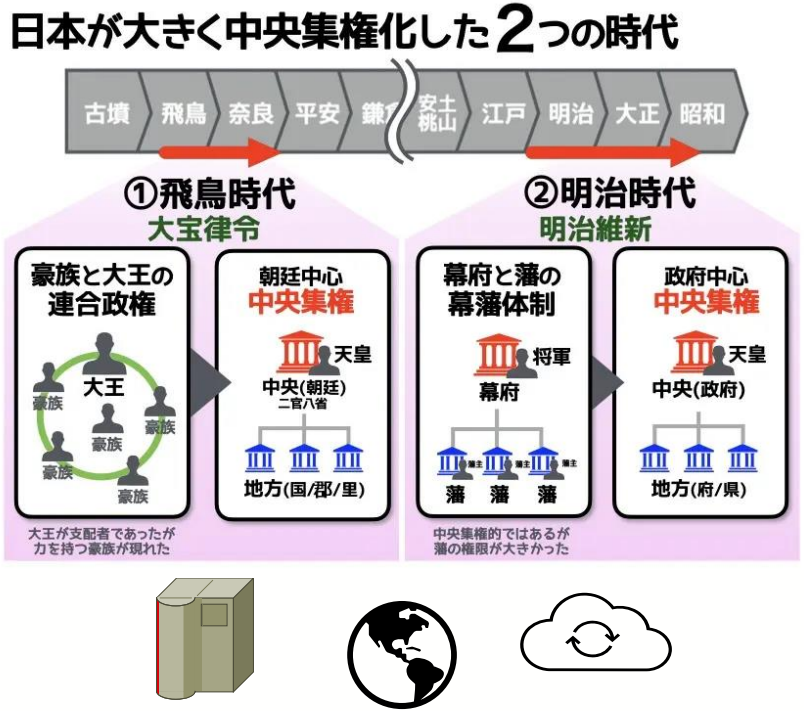
分散か?集中か?

-centralized vs distributed-

Shishio Tsuchiya
shtsuchi@arista.com

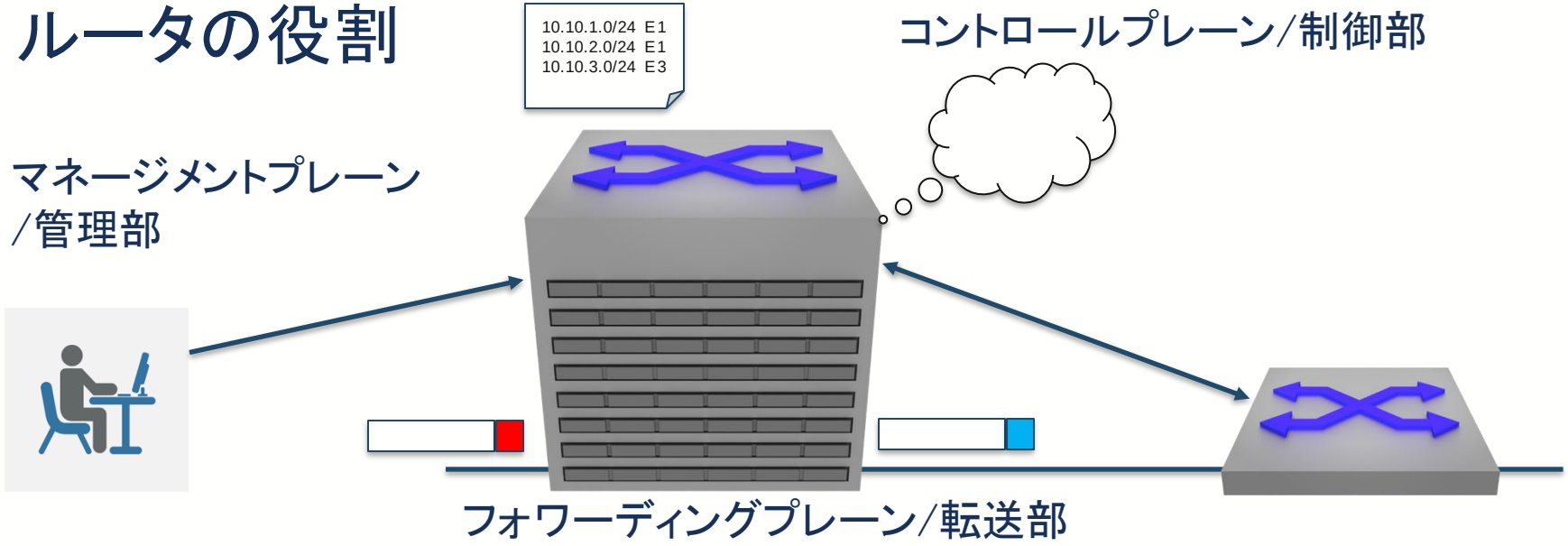
日本における中央集権と地方分権

https://katekyo.mynavi.jp/juken/34530



- 日本においては飛鳥時代の大宝律令と明治維新が大きく中央集権した時代と言われてる
- 歴史的にもこれらは繰り返される
- ネットワークにおける中央集権(集中管理）と地方分権（分散処理）について考えてみよう

