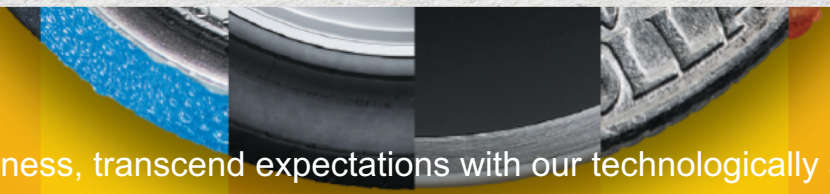
NTT Communications
Transform. Transcend.

エンタープライズ向けクラウドのSDN基盤の安定化への挑戦

SDN Tech Lead 川上 雄也
SDN Engineer 梶浦 悠生



Transform your business, transcend expectations with our technologically advanced solutions.

Enterprise Cloud 2.0 (ECL2.0)

NTT Communications
Transform. Transcend.

Enterprise Cloud Knowledge Center

English　お問い合わせ　ポータルへログイン

特長　サービス　料金　導入事例　お知らせ　サポート情報

キーワード検索

ホーム > 故障・メンテナンス情報（サービス稼働状況）

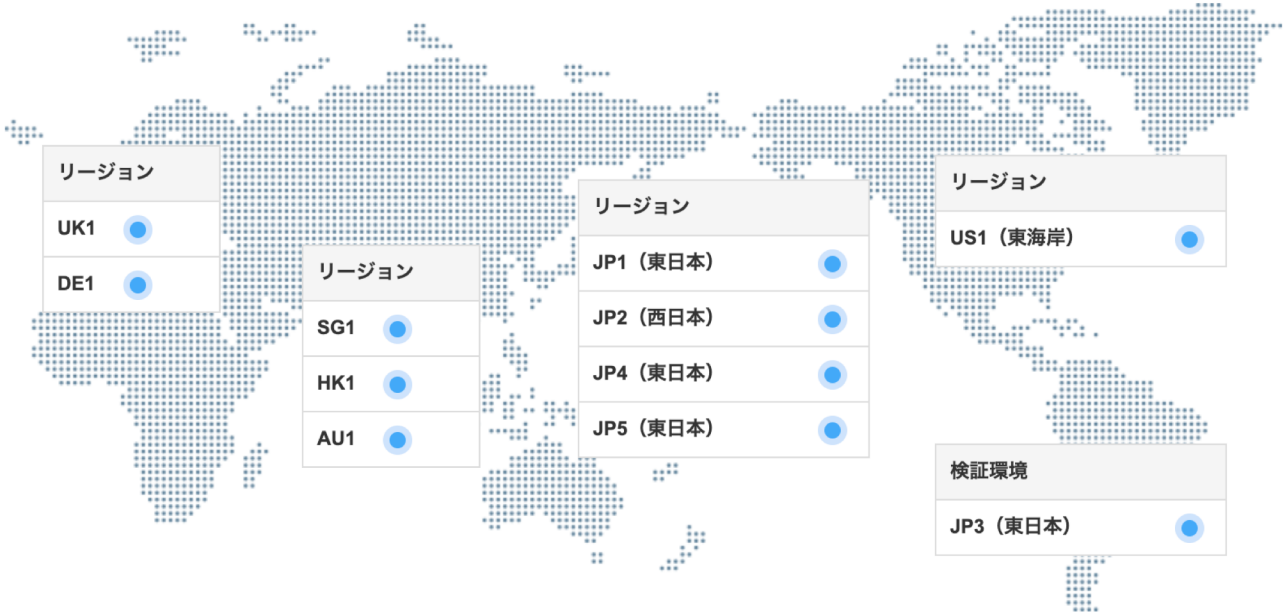
現在の状況

履歴

正常稼働中

現在、Enterprise Cloud 2.0の各サービスに故障情報はありません。

JP4リージョンは、ただいま定期メンテナンス時間帯です。



最終更新日時　2019-07-16 04:24:02

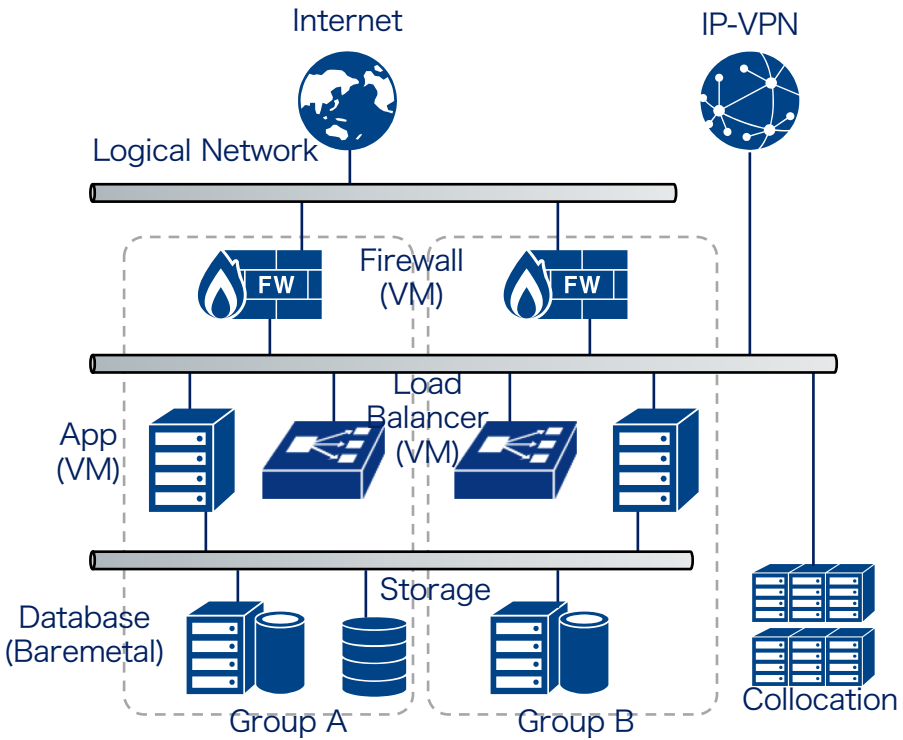
● 正常稼働中 ● メンテナンス中 ● ポータル・APIによる参照／操作不可 ● 故障発生中

NTTコミュニケーションズのクラウド

	サービス	サービス開始	提供方式
Cloud n NTT Communications	Cloud n	2012年3月	Public 個人/法人
Enterprise Cloud	Enterprise Cloud 1.0	2012年6月	Closed 法人向け
	Enterprise Cloud 2.0	2016年3月	Closed 法人向け

Enterprise Cloud 2.0 の特徴

オンプレミスのネットワークをそのまま持ち込めるクラウド



特徴

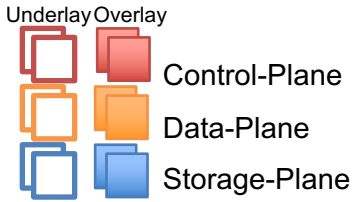
- 仮想的なL2ネットワークを自由に構築
- IPアドレスも自由に利用可能
- マルチキャストを利用可能
- グループ単位の物理的な冗長が可能

