詳解 VMware NSX

ネットワーク仮想化の基礎と応用

VCP-NV
資格取得の必携書

ヴイエムウェア株式会社
進藤資訓+水本真樹+田中 洋+横井智樹+髙田和美+小椋康平[共著]

プラットフォームとアーキテクチャを徹底解説!

OpenFlow、OpenStack、SDDCなど、
技術背景から設計・運用まで体系的にわかる。
ヴイエムウェア社の第一線エンジニアによる
待望の日本語ドキュメント。

インプレス

■本書の内容と商標について

・本書の内容は、2014 年 12 月の情報に基づいています。記載した URL やサービス内容などは、予告なく変更される可能性があります。
・本書の内容によって生じる直接的または間接的被害について、著者ならびに弊社では一切の責任を負いかねます。
・本文中の社名、製品・サービス名などは、一般に各社の商標または登録商標です。本文中に©、®、™ は表示していません。

SDN and Network Virtualization have changed networking. Yet to date, there has not been a comprehensive treatment of NSX which is a leading product in the space. This is that book. It is a definitive first source on NSX and provides the context that can only be provided by the team who helped make NSX's success a reality. If you read one book on the subject, this should be it.

VMware, Inc. VMware Fellow
Network & Security Business Unit SVP and General Manager
Martin Casado

【日本語訳】

SDNとネットワークの仮想化はネットワークに変革をもたらしました。この分野を牽引するNSXについて包括的にまとめたものはいままで存在しませんでしたが、本書はまさにそのような一冊です。本書はNSXに関するはじめての信頼のおける情報源であり、それはNSXを成功に導いているメンバーによってのみ書き得るものだと思います。この分野の本をなにか一冊読みたいのなら、この一冊で決まりでしょう。

VMware, Inc. VMwareフェロー
ネットワーク・セキュリティビジネス部門担当　上級副社長兼ゼネラルマネージャ
マーティン カサド

はじめに

2011年頃、ネットワーク構成をソフトウェア制御する「OpenFlow」に注目が集まり始め、ネットワーク世界に新しい流れが生まれました。それは「SDN（Software-Defined Networking)」というより大きな概念となり、現在さまざまな製品やサービスが市場に流通しています。

一方、「クラウド」と呼ばれるようになったサービスも本格的に普及し始め、IT環境の提供モデルが大きく変貌しました。多様な形態のサービスが出現し、今までのデータセンターのあり方が根本から変わろうとしています。

2012年8月、ヴイエムウェアはネットワーク仮想化のリーダー的存在であったニシラを買収しました。既にサーバ仮想化をリードしていた同社は、その技術を余すことなく取り込み、2013年8月にネットワーク仮想化ソフトウェア「NSX」をリリースしました。

NSXは多くのユーザーに認められ、利用されています。しかし、これらのまとまった技術情報は日本語はもちろん、英語の情報でさえ少ないのが現状です。

このような状況を打開したいという強い思いから、ヴイエムウェアの第一線エンジニアが立ち上がりました。

本書は、6人のNSXエンジニアによって執筆されています。各人の得意分野を活かし、ネットワーク仮想化の技術背景・周辺知識から、NSXのアーキテクチャ、OpenStack連携に至るまで、非常に幅広く解説しています。

この10年で、サーバ仮想化がIT基盤として必要不可欠になったように、今後、ネットワーク仮想化もなくてはならないものになります。今までのインフラエンジニアにとって、ネットワーク仮想化を理解することは大変かもしれません。本書がそのようなエンジニアの理解に少しでも役立ち、NSXをより身近なものに感じていただければと願います。

2014年10月

筆者一同

■本書の対象読者

本書は、ヴイエムウェア株式会社が提供するネットワーク仮想化ソフトウェア「NSX」の技術知識を深めたいユーザーを対象にしています。前提として、IPルーティングやスイッチの動作など、基本的なネットワーク知識が必要となります。

■本書の構成

Chapter 1 技術背景と定義

ネットワークアーキテクチャの歴史と課題、OpenFlowをはじめに、SDN、ネットワーク仮想化の概要を説明します。

Chapter 2 標準化とメリット

ネットワーク仮想化の標準化状況とメリットを解説します。

Chapter 3 既存ネットワークの課題

既存ネットワークに関する技術的な課題の詳細を解説します。

Chapter 4 ネットワーク仮想化のAPI

ネットワーク仮想化で利用されるAPIの背景と詳細を解説します。

Chapter 5 NSXの技術解説

NSXプラットフォームの「NSX for Multi-Hypervisor」と「NSX for vSphere」に関して、アーキテクチャやコンポーネントを解説します。

Chapter 6 OpenStackとNeutron

クラウド管理プラットフォームとして、OpenStackとNSXとの連携を概要から詳細に至るまでを解説します。

Chapter 7 SDDCとNSX

NSXプラットフォームが生み出す、さまざまなユースケースやインテグレーション、SDDCコンセプトにおけるNSXの位置付けを解説します。

マーティン・カサドから …… iii

はじめに …… i

Chapter 01 技術背景と定義 田中 洋 …… 001

1-1 ネットワーク仮想化とは？ …… 002

1-1-1 ITインフラ基盤における仮想化の歴史 …… 003

1-1-2 サーバ仮想化の躍進 …… 006

1-2 ネットワーク仮想化の歴史 …… 008

1-2-1 ネットワーク技術の発展 …… 009

1-2-2 ネットワークベンダーにおける技術開発 …… 014

1-3 既存ネットワーク仮想化の問題点 …… 017

1-3-1 ネットワークの複雑化と課題 …… 017

1-3-2 クラウドの躍進に伴う新たな課題 …… 018

1-4 SDN と OpenFlow …… 023

1-4-1 SDN の成り立ち …… 023

1-4-2　OpenFlowの成り立ち …… 028
1-4-3　OpenFlowを利用したネットワークアーキテクチャ …… 032
1-4-4　SDNとOpenFlowの関係 …… 037
1-4-5　ハードウェア定義かソフトウェア定義か？ …… 039

1-5　ヴイエムウェアのネットワーク仮想化 …… 042
1-5-1　ネットワーク仮想化の定義 …… 042
1-5-2　ネットワーク仮想化で解決される問題 …… 046

Chapter 02　標準化とメリット　横井 智樹 …… 049

2-1　ネットワーク仮想化の標準化状況 …… 050
2-2　ネットワーク仮想化によるメリット …… 053
2-2-1　ビジネス上のメリット …… 053
2-2-2　システム構成上のメリット …… 058
2-2-3　ネットワーク仮想化に伴う変化 …… 063

2-3　ネットワーク仮想化が変えるインフラ基盤 …… 065
2-3-1　インフラ基盤におけるオープンプラットフォーム …… 065
2-3-2　オープンプラットフォームとエコシステム …… 066

2-3-3　クラウド化とともに変わるインフラエンジニアの役割 …… 068

2-3-4　求められるインフラ基盤のデリバリースピード …… 070

2-3-5　DevOpsに対応できるインフラ基盤 …… 073

Chapter 03　既存ネットワークの課題　田中 洋 …… 079

3-1　既存ネットワークと仮想化技術 …… 080

3-1-1　OSI参照モデル …… 081

3-1-2　既存ネットワークの管理性 …… 083

3-2　VLAN 問題 …… 086

3-2-1　アクティブVLAN …… 086

3-2-2　MACアドレステーブルの上限値 …… 087

3-2-3　VLAN識別上限値の問題 …… 088

3-2-4　レイヤ2要件 …… 092

3-3　インフラの要件とネットワーク基盤の複雑性 …… 096

3-3-1　ロジカルポート …… 098

3-3-2　ACL …… 099

3-3-3　ファイアウォール …… 100

3-3-4　ファイアウォールルール …… 101

3-3-5　仮想ルーター …… 102
3-3-6　仮想ロードバランサー …… 103

3-4　セキュリティアーキテクチャ …… 104
3-4-1　ゼロトラストモデル …… 104
3-4-2　境界セキュリティ …… 105

3-5　インフラ基盤のライフサイクル …… 109
3-5-1　計画（Plan） …… 109
3-5-2　設計（Design） …… 110
3-5-3　実装（Implementation） …… 110
3-5-4　運用（Operation） …… 110
3-5-5　インフラ基盤ライフサイクルへの考察 …… 111
3-5-6　PoC実施の課題と解決策 …… 112

Chapter 04　ネットワーク仮想化の API　横井 智樹 …… 115

4-1　WEB API …… 116
4-1-1　XML-RPC …… 117
4-1-2　SOAP …… 117
4-1-3　RESTful API …… 118

4-2　従来のネットワーク基盤における API 121

4-2-1　NETCONF 122

4-2-2　NETCONF 未対応のケース 125

4-3　ネットワーク仮想化基盤における API 127

4-4　ネットワーク仮想化基盤での API の利用例 129

4-4-1　API 経由での設定シナリオ 129

Chapter 05　NSX の技術解説　進藤 資訓、髙田 和美 139

5-1　NSX のアーキテクチャ概要 140

5-2　NSX 各種コンポーネント 143

5-2-1　コントローラクラスタ 143

5-2-2　トランスポートノード 148

5-2-3　ハイパーバイザーと Open vSwitch 148

5-2-4　Service Node 152

5-2-5　Gateway 155

5-2-6　L2 Gateway 156

5-2-7　L3 Gateway 158

5-2-8　NSX Manager …… 159

5-2-9　システム要件 …… 163

5-3　NSXの機能と概念 …… 168

5-3-1　論理スイッチ …… 168

5-3-2　論理ポート …… 168

5-3-3　アタッチメント …… 171

5-3-4　論理ルーター …… 172

5-3-5　トンネル …… 177

5-3-6　トランスポートゾーン …… 186

5-3-7　QoS …… 187

5-3-8　Security Profile …… 189

5-3-9　ACL …… 191

5-3-10　Gateway Service …… 193

5-3-11　ポートアイソレーション …… 195

5-3-12　ポートセキュリティ …… 196

5-3-13　ポートミラーリング …… 197

5-4　NSX for vSphere 技術解説 …… 199

5-4-1　機能とコンポーネント …… 199

5-4-2　システム要件 …… 203

5-4-3　NSX環境の導入と利用 …… 205

5-5　NSXが提供する機能 …… 208

5-5-1　論理スイッチ …… 208

5-5-2　論理ルーター …… 223

5-5-3　論理ファイアウォール …… 241

5-5-4　論理ロードバランサー …… 246

5-5-5　論理VPN …… 248

5-5-6　物理環境への接続 …… 254

5-5-7　その他、各種サービス機能 …… 259

Chapter 06　OpenStackとNeutron　水本 真樹、小椋 康平 …… 261

6-1　OpenStack …… 262

6-1-1　Neutronネットワークサービスの意義 …… 263

6-1-2　ネットワーク仮想化とOpenStack連携のメリット …… 264

6-1-3　インスタンスとネットワークのアタッチ …… 268

6-1-4　インスタンス起動時のL2/L3ネットワークサービスのワークフロー …… 270

6-2　多階層アプリケーション環境の構成 …… 276

6-2-1　多階層アプリケーション環境の構成手順とOpenStack環境の概要 …… 276

6-2-2　論理ネットワークの作成 …… 278

6-2-3　セキュリティグループの設定 …… 284

6-2-4　LBaaSの設定 …… 293

6-2-5　FWaaSの設定 …… 300

6-3　OpenStackとNeutron …… 304

6-3-1　OpenStackの主要なコンポーネント …… 305

6-3-2　Neutronの歴史 …… 308

6-3-3　Neutron とNSX プラグインの機能 …… 309

6-3-4　プラグインの開発 …… 316

6-3-5　Neutron プラグインの比較 …… 318

6-4　NeutronとNSXの連携 …… 320

6-4-1　プロバイダネットワークとルーター作成 …… 320

6-4-2　Neutronのインストール …… 322

6-4-3　Neutron のインストール手順 …… 322

6-4-4　NSXプラグインインストール後の設定手順 …… 330

6-4-5　NSXの管理機能の初期設定概要 …… 338

6-4-6　OpenStackとNSXを組み合わせた際の利点 …… 339

6-5　OpenStackの構成例と運用 …… 344

6-5-1　OpenStackの試験環境のインストール …… 344

6-5-2　サーバ、ネットワーク構成のノウハウ

（リファレンスアーキテクチャ、最小、典型的な構成） …… 347

6-5-3　Open vSwitchの運用とESXiの仮想スイッチNSX vSwitch …… 349

6-5-4　運用とトラブル対応例 …… 350

6-6　VMware OpenStack ソリューション …… 366

6-6-1　VMwareが提供するクラウド基盤のコンポーネント …… 366

6-6-2　vSphere ESXiハイパーバイザー用ドライバ（vCDriver） …… 367

6-6-3　Cinder VMDK - Virtual SAN（VSAN）の構成 …… 371

6-6-4　vSphereコンポーネントの運用と設計例 …… 375

6-6-5　ヴイエムウェアのOpenStackへの取り組み …… 377

6-6-6　ヴイエムウェアのテクノロジーとOpenStack …… 379

Chapter 07　SDDC と NSX　水本 真樹 …… 387

7-1　NSX のユースケース …… 388

7-1-1　NSXにより解決する問題ともたらすメリット …… 390

7-1-2　セキュリティのユースケース …… 391

7-1-3　クラウドのユースケース …… 394

7-1-4　エンタープライズのユースケース …… 400

7-2　NSX の設計と構築 …… 403

7-2-1　ステップ1 - L2/L3ファブリックのデプロイ …… 403

7-2-2　ステップ2 - NSX Manager ＆ NSX Edgeの配置 …… 422

7-2-3　ステップ3 — リソースの使用 …… 431

7-3　NSXのネットワークセキュリティサービス …… 439

7-3-1　VMware NSXのネットワークセキュリティサービス概要 …… 439

7-3-2　NSXのセキュリティポリシー …… 443

7-3-3　分散ファイアウォールとファイアウォールサービス連携 …… 445

7-3-4　ロードバランサーサービス …… 452

7-3-5　ハードウェアVTEPとのOVSDB連携 …… 453

7-3-6　VPN接続 …… 458

7-4　VMware NSXエコシステム …… 459

7-4-1　ネットワーク・セキュリティの統合ソリューション …… 460

7-4-2　パートナーアプライアンスのサービスと連携 …… 461

7-4-3　パートナーソリューションの登録 …… 465

7-5　NSXの運用 …… 470

7-5-1　ネットワークの自動化と監視 …… 470

7-5-2　各コンポーネントの管理ツール …… 472

7-5-3　ネットワークサービスの動作確認 …… 475

7-5-4　データ管理 …… 477

7-5-5　トラブルシュートの手順とツール …… 479

7-6　VMware NSX と SDDC …… 485

7-6-1　NSX により解決されるデータセンターの課題 …… 485

7-6-2　クラウド管理プラットフォーム

- vRealize Automation Center と OpenStack …… 487

7-6-3　サービスプロバイダやエンタープライズクラスのサポートモデル …… 494

索引 …… 496

謝辞 …… 510

著者プロフィール …… 511

Chapter 01

技術背景と定義

田中 洋

1-1 ネットワーク仮想化とは？

ITインフラ基盤における歴史的に最も大きな変革は、サーバ仮想化であることに間違いはありません。x86サーバのCPUやメモリなど、リソースを抽象化してプール化し、仮想マシンの配置やITインフラ管理の大部分を自動化することに成功しています。ITサービスの拡大はクラウドに発展し、この変革はオープン化の波との相乗効果により大きく加速している状況です。