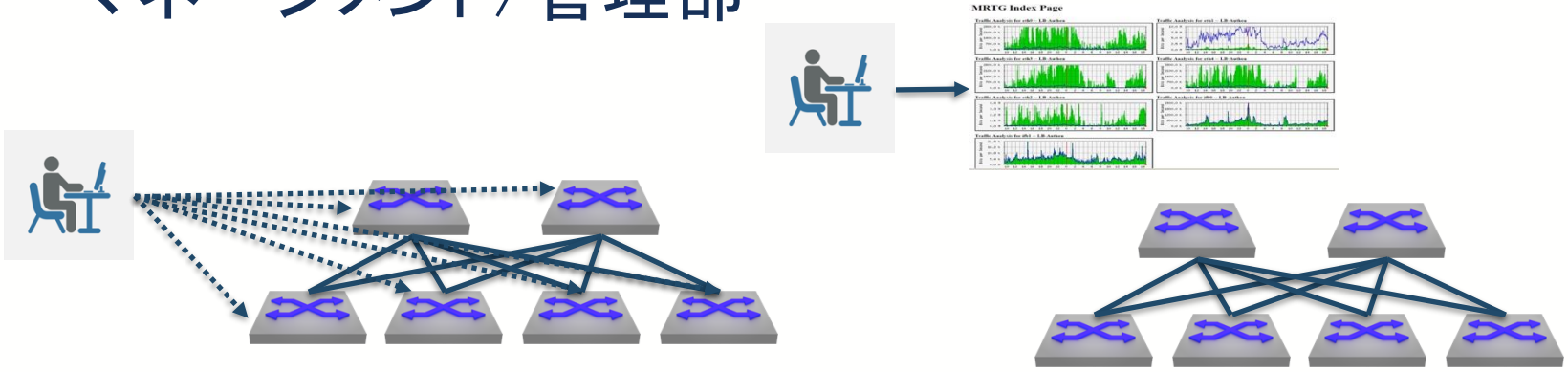
ルータの役割



- コントロールプレーン/制御部
  - 隣接ノードなどとやり取りをし、ネットワーク全体のテーブルを作成する
- フォワーディングプレーン/転送部
  - パケットが到着すると、ヘッダーを付け替えFIBに従いパケットを転送する
- マネージメントプレーン/管理部
  - ルータを管理制御を行う/ステータス情報をアップデートする

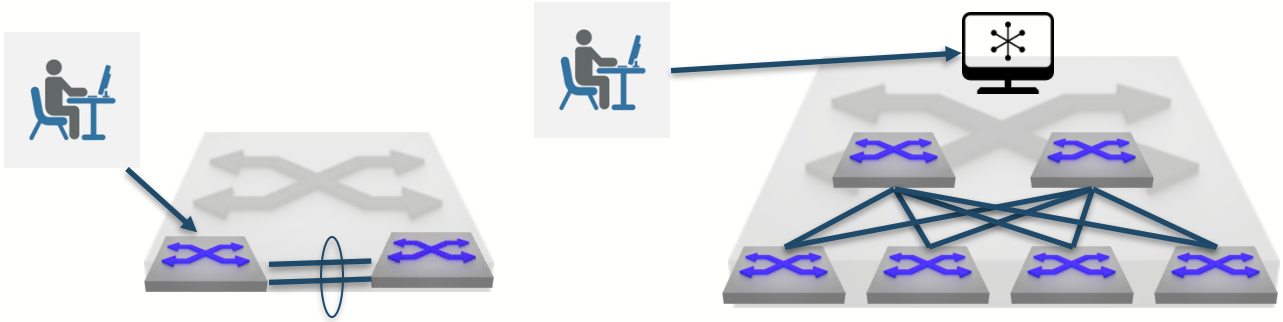
マネージメントプレーン/制御部

マネージメント/管理部



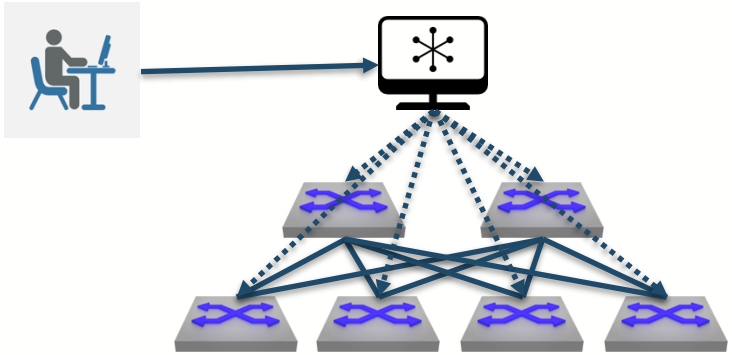
- それぞれのノードにアクセス/設定などを行う
- 各ノード/各ポートでの使用率などの把握は必要