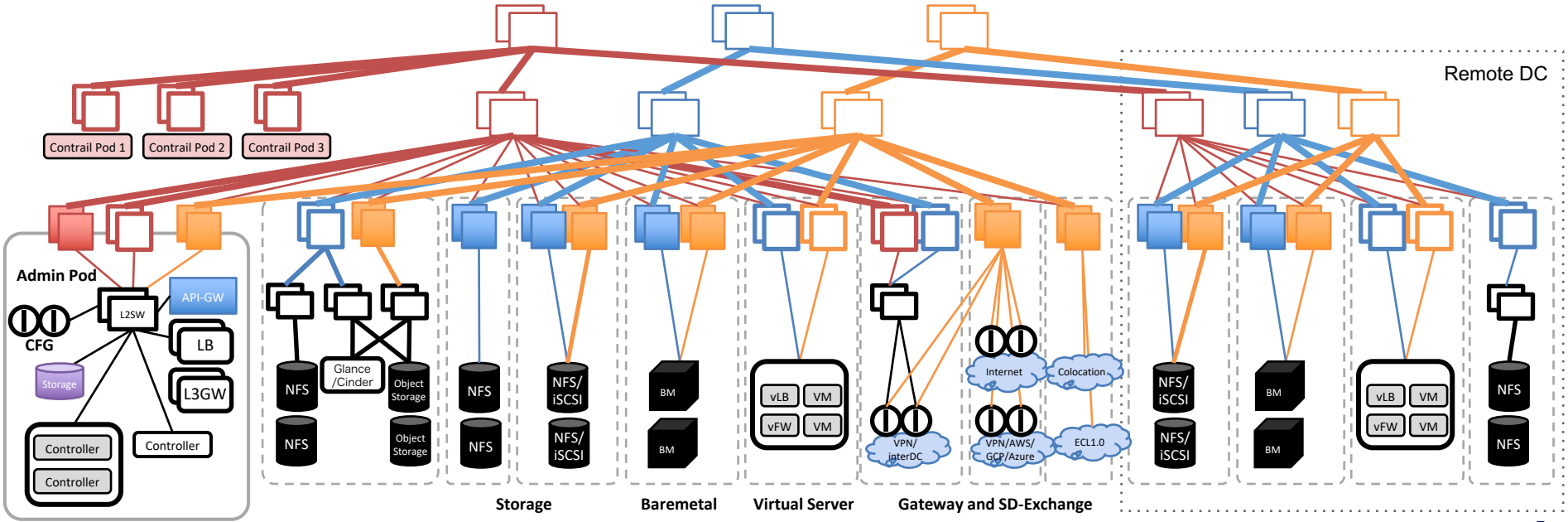
提供メニュー

- ベアメタルサーバ
- 仮想サーバ
- ストレージ
- インターネット接続
- VPN接続
- コロケーション接続
- 他社クラウド接続　　など

ECL2.0 Network Overview

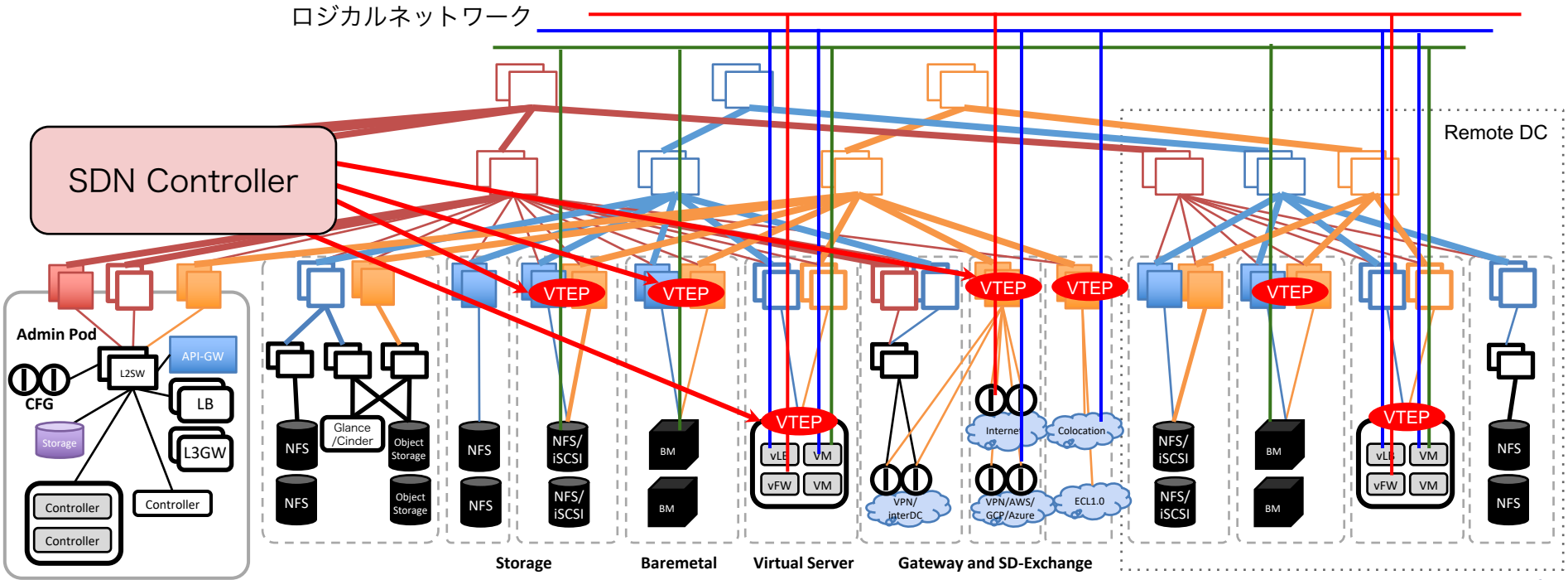
- Control/Data/Storageの3-planeに分離
- 各Planeは5-stage CLOSトポロジーでeBGPでマルチパス
- LeafスイッチはVirtual Chassisで筐体冗長
- スケールアウトしていく構成





ECL2.0 SDN Overview

- SDNコントローラがハードウェアVTEPおよびソフトウェアVTEPを動的に設定
- VXLANの仮想L2ネットワークをオンデマンドに作成する
- VTEP同士がEVPNでMACアドレスの経路を広告し合いL2到達性を実現する



SDN Controller (OPENCONTRAIL)

NTT Communications
Transform. Transcend.

tungstenfabric

- SDNコントローラとしてJuniper社のContrailを採用
  - オープンソースのOpenContrailと同一実装
  - 現在はLinux Foundation下でTungsten Fabricとして開発
- Neutron互換のネットワーク・オーケストレーターがContrailを含めたネットワークサービス機能を管理

Version	Release	Note
R5.1 (R1907)	2019-07	EVPN Route Reflectorのサポート
R5.0	2018-04	完全コンテナ(Microservices)化
R4.1	2017-11	Hardware VTEPに対するEVPNのサポート
R4.0	2017-10	コントローラのコンテナ化
R3.2	2016-12	ISSU/Multi Queue/Graceful Restartのサポート