ネットワーク仮想化も上述のサーバ仮想化と同じか、それ以上のインパクトをもたらす重要な技術と考えられています。ネットワーク仮想化そのものがもたらすさまざまな価値のみならず、ネットワーク関連のハードウェア製品やサーバをはじめ、ITインフラ基盤のコンポーネントを提供している業界のトップベンダーにも変革を促すからです。

ここ数十年で最も大きな変化といえ、そして新たなオープンネットワーキングとも呼べる潮流が押し寄せています。何故ならこの変化は、大手サービス事業者のデータセンターでは既知となっています。クラウド時代の幕開けと共にネットワーク仮想化に端を発した、ITインフラ基盤の大きな変化は、多くの人の目に触れない場所で既にもたらされているのです。

これまではネットワーク仮想化が世の中を変えることはなかったのでしょうか。

実はネットワーク仮想化と呼ばれた技術は、過去にも存在していました。しかし、VLANなど個別リソース活用の域を出ず、外部からのコントロール性にも乏しかったため、リソースを一元的に扱うには極めて不向きでした。

また、ネットワーク自体も歴史的に成功を収めたインターネット向けのアーキテクチャや、自律分散型のハードウェア機器を中心としたプロトコル互換性の問題に縛られていたため、データセンター中心の時代から取り残されがちでした。一度成功を収めたネットワークアーキテクチャが長期間にわたり見直されないなど、サーバ仮想化によるコンピュートリソースの集約化と比較して、明らかにアンバランスな構成が存在してきました。

こうした状況下でも、実際に多くのネットワークベンダーの製品開発は、ユーザーが求めるオープン化への要求に応える方向に舵を取ろうとはしませんでした。そのため、これに異を唱えた一部の大規模なユーザーは、独自にネットワークのオープン化を進めていったのです。しかし、ITインフラ基盤を構成し運用している数多くのユーザーは、データセンター中心の時代になっても、サーバ仮想化やオープン化の恩恵を受けられない状態が続きました。

そこに突如ムーブメントをもたらしたのが、2012年のSDN[※1]とOpenFlow[※2]です。ヴイエムウェアは、Open vSwitchなどのオープンネットワーキングと組み合わせることで、実現性の高い製品レベルまで高めてイノベーションを起こし、ネットワーク仮想化と呼ばれる新しい分野を生み出しました。そして、今後さらに大きな変化をもたらすと期待されています。

しかし、残念ながらネットワーク仮想化に関する資料は、日本語はもちろん英語でも潤沢といえる状況ではありません。本書の目的は、ネットワーク仮想化の課題と背景、VMware NSXアーキテクチャ、そして各種ユースケースや使用方法に関わる情報を幅広く提供することです。

本章では、上記のネットワーク仮想化のコンセプトとアーキテクチャにフォーカスして、サーバ仮想化やネットワーク技術の歴史を通して、最近のクラウド環境における既存ネットワークの問題点、OpenFlowをはじめSDNやネットワーク仮想化の定義を解説します。

1-1-1 ITインフラ基盤における仮想化の歴史

最古の仮想化技術の1つといえば、マルチユーザーアーキテクチャやタイムシェアリングなどが該当します。これらはCPUサイクルのシェアが目的で、限られたコンピューティングリソースをどのように共有して利用するか、その方法が考えられ使用されてきました。本項では、これらの技術を考えつつ、ITインフラ基盤の歴史を振り返ります。

1980年頃まではIBMや国産のメインフレームが全盛で、続いてDECなどのミニコンの流れが出現し、1990年代に入ると、UnixやWindowsなどのクライアント・サーバモデルのアーキテクチャが主流となりました。1990年代～2000年代は、ホストコンピューターがすべてを処理するアーキテクチャではなく、クライアント側がある程度の計算処理を実行するアーキテクチャが安価で最も高性能となりました。クライアント・サーバモデルのアーキテクチャが全盛期となり、多種多様なアプリケーションがクライアント向けに開発されました。

しかし、2000年代中盤以降となると、クライアント・サーバモデルのアーキテクチャから、Webサービスを中心に大規模サービスを利用する形式へと変遷していきます。メールや検索サービスなど、多数のユーザーを抱えるWebサービスで、多くのベンダーが従来と比較すると非常に低価格もしくは無料のサービスをリリースしました。例えば、Google社が提供する「Gmail」

※1 SDN = Software-Defined Network, Software Defined Networkingの略。ソフトウェアで定義されるネットワーク

※2 「1-4-2 OpenFlowの成り立ち」で後述する、スイッチやルーターのフォーワーディングプレーンにアクセスしさまざまな制御を可能にするプロトコル。ソフトウェアでの実装が特徴で、ローカルにデータベースを持つエージェント（仮想スイッチ、もしくは物理スイッチ）とコントローラが存在する。OSS（Open Source Software、オープンソースソフトウェア）の実装として「Open vSwitch」などがある。

も当初は他のメールサービス同様、50MB 程度のストレージ容量でした。しかし 2004 年には一挙に 1GB 容量を提供することを発表し、リリースされています。実に 20 倍におよぶストレージ容量の拡充です。

大規模に展開されるサービスを多数のユーザーが利用する Web サービスの登場はクラウド時代の幕開けといえます。これ以降、大規模クラウド型サービスとモバイルなどの軽量クライアントの組み合わせが、現在のアーキテクチャを形成しています。

IT アーキテクチャは、集中型（メインフレーム時代）から分散型（クライアント・サーバー時代）へ、そしてまた、集中型（クラウド時代）への移行を繰り返しています。

なお、再び集中型アーキテクチャが流行した背景には、サーバの仮想化によって Unix などの専用ハードウェア・専用 OS を使用しなくても、x86 サーバとハイパーバイザーを用意すれば手軽にサーバ基盤を用意できるようになったことや、1 つの物理サーバで多数のサービスを搭載可能となった状況などがあります。また、ムーアの法則下における x86 サーバのさらなる低価格化で、大量リソースを用意することが可能になったことなども要因に挙げられます [※3]。さらに、Linux の一般化や分散技術を利用したミドルウェアの発展などさまざまな背景が関連しています。

一方、ネットワーク分野（もしくはインターネット分野）における仮想化は、回線やセグメン

表 1.1 コンピューティング・ネットワーキングアーキテクチャの変遷

カテゴリー	～1990 年	1990 年代	2000 年代	2010 年代
コンピューティング				
アーキテクチャ	集中型	分散型	分散型	集中型
OS	専用	汎用 Unix/Windows	汎用 Unix（Linux）/Windows	汎用 Linux/Windows
ハードウェア	専用	専用	汎用（x86）	汎用（x86）
ネットワーキング				
アーキテクチャ インターネット （例：OSPF、BGP）	自律分散型	自律分散型	自律分散型	自律分散型
アーキテクチャ データセンター （例：ISL、802.1q 各社 TRILL ベースなど）	専用	専用	専用・汎用混在	専用・汎用混在
OS	専用	専用	専用	専用・汎用 （OpenFlow）※4
ハードウェア	専用	専用	専用	専用・汎用 （OpenFlow）※4

※3　ムーアの法則：Intel CEO のゴードン・ムーアが提唱した、CPU の性能は 18 ヶ月に 2 倍になるという法則。
※4　OpenFlow は全体のパーセンテージではまだ極少数。

ト（ネットワーク）を対象としたリソース仮想化として実装されてきました。その意味では、コンピューター業界と同様に、古くから仮想化が存在しています。

ネットワークベンダーは、ネットワーク機器に1990年代～2010年代まで一貫してネットワークベンダー固有のハードウェアアーキテクチャ上に、IEEEやIETF等によって定められた各種ベンダー中心で策定された標準化プロトコルを実装する形で製品を発展してきました。IEEEやIETFなどは標準化を行っている非営利の団体であり、その意味では、団体もプロトコル標準化の方法もオープンといえます。しかし、ルーターやスイッチなどの実際のネットワーク機器は、相互接続性やプロトコルの実装、またOSIの低レイヤで独自のプロトコルと実装が残っているなど多くの課題があり、簡単に異なるベンダーの機器でも接続できるわけではありません[※5]。

前表から分かる通り、ネットワークアーキテクチャは、専用ハードウェアと専用OSの上に単一のアーキテクチャを各ネットワークベンダーの中で競争を行い、実装するプロトコルに関しては標準化して、搭載してきたといえます（表1.1）。これはコンピューター業界とは異なり、一貫して分散型のアーキテクチャトレンドといえ、このトレンドは2010年代に入ってもこの傾向は変わっていません。

また、ネットワーキング、特にIPルーティングの分野では、自律分散アーキテクチャが基本です。自律分散アーキテクチャでは、各ネットワーク機器がルーティング情報を独立して保持し、それぞれが独自にパケット転送の処理を行うことを意味します。IPルーティングとは、どのネットワーク機器にパケットを転送する必要があるのかを決めているプロトコルで、OSPFやBGPなどダイナミックルーティングと呼ばれるプロトコルでは、リンク状態などネットワーク状態が変化した際に、動的にステータスが接続先ネットワーク機器へ通知されます。

つまり、ネットワーキングの分野は、年代を問わず一貫して分散型であり、コンピューティング分野とは異なりアーキテクチャの変遷はありません。また、プロトコル自体は標準化されていますが、専用ハードウェアや専用OSが必要とされるのが現状です。

IPルーティングなどWANやインターネットに関わる分野は標準化もされているものの、データセンターなどの分野を考えると、レイヤ2ネットワーク技術など、各社独自のプロトコルとそのハードウェアは密接に関わっています。この分野は専用ハードウェアと専用OS、かつ独自プロトコルといえるため、メインフレームと変わらず、ネットワークベンダーからハードウェア、ソフトウェア、保守、それらに関わる教育など、さまざまな製品情報やサービス、保守を該当ネットワークベンダーから受ける必要があります。

※5　ネットワーク業界・インターネットの分野では、IEEEによる標準化、IETFによるRFCなどが標準化、相互接続性に関わるルールを定めてきました。IETF RFCは膨大で、2014年8月現在文書番号7359が最新になっています。これらは非営利団体であり、ネットワークベンダー、研究者、ボランティアの膨大な労力によってこの標準化作業が行われてきています。

したがって、ネットワーク業界全体、特にデータセンターなど近年大規模化している分野を考えると、IP ルーティング以外にウェイトが大きくなっているため、あまりオープンな環境とはいえない状況です。コンピューティングと違いあまりオープンではないネットワーク業界との相違点が大きくなっています。

また、標準化が進んでいる IP ルーティングだけなど一部だけ切り離すことも難しいので、依然としてネットワーク機器は専用ハードウェア、専用 OS と独自プロトコル実装を前提として考え、利用していくしかありません。

1-1-2 サーバ仮想化の躍進

前項「1-1-1 IT インフラ基盤における仮想化の歴史」では、コンピューティング分野におけるメインフレームでのマルチユーザーアーキテクチャやタイムシェアリングなど、リソース仮想化に言及しましたが、これらとサーバ仮想化は同じ意味合いといえるでしょうか。

技術的にはもちろん同じ意味だと説明されることも多いと思われますが、今一度それぞれの意味を熟考してみてください。この回答は技術的には正解ですが、サーバ仮想化の重要な点を見落としているといえます。

サーバ仮想化は、タイムシェアリングで CPU やメモリリソースを分割して各仮想マシンで利用する技術です。その分割単位は異なりますが、技術的な意味ではメインフレームのタイムシェアリングと大きく変わりません。しかし、サーバ仮想化が現代の IT インフラ基盤にもたらす価値で

表 1.2 サーバ仮想化以前の技術とサーバ仮想化技術の分解点の違い

レイヤ / 技術	タイムシェアリング	仮想メモリ	Solaris コンテナ※6 (2000 年頃)	サーバ仮想化
アプリケーション	分離	分離	分離	分離
OS	-	-	-	分離
仮想マシン	-	-	-	仮想マシンによる抽象化
CPU	CPU サイクルを時分割	-	CPU サイクルを時分割	CPU 支援機構（※7）を利用してゲスト OS 向けの権限付き命令を実行可能。CPU サイクルを時分割
メモリ	-	仮想メモリアドレス空間を定義して物理メモリにマッピング	仮想メモリアドレス空間を定義して物理メモリにマッピング（CPU 支援機構はない）	CPU 支援機構を利用して仮想メモリのオーバヘッドを低減

※6　コンテナ技術は 2013 年以降ごく最近注目を浴びているが、ここでの記載は 2000 年頃のかつての Solaris コンテナとして記載。
※7　CPU 支援機構 Intel VT-d, AMD-V

最も重要な点は、この「リソースシェアリング」技術そのものではありません。

仮想マシンをハードウェアから分離して抽象化し、俊敏性（アジリティ）や可搬性（モビリティ）などさまざまな特性を持つことを可能にすることが大きな意味を持ちます。サーバ基盤はコモディティな x86 サーバで構わず、ベンダーの交換も OS 選択の制限もありません。ハイパーバイザーを搭載してその仮想化基盤の上で自由に仮想マシンとゲスト OS を展開できます。

サーバ仮想化には、次に挙げるメリットがあります。

- 俊敏性（アジリティ）：
  定義ファイルとディスクのみで仮想マシンを展開できるため、迅速なアプリケーション展開が可能になる。
- 可搬性（モビリティ）：
  ハードウェアと紐付いた OS、ハードウェアと紐付いたストレージなど、移動が困難なシステムから、VMDK などのファイル単位のシステムとなることで、高い可搬性を提供できる。
- 管理性：
  ハードウェアと紐付いた OS ではなく、ハイパーバイザーから提供される高度な情報を利用可能になり、管理が容易になる。
- 互換性（コンパチビリティ）：
  ハードウェアと紐付いた OS とは違い、サーバ各社間で互換性がない状態から、x86 サーバと各ハイパーバイザーが取り扱うことができる標準的な OVA・OVF ファイルになることで、サーバ各社間の仮想マシン互換性が飛躍的に向上する。

サーバ仮想化は、IT インフラ基盤のサービスの基本単位･管理単位である OS やアプリケーションを、ハードウェアから抽象化することによって、俊敏性や可搬性、管理性、互換性などの向上、それに伴うコスト削減など、さまざまなメリットが生じることが重要です。サーバ仮想化により、サーバの管理者は、工数を要するサーバの展開や追加など、アプリケーションに関する要件変化に迅速に対応することが可能になります。

