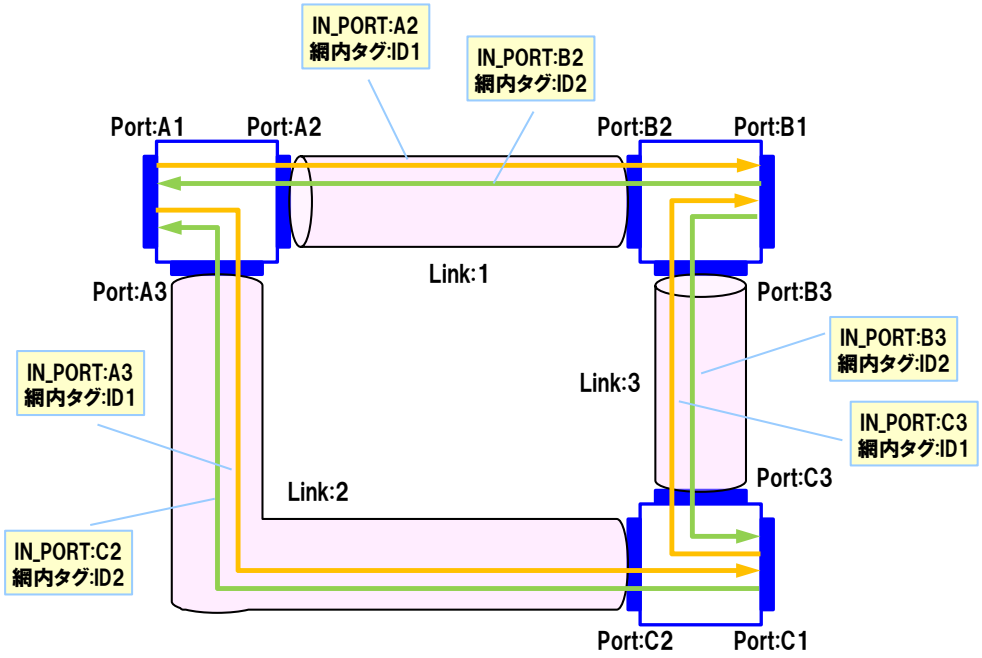

図 2-25　網内タグをリンク内でユニークに付与

6. 中継パスの適用

6.1. 中継パスの基本的な考え方

ネットワークに多数のユーザを収容した場合、多数の回線やフローを管理する必要があり、大量のパケットが通過するネットワーク内のコアノードでは、処理負荷の増加やフローエントリ数の制限が課題となる懸念がある。そのため、5 項に示すユーザフローの分離の考え方を応用して、複数の回線を集約して中継パスとして管理し、設計や運用の効率化を図る方法を考える。

具体的には宛先に応じて複数の回線を束ね、束ねた回線を中継パスとして管理し、中継パス単位に網内タグを再設定する。SDN ノードをエッジとコアで機能分担し、エッジノードは回線を識別して宛先に応じた中継パスに収容する（中継パス用の網内タグを付与する）。コアノードは中継パスを識別する網内タグのみをみて宛先のエッジノードまでパケットを転送する。