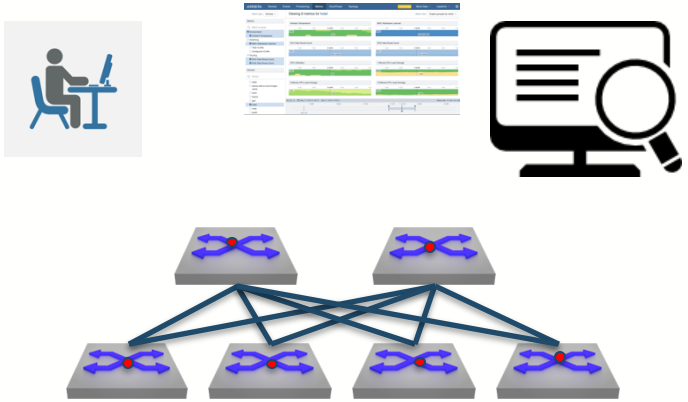
多ノードを抽象化し一台の様に見せる



- 制御部はActiveノードに渡し、Standbyはあくまで準ずる(nv Cluster/Stackなど)=>アクティブノードがコントローラーの役割を行う
- コントローラーの役割を持つものと管理者がやり取りを行う。コントローラーは各ノードを制御する(OpenFlow)
- バーチャルシャーシを構築する為のHigigプロトコルも存在する
  - https://docs.broadcom.com/doc/12358298