1-2 ネットワーク仮想化の歴史

ネットワークにおける仮想化の歴史を技術的に振り返ってみましょう。サーバの仮想化と同様、少ない帯域と貴重なネットワーク機器のリソースを分割して利用することが、ネットワークにおける仮想化として長く利用されてきました。

例えば、ATMにおける仮想回線（VC）、EthernetにおけるVLAN（仮想LAN）等の技術です。ネットワーク機器のリソース（回線やセグメント）を1つの物理ネットワーク機器で利用する技術ですが、ネットワーク機器のアーキテクチャが自律分散型で、各ネットワーク機器がパケット転送の処理を独立して行うため、仮想化の対象はその技術が適用される1つの物理ネットワーク機器のみです。また、仮想化が適用されたネットワーク機器がその情報をネットワーク基盤上の他のネットワーク機器に伝えるには、パケットのヘッダになにかしらの識別子を付与して、転送させる相手側のネットワーク機器に必要な処理を実行させる動作が必要です。

このように、自律分散型アーキテクチャは、ネットワーク機器が独立して情報（データベース）を持ち、それぞれ独立してパケット転送の決定をします。そのために、パケットに識別子を付加する必要があります。新しいヘッダ処理が必要になった場合には、該当するヘッダに対応した新しいネットワーク機器が必要になることがよくあります。また、処理できたとしてもハードウェ

表 1.3 ネットワークにおける仮想化技術の歴史

主な年代	技術カテゴリー・名称	技術略語	技術正式名	仮想化対象	説明
1980年代	WAN・X25	VC Virtual Circuit	Virtual Circuit	回線	WAN上での回線仮想化
1990年代	WAN・フレームリレー	VC	Virtual Circuit	回線	フレームリレー回線の仮想化
1990年代中盤	WAN・ATM	VP	Virtual Path	回線	ATM回線の仮想化
		VC	Virtual Channel	回線	ATM回線の仮想化
1990年代後半	キャンパス・VLAN	VLAN	Virtual LAN	セグメント	スイッチによるブリッジの統合
1990年代後半	WAN・VPLS	VPLS	Virtual Private LAN Service	拠点セグメント	MPLSを利用したVPN
2000年代	WAN・多重タギング	PBB	Provider Backbone Bridging	VLAN	キャリアのためのタギングを二重化
2010年代	データセンター・拡張VLAN	VXLAN	Virtual eXtensible LAN	セグメント	VLAN数の制限を超えるために策定

アでオフロードできずソフトウェア処理となる場合もあり、この場合は高価なASICなどが生かせません。

表にネットワーク分野における仮想化技術の歴史を示します（表1.3）。仮想化対象はいずれもネットワークリソース（回線、セグメント情報）等のネットワークリソースであり、ハードウェアからネットワークを抽象化する要素のある技術ではありません。

1-2-1 ネットワーク技術の発展

ネットワーク技術の仮想化の歴史は、前節「1-2 ネットワーク仮想化の歴史」に記述した通り、リソース仮想化の歴史ともいえます。限りある帯域を有効活用する点に重点が置かれています。また、ハードウェアの革新により通信速度と帯域が短期間に向上すると、そのハードウェアに合致するリソース仮想化のための新たなヘッダを定義し処理することで、ネットワークベンダー間での熾烈な競争を生き抜く必要がありました。特に、1990年代～2000年代はネットワーク技術のベースとなる通信速度や帯域の向上が著しく、ハードウェアに搭載されるソフトウェアが成熟できるほどの期間が与えられなかったともいえます。

また、ネットワーキングは、IPルーティングが主なプロトコルであるインターネット・WANと、独自のプロトコルが残るデータセンター・LANの間では状況が違うため、分けて考える必要があります。インターネット・WANは、WAN（網）の単位で1つのネットワークリソースとして捉えられるのに対して、データセンターでは、使用されているプロトコルが独自かつ各社で複数存在し、1リソースとして捉えることは困難です。ネットワーク技術では、適用される場所で特性が大きく異なることとプロトコルの存在が、サーバとサーバ仮想化の歴史との差異となります。

本項では、リソース仮想化に関するネットワーク技術の発展をもう少し掘り下げて説明します。

1980～1990年代には、WANで利用されていたフレームリレー網の有効活用としてフレームリレーの仮想回線をVC（バーチャルサーキット）と呼び、物理的な回線を共有利用していました。

また、キャンパスネットワークでも、VLAN（仮想LAN）を使用できるレイヤ2スイッチが、複数のブリッジ機器を1つにまとめられるネットワーク技術として、既に20年近く基本的な技術として利用されています。

以降、ネットワークにおけるリソース仮想化の例を紹介します（歴史を踏まえて既に利用されていない技術も含まれています。

・フレームリレーVC（Virtual Circuit）

- フレームリレーは、OSIレイヤ2に相当するプロトコルです。重厚なX.25に変わりX.25

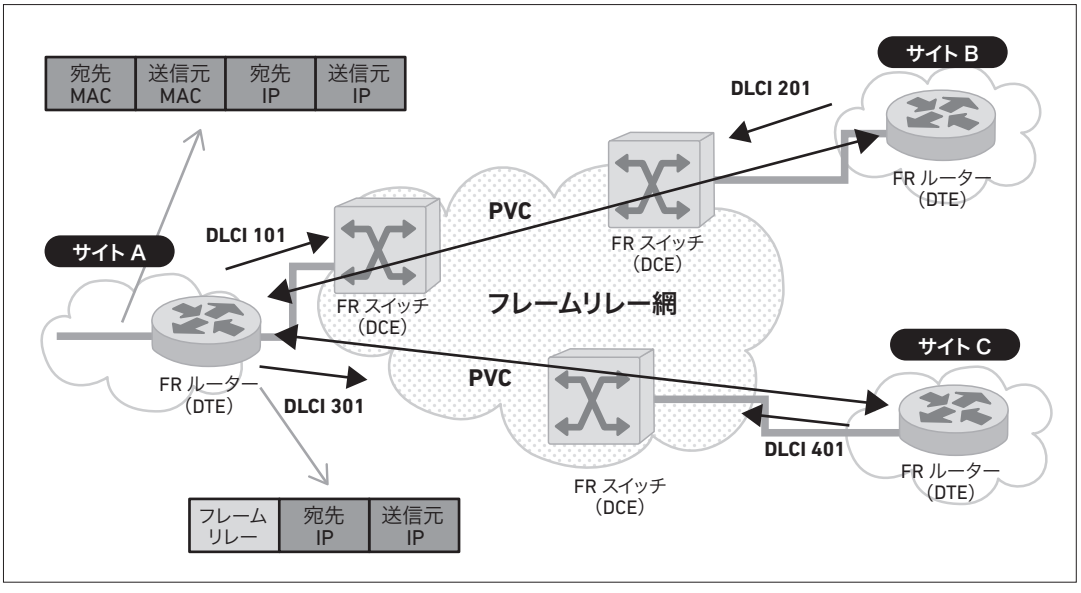


図 1.1 フレームリレー概要

よりも誤り訂正などの簡素化による高速化が図られたパケット交換技術で、1990 年代に流行しました。2000 年代初頭にほとんど使用されなくなっています。

- フレームリレーでは、キャリア側の装置を DCE、ユーザー側の装置を DTE と呼び、LMI（Local Management Interface）によるシグナリング交換を行い接続を確立します。例えば、DTE へ Ethernet フレームが送られてくるとフレームリレーヘッダとして DLCI などが付加されフレームリレー網にパケットが転送されます。仮想回線としては、PVC（Permanent Virtual Circuit）、SVC（Switched Virtual Circuit）の 2 種類があります。日本では、PVC で利用されるケースがほとんどでした。この仮想回線上でローカルのフレームリレースイッチの識別子を DLCI と呼び、IP アドレスと組み合わせて仮想回線を実現しています（図 1.1 参照）

・ATM VP（Virtual Path）/ VC（Virtual Channel）

- ATM は、前述のフレームリレーと同様に WAN で利用される技術です。ATM はフレームリレーや Ethernet とは異なり固定長のセルと呼ばれるデータ単位でハードウェアスイッチによって高速にデータ転送します。1 セルは 5 バイトの ATM ヘッダと 48 バイトのペイロードで合計 53 バイトです。他の方式と異なり固定長かつ小さなデータ単位なので転送時間に時間差も発生せず、単一の固定長のみの処理に特化できるため当時でも高速なハードウェアを作成可能でした。現在では、キャリア向けも含めほとんど利用されていません。

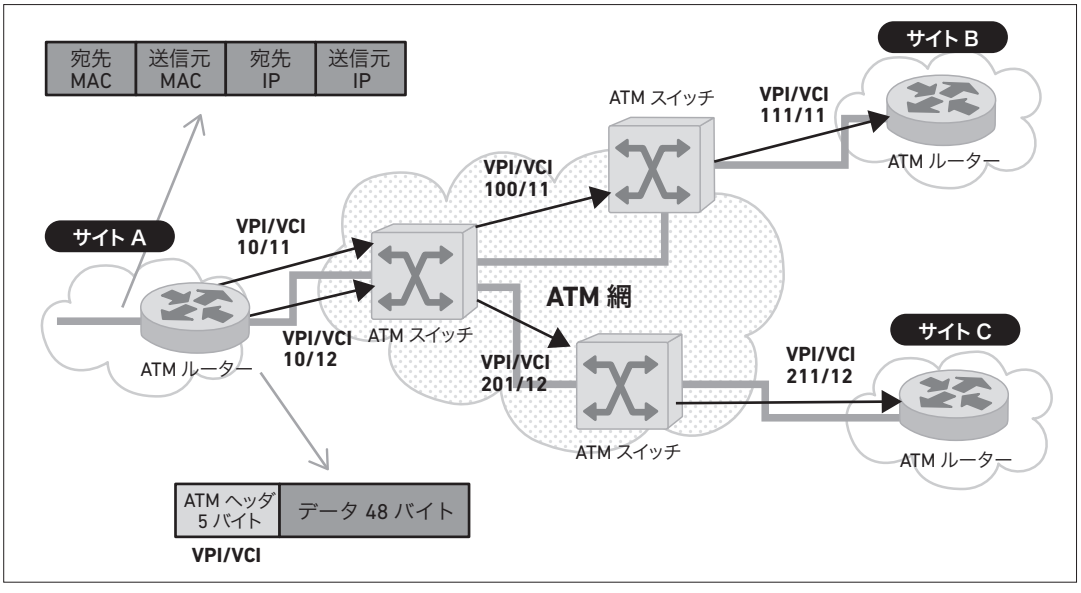


図 1.2 ATM 概要

- ATM では、OSI レイヤ 2 に該当する AAL(ATM Adaption Layer)が複数定義されており、上位層には TCP/IP が利用可能です。ATM の仮想技術もフレームリレー同様に回線を対象とした技術で VCI、VPI と呼ばれます。VC、VP の識別子ですが、VCI（16 ビット）がより下流の単位となり VCI をいくつかまとめた単位を VPI（8 ビット）として割り当て可能です。1 つの物理回線に複数の VPI を持つことができきます（図 1.2 参照）

・VLAN VLAN タグ

- VLAN は、IEEE802.1Q として標準化されています。VLAN は物理的なセグメントを持つブリッジと呼ばれるデバイスを、1 つのデバイスに収容するために考案された仮想セグメントを構築する技術です。ブリッジに対して、この VLAN を利用して複数のブリッジをまとめることが可能な装置をスイッチと呼びます。
- VLAN を識別する VLAN ID は 12 ビット長で、最大 4096 個の VLAN を識別可能です。規格上使用可能なのは 4094 個です。また、各ベンダー固有の予約 VLAN 番号があり、実際に利用できる VLAN はさらに少なくなります（図 1.3 参照）

・MPLS VPLS（Virtual Private LAN Services）

- MPLS はラベルと呼ばれる固定長ヘッダにより転送する技術で、ATM から更改されたキャリア網などで利用されています。当初キャリア向けの高価なハードウェアのみで実装され

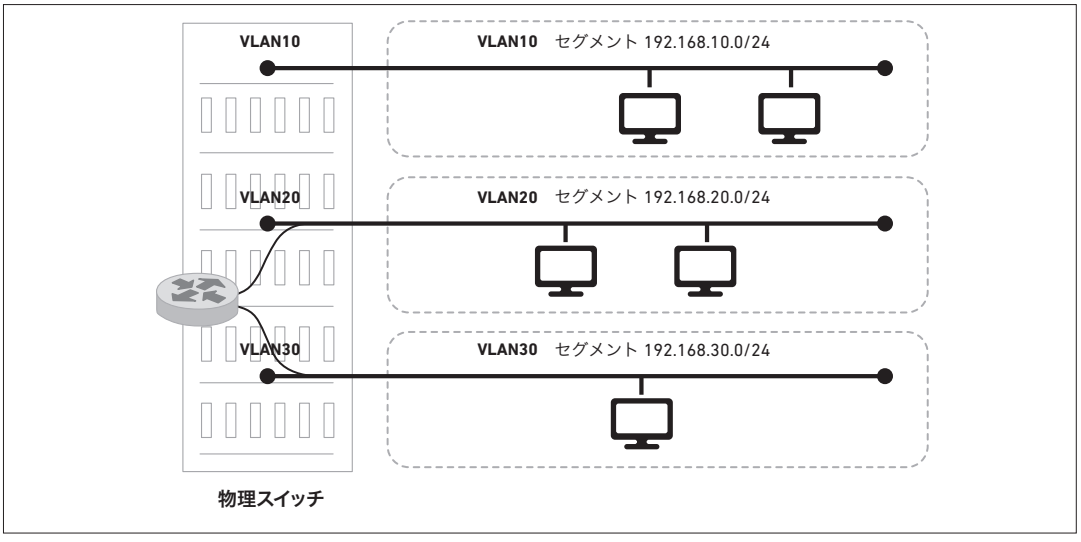


図 1.3 スイッチ内の VLAN イメージ

ていましたが、現在ではルーターの高機能化と ASIC の高速化により、一般的なルーターでも使用できます。現在の MPLS では、柔軟なトラフィックエンジニアリングや FRR（Fast Reroute）など、高可用性や運用性などの機能が重要視されています。

- VPLS にはいくつか種類がありますが、本項では最も基本的な MPLS を利用して拠点間のレイヤ 2 ネットワークを仮想的に接続する技術を紹介します。当初拠点から転送されたパケットが VPLS 網に戻ってくる DC 間のレイヤ 2 ループがありましたが、ベンダー実装により解決しています。仮想ネットワークセグメントを識別するために、VFI（Virtual Forwarding Instance）識別子を利用して、PE（ルーター）が拠点側の VLAN と MPLS/IP ネットワーク対向 PE（ルーター）とのマッピングを定義します（図 1.4 参照）。