SDNのスケール

- Contrailのユースケースとしては世界最大スケール
- L2 Forwardingをメインで使用
- ハードウェアVTEPを多用

ソフトウェアVTEP(Compute)数	660
ハードウェアVTEP(ToR SW)数	62
EVPN経路数	140,000
仮想ネットワーク数	7,000

※１リージョンあたりの最大数の概数
※罹障範囲分割のためスケール制限を実施中

SDNのスケールとクラウドビジネス

リージョンをIPパケットに例えるなら…

➔コントローラはヘッダ
➔テナントリソースはペイロード

如何にペイロードを大きくできるかがビジネスの鍵になる

そのためにはSDN基盤の**Scalability**が重要になる

そのためにはSDN基盤の**Stability**が超重要になる

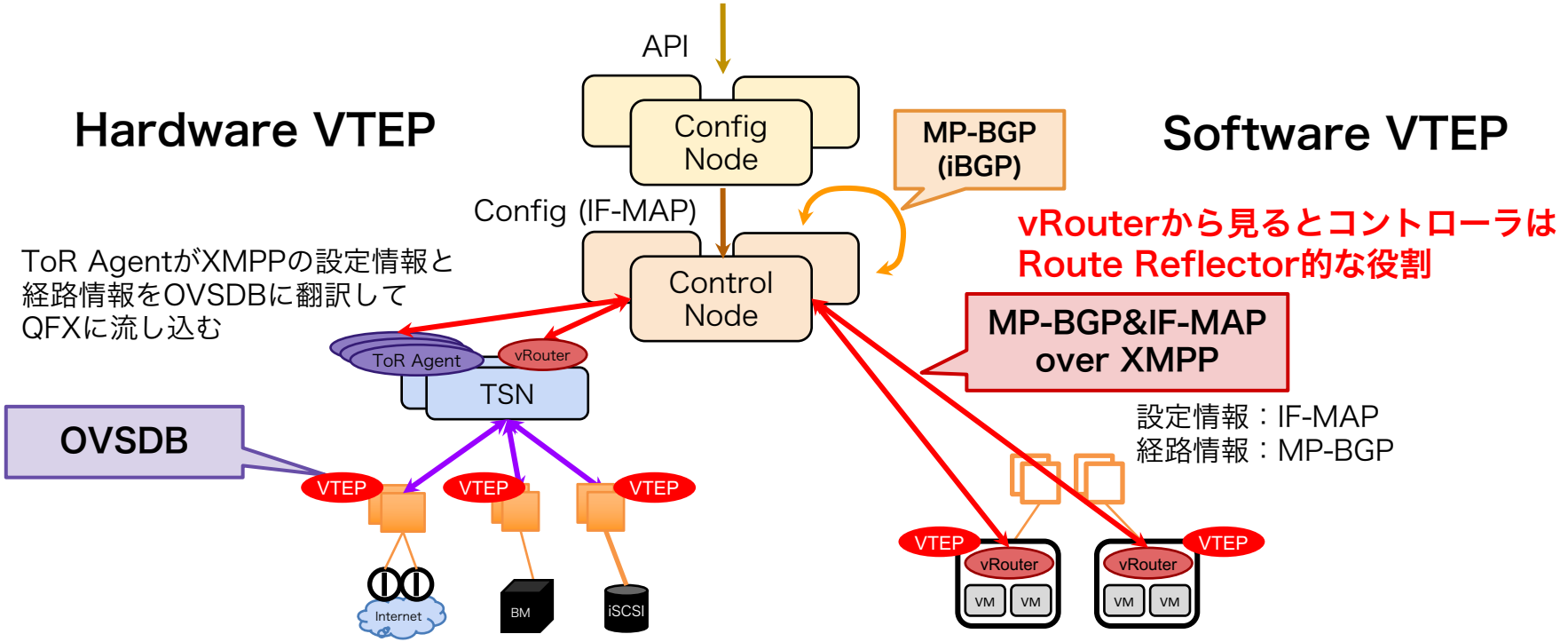
挑戦する品質

まずは…
　SDNコントローラの障害で通信断が起きないようにする

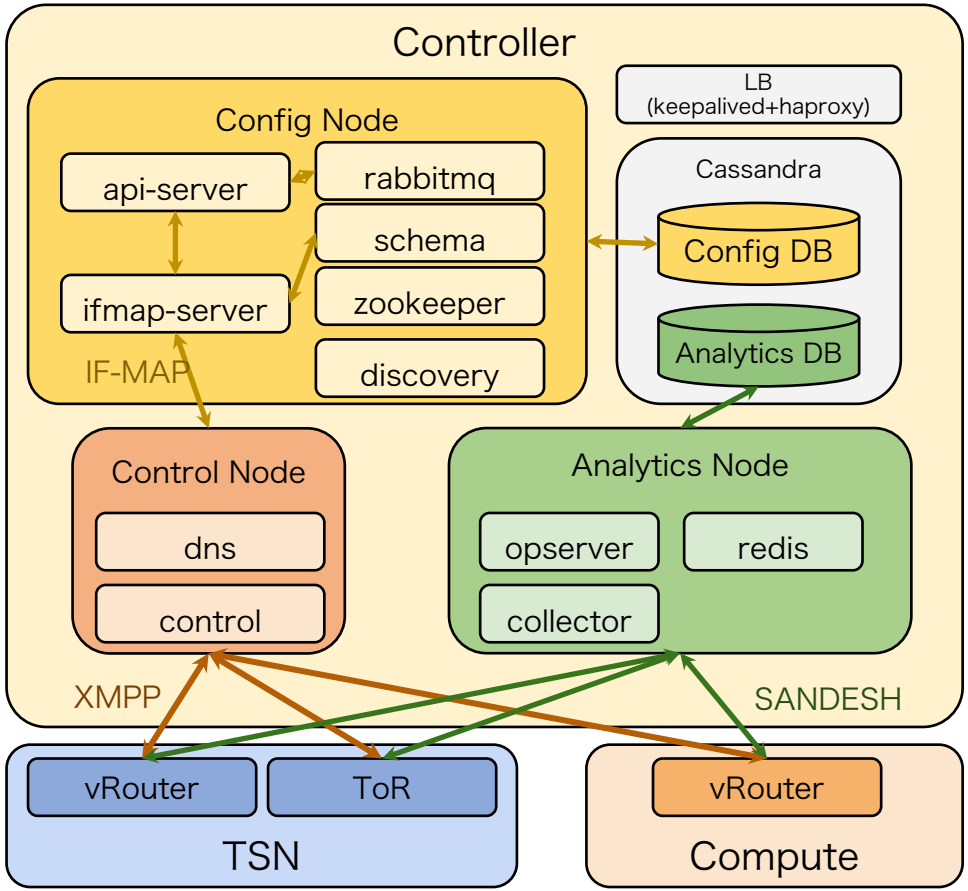
	ユニキャスト通信断	BUM通信断
SDNコントローラ障害・作業	1秒未満	30秒未満
NW機器・リンク障害・迂回	1秒未満	30秒未満
Compute障害・作業	当該機器 30秒未満	当該機器 30秒未満
HW VTEP障害・作業	当該機器 30秒未満	当該機器 30秒未満

Contrail C-Planeアーキテクチャ

- MP-BGP(EVPN)でMACアドレスのL2経路を交換することでL2到達性を確保
- ComputeやQFXとの間の経路交換や設定投入はXMPP/OVSDBを使用