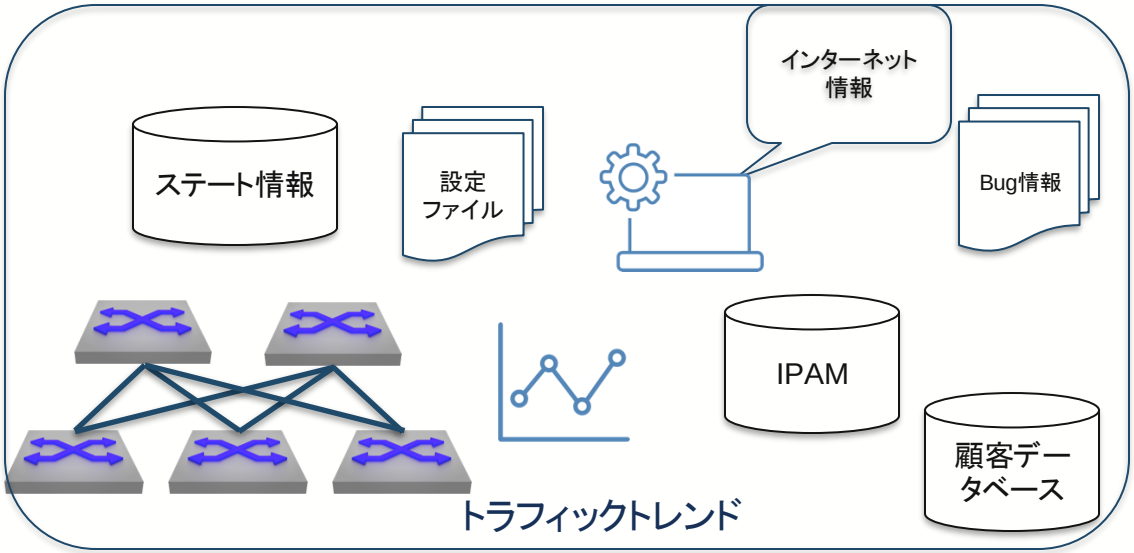
集中管理をしたい





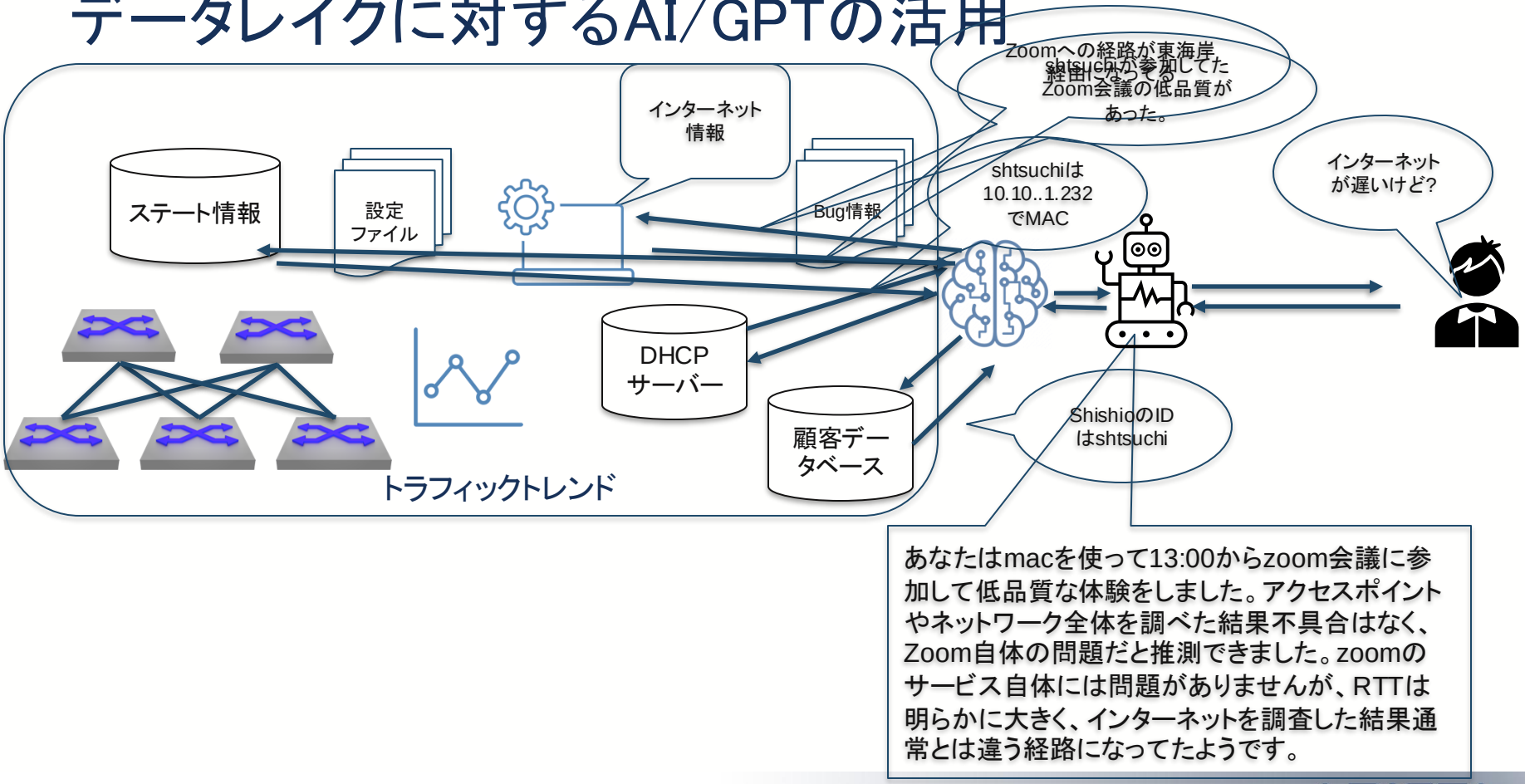
- 一括に設定を可能であること
- ステート情報を一元管理できること
- やりたいのは一台のノードにしたいわけではない

データレイク

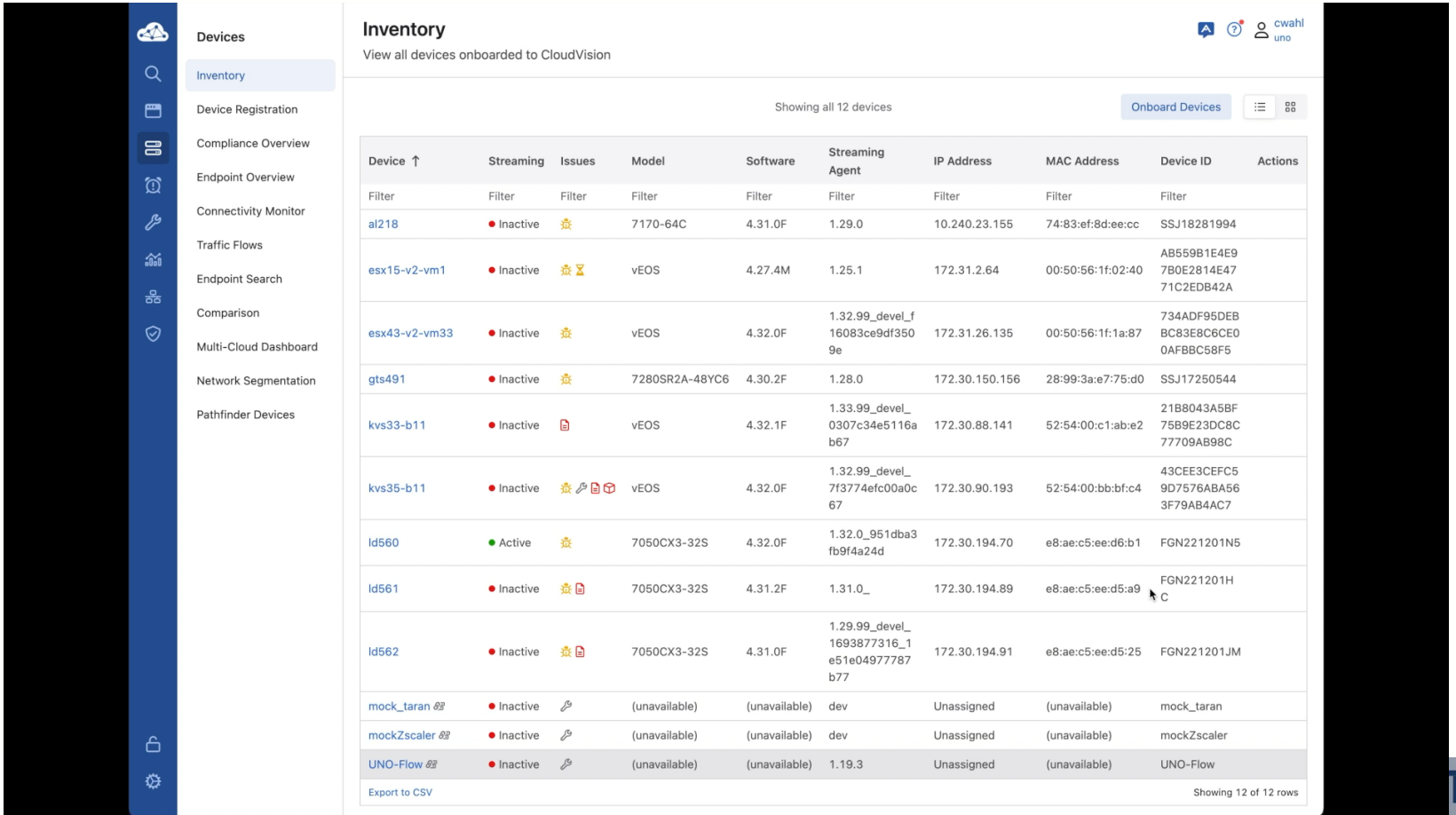


- データレイクは、わかりやすく定義すると、多数のソースからのビッグデータを元のままの多様な形式で保持する中央ストレージリポジトリのことです。**構造化データ、半構造化データ、非構造化データ**を格納できるので、将来の使用のためにデータをより柔軟な形式に保持できます。データレイクは、データを格納する際に識別子とメタデータタグを関連付けることで、検索を高速化します。
  - Talend https://www.talend.com/jp/resources/what-is-data-lake/

データレイクに対するAI/GPTの活用

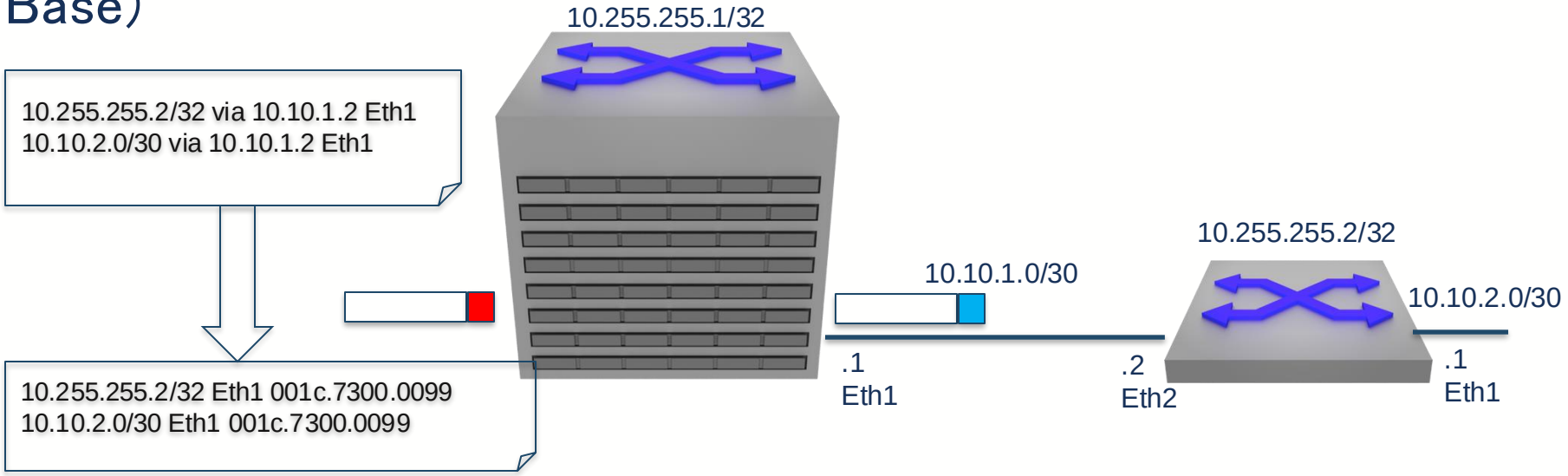


AVA(Autonomous Virtual Assistance)の例



フォワーディングプレーン/転送部

RIB(Routing Information Base)とFIB(Forwarding Information Base)



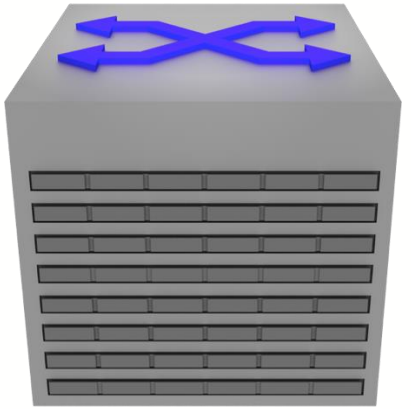
- ルーティングプロトコルによってRIBを作成し、転送のためのFIBを作成する

RIB/FIB

```
R2#show ip route 172.16.0.3/32 detail

VRF: default
Codes: C - connected, S - static, K - kernel,
       O - OSPF, IA - OSPF inter area, E1 - OSPF external type 1,
       E2 - OSPF external type 2, N1 - OSPF NSSA external type 1,
       N2 - OSPF NSSA external type2, B - Other BGP Routes,
       B I - iBGP, B E - eBGP, R - RIP, I L1 - IS-IS level 1,
       I L2 - IS-IS level 2, O3 - OSPFv3, A B - BGP Aggregate,
       A O - OSPF Summary, NG - Nexthop Group Static Route,
       V - VXLAN Control Service, M - Martian,
       DH - DHCP client installed default route,
       DP - Dynamic Policy Route, L - VRF Leaked,
       G  - gRIBI, RC - Route Cache Route,
       CL - CBF Leaked Route

 B E      172.16.0.3/32 [200/0] via 172.16.200.18, Ethernet50/1 R3
```



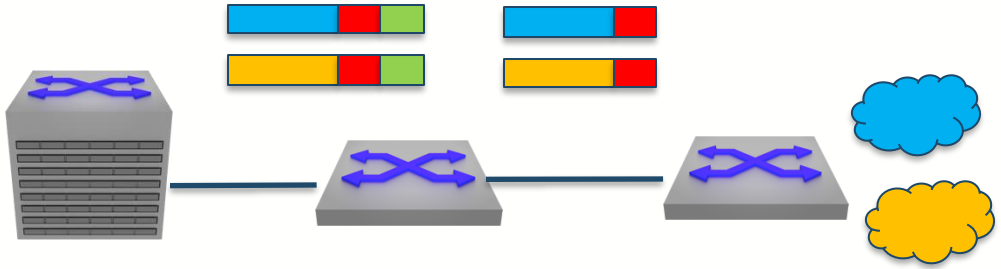
```
R2#show ip hardware fib routes 172.16.0.3/32
VrfName : default
* - Routes are compressed
  Prefix                                  Type            FecId                FecType                Vias
-------------------------------------------------------------------------------------------------------
  172.16.0.3/32                           ebgp            1297036705567604740  forwardRoute              1
 FecId                FecType            Vias       Next Hop                          Weight                                  l3Intf
-------------------------------------------------------------------------------------------------------------------------------
 1297036705567604740  forwardRoute       1          172.16.200.18                       1                                  Ethernet50/1
R2#
```

FEC(Forwarding Equivalence Class)

- FEC(forwarding equivalence class、FEC) とは、MPLSにおいて使用される用語で、類似または同一の特性を持つパケットの集合であり、同じように転送される可能性がある、つまり同じMPLSラベルにバインドされる可能性があるパケットの集合

Wikipedia

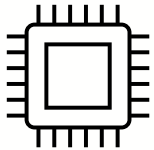
- 今だと一般的なフォワーディングコンポーネントのセットの様に使われる
- RFC3031 MPLS Architecture
  - https://datatracker.ietf.org/doc/html/rfc3031#section-2.1



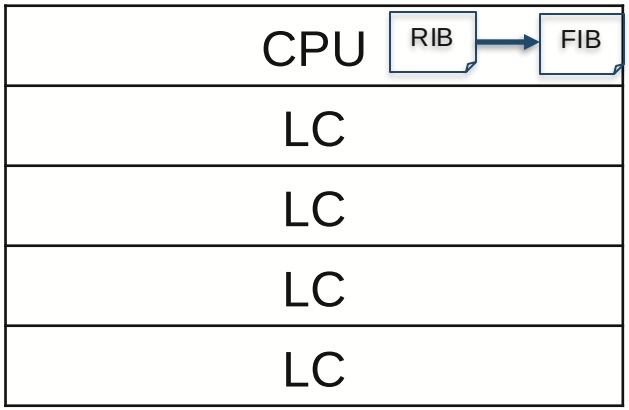
FEC(Forwarding Equivalence Class)

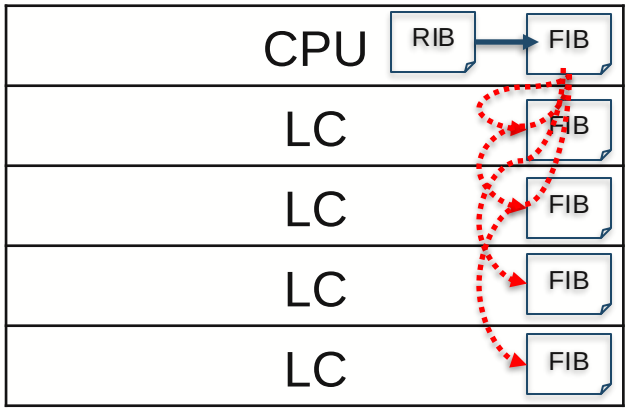
```
R2#show platform fap ip route  172.16.0.3/32
Tunnel Type: M(mpls), G(gre), MoG(mpls-over-gre), MoU(mpls-over-udp), IPoU(ip-over-udp)
             vxlan-o(vxlan outer-rewrite info), vxlan-i(vxlan inner-rewrite info)
CW  - Control word
FL  - Flow label
EL  - Entropy label
ELI - Entropy label indicator
*   - Routes in LEM
D   - ECMP is divergent across switching chips
 ------------------------------------------------------------------------------------------------------------
|                                Routing Table                                                 |             |
|------------------------------------------------------------------------------------------------------------
|VRF|   Destination    |     |                   |     |        |                  | ECMP|  FEC | Tunnel
| ID|      Subnet      | Cmd |     Destination   | VID | Outlif |   MAC / CPU Code  |Index| Index|T Value
 ------------------------------------------------------------------------------------------------------------
|0  |172.16.0.3/32     |ROUTE| Et50/1            |1012 |8188    | 44:4c:a8:25:8e:65 |  -  |32772 |   -
```

```
R2#show platform fap fec 32772
Tunnel Type: Mpop(mpls pop), Mpush(mpls push), Mswap(mpls swap),
             MoG(mpls-over-gre), T(IPv4 tunnels GRE/GUE/VXLAN),
             N(Ipsec tunnel NAT-T [IP,SPORT,DPORT])
CW  - Control word
FL  - Flow label
EL  - Entropy label
ELI - Entropy label indicator
D   - ECMP is divergent across switching chips
 ------------------------------------------------------------------------------------------------
|                                        FEC Entry                                               |
 ------------------------------------------------------------------------------------------------
|     |      |     |                   |     |        |                  |
| ECMP|  FEC |     |                   |     |        |                  |
|Index| Index| Cmd |     Destination   | VID | Outlif |   MAC / CPU Code |    Tunnel Value
 ------------------------------------------------------------------------------------------------
|  -  |32772 |ROUTE| Et50/1            |1012 |11      | 44:4c:a8:25:8e:65 |   -
```



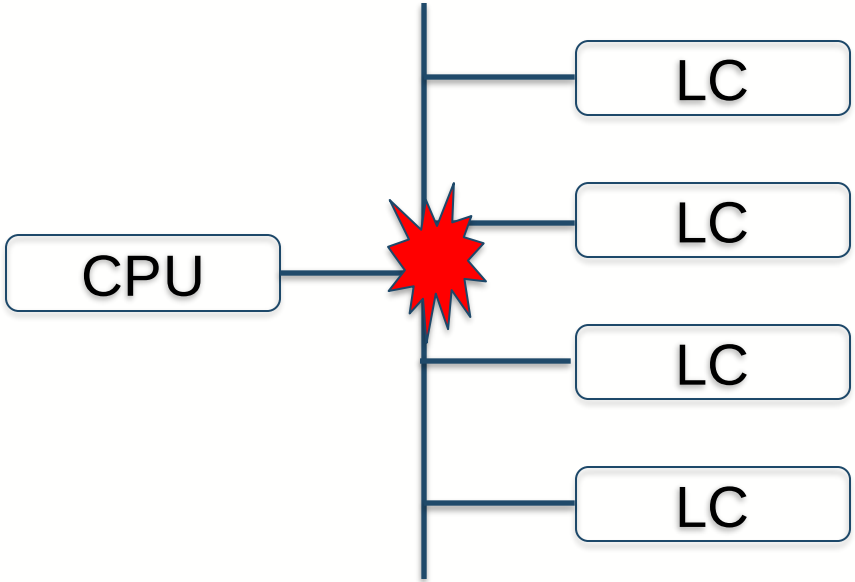
Distributed Forwarding





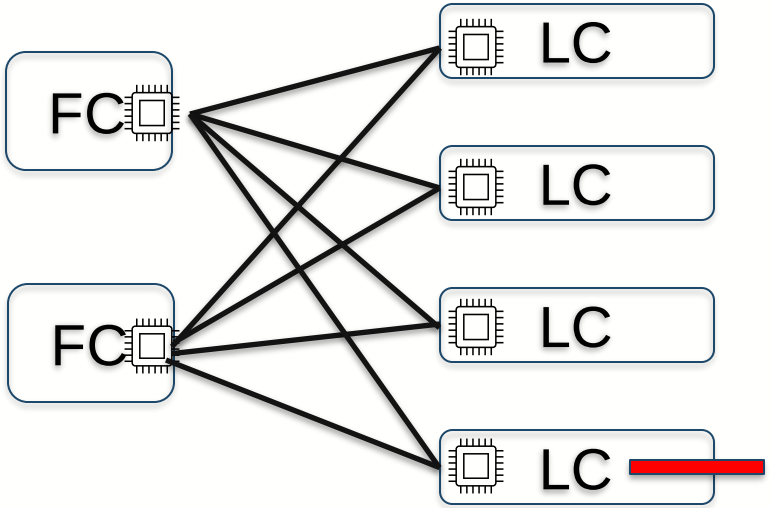
- CPU(スーパーバイザー)で作成されたFIBをそれぞれのラインカードにコピーをする
- ラインカードがそれぞれFIBを持つことになる

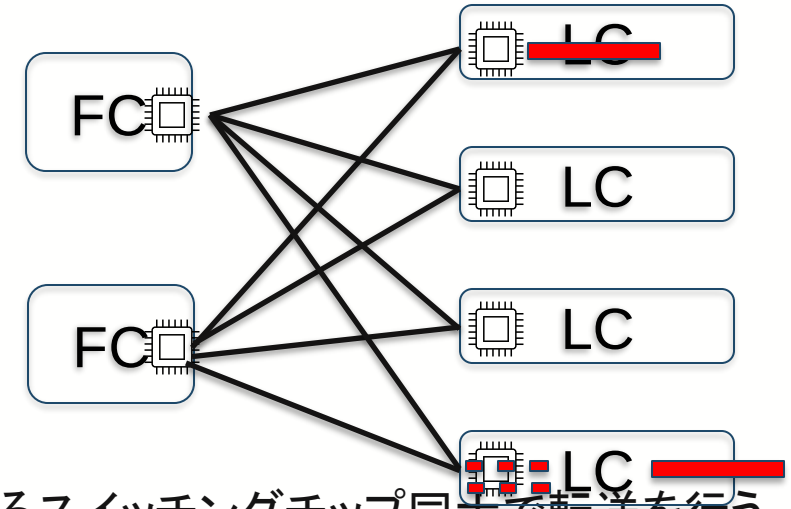
Distributed Forwarding FIBの配送



- FIBの構造が階層化されおらずパッチワーク的なアーキテクチャーの場合、内部メッセージのみで輻輳などが起こることもありえる（まぁありえないのですが）

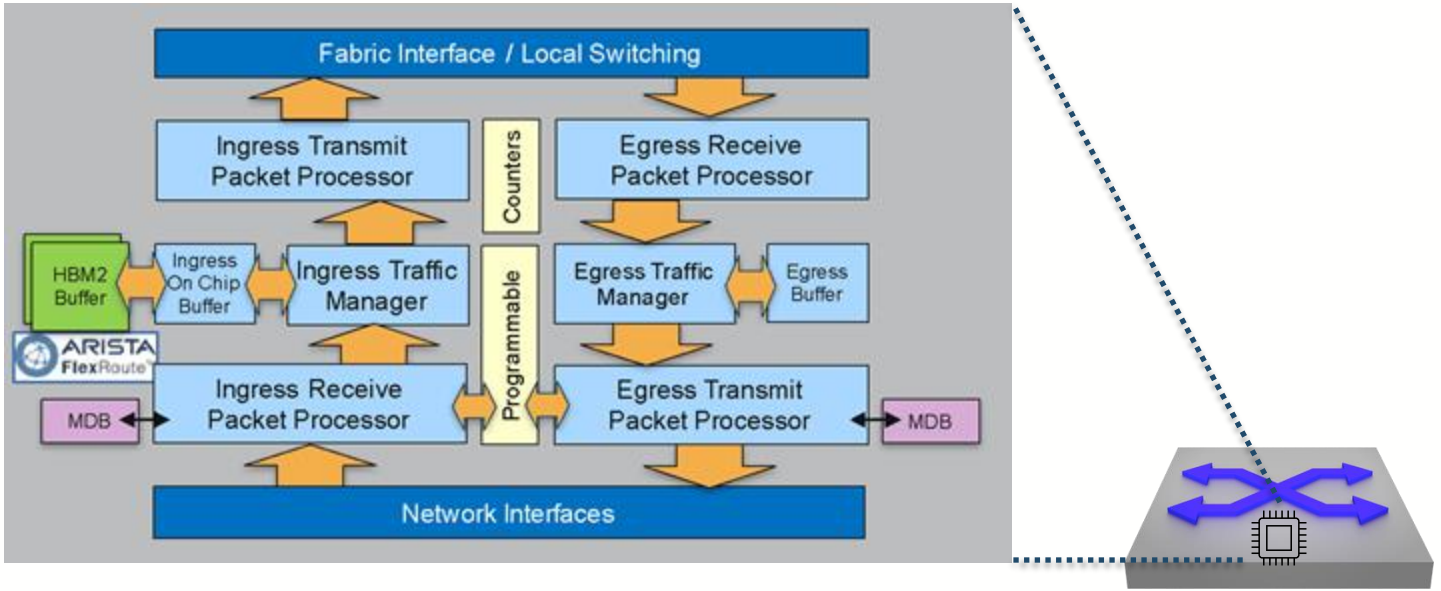
Distributed Forwarding