・VXLAN（Virtual eXtensible Local Area Network）

- サーバ仮想化により利用 VLAN が爆発的に増加して、約 4000 個の VLAN 数では不足するため、VLAN の拡張プロトコルとして、シスコシステムズ社やヴイエムウェア、ブロードコム社などで考案されたプロトコルです[※8]。VLAN に変わる新たな識別子は VNI（VXLAN Network Identifier）と呼ばれ、24 ビットで約 1670 万個の識別が可能です。
- VXLAN は仮想化環境で多く利用されます。仮想化環境を想定すると、仮想マシンなどから送信される Ethernet フレームはインナーフレームと呼ばれます。VXLAN を、終端する VTEP（VXLAN Termination End Point）でカプセル化を行い、宛先仮想マシンを持っ

※8　RFC 7348: Virtual eXtensible Local Area Network（VXLAN）: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks として、2014 年 8 月に承認（提案は 2011 年）。

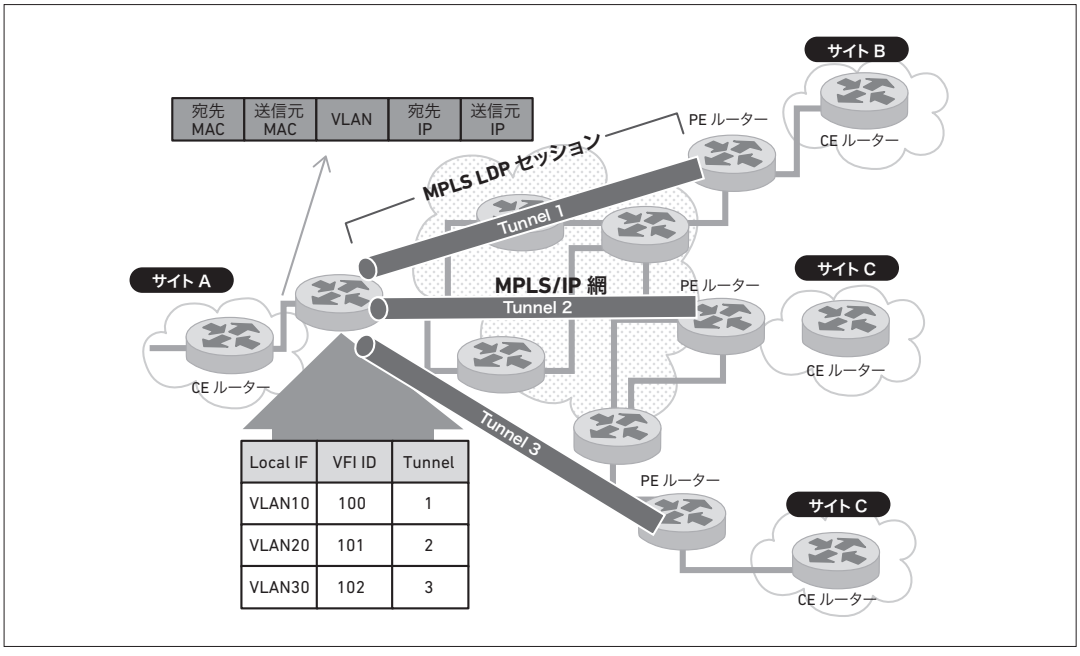


図 1.4 VPLS 概要

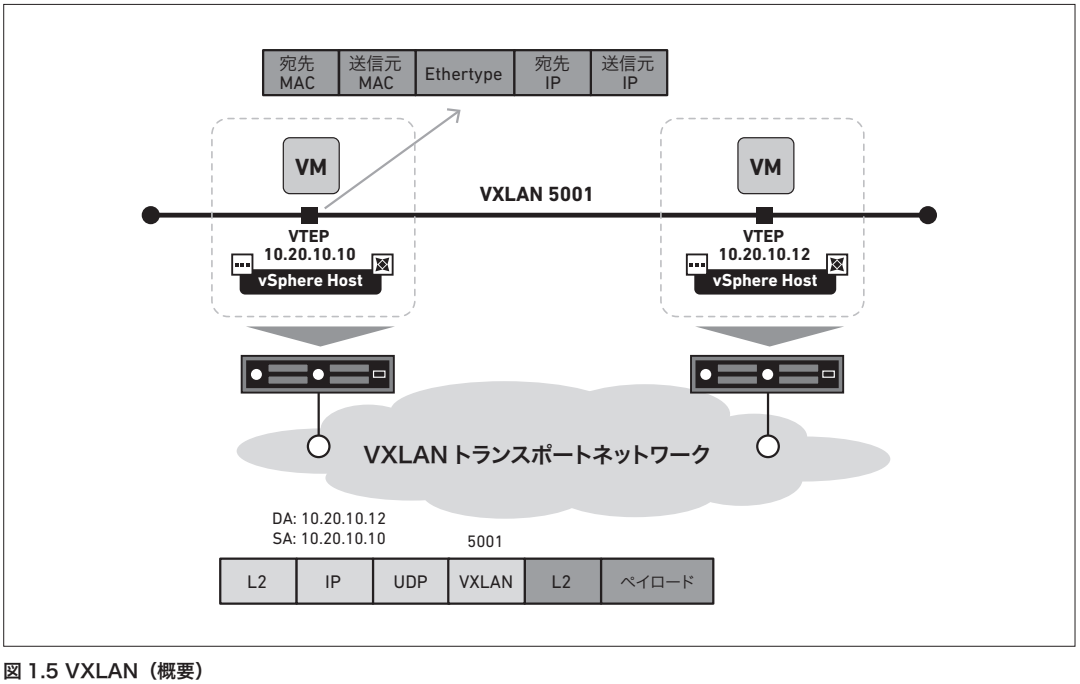


図 1.5 VXLAN（概要）

ているハイパーバイザーの VTEP へ送信されます。外側に付いている VXLAN フレームは、インナーフレームに対してアウターフレームと呼ばれます（図 1.5 参照）。

本項で確認した通り、ネットワーク技術の歴史では、リソース仮想化と呼ばれる 1 つの物理ネットワークリソースを共有する仮想化が行われています（Chap.3 参照）。サーバ仮想化とは異なり、ハードウェアとソフトウェアを抽象化して分離する要素はありません[※9]。

また、近年サーバ仮想化環境ではハイパーバイザーの機能として、仮想スイッチが重要な役割を果たしています。当初は仮想マシンへの接続機能だけでしたが、QoS や SPAN などアクセススイッチの機能を代用でき、確実に仮想マシンに接続されるので、論理上のアクセススイッチといえます。

この仮想スイッチの論理的な配置は、データセンターネットワーク、またネットワーク仮想化の意味を理解する上でも重要な意味を持ちます（詳細は「1-4 SDN と OpenFlow」を参照）。

1-2-2 ネットワークベンダーにおける技術開発

前節ならびに本節では、ネットワーク技術の分野で行われてきた、主にリソースの仮想化と呼ばれる技術革新の歴史を振り返りました。本項では、ネットワーク機器ベンダーの観点でこれら

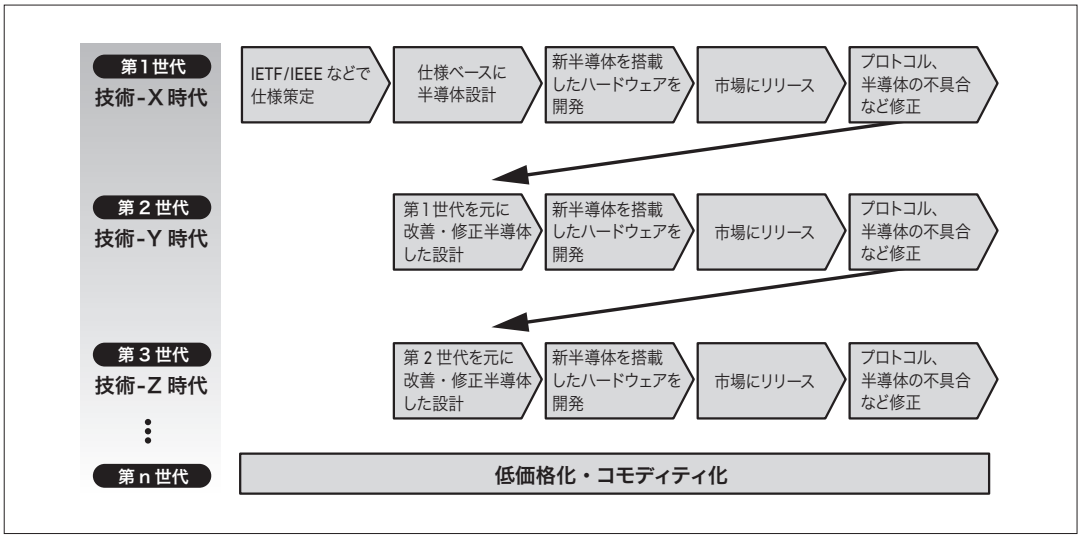


図 1.6 ネットワーク機器ベンダーの開発サイクル

※9　各ネットワークベンダーの独自技術では、「バーチャルシャーシ」に該当する技術が唯一、ハードウェアとソフトウェアを抽象化して分離する意味合いに近いといえます。バーチャルシャーシとは、複数のネットワーク機器を仮想的に 1 つの筐体とみなし、その仮想的な筐体単位でレイヤ 2、レイヤ 3 の隣接関係を構築できる技術です。しかし、近年やっと登場してきたこととまったくの独自規格であるため本書では紹介していません。また、あくまで単一のレイヤ 2、レイヤ 3 ノードとして動作するだけで、ネットワーク仮想化と違い、ネットワーク基盤全体に対して何らかの操作を行うものではなく、あくまで局所的に利用します。したがって、バーチャルシャーシは特定箇所のレイヤ 2、レイヤ 3 の隣接関係の簡略化が目的です。

の技術の変遷を考えてみましょう。

前項で説明したフレームリレーやATM、MPLS/VPLS、VLAN、VXLANなど、各種ネットワーク技術の歴史を振り返ると、1990年代から2010年代まで常にハードウェア開発を伴った新たな変化が起こっています。

1990年代には、既にフレームリレーでVC（Virtual Circuit、仮想回線）を扱える機器が出現し、数100kbps程度の限られた帯域で複数の仮想回線を構築し、回線リソースを共有しています。その後、ATMやMPLSなど固定ヘッダをハードウェア処理する方式のデータ転送方式が出現し、固定長のヘッダのみを見て処理することで、揺らぎの少ないデータ転送を実現しています。

この際も前述のフレームリレーと同じく回線リソースを共有し、セルをハードウェアで処理するために、新たな半導体やスケジューリング機構が開発されています。その結果、数100Mbpsの帯域と従来よりも格段に高速なパケット転送が実現しています。

ここで説明した技術は、各社独自のハードウェアとパケット、フレーム、セルなどの開発に加えて、各社が定義した技術の標準化を試み、それぞれ独自の半導体やASIC等を開発するといったサイクルで、新たな製品をマーケットへリリースしています。現在生き残っているネットワーク機器の大手メーカーは数社と限定されていますが、1990年代～2000年代は、100以上のネットワーク機器ベンダーが存在していました[※10]。

本項で説明した新たな技術の導入には、新ヘッダへの対応や旧来の技術との低い互換性のために、その都度、新規のハードウェアネットワーク機器の開発が必要とされました。しかし、当時はそれぞれ従来とは異なり、効率的なパケットスケジューリングや転送性能の向上、リソース仮想化などが実現できれば、新たに高価なハードウェアの調達する必要があっても、デメリット以上の大きなメリットが得られるため、十分に意味がありました。

また、別の観点では、1990年代はインターネットの規模が爆発的に大きくなり、これらネットワーク機器ベンダーの製品が数多く必要とされていたこともうかがえます。総務省情報通信統計のデータを示します（図1.7）。

商用インターネット黎明期の1994年までの売り上げ水準から数倍の規模になり、ITバブルとその崩壊（2000年）、リーマンショック（2009年）など需要増減の大きな要因がありますが、この15年ほどは平均2兆5,000億円程度で推移しています。なお、5年程度のサイクルで売り上げが増減していることから、ネットワーク機器更改のタイミングと判断できます。

※10 2014年現在では、シスコシステムズ社やジュニパーネットワークス社、ブロケード社に加えてファーウェイ社等の新興メーカーが存在しますが、1990年代には、ベイネットワークス社（後のノーテル）、3Com社、Ascend社等多数のメーカーが存在していました。例えば、Interop Tokyoの会場の大きさを比べれば自明で、2014年はHall5＋αのスペースですが、2000年頃ではHall 5～10のすべてを占めていました。これは景気が低迷しているのではなく、当時はネットワーク機器ベンダーが多数のカテゴリーに別れ、さまざまなプロトコルやスケジューリングに関するアイデアとハードウェアで製品を作り、覇権を競っていたハードウェア競争全盛の時代といえます。

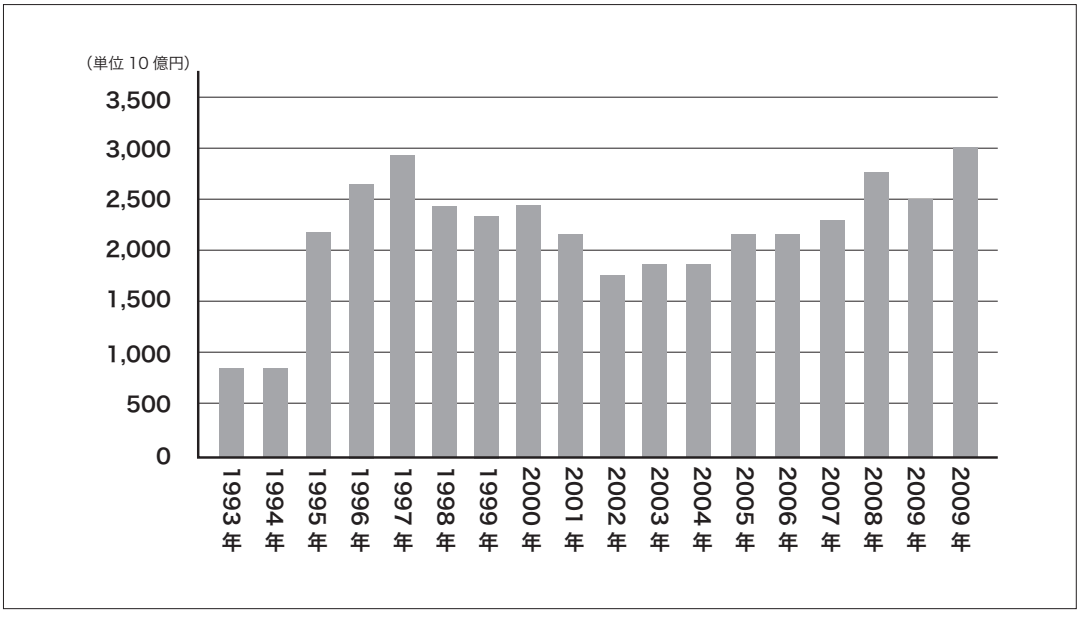


図 1.7 総務省データ 国内通信機器売り上げ推移（引用：総務省情報通信統計データベース）

データセンターへの集中化が進んできた 2000 年代後半以降はどうでしょうか？

ネットワーク技術として PBB や各社 TRILL 系マルチパスレイヤ 2 技術、VXLAN が出現していますが、マーケットシェアからはまだまだ IP ルーティングと VLAN で構成されているケースが多い印象があります。