中継パスのイメージを図 2-26、図 2-27 に示す。

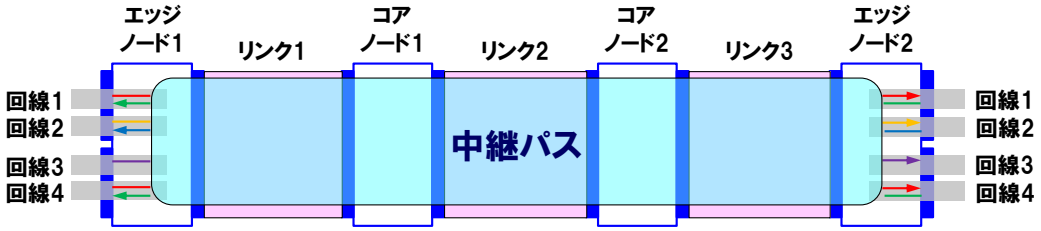

図 2-26　中継パスに回線を集約

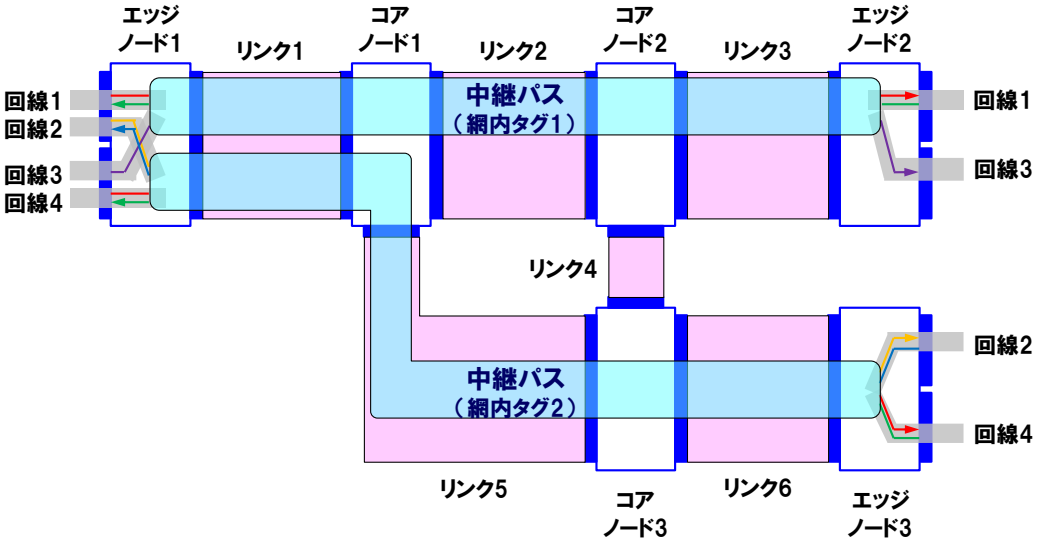

図 2-27　宛先別の中継パスを設定

6.2. 中継パス適用の効果

中継パスを適用した場合、コアノードは中継パスのみをハンドリングすればよく、コアノードの負荷軽減が期待できる。中継パスを適用した際のコアノードに対する効果を以下に示す。

- 中継パスを識別するタグのみをみて中継。
  - ➢ ノード装置の負荷軽減
- フロー集約によるフローエントリ節減。
- 中継パス単位の監視、保守、切り替えが可能。
  - ➢ 運用の効率化
- 処理が単純でありハード SW に向く。

第2節 情報の管理

1. 情報モデル

通信事業者が回線サービスを提供するにあたり、必要となる情報の管理の考え方について例を用いて示す。

通信事業者が管理する情報を、「ユーザ」「契約」「回線」「フロー」「ポート」「リンク」「ノード」のオブジェクトで構成されるモデルとして表現する（図 2-28）。「ユーザ」「契約」「回線」についてはサービスを管理するために必要な情報、「フロー」「ポート」「リンク」「ノード」についてはネットワークを管理するために必要な情報である。なお、本モデルはユーザに提供する回線を管理するための情報を示し、通信事業者が内部で使用する回線（管理用など）は考慮していない。