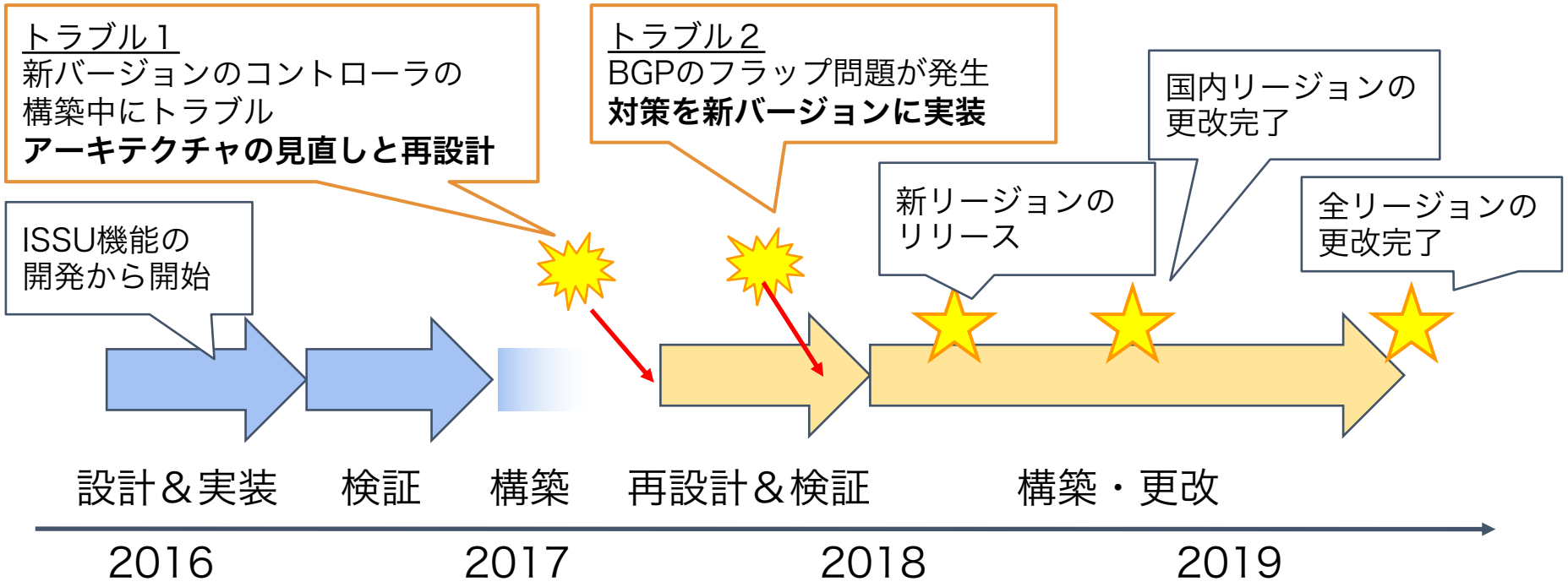


Contrail内部アーキテクチャ



SDN基盤の更改

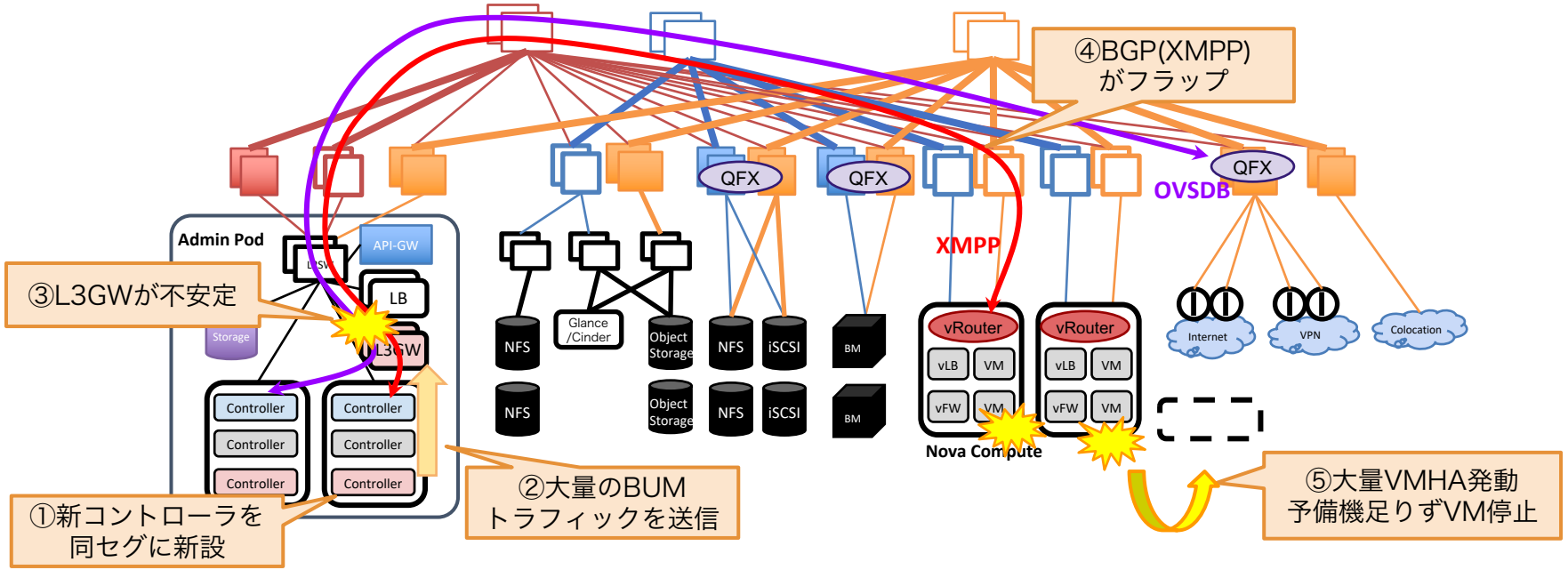
新バージョン構築中のトラブルにより設計・アーキテクチャを見直し、
全リージョンの更改を完了



事例1: 新コントローラ構築でBGPフラップ

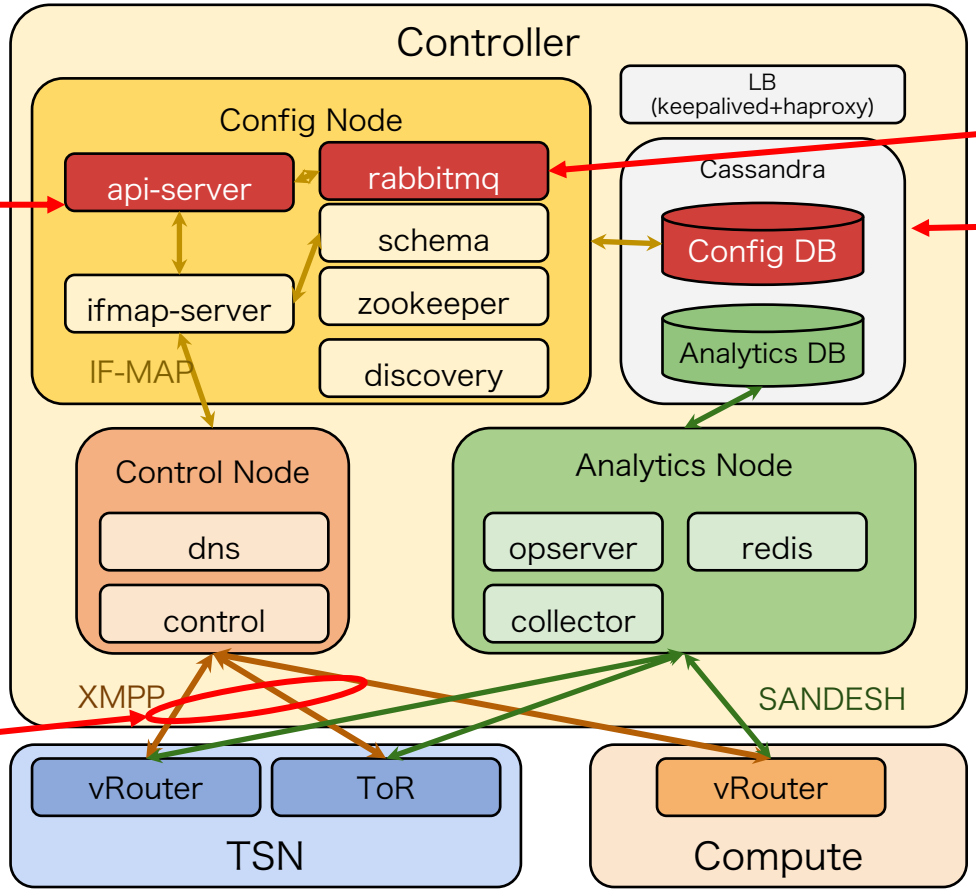
新設のコントローラの設定ミスでkeepalivedの不具合を引き、大量のBUMトラフィック(GARP/VRRP)が送信されたためGWルータが不安定になり、そこを通る既設コントローラとComputeやQFXとの間のネットワーク制御通信 BGP(XMPP)/OVSDB がフラップした。