また、プロトコル以外にバーチャルシャーシや MLAG などの技術もありますが、ハードウェアやパーツ、搭載するネットワーク機器の OS バージョン、サポートする構成などに制限が大きいため、局所的に使用されつつも、ネットワーク基盤の更改を全面的に後押しする技術ではないと判断できます。

いずれにしろ 2010 年代は 1990 年代の爆発的なインターネット需要やハードウェアが進化していた時代とは異なり、IT インフラ基盤としてのネットワーク整備が一段落しているといえます。

ネットワークベンダーのライフサイクルとキャズム

ネットワーク機器ベンダー、シスコシステムズ社の製品開発をモデルにした書籍『ライフサイクルイノベーション』（ジェフリー・ムーア著 / 翔泳社）では、製品のライフサイクルを定義し、その時期によって取るべきマーケティング施策が示されています。本書で説明しているネットワーク業界全体をこの書籍の考え方に当てはめて考えると、より深い洞察が得られるかもしれません。また、ジェフリー・ムーアは、破壊的イノベーションとその市場浸透段階におけるマーケットとユーザー受け入れのギャップを議論している、『キャズム』（翔泳社）も執筆しています。

1-3 既存ネットワーク仮想化の問題点

ネットワークにおける技術開発は、前節で説明した通り、さまざまな要因からハードウェア中心の技術として発展してきました。したがって、ソフトウェアによるライブラリや管理などに重点を置くのではなく、ハードウェアをより細かくユーザーが操作するCLIベースの利用が主でした。ハードウェアが持つ細かいパラメータを操作する意味では、適したインターフェースといえるため、これまでは特に問題になっていませんでした。

一方、2000年代も後半になると、ネットワークベンダーからさまざまな技術がリリースされていますが、マーケット側の通信速度や通信帯域に関する需要も以前ほど切迫したものではありませんでした。実際サーバに搭載されるNICで10GbEは既に十分に安価ですが、必ずしも搭載はされていません。また、ネットワーク構成も依然としてIPルーティングとVLAN構成のままであるケースもよく見かけます。ハードウェアの進化によりもたらされてきたメリットが少なくなってくると同時に、クラウドによる大規模化、管理の困難さ、自動化への対応など、新たな課題が目立ってきています。本節ではこの問題点を掘り下げて説明します。

1-3-1 ネットワークの複雑化と課題

各種のネットワークリソース仮想化技術の利用が大規模に進んだことや、2000年代以降のデータセンターへの集中により、ネットワークが非常に複雑で管理が容易でなくなってきています。その一方で、前項で触れた通り、ネットワークアーキテクチャの進化のスピードも緩慢になり、今なおデータセンターでは、IPルーティングとVLANを中心としたネットワーク技術でネットワーク基盤が構成されています。

2000年代中盤、サーバ仮想化が適用され始める前のデータセンターネットワークは、以下の状態です。

- データセンターネットワークはVLANで構築
- IPルーティングはBGPやOSPF等で構築済み
- サーバ追加・削除ごとにVLANやネットワークを人手を介して追加・削除
- トポロジーなどネットワークを静的に扱い、操作手順書などを逐次作成・改定して管理

・MAC アドレス数や VLAN 数などデータセンター規模も大きくないため問題となっていない
・帯域は 1Gbps と 10Gbps の併用で、サーバに 10GbE が搭載されるケースはまだ多くない

2000 年代中盤以降、サーバ仮想化が適用され始めると、以下の問題点が露呈しています。

・VLAN の追加・削除が日常的になり、関連するトランクやルーティング等の設定変更作業も合わせると、負荷が大きくなる
・仮想スイッチの変更を強いられ、またその変更管理が大きな負担となり始める
・一部大規模なデータセンターでは、MAC アドレス数の消費がスイッチの上限値に近くなり、増設計画などに影響を及ぼし始める

上記はサーバ仮想化が標準となり拡大していく中では大きな問題となり得ます。次項では、問題をさらに具体的に確認しましょう。

1-3-2 クラウドの躍進に伴う新たな課題

前項では、ネットワークアーキテクチャや基本構成に大きな変更がない状況で、サーバ仮想化の一般化が進んだ、データセンターの環境変化を紹介しました。本項では、クラウド化に伴い新たに求められるネットワーク要件やネットワークにおける課題を具体的に考えてます。

サーバ仮想化の躍進に伴い、以下の問題が浮かび上がりました。

・vMotion などデータセンターに広範囲なレイヤ 2 要件が求められます。しかし、一般にデータセンターネットワークでは、広範囲なレイヤ 2 ネットワークはスパニングツリーや VLAN 構成の大規模化を意味します。ネットワーク機器としても過負荷、障害範囲の大規模化など問題があり、従来は推奨されるデザインではありませんでしたが、サーバ仮想化以降では大規模なレイヤ 2 ネットワークを構築しなければなりません。

・仮想マシンの可搬性に関して、VXLAN など比較的標準に近いプロトコル以外にも、VNTag や VEPA など各社が独自にプロトコルを策定し実装しています。ユーザーの立場では、どのプロトコルが標準かつ利用しやすいのか分からず、また、既に通信帯域に関する大きな需要がないにも関わらず新規のハードウェア購入が求められます。

・IT インフラ基盤のクラウド化に伴うネットワーク構成変更の回数が非常に増加しています。

・仮想マシンのデプロイが日常的になり、ネットワークの設計、実装の追加や変更が求められます。この変更管理を手動で実施している場合が多いため、必然的に負荷が高くなります。
  - ネットワーク構成図、VLAN、IP アドレスなどを手動で管理しているケースも多く、追加・削除に伴い、管理ドキュメントも最新版にアップデートする必要があります。ネットワーク管理者のプロセス上でも大きな工数を占め、単にネットワーク機器の設定だけを変更すればよいわけではありません。

上記の問題は、サーバ管理者がネットワーク管理者へ仮想マシンのネットワークを申請するケースで、手順を詳細に考えると理解できます。

1. サーバ管理者が、仮想セグメントサブネットと数量を申請します。
2. ネットワーク管理者は、VLAN とゲートウェイ、冗長化の有無、IP アドレスを考え、払い出しを行う（1 週間〜 2 週間）。
   a. VLAN：4000 までしか使用できない。また、vMotion など広範囲で利用する場合、疎通させたいエンドノードの範囲で共通化させる必要がある。複数のネットワーク機器間で一貫性（Consistency）を保つ必要があり、影響の大きい VLAN 構成とスパニングツリーなどの設定を慎重に検討します。
   b. IP アドレスの払い出し：ここで問題なのは IP アドレスの払い出し（IPAM）ではなく、払い出される IP アドレスの種類です。払い出される IP アドレスには、ネットワーク管理者が管理している IP アドレスから一意になるように割り振った IP アドレスと、仮想ネットワークで自由に払い出されて重複する可能性がある IP アドレスの 2 種類があります。一般的なネットワークに関する要望として、自由なプライベートアドレス（クラス A、B、C）を利用したい状況はよくありますが、これは次の理由で難しいため、エンドユーザーはネットワーク管理者から払い出される IP を利用することになります。
   なお、上記は複雑なネットワーク設計・設定・機能が必要なためです。詳細は「3-3-5 仮想ルーター」を参照してください。
3. 各要件をネットワーク機器のコンフィグレーションに変換して確認します（場合によっては検証も行います）。
4. コンフィグレーションをネットワーク機器に投入します。

手順 2 でネットワーク管理者が時間を要していると思われますが、既存のネットワーク作業者の手順を考慮すると明白です。各機能には制限があるため、マニュアルでコンフィグレーションを作成します。仮にスクリプトなどで自動化されているとしても、複数ネットワーク機器間の設

計一貫性を担保するのは設計者であるため、マニュアル管理・操作といえます。

つまり、サーバが仮想化され自由自在に仮想マシンが展開できる IT インフラ基盤でも、ネットワークの機器、設計、実装が既存の方法のままでは、ネットワーク機器に依存する制限、マニュアル操作、マニュアル管理からは逃れることができません。

・ネットワーク機器依存：

VLAN 数の上限、仮想ルーティングテーブルの有無など、ネットワーク機器に依存した機能が多い。慎重に検討する必要があります。

・マニュアル操作：

ユーザーが希望すること（ネットワークを作成して、仮想マシンを接続したいこと）と、ネットワーク機器のコンフィグレーションの間にはギャップが大きく、正しいコンフィグレーションを作成するには多くのパラメータへの配慮が必要です。

・マニュアル管理：

すべてのネットワーク機器で設計の一貫性を保つ必要があります。具体例を挙げれば、アクセススイッチのトランクポートで Allowed した VLAN をアグリゲーションスイッチでも Allowed する必要があります。また、他のアクセスする可能性のあるアクセススイッチでも同様に Allowed する必要があり、ある VLAN に対する VLAN 設定の一貫性をアグリゲーションスイッチ、複数のアクセススイッチすべてで保つ必要があります。ネットワークの設計・実装が難しいのは、ネットワーク基盤すべてのネットワーク機器で「設計全体の一貫性」を保ち続ける必要があることです。仮にこの一貫性が失われた場合、通常は即通信障害として顕在化します。

VLAN や IP アドレスの払い出しだけを検討しただけでも多数の考慮すべき項目があり、マニュアル管理やマニュアル操作が求められます。実際には、上記以外のネットワーク要素に関しても、検討、設計、実装、運用する必要があり、作業負荷が高いのは明白です。

また、クラウド化により、さらに次の要素も求められます。

1. データセンター間のレイヤ 2 延伸、オンプレミス連携、ハイブリッドクラウドなどレイヤ 2 に関するネットワーク要件

   単純にレイヤ 2 をデータセンター間で延伸した場合、例えば、ゲートウェイの冗長化に関

して、各HSRP/VRRPグループごとに想定するゲートウェイがActiveになるようにフィルターするなどの考慮が必要です。こうした作業をHSRP/VRRPグループごとに動的に作成、削除、メンテナンスを継続することは非常に困難です。また、このフィルターが誤っている場合は、ゲートウェイへの接続性がなくなります。詳細は「3-2-4 レイヤ2要件」を参照。また、オンプレミス連携やハイブリッドクラウドなどレイヤ2延伸を考えた場合、通常はL2-VPNと呼ばれるトンネルを作成する方法で実現します。従来、複数のトンネルを冗長化のために作成した場合は、トンネル間でループするなどの新たな問題が発生していました。これらの状況への対応として、各社はクラウドに特化したデータセンター間のトンネルプロトコルを策定してきました。各社独自の技術で実装されているため、新規のハードウェアが必要であることを意味します。

2. マルチテナント化への対応。VLAN利用数など既存のネットワーク技術のより大規模な消費
ITインフラ基盤のクラウド化によりマルチテナント化が求められます。これは、より多くのセグメント、つまりVLANや仮想ルーターなどのネットワーク基盤のリソースを複数同時に要求します。したがってVLANでは、上限値の4000個を上回る必要性も発生しており、各種ハードウェアの制限等も含めると非常に複雑なネットワーク基盤の設計、実装、運用が求められます。(Chap.3参照)

3. より迅速なネットワーク提供と自動化への対応
ネットワーク基盤にも仮想マシンの日常的な展開と動的な変更に適応した変更が求められますが、前述した通り、ネットワーク機器に依存した制限、マニュアル操作、マニュアル管理が必要になるため、スクリプト作成だけでは、自動化などへの対応は難しく、簡易なAPIが提供されるなど、大きな要素追加がなければ変更の迅速性、自動化は困難です。

全体に関わる問題として、ネットワーク基盤全体の設計で「一貫性」を保つことが難しく、特にセキュリティに関する設計、実装の一貫性を保つことが困難です。

一般的に、既存のルーターやスイッチに関しては、ネットワーク管理者がネットワーク全体のデザイン・設計の一貫性を維持しています。一貫性の担保には機器がなにも関連していないため、保守するには多くの手間が掛かります。サーバ基盤とは異なりネットワーク基盤に特有な点は、影響範囲が新規に展開したサーバ基盤だけにとどまらず、例えば、OSPF、BGP設定パラメータを1つ間違うだけで、ネットワーク基盤全体で大きな通信断が発生します。

また、設計・実装の一貫性はセキュリティに関しても保つ必要があります。特に仮想マシンがvMotionにより移動する場合でも、セキュリティ設定の一貫性を複数のネットワーク機器間で維

持する必要があります。セキュリティに関する設定は、日常的に変更する頻度も高く、ルール数も多いことから、関連するネットワーク機器すべてで設計の一貫性を保ち続けることは、非常に負荷が高い作業です。

ネットワーク基盤は、リソース仮想化技術や冗長性、ルーティングプロトコルなどを組み合わせて設計・実装されていますが、サーバ仮想化やクラウド化の普及に従い、複雑性が増している現状があります。下図にデータセンターネットワークの複雑性を示します（図 1.8）。

本節で説明した通り、データセンターネットワークの問題点は、クラウド化が進んだ IT インフラ基盤では以前にも増してより顕在化しており、従来のネットワーク技術はもちろん、設計や実装のあり方を見直すことが迫られています。次節では、こうした現状を踏まえて新たに出現した SDN（Software-Defined Networking」の流れを説明します。