次項からは各オブジェクトについて解説する。

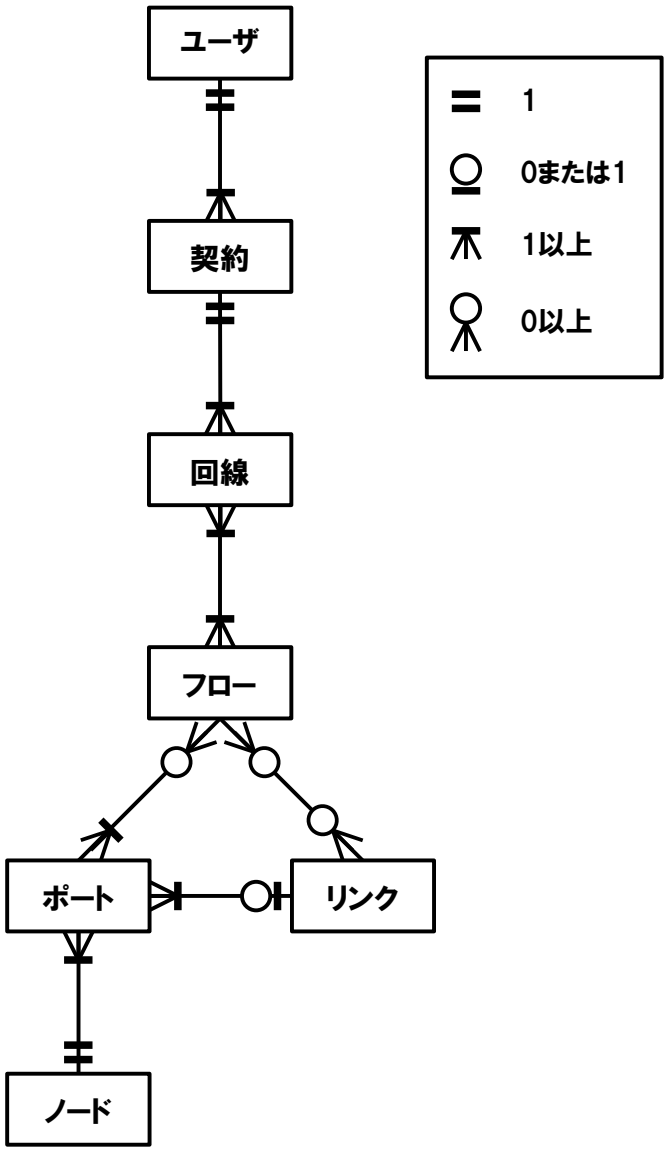

図 2-28　情報モデル

2. ユーザオブジェクト

ユーザは回線サービスの利用者である。契約のオブジェクトで構成される（図 2-29）。

1 ユーザに対して複数の契約情報が取り得る。

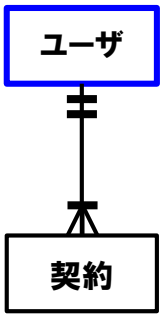

図 2-29　ユーザオブジェクト

3. 契約オブジェクト

ユーザは回線サービスを利用するために通信事業者と契約を結ぶ。ユーザ、回線のオブジェクトで構成される（図 2-30)。

1 契約に対して複数の回線情報が取り得る。

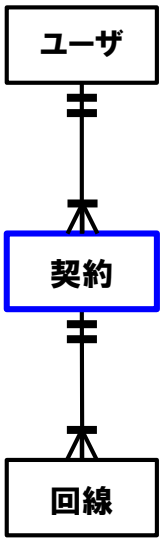

図 2-30　契約オブジェクト

4. 回線オブジェクト

回線は 1 つ以上のフローで構成される Point-to-Point の通信であり、通信事業者のサービスの提供単位である。契約、フローのオブジェクトで構成される（図 2-31）。

1 回線は複数のフロー情報で構成される場合がある。また 1 フローは複数の回線で使用する場合がある。

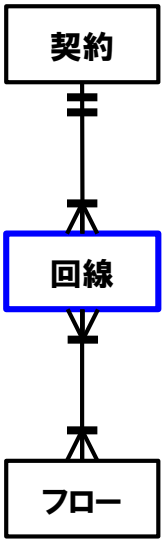

図 2-31　回線オブジェクト

回線オブジェクトがもつデータ属性の例を表 2-7 に示す。

表 2-7　回線オブジェクトのデータ属性の例

データ属性	概要
回線-ID	・ 回線にユニークに割り振られた識別子。
ロケーション	・ 回線の始点と終点を示すユーザの拠点を示す情報。
関連フロー	・ 回線を構成するフローを示す情報。
回線オプション	・ 回線オプションの有無や、オプションの内容を示す情報。
帯域	・ ユーザが契約した回線の帯域を示す情報。
関連契約	・ 回線情報と契約情報を紐付けるための情報。

5. フローオブジェクト

フローは回線を構成するノードとリンクを通過するパケットの通り道であり、End-to-End で片方向に設定される。回線、ポート、リンクのオブジェクトで構成される（図 2-32)。

フローと回線の関係は、1 フローは複数の回線で使用する場合がある。また 1 回線は複数のフローで構成される場合がある。

フローとポートの関係は、1 フローは複数のポートの情報で構成される。また 1 ポートには複数のフローが設定される場合があるが、フローが設定されていないポートも取り得る。

フローとリンクの関係は、1 フローはリンクなし、または複数のリンクの情報で構成される。リンクのないフローとは、ネットワーク間のタグの付け替え処理のみのフローなど、フローがノード内の IN_PORT と OUT_PORT で終端するような場合である（図 2-33)。また 1 リンクには複数のフローが設定される場合があるが、フローが設定されていないリンクも取り得る。