BGPのフラップを検知してComputeノードの異常回復機能であるVMHAが働いて予備機にコールドマイグレーションを試みるもその数が多すぎて予備機が足りなくなり、大量のVMが停止状態になった。



事例2: Config機能の停止でBGPフラップ

治すために
ここを再起動
したり



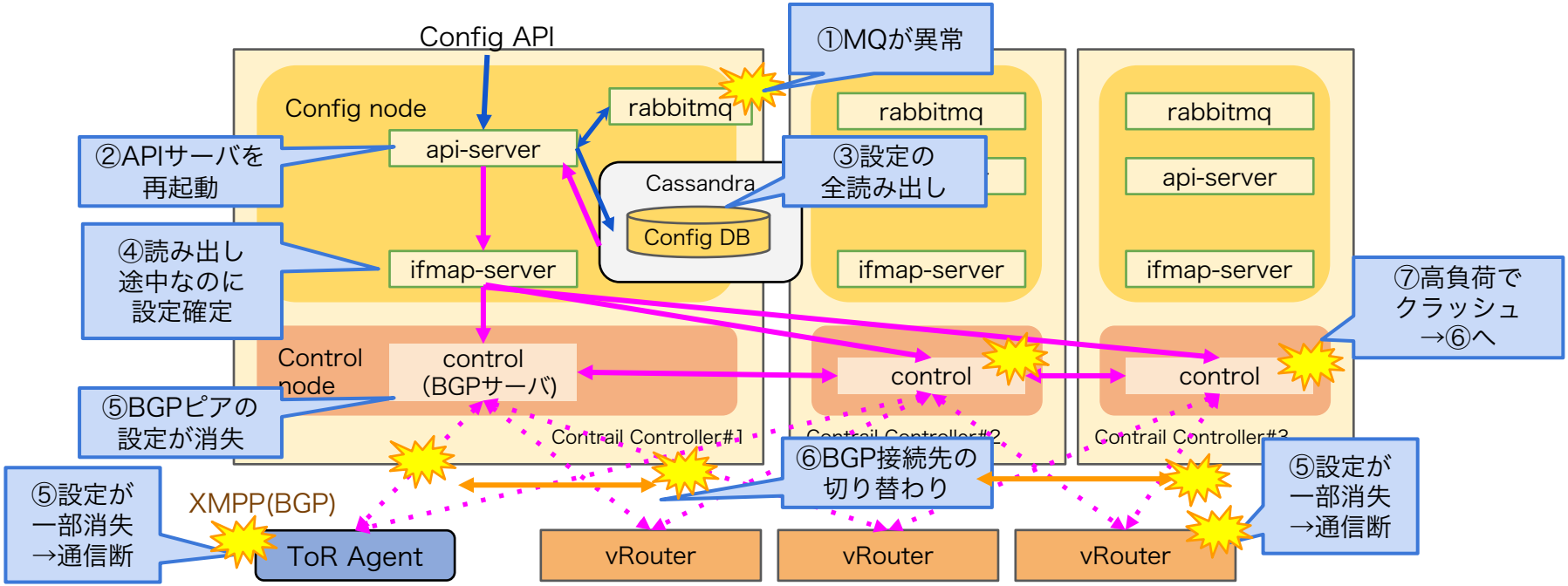
ここが壊れたり

ここが止まったり

すると、
ここが切れる

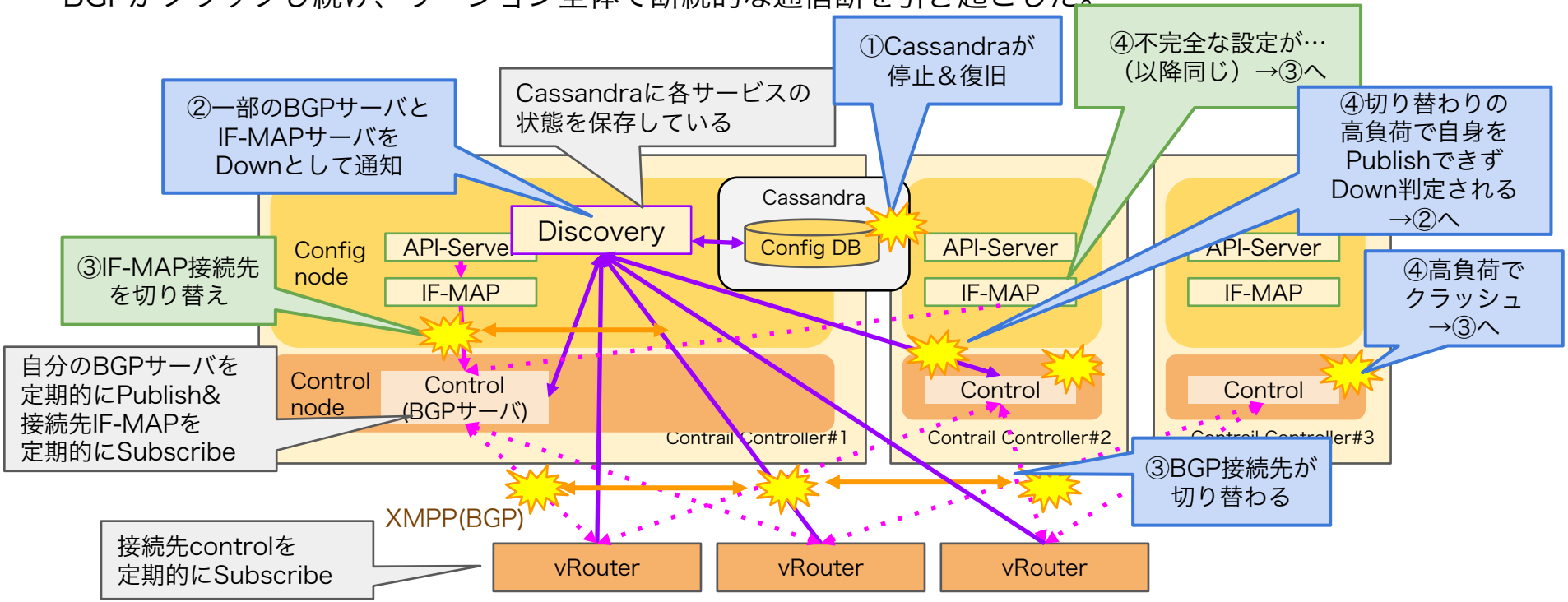
2-1: APIサーバ再起動でBGPフラップ

RabbitMQの状態異常でAPIサーバとの接続が機能不全になり、設定不能状態に陥ったため、APIサーバを再起動したところ、Cassandraから読み出し途中の設定が高負荷により不完全な状態でControlノードのBGPサーバに伝わり、BGPピアの切断やvRouterの設定の欠損が発生。さらにピアの切断が更なる高負荷を誘発しvRouterの接続先切り替わりや、BGPサーバ自身の高負荷によるクラッシュを誘発し、系全体でBGPのフラップが続き、リージョン全体で断続的に通信断を引き起こした。



2-2: Cassandra停止でBGPフラップ

Cassandraがハードウェア故障で停止＆自然復旧したが、サービス発見のためのDiscoveryサーバが中途半端に一部のサービスをDown状態として応答したため、それを受け取ったBGPサーバやvRouterがIF-MAPやXMPPの接続先を一斉に切り替えた結果、高負荷や設定の欠損を誘発して前ページ同様にBGPがフラップし続け、リージョン全体で断続的な通信断を引き起こした。