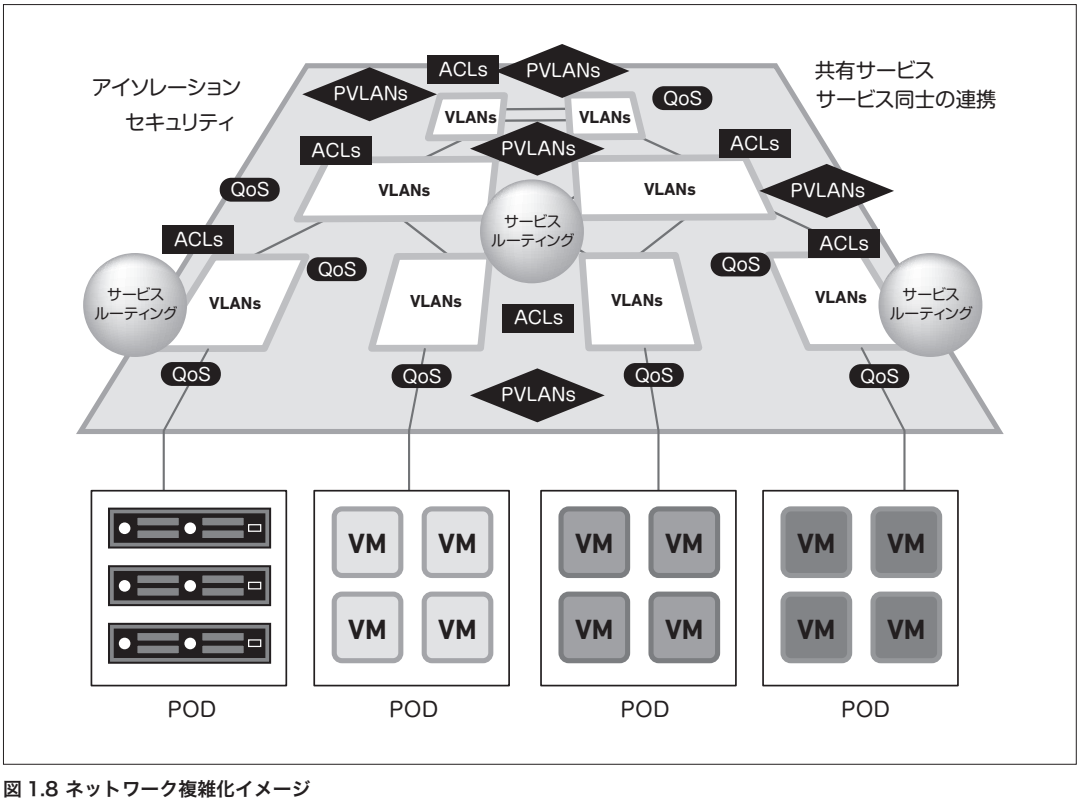


図 1.8 ネットワーク複雑化イメージ

1-4 SDN と OpenFlow

本節では、前節までに説明したネットワーク関連の状況から、なぜ SDN や OpenFlow が誕生したのか、その経緯やプロトコル、そして関連する技術を説明します。なお、各ネットワーク機器ベンダーで使用されているコンテキストやその製品、ソリューションを理解する上でも、本節で説明する技術背景を把握しておくことが有効です。

1-4-1 SDN の成り立ち

「SDN」の初出は、2011 年の MIT（Massachusetts Institute of Technology）の論文です。SDN（Software-Defined Networking）、すなわち「ソフトウェアで定義されるネットワーク」は一般用語の組み合わせですが、現在の注目度とムーブメントの源流は間違いなくスタンフォード大学といえます。2006 年頃から、巨大なデータセンター利用者（Google や Facebook）が協力して、スタンフォード大学で次世代ネットワークに関する研究が活発になり、解決すべき状況として認識されたのが、複雑化したデータセンターネットワークの状況です。

前述の通り、ネットワーク技術・業界では、サーバ仮想化で起こった迅速性・可搬性などを高める抽象化に区分されるアーキテクチャの変化はなく、各種ネットワークのリソース仮想化技術の組み合わせでネットワーク基盤が構築されるため、その複雑性を管理できる人・組織だけがネットワークを使いこなせる状況でした。「クラウド」が話題になり始めた当時は、複雑化したネットワークはデータセンターネットワークの効率性を大きく阻害していると認識され、研究対象となりました。

SDN と OpenFlow は、スタンフォード大学 Nick McKeown（ニック・マッキューン）の研究室からそのムーブメントが始まります。大学の研究者である彼らもネットワーク基盤の複雑性を疑問に感じていたのです。ちなみに、Nick McKeown（ニック・マッキューン）と研究者仲間カリフォルニア大学バークレー校の教授 Scott Shenker（スコット・シェンカー）は、このネットワークの複雑性を使いこなすことを「Master of Complexity（複雑性の支配）」と呼んでいます。

● 複雑性の例

この「複雑性の支配」をより理解するために、実作業と付き合わせるために例を挙げましょう。

データセンターにサーバ管理者とネットワーク管理者がいて、サーバ管理者がいくつかのサービスをリリースするため、ネットワーク管理者に、セグメントやIPアドレス、ロードバランサーとその設定、VIPアドレスの個数などを要求したケースを例示します。ネットワーク管理者は、数が多いため2週間ほど要すると返答します。ネットワーク管理者の実作業を見てみましょう。

ネットワーク管理者は、まず使用するセグメント、ゲートウェイのIPアドレス、VIPアドレスなどをどのように払い出すかを考えます。管理帳から払い出せるリソースを確認し、払い出せるか検討します。場合によっては一部のネットワーク機器で上限を超える可能性があります。

また、特別なIPアドレスの要望であれば、仮想ルーティングなどのさらに込み入った技術が必要になる場合もあります。準備から設計、そしてネットワーク機器への実際のパラメータ決定など、確実に相応の時間が必要になります。

次に、ロードバランサーなど高レイヤのサービスはリソースに指定や制限が大きいため、調達なども含めてさらに周到な準備が必要です。最終的には、ping、traceroute、showなど各種コマンドで正しく設定されているか1つ1つ確認する必要があります。例え、管理スクリプトを利用してもネットワーク管理者の作業には相応の時間を要します。

さらに、現在のネットワーク基盤はネットワークベンダーごとに異なったOS、独自のCLIを搭載しているため、サーバ管理者が自由に触ることはできません。また、一部のパラメータ間違いや抜けなどで致命的な障害に通じるため、慎重な設計・設定が必要です。担当者が誰であっても専門エンジニア（つまりネットワークエンジニア）が担当せざるを得ない状況です。

一方、サーバ管理者にとって、サービス開始に伴うネットワーク準備に2週間を要するのは、長過ぎると思わざるを得ません。しかし、ルーティングに関する知識があっても、OSはもちろんCLIも違い、各ベンダーのパラメータに精通しているわけではないため、専門部隊に任せるしかありません。緊急の場合は急ぐように要求できても、ネットワーク管理者からの作業完了通知を待つしかありません。

つまり、前述の「複雑性の支配」とは、シンプルにネットワークサービスを受けたい、使いたいと考えても、実際にはそうならず、管理できる人間・組織のみがネットワークを使いこなせる、そういったフラストレーションの溜まる状況も表しています。

● Nick Mckeown と Scott Shenker

前述のNick Mckeownは現在スタンフォード大学でコンピューター･サイエンスの教授ですが、その中でも一貫してネットワーク関連技術を専門としています。例えば、カリフォルニア大学バー

クレー校では、博士論文「セルスイッチにおけるインプットキューのスケジューリング」で Ph.D を授与されています。

また、シスコシステムズ社 GSR12000 ルーターのアーキテクトチームのチームメンバーとして働いた経験があり、実業界でも数々のスタートアップに関わり、アブリズオ社（PMC-Sierra が買収）では CTO、サンダーレイヤー社の共同創業者、ネモシステムズ社（シスコシステムズ社が買収）の共同設立者として活躍して、シリアルアントレプレナーとして名を残しています。

そして、Scott Shenker はカリフォルニア大学バークレー校の教授で、ACM と IEEE（米国電気電子学会）のフェローでもあり、コンピューター・サイエンスの研究者です。最近では特に分散データベースの基礎になる分散ハッシュの論文で著名で、コンピューター・サイエンスの研究者の中でも、被引用論文数が上位に位置付けられる研究者です。

● Martin Casado のアイデア

ニシラ共同設立者である Martin Casado はその設立前に、前述の Nick Mckeown 研究室在籍時にあるアイデアを考え付きます。ネットワークの複雑性を解決するには、ネットワーク機器の状態をオペレーショナルステートとルーティングステートに分離すれば、オペレーショナルステートに関しては一元管理が可能になる、というアイデアです。

Casado のアイデアでのネットワーク機器で保持すべきステート（状態）は次の通りで、モデル化すると下図となります（図 1.9）

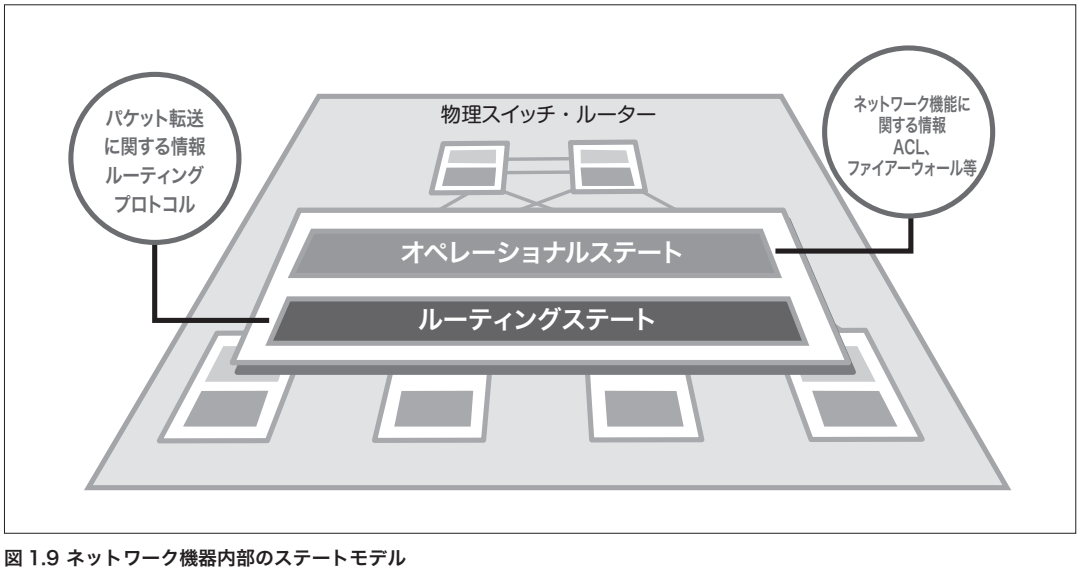


図 1.9 ネットワーク機器内部のステートモデル

・ルーティングステート：

パケットを転送するための情報。

- ネットワーク機器はパケット転送に関する情報を所持する必要がある。

・オペレーショナルステート：

ACL、ファイアウォール、サービスルーティングに関する各レイヤを操作して機能するための情報。

- 追加したいセキュリティやネットワークサービスによって加えられるパケット転送に関するルール。

上記のルーティングステート、つまりルーティングに関する状態は問題ではありません。本章前半で触れた通り、ネットワークアーキテクチャは自律分散型で進化し、2006年頃にはすでに各種のIPルーティングプロトコル、OSPFやBGPで大きな問題になることもありません。非常に枯れたプロトコルで、Zebra/Quaggaなどのオープンな実装もあり、実装にも問題は少ないといえます。また、変更頻度もオペレーショナルステートと比較すると少なく問題になりません。

一方、オペレーショナルステートは先程のネットワーク管理者の作業例からも分かる通り、セキュリティ関連など日常的かつ動的な変更なども多く、変更頻度も高いといえます。

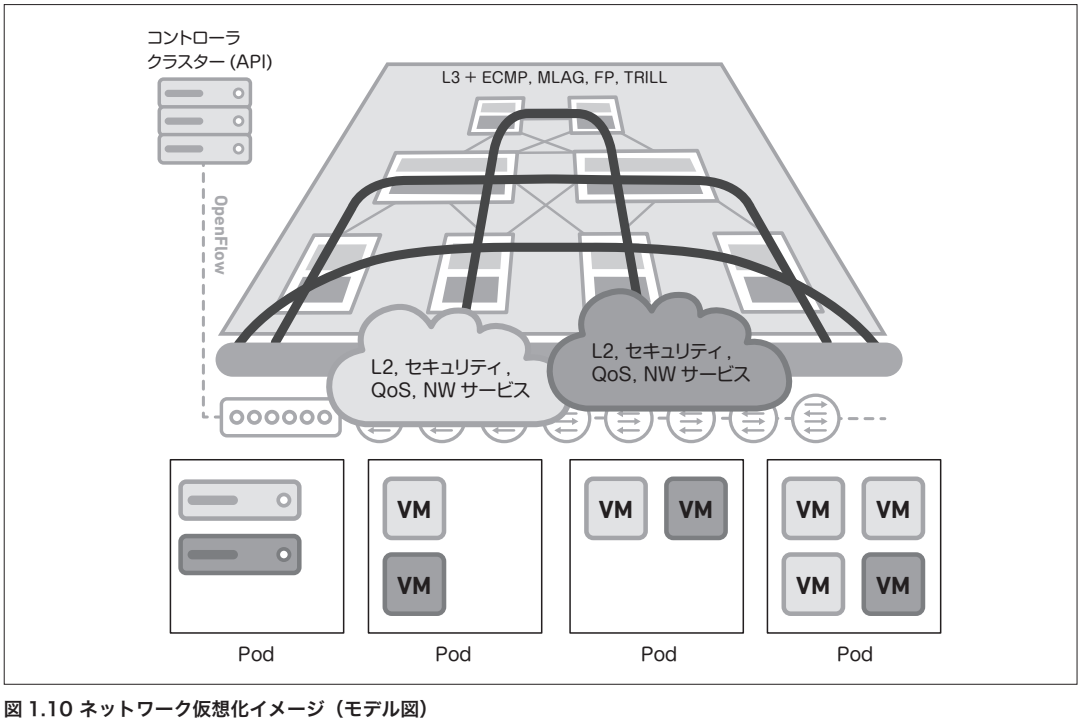


図 1.10 ネットワーク仮想化イメージ（モデル図）

Casado のアイデアは、ネットワーク機器にルーティングステート（＝自律分散なアーキテクチャのパケット転送）は残し、オペレーショナルステートを一元管理するネットワーク OS を作ることで、データセンターネットワークの複雑性を回避するものです。

同時に物理ネットワークはルーティングステートのみを実装し、最初にステートを持つ必要があるハイパーバイザー上でオペレーショナルステートとルーティングステートを分離して一元管理し、そのハイパーバイザー間はカプセル化プロトコルで接続する、NVP およびその後の NSX の製品コンセプトが完成しました。

元々「SDN」とは、Casado や Mckeown、Shenker にとって多様な意味を持つ用語ではなく、明確な課題であるデータセンターネットワークの複雑性を、「ルーティングステートとオペレーショナルステートに分離し、物理ネットワークにルーティングステートの役割を持たせ、オペレーショナルステートは新規のネットワーク OS で一元管理する」ことで解決する回答です。

● Nicira Networks 設立とヴイエムウェアによる買収

Casado は、このアイデアを元に Mckeown と Shenker 両名と共に、2007 年に Nicira Networks を設立します。Mckeown は最初の CEO、Casado は CTO として、OpenFlow や Open vSwitch、SDN、そしてネットワーク仮想化コンセプトを実現する製品開発を担当します。

実験的に開発していたネットワーク OS「NOX」から「ONIX」を設計し直し、NSX の前身である「NVP」へと発展させます。同時に OpenFlow と Open vSwitch のプロトタイプを作成します。2007 年夏のことで OpenFlow バージョン 1.0、Open vSwitch が完成する 2009 年よりも 2 年も前のことです。

ニシラはヴイエムウェアに約 1,200 億円で買収され（2012 年）、「SDN」は当初の想定を大きく超えネットワーク業界で一大ムーブメントとなります。特にアメリカではこの「ネットワーク仮想化」のコンセプトが大きな注目を集めました[※11]。

また、Mckeown と Shenker、Casado の 3 人は、ONF（Open Networking Foundation）の設立にも深く関わり、Mckeown と Shenker の 2 人は現在も、取締役（Board of Directors）です。

なお、日本では SDN よりもまず OpenFlow と説明されることが多々あり、結果的に SDN の意味が多義的に理解される要因となっています。SDN は Software-Defined Networking と多義的に捉えられる、一般用語の組み合わせであるため、新しい意味を与えることも考えることも可能です。しかしながら、本項で説明した通り、SDN を意味のある形で定義し、流行した「ネットワーク仮想化」のコンセプトを深く理解することは非常に有用です。

※11 Is Nicira's Network Virtualization a Game-Changer?（ニシラはゲームチェンジャーなのか？）
http://www.wired.com/2012/04/nicira-game-changer/（Wired, 2012, Acessed 2014）

Nick Mckeown 研究室の人々

Nick Mckeown 研究室から発したSDNのムーブメントでは、SDNの提唱者のNick Mckeown教授、Scott Shenker、Martin Casadoの3名が設立したNicira Networks（共同設立者兼CEO：Mckeown、共同設立者兼CTO：Casado、出資者：Shenker）が最も大きな流れといえます。また、この研究室の出身者は優秀で、この研究室で学んだ他の教え子はいずれも各SDNに関連した会社、製品を担当しています。その例としては、Big Switch Networks[※12]、NECなどの会社があります。しかし、現在有名なSDN製品の会社の源流がNick Mckeown研究室1つに行き着くことは面白く、珍しいことといえます。ちなみに、Nick Mckeown研究室と関連がない後発組には、シスコシステムズ社の出身者が多いPLUMgrid、日本にはミドクラ社、ストラトスフィア社などがあります。

1-4-2 OpenFlowの成り立ち

OpenFlowは、Nick Mckeown研究室で誕生し、Open vSwitchのコントロールプレーンプロトコルとして実装されています。Open vSwitchはパケット転送のソフトウェア実装です。

パケットのソフトウェア実装はネットワーク技術の歴史では珍しく、1984年にシスコシステムズ社が世界初のルーターを開発していますが、その後はほとんどハードウェア上のASICに実装されてきた歴史があります[※13]。最近ではヴイエムウェアのvSwitch（仮想スイッチ）がパケット処理のソフトウェア実装として存在しますが、あくまで物理スイッチの代替として、機能性・管理性にフォーカスした実装です。

ちなみに、vSwitchはPVLAN、RSPAN、IPFIXの機能を順番に実装し、また複数の仮想スイッチを管理可能なvDS（vSphere Distributed Switch）等を実装しています。

2000年代後半にOpenFlowが開発された経緯は、Linuxの普及とvSphere ESXiによるサーバ仮想化の隆盛とその一般化が背景にあります。Linuxやサーバ仮想化が普及した結果、研究目的でもハイパーバイザーによるサーバ仮想化が一般的となりましたが、当時のLinuxはLinux Bridgeのみしかなく、ネットワーク機能はほとんどなくVLANへのマッピングなどを行うのみで、高レイヤのネットワークサービスはすべてネットワーク基盤側で実装されていました。

こうした背景でさらに一般化したハイパーバイザー上の仮想マシン間で自由なネットワーク構築や、高レイヤのネットワーク機能の追加を求める流れが発生します。ハイパーバイザーはもちろんコンピューティング能力を仮想化する機能として存在しますが、同時にすべての仮想マシン

※12 Big Switch Networksの共同設立者Guido Appenzellerは、2014年10月ヴイエムウェアにCTOとして入社しています。

※13 シスコシステムズ社が最初に販売したマルチプロトコルルーター「AGS」（1986年発売）は、アメリカ・カリフォルニア州マウンテンビューのコンピューターミュージアムに収蔵されています。実物を見ると、この時代のルーターが非常に原始的で、パーソナルコンピュータに近いコモディティハードウェアとソフトウェアの組み合わせで実装されていることが分かります。当時マルチプロトコルルーターとして有名でしたが、マルチプロトコルの指す意味はOSPFやBGPなどルーティングプロトコルではなく、ルーテッドプロトコル（つまりIP、もしくはIPX、DECNetなどのこと）を指しています。最新の状況からは想像できないほど昔の話です。また、製品名の「AGS」はAdvanced Gateway Serverの略で、「ルーター」とすらいわれていません。

からのトラフィックの入出力が必ずハイパーバイザーを経由することも事実で、ハイパーバイザー上のソフトウェアで自由に制御したいという流れも自然なものです。つまり、サーバ仮想化が定着する 2000 年代半ばまでは、こうした需要は大きくなかったといえます。

上記の歴史的な経緯で OpenFlow は Nick Mckeown 研究室で考案され、コントローラ「NOX」・「ONIX」、仮想スイッチ Open vSwitch（OVS）として開発・実装されます。Casado をはじめ Ben Pfaff や Justin Pettit など、初期の OVS コントリビュータはすべて旧 Nicira Networks 社の開発者です。現在でも Open vSwitch の大部分のコミットは、ヴイエムウェアの NSX 開発部門の OVS チームが行っています。

● OpenFlow の特徴

OpenFlow とは、ハードウェア機器中心で静的に設計・設定されていたネットワークを、サーバ仮想化基盤に相応する動的変更を可能にすべく考えられた、パケット転送のためのコントロールプレーンプロトコルです。フローベースのエントリーで、宛先セグメントごとにエントリーを持っている現在のルーティングテーブルを利用した物理的スイッチよりも粒度が細かいことが特徴です。また、Open vSwitch は OpenFlow の最も有名な実装で、Linux に標準採用されています。

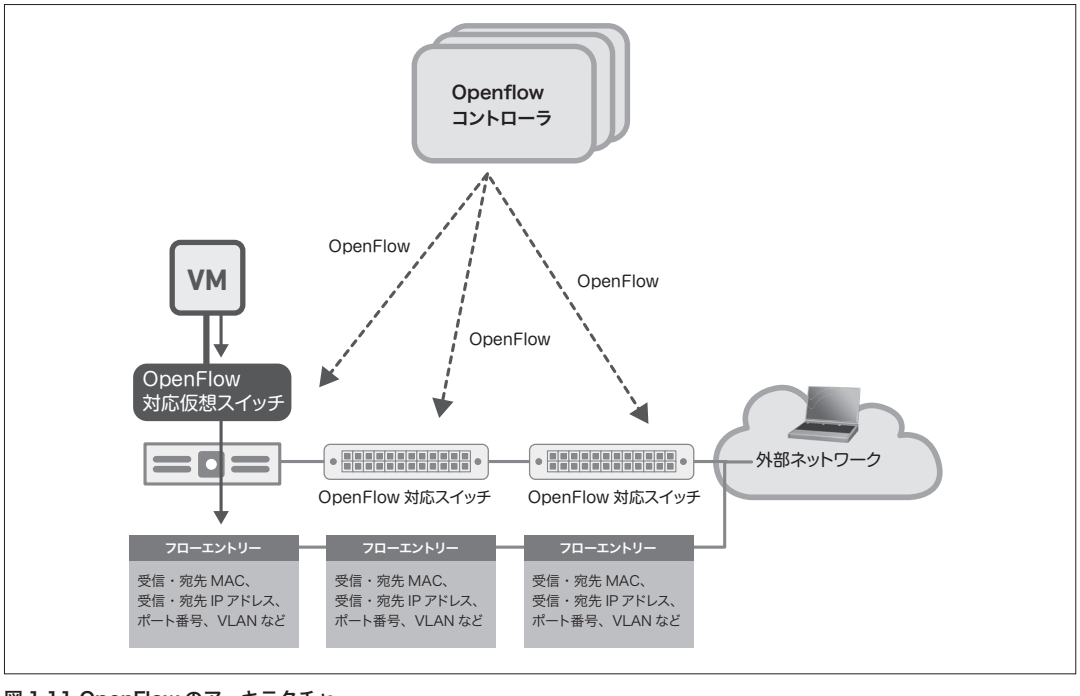


図 1.11 OpenFlow のアーキテクチャ

このフローベースの特徴のため、物理スイッチで OpenFlow を利用すると膨大なエントリーとなり、現在のスイッチと同様のパケット転送を行うには高価な TCAM [※14] を搭載する物理スイッチが必要となります。これは何の用途のために OpenFlow が生まれ、どのように利用されるべきかを考える際に重要な特徴です。

OpenFlow のフローエントリーは、送信元・宛先 MAC、送信元・宛先 IP アドレス、ポート番号などヘッダ情報をタプルと呼びます。従来のレイヤ 2 スイッチング、L3 ルーティングなどとは異なり、OSI レイヤの階層の区別なく同時にタプルを読み込み、処理にしたがいパケットヘッダの加工が行われます。最終的に決定した処理内容にしたがって転送処理されます。レイヤ 2-4 のフローベースの処理のため、従来と比較するとエントリーは細かい粒度となります。また、ロー

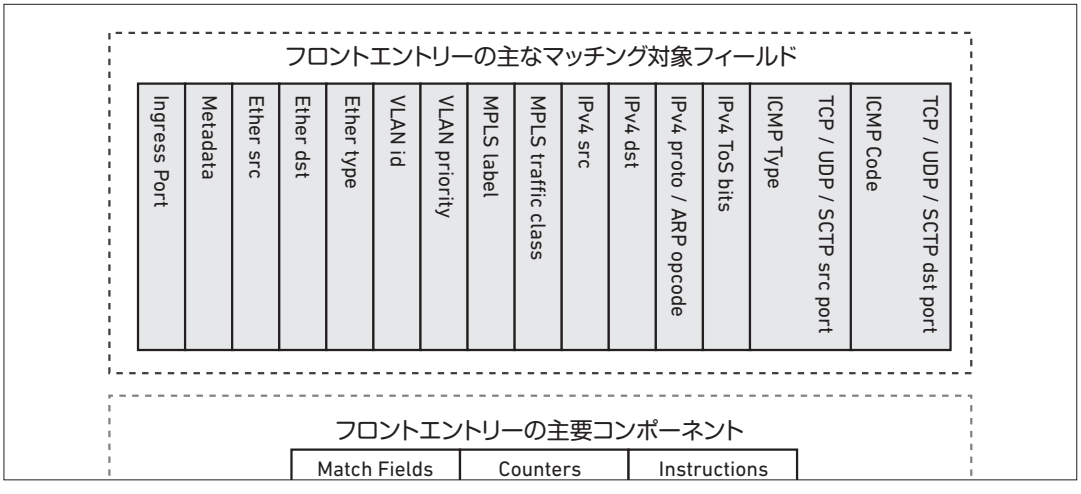


図 1.12 OpenFlow プロトコル - フローエントリー

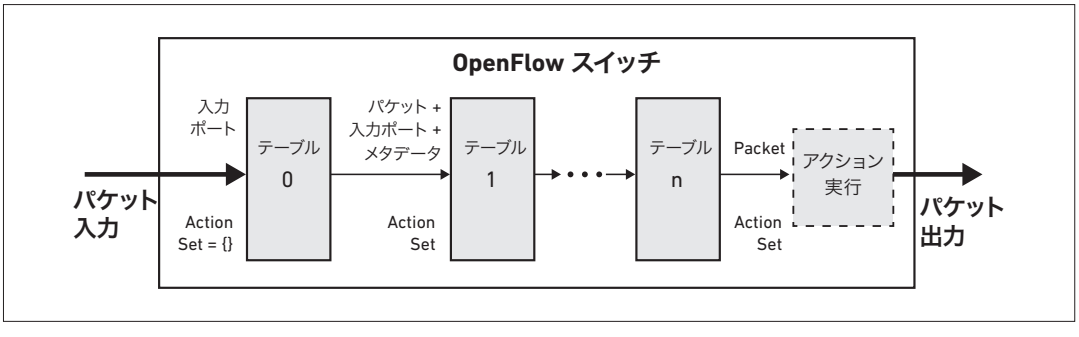


図 1.13 OpenFlow スイッチ内のパケット処理

※14 TCAM = Ternary Content Addressable Memory の略。マルチレイヤスイッチの MAC アドレス転送、QoS、ACL などに使われるメモリのこと。単機能で非常に高速に利用できるが、非常に高価なメモリ。

カルにデータベース（Open vSwitch では conf.db）を持ち、OpenFlow コントローラと呼ばれる管理ノードから OpenFlow スイッチへ向けて、フローエントリーがアップデートされます。コントローラはすべての OpenFlow エージェントを管理するため、集中した一元管理が可能です。

プロトコルの定義はバージョン 1.1 で、次のヘッダ情報（タプル）の処理が可能です（図 1.12）。

OpenFlow スイッチは受信したパケットを内部のフローテーブルから生成されたルールを逐次処理します。すべてのルールにマッチしない場合、パケットは廃棄（バージョン 1.0）、もしくは OpenFlow コントローラへ問い合わせします。（バージョン 1.x）

フローエントリーを獲得する方式には、リアクティブ方式とプロアクティブ方式があります。その名前の通り、フローエントリーをいつ獲得するかによって決まっています。

リアクティブ方式は、パケット受信後そのタプル処理をコントローラへ問い合わせます。最初のパケットもしくは宛先処理ができないパケットは、すべてコントローラへ問い合わせることになります。一方、プロアクティブ方式は、コントローラへ設定された時点で宛先を決め、事前にフローエントリーが配信されます。受信パケットは最初からその決められた宛先へ転送されます。

このフローエントリーの実装の方式は、何に対して OpenFlow を使い、どのようなユースケースに対して使いたいか、その目的に強く依存します。具体的には、オーバーレイモデルは事前に仮想ネットワークが定義されているため、プロアクティブ方式によって事前にフローエントリーが配布されます。反対に、Hop-by-Hop モデル（1-4-3-2 で後述）はネットワークそのものに重点が置かれ、エンドノードの情報（IP アドレスや MAC アドレス）を知っていることを前提とはしないため、必ず “Unknown”、つまり OpenFlow スイッチが宛先を知らない場合が発生します。

この場合、通常のネットワークのブロードキャストに相当するコントローラに問い合わせることでパケットの宛先を決定します。OpenFlow は、本項で述べた特徴から、実際に適用してメリットのある場合だけでなくデメリットもある場合もあり、ユースケースに用途の向き不向きがあります。この点は次項で説明します。

2009 年の誕生以来 OpenFlow は継続して成長しています。既に、ONF 参加企業も約 150 社におよび、プロトコルバージョンも 1.4 となっています（2014 年 10 月現在）。

OpenFlow 出現と SDN でのコンテキスト

SDN が話題になり始めた 2011 年に最初の Open Networking Summit が開催されています。Nick McKeown が行った基調講演では SDN の明確な定義が語られています。英語による基調講演ですが、一見の価値があります（本章で述べている点が改めて確認できます）。

ONF 2011 の基調講演「How SDN will Shape Networking」（http://www.youtube.com/watch?v=c9-K5O_qYgA、2011, Accessed 2014）

1-4-3 OpenFlowを利用したネットワークアーキテクチャ

OpenFlow/Open vSwitchの策定・開発をリードしたニシラの場合、OpenFlowのソフトウェア実装 Open vSwitchをオープンソースとして公開しました。ニシラの唯一の製品はNVPで、OpenFlowやOpen vSwitchは製品化されていません。OpenFlowとOpen vSwitchは、NVP内部で製品の一部（サウスバンド）として利用しています。

NVPの主たる製品価値は、OpenFlowコントローラによりハイパーバイザーに搭載される仮想スイッチを高度に管理し、カプセル化プロトコルによりオーバーレイモデルで接続し、自由に論理ネットワークを構成することです。世界初のネットワーク仮想化製品（かつOpenFlow・SDN製品）として2011年に正式リリースしています[※15]。なお、NSXのアーキテクチャは「Chap.05 NSX技術解説」で詳述します。

OpenFlow/Open vSwitchが話題になった当時では、ソフトウェアによるデータプレーンの実装を利用したネットワークアーキテクチャには2つの方法が存在しました。既出のオーバーレイモデルとHop-by-Hopモデルです。本項ではそれぞれのアーキテクチャの特徴を比較します。

●オーバーレイモデル

オーバーレイモデルの特徴は次の通りです。

- ハイパーバイザー上の仮想スイッチは仮想マシンのトラフィックが必ず経由するコンポーネントであるため、エッジデバイスであるハイパーバイザーへインテリジェンス（機能性つまりオペレーショナルステート）を搭載することを主眼とする。
- 物理ネットワークトポロジーは、ルーティングステートだけを管理するシンプルな構成となる。
- 中継するネットワーク機器でOpenFlow処理等、何か特別な機器は必要としない（IP接続性のみが必要）。
- フローテーブルへのアップデートも設定を施した時点で可能なため、事前にフローエントリーを作成することが可能（プロアクティブ型）。

● Hop-by-Hopモデル

Hop-by-Hopモデルの特徴は次の通りです。

※15 Nicira NVP（Network Virtualization Platform）2011年8月に1.0を正式リリース。現在のVMware NSX Multihypervisor。

1. パケット転送する主体を、従来と同じネットワーク機器（スイッチ）を対象とする。
2. OpenFlow を搭載したスイッチは、入力されたパケットを OpenFlow テーブルにしたがい処理し、次の宛先ポートを決定する（従来のルーターやスイッチと同様の役割）。
3. 上記 2 の理由からどのルールにもマッチしないパケット（最初に入力されるパケット、もしくは宛先の分からない "Unkown" なパケット）は、OpenFlow コントローラへ問い合わせが必要になる [※16]。
4. したがって、上記 1 の特徴から、中継するネットワーク機器をすべて（少なくともインテリジェンスを持つコア網は）OpenFlow 機器への置き換えが必要となる。トンネルを利用して物理的に既存スイッチを利用することも可能だが、その場合は該当スイッチでしたい OpenFlow 処理が行えない。

表 1.4 オーバーレイモデルと Hop-by-Hop モデルの比較

比較項目	オーバーレイモデル	Hop-by-Hop モデル
アーキテクチャの主体	エッジであるハイパーバイザー	ネットワーク機器
既存ネットワーク機器との互換性	あり。 既存の IP ネットワークをそのまま利用する	なし。 OpenFlow 対応機器へすべての機器を変更する
ハイパーバイザー	ハイパーバイザーは仮想ネットワーク上必須コンポーネントなので変更が最小	ハイパーバイザーには手を加えない
フロー単位でパケットをコントロールするか	しない。 トラフィックエンジニアリングは物理層で行う	フローベースのトラフィックエンジニアリングを行う
拡張性・パフォーマンス	パケット転送しているだけなので、OpenFlow のエントリーはアグリゲーション可能。拡張性、パフォーマンスに対する不安は少ない	フローベースでパケット処理、トラフィックエンジニアリングできることが価値なのでアグリゲーションは難しい。拡張性、パフォーマンスに対する不安が大きい
拡張性	プロアクティブ型なのでフローエントリーは事前に配布済み。したがって宛先は学習済み	ネットワーク機器に実装する以上、最初のパケットは Unknown となり、コントローラが回答する必要がある。したがって、処理の拡張性、処理方法について十分な考慮が必要
トラブルシューティング	シンプルな送受信先の IP のみの管理をエッジ 1 箇所に実装しているだけなので比較的容易	最終的にはネットワークのトポロジーとそのルーティングをフローベースで確認する必要がある。IP ルーティングのトラブルシューティング方法を OpenFlow ネットワークでも適用する必要がある

※ 16 上記 3 の指摘は、OpenFlow スイッチの存在意義を考える上で重要です。
OpenFlow スイッチは、通常のネットワークと同様にパケットを受信して処理しているため、ARP などブロードキャストや Unknown ユニキャストなどによる攻撃を受ける可能性があります。通常のネットワーク機器でも 1990 年代に問題になったことがあります。当時ではまだ性能の高くないスイッチ（当時まだ ASIC で処理されるスイッチでもノンブロッキングスイッチが存在）の CPU 処理を専有し、スパニングツリー処理ができない状態が発生することがありました。OpenFlow スイッチも同様で、ネットワーク上に配置するネットワーク機器であるため、最初のパケット、外部から入力される Unknown なパケットを避けることはできません。また、OpenFlow コントローラはソフトウェア実装のため、そのパフォーマンスも問題になります。コントローラの拡張を考える場合でも、フローエントリーの粒度が細かく膨大なエントリーを処理する必要があるので、複数のコントローラ間でのエントリーの一貫性を保つことが難しいといえます。

対象となるユースケースと解決すべき課題で最適なアーキテクチャは変わりますが、問題の所在をデータセンターとして、前節で言及したクラウドの躍進に伴う新たな課題を考慮すると、以下の観点からオーバーレイモデルが問題解決に有用と判断できます。

- そもそもデータセンターの課題を解決するために、IP プロトコルベースの既存機器すべてを交換するアプローチは現実的ではない。
  - ネットワークを OpenFlow 対応とする場合、既存の IP ベースのネットワーク機器をすべて（もしくは OpenFlow 導入の意味がある相応規模の機器を）OpenFlow 対応スイッチに変更する必要があり、ネットワーク基盤の大規模な変更が必要となる。
  - TRILL などの新技術を搭載したネットワーク機器を新規に導入する場合よりも、エンジニアがプロトコルやオペレーション以外に、OpenFlow やエントリー処理の仕組みなど、新しい技術を基本から理解する必要があるため負荷が高い。

- トラフィックを自由に操作できることは、解決したり定義された問題の解決そのものではない
  - Hop-by-Hop モデルの場合、MPLS-TE のようなトラフィックエンジニアリングが可能だが、ネットワークのみに注目してトラフィックエンジニアリングできること自体は、データセンターネットワークの解決すべき問題とは関連がないため、優位なメリットではない。

- OpenFlow がフローベースの処理だからトラフィックエンジニアリングができ、セキュリティポリシーが適用可能であることと、その結果膨大に増える拡張性を同時に兼ね備えることが難しいこと（OpenFlow 対応の特別な高価な TCAM かフローエントリーの集約が必要）
  - トラフィックエンジニアリングと類似する観点だが、OpenFlow はフローベースの粒度でエントリーを持っているため、セキュリティに関しても同様に細かい適用が可能とされる。しかしながら、ネットワーク基盤に適用する Hop-by-Hop モデルでは膨大なフローエントリー処理を要し、既存ネットワーク機器と比較して膨大な量のエントリーを持つ必要がある。
  - 高価な TCAM を搭載した OpenFlow スイッチのみで OpenFlow ネットワークを構築するか、フローエントリー集約のいずれかが必要となる。前者は、OpenFlow スイッチの価格が 2 ～ 3 倍になることが予想され非常に高価になる。後者はそもそもフローエントリーをネットワークサマライズ同様に集約することになるので、フローベースの処理ができなくなる。

- 問題が発生した場合、Hop-by-Hop モデルでは最終的にネットワークのトポロジーとそのルーティングをフローベースで確認する必要がある（オーバーレイモデルではシンプルな送受信先の IP アドレスのみの管理をエッジ 1 箇所に実装しているため格段に容易）。

前述の通り、ネットワーク基盤のみを対象とする Hop-by-Hop モデルは、既存ネットワーク技術と比較するとメリットが少ない割りに課題が多く、一般的な IaaS 基盤のユースケースを想定した場合では導入は難しいといえます。一方、オーバーレイモデルのソリューションは、仮想化されたサーバ基盤とそのネットワーク基盤の問題点を解決する点で有効です。

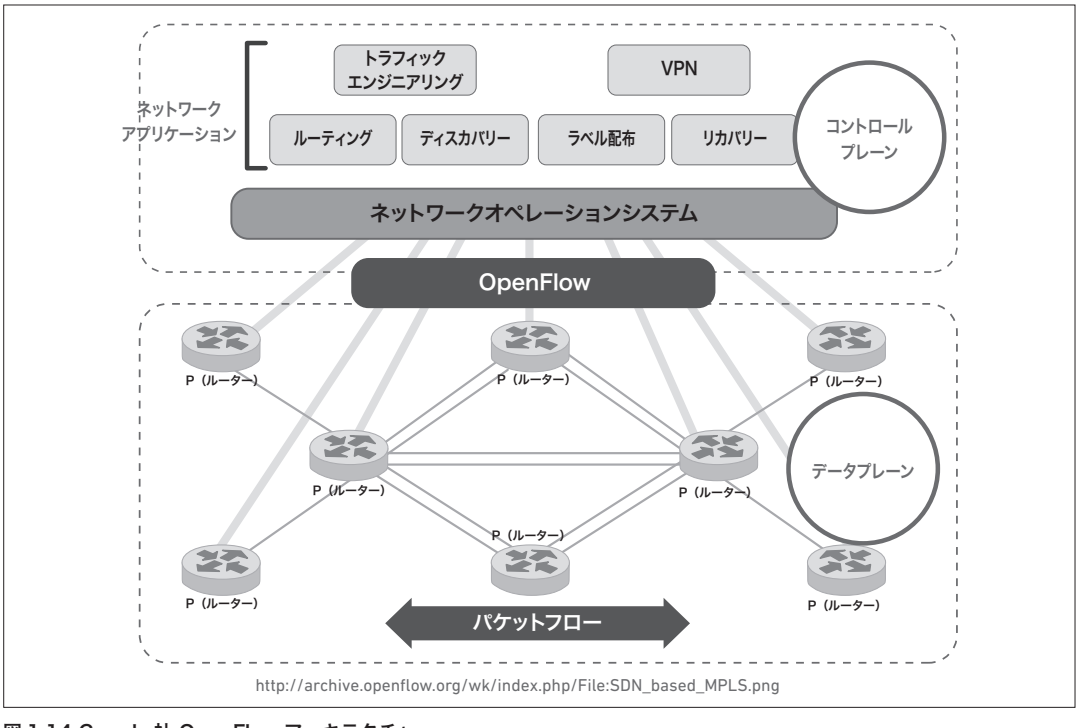
● SDN のその他のユースケース

その他のユースケースを考えた場合、Hop-by-Hop モデルが有利かもしれません。2 例ほど掘り下げて考えてみます。

データセンター間通信のレイヤ 2 延伸

データセンター間の通信を OpenFlow スイッチで自由にトラフィックエンジニアリングが可能であれば、データセンター全体の負荷分散や BC/DR [※17] などさまざまな用途に利用できそうです。

Google が 2012 年にデータセンター間を OpenFlow を利用して接続し、データセンター間で負荷分散をはかっていることが話題になりました（図 1.14）。



http://archive.openflow.org/wk/index.php/File:SDN_based_MPLS.png

図 1.14 Google 社 OpenFlow アーキテクチャ

※17 BC/DR = Business Continuity/Disaster Recovery の略。事業継続・災害対策

Google では、OpenFlow をマネージメント制御として利用しており、その配下には通常のルーターに対する BGP の設定を作成していることが分かります。つまり、コントロールプレーンへの操作であり、データプレーンの動作は行っていません。データセンターでは BGP の設定などにより作り出されたルーターの FIB テーブルでハードウェア転送が行われています。

上記に類似したアーキテクチャを採用した SDN 製品があります（ジュニパーネットワークス社「Contrail」、Nuage Networks 社など）。これらの製品も含めて、なぜ OpenFlow をデータプレーンで利用しないのでしょうか。OpenFlow はフローベースのエントリーであるため、フロー数によってエントリーが莫大に増大し、想定しているユースケースがデータセンター間のため、最も集積したトラフィックフローとなり、現実的に処理できない可能性があるためです。

Google 社をはじめ、ジュニパーの Contrail や Nuage Networks 社は、自由化が主目的であり、既存プロトコル利用における安定性とフロー数双方の観点で、同様のアーキテクチャを採択していると考えられます。OpenFlow スイッチのみで構成するデータセンター間ネットワークは、フローエントリーを集約すれば同様のことは可能ですが、その場合、OpenFlow が本来処理できるフローの粒度で処理できないため、わざわざ OpenFlow を採用する意義が少ないといえます。

キャンパスネットワークなどより小さなスイッチの統合管理

キャンパスネットワークなどの場合は、上記で説明した争点が該当しないケースもあり、メリットとデメリット双方を深く検討する必要があります。キャンパスネットワークの場合、小規模で OpenFlow スイッチとフローのエントリーを抑えることができ、管理が容易になるというシナリオです。この場合、フローベースでは複数の OpenFlow 機器のトラブルシューティングに手間が掛かります。また可視化のためのトラブルシューティング補助ツールなどが存在するかどうかが重要になってきます。しかし、これらはまだこれからの分野です。

また、フローベースのトラブルシューティングと、対応可能なエンジニアの育成には手間が掛かるため、そのような運用までを見据えたメリット・デメリットを想定し、検討することが必要です。

ここに挙げた例では、そもそも対象としている課題がデータセンターネットワークの複雑性ではありません。特に Google 社は、データセンター間のネットワークトラフィック分散と効率的な運用がネットワーク上の課題であることは明らかです。ただし、本書では、「1-4-1 SDN の成り立ち」で説明した通り、データセンターネットワークの複雑性を解決すべき課題と考えています。言い換えると、ヴイエムウェアがフォーカスするのは仮想化されたデータセンター（つまり SDDC）であり、ネットワークそのものではありません。