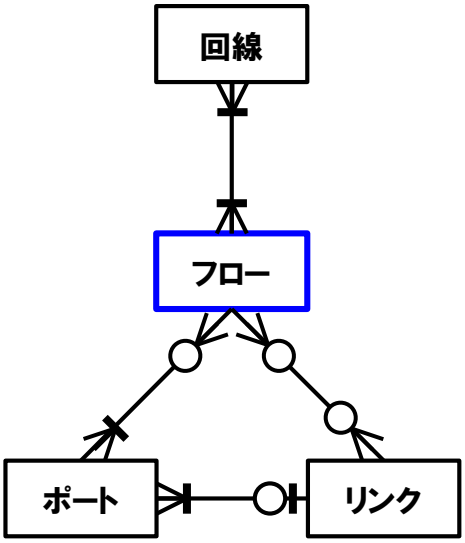

図 2-32　フローオブジェクト

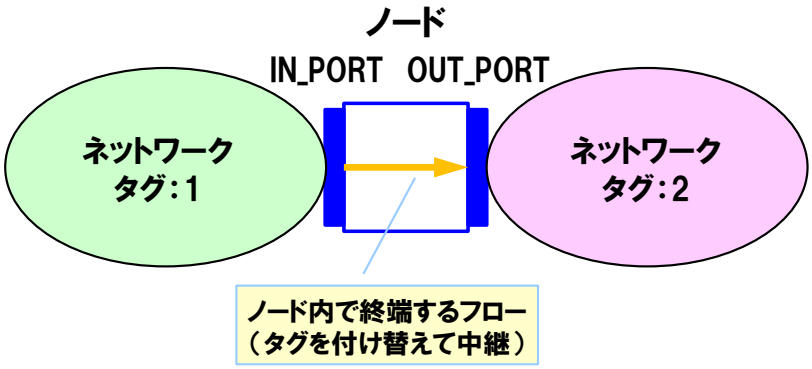

図 2-33　リンクのないフローの例

フローオブジェクトがもつデータ属性の例を表 2-8 に示す。

表 2-8　フローオブジェクトのデータ属性の例

データ属性	概要
フロー-ID	・ フローにユニークに割り振られた識別子。
経路情報	・ フローが通る経路を示す情報。 ・ 経由するノード、ポート、リンクで構成される。
Match 条件	・ パケットを識別するための Match 条件を示す情報。 ・ 条件と該当ノードの情報で構成される。
Meter 情報	・ フローの統計情報や、統計量に応じた制御方法を示す情報。
関連回線	・ フロー情報と回線情報を紐付けるための情報。

6. ポートオブジェクト

ポートはノードが 2 つ以上有してリンクを接続する。またフローの端点となる。フロー、リンク、ノードのオブジェクトで構成される（図 2-34）。

ポートとフローの関係は、1 ポートには複数のフローが設定される場合があるが、フローが設定されていないポートも取り得る。また 1 フローは複数のポートで構成される。

ポートとリンクの関係は、1 ポートに対してリンクが存在する場合としない場合がある。また 1 リンクは複数（2 つ）のポートで構成される。

ポートとノードの関係は、1 ポートに対して一意のノードが定まる。また 1 ノードには複数のポートが存在する。

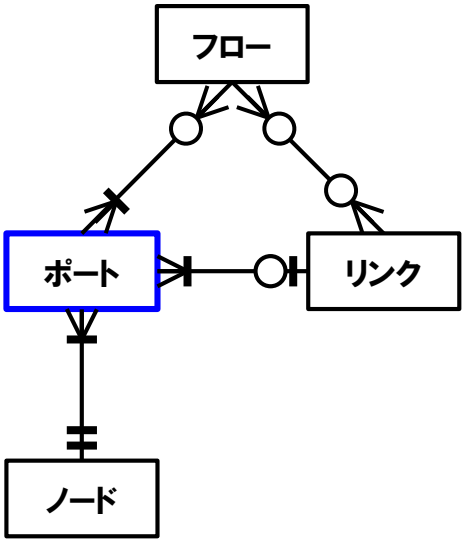

図 2-34　ポートオブジェクト

ポートオブジェクトがもつデータ属性の例を表 2-9 に示す。

表 2-9　ポートオブジェクトのデータ属性の例

データ属性	概要
ポート-ID	・ ポートにユニークに割り振られた識別子。
関連ノード	・ ポートが存在するノードを示す情報。
関連リンク	・ ポートが接続するリンクを示す情報。
関連フロー	・ ポート情報とフロー情報を紐付けるための情報。

7. リンクオブジェクト

リンクはノード - ノードのポート間を繋ぐもの (物理／論理) である。フロー、ポートのオブジェクトで構成される（図 2-35)。

リンクとフローの関係は、1 リンクに対して複数のフローが設定される場合があるが、フローが設定されていないリンクも取り得る。また 1 フローは複数のリンクで構成される。