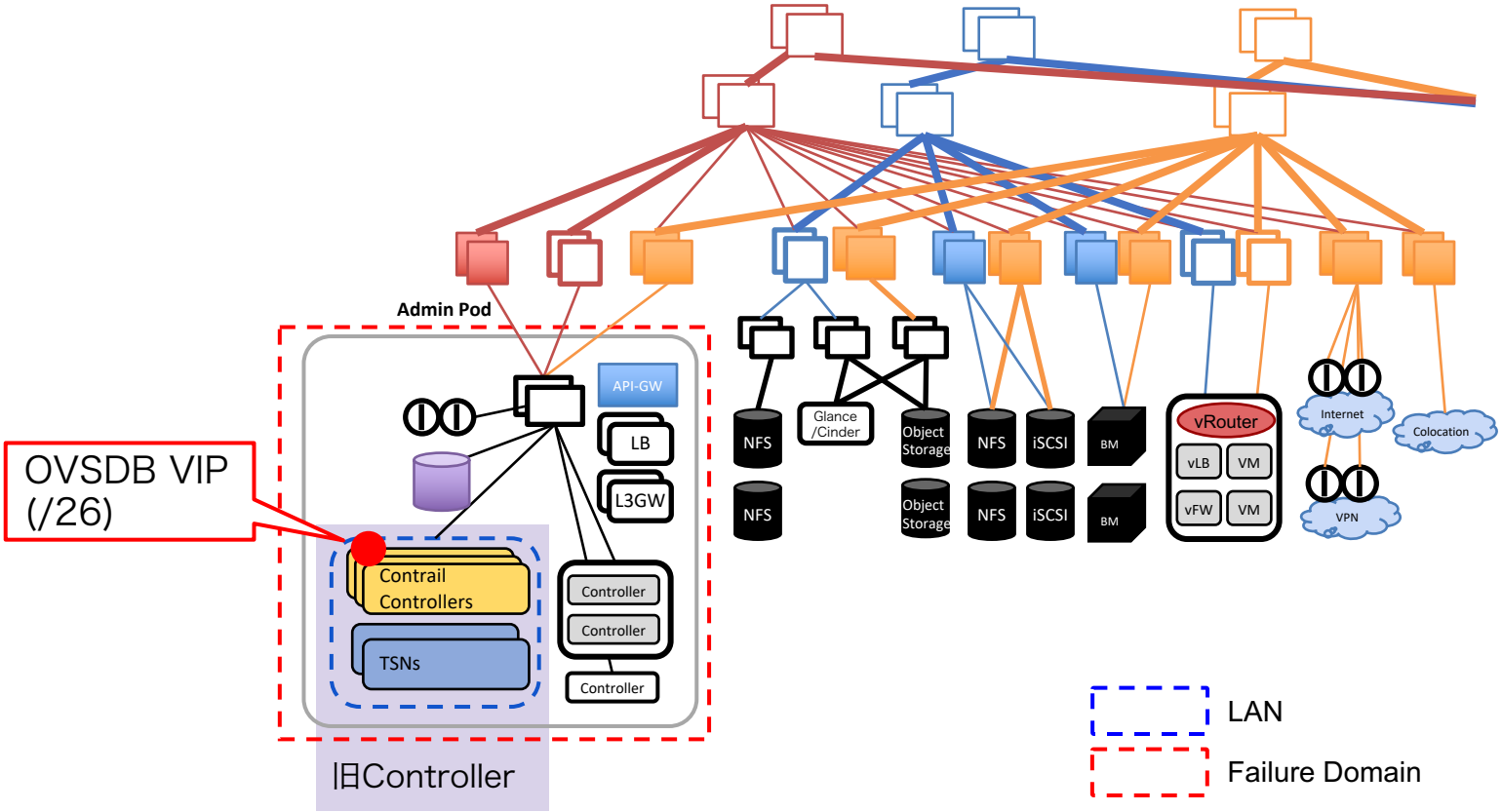


問題点と反省

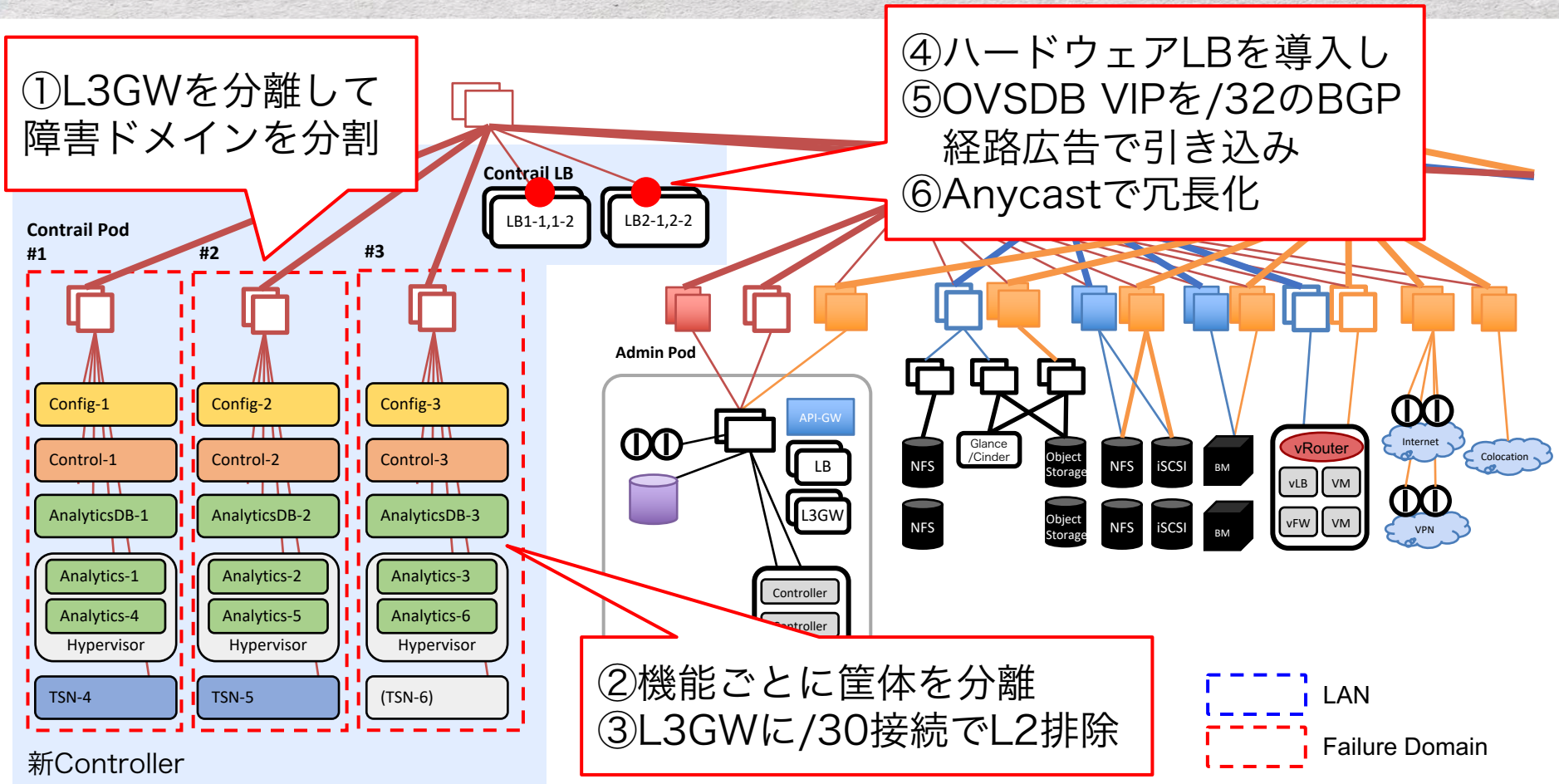
1. すべての機能が１つの筐体に存在した
   - ➢ Analytics(統計)の負荷がControl(BGP)に悪影響を与えていた
2. 旧コントローラと同じL2に新コントローラを構築した
   - ➢ OVSDB VIPを同番移行する必要があった
3. ３冗長のコントローラが１つのL3GWの配下にあった
   - ➢ VRRPで冗長を取る必要があった

プロダクトの当初設計に従ってそのまま構築していた

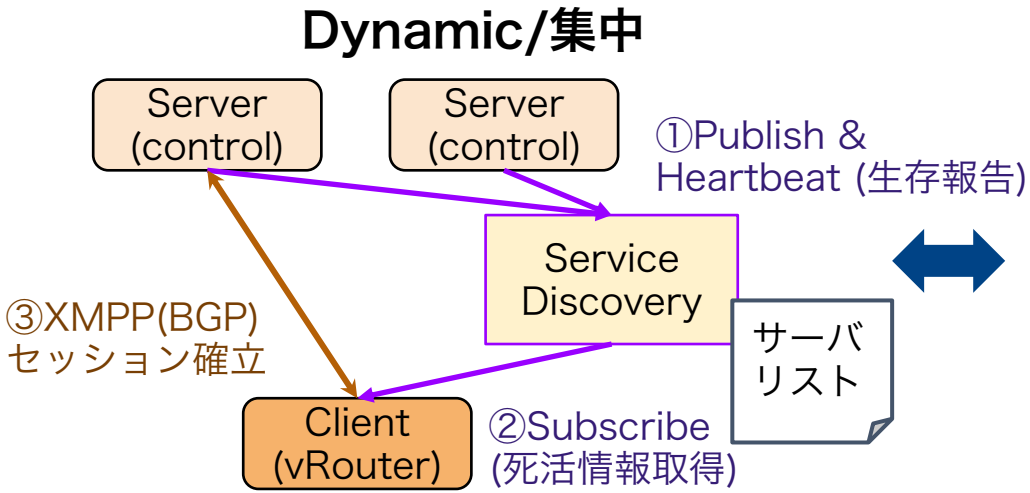
旧バージョンのSDNコントローラ構成

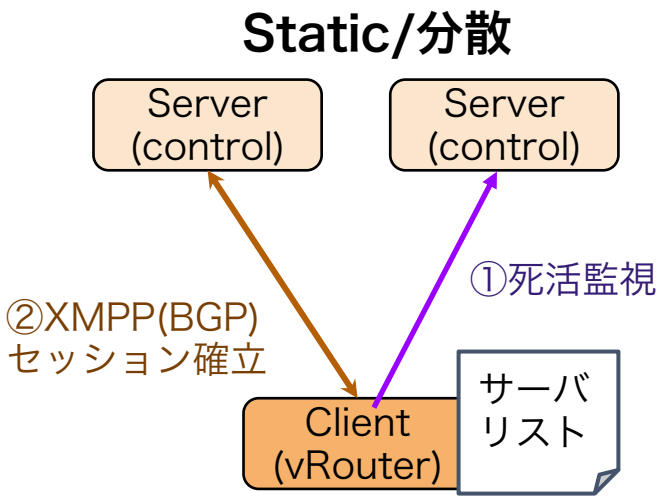


再設計後の新バージョンのSDNコントローラ構成



Service Discoveryの難しさ (1/2)





利点

- 動的にサーバを追加変更しても クライアントの設定変更が不要
- セッション迂回がSDの操作で実現可能

- 自律分散的に動作するので系全体が不安定になりにくい

欠点

- Service Discoveryが不安定だと 系全体が影響を受ける
- Heartbeatタイマーの調整が難しい
- 死活監視とセッションが別のパスになる

- リストの更新作業がスケールしない
- 迂回がセッション切断になる

Service Discoveryの難しさ (2/2)

BGPフラップの事例2では…

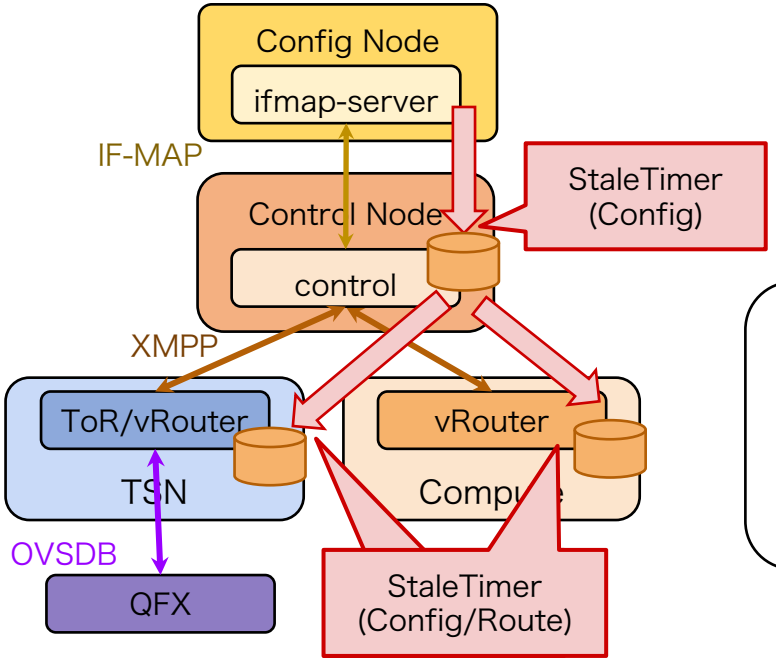
1. BGP経路処理とDiscovery Heartbeat(生存報告)を同じ優先度で行っていたために、BGPの負荷が高くなるとHeartbeatを送信できない状態に簡単に陥った
2. タイマーが短かったため(15秒)高負荷になるとすぐにDownとして扱われてしまい、BGPがフラップした。

Heartbeatの優先度を上げて、タイマーを伸ばした(10分)

そもそもそんなに頻繁にサーバ(RR)の台数は変わらない…

※Contrail 4.0からはService Discoveryの機能が除外されすべてStaticな設定になりgraceful restart機能が実装された

Stale Timerの難しさ (1/2)



BGPのGraceful Restartで使われている仕組みがBGP経路だけでなく設定情報の伝達でも使われている

サーバ再起動や接続変更の後も、元の設定・経路をStale状態で持ち続け、一定時間内に受信した新しい設定・経路を改めてCommitし、Stale状態の古い設定・経路を破棄する。

Stale Timerを長くすると…
古い設定・経路を参照し続けて、新規通信や経路変更に対応できない

⟷

Stale Timerを短くすると…
設定・経路を削除しやすくなりコントローラ障害時に通信断になりやすい

Stale Timerの難しさ (2/2)

BGPフラップの事例2では…

1. IF-MAPにはGraceful Restartで定義されているEnd-of-RIBのマーカーの仕組みが存在しないため、タイマーを使わざるを得なかった
2. タイマーがハードコードされていて、その値が我々のユースケースに合わなかった

タイマーをすべて設定可能にして、
我々のスケールに合わせて長くする方に調整した。

いろいろなパターンで系全体のバランスを見る必要があるので難しい…

※Contrail 4.0からは設定情報の伝達からIF-MAPの機能が除外され、RabbitMQを直接ControlがConsumeするようになる

SDN基盤バージョンアップの難しさ1

Contrailの依存関係

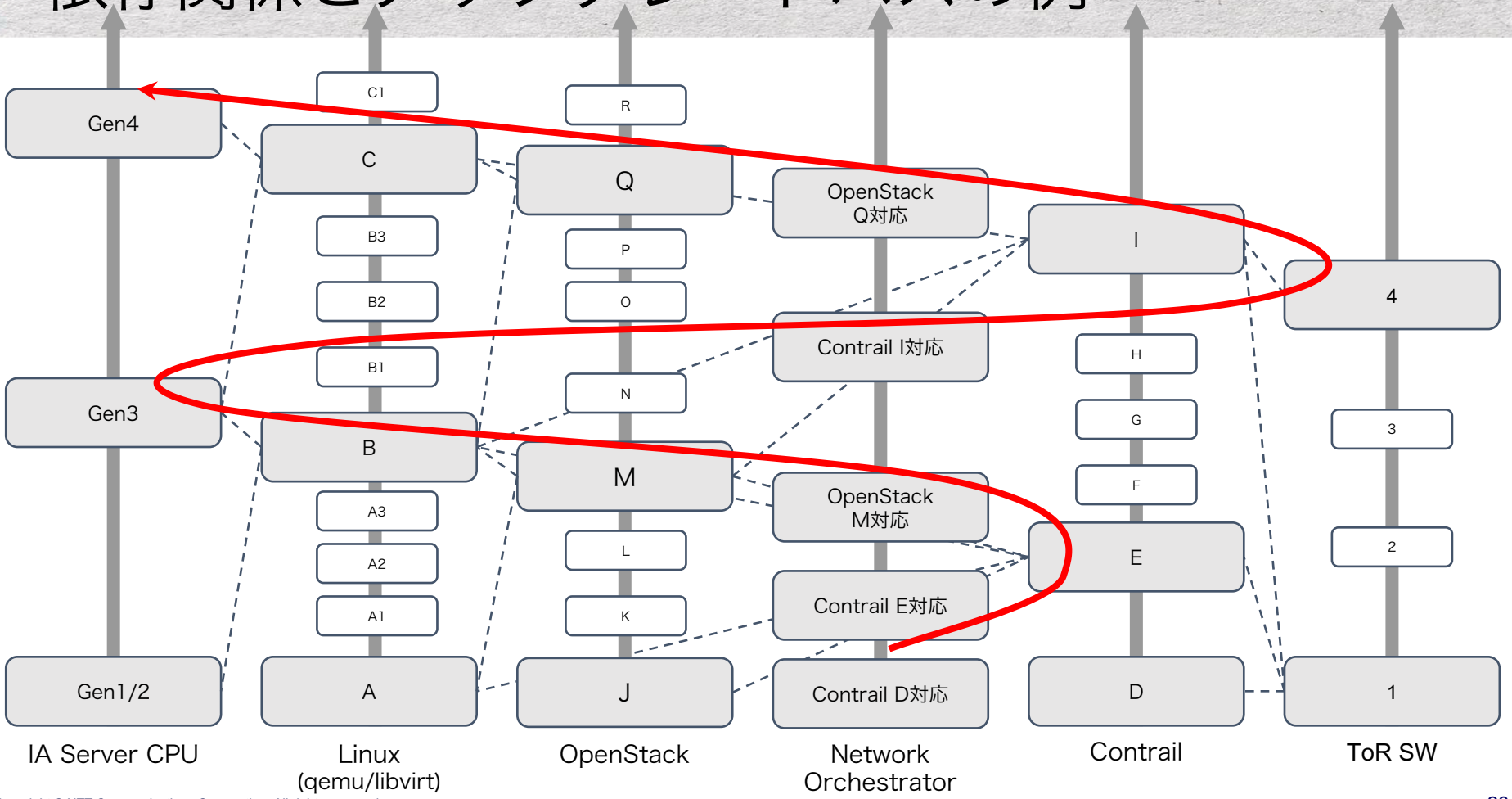
1. Linux Kernel
2. Host OS
3. OpenStack
4. QFX

外部要因

1. IAサーバ調達(Linux Kernel)
2. Guest VMの対応(qemu/libvirt)
3. 各種EoS/EoL/EoE

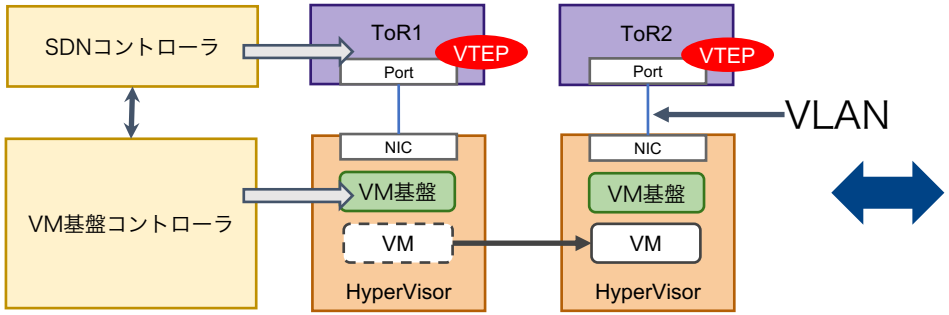
調達やEoSの時期を鑑みつつ、
すべての要素についてサポート関係・依存関係を考慮しながら
アップグレードパスを検討し（なければ作り）、
バージョンアップを実行するのが非常に大変

依存関係とアップグレードパスの例



VM向け仮想ネットワークのアーキテクチャ

Hardware VTEP/疎結合

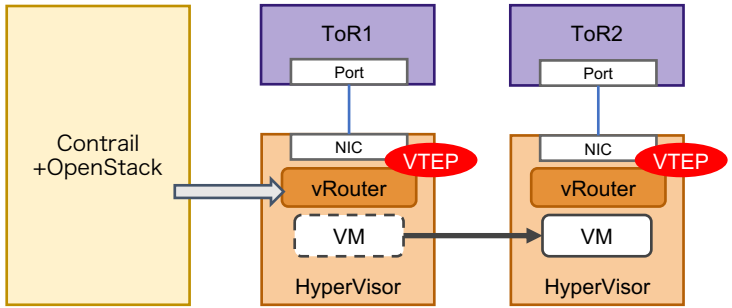


機能を分散させて
複雑性はコントローラ間に持たせる

利点
- SDN基盤とVM基盤が疎結合なので
  バージョンアップ・運用しやすい

欠点
- Security GroupなどのL3/L4機能の
  Live Migrationへの追従が複雑になる

Software VTEP/密結合



機能を一箇所に固めて
複雑性を集中させる

- Security GroupなどのL3/L4機能の
  提供が容易になる
- 依存関係が強くなるので
  バージョンアップが大変

OpenSource と Proprietary

Juniper ContrailがOpenSourceであることのメリット

- コードレベルで動作が理解できる
- トラブル時の対応が早くなる
  - 何が原因で不具合が起きたのかわかるのでメーカが解析に必要なデータを集めやすい
  - 自分たちでgdbして例外が出ているポイントを見つけられるので、想定できる原因をメーカに伝えることができる
    （使い方を熟知しているのは自分たち）
- 軽微なバグにはパッチを当てて対応できる
  - メーカーからの提供を待たずにパッチを当てて自前でビルドしてバイナリやカーネルモジュールを差し替えることができる

新SDN基盤の運用開始から１年

- 紹介した不具合やアーキテクチャ上の問題はすべて修正してUpstreamにも取り込まれた
- SDNコントローラの商用でのデプロイ・運用・アーキテクチャのベストプラクティスを導き出すことができた

大きなトラブルもなくSDN基盤の安定を実現できた
(**Stability**達成)

次の挑戦

Scalabilityの達成

- 2倍のスケール
- 罹障範囲の局所化

SDNのSREの実現

- テストの自動化
- デプロイの高速化
- コードのコントリビューション

現在の取り組み

➔ vRouterやToRをVMやコンテナで模擬しスケール検証環境構築
➔ Ansible等での機能試験やスケール＆障害試験の自動化
➔ パッチ作成やビルドの環境＆体制の構築
➔ 作成したパッチのメーカへの提供

求められるスキルが広範囲に及ぶので、
これらができるエンジニアがなかなか見つからない…

さいごに

NTTがしっかり使い込んで、もっと安定化させていくので、
みんなもContrail / Tungsten Fabricでクラウドを作ろう！