リンクとポートの関係は、1 リンクは複数（2 つ）のポートで構成される。また 1 ポートに対してリンクが存在する場合としない場合がある。

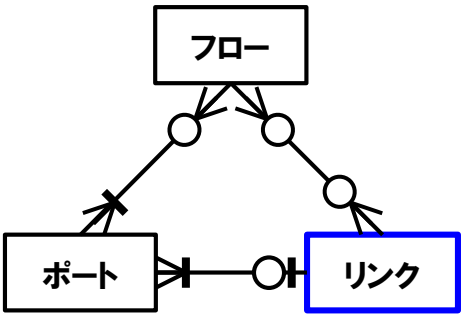

図 2-35　リンクオブジェクト

リンクオブジェクトがもつデータ属性の例を表 2-10 に示す。

表 2-10　リンクオブジェクトのデータ属性の例

データ属性	概要
リンク-ID	・ リンクにユニークに割り振られた識別子。
関連ポート	・ リンクが存在するポートを示す情報。
帯域	・ リンクの帯域を示す情報。
関連フロー	・ リンク情報とフロー情報を紐付けるための情報。

8. ノードオブジェクト

ノードはデータプレーン機能を搭載する SDN ノードであり、OpenFlow スイッチなどの SDN 対応スイッチが該当する。ポートのオブジェクトで構成される（図 2-36）。

1 ノードは複数のポートをもつ。また 1 ポートに対して一意のノードが定まる。

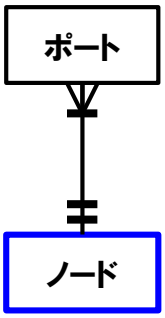

図 2-36　ノードオブジェクト

ノードオブジェクトがもつデータ属性の例を表 2-11 に示す。

表 2-11　ノードオブジェクトのデータ属性の例

データ属性	概要
ノード-ID	・ ノードにユニークに割り振られた識別子。
関連ポート	・ ノードに存在するポートを示す情報。
関連フロー	・ ノード情報とフロー情報を紐付けるための情報。

第3編 運用・監視フェーズ

SDN を用いたネットワークの運用・監視に関して、基本的には既存のネットワークの考え方と大きく変わることはない。マルチドメインかつマルチレイヤのネットワークの状態を常時把握し、異常発生時には、その検出と対処を行うことがポイントとなる。また下位レイヤを含めて、リソース管理や収容管理のデータが一元的に管理できれば、レイヤにまたがるリソース配分の最適化や、故障発生時の切り替えや被疑箇所推定などの運用の効率化が可能となる。

運用・監視フェーズの詳細については、今後の課題とする。

第1章 運用・監視フェーズの基本的な考え方

運用・監視フェーズの基本的な考え方については、今後の課題とする。

第2章 SDN NW の運用・監視

SDN NW の運用・監視については、今後の課題とする。

第3章 仮想 NW の運用・監視

仮想 NW の運用・監視については、今後の課題とする。

第4編 付属資料

第1章 用語の定義

本ガイドラインで定義した主な用語について表 4-1 に整理にする。

表 4-1　用語の定義

用語	定義
第 1 編 第 3 章 SDN を用いたネットワークモデル	
仮想 NW	・ SDN NW 上に構成される L2 以上の論理ネットワーク。
SDN NW	・ SDN ノードと SDN ノード間のリンクで構成され、SDN コントローラから制御されるネットワーク。
Optical/Transport NW	・ SDN ノード間にリンクを提供するネットワーク。
第 2 編 第 3 章 フローと回線の定義	
フロー	・ リンク及び SDN ノードを通過するパケットの通り道。 ・ End-to-End のポート間に片方向で設定。
回線	・ ユーザに対してネットワークサービスを提供する単位。 ・ 1 つ以上のフローで構成。
基本回線	・ ユーザに提供する基本的な回線。
回線オプション	・ 基本回線に対して機能を追加するもの。

第2章 参照資料の一覧

本ガイドラインで参照している主な資料について整理する。

- 第 1 章
  - NTT 持株会社ニュースリリース「世界最高性能の SDN ソフトウェアスイッチをオープンソースソフトウェアとして公開」

    http://www.ntt.co.jp/news2014/1406/140606a.html
  - O3 プロジェクト ホワイトペーパー

    http://www.o3project.org/ja/download/document/O3whitepaper_ja.pdf
  - O3 発表資料 O3 シンポジウム 2014「ネットワークビジネスを変革する最新（世界初）ソフトウェアテクノロジ」

    http://www.o3project.org/ja/download/index.html
  - O3 プロジェクト OSS のダウンロード
    『O3 Orchestrator Suite(ODENOS)』（SDN 共通制御フレームワーク）

    http://www.o3project.org/ja/download/index.html
- 第 2 章
  - Ryu Certification

    http://osrg.github.io/ryu/certification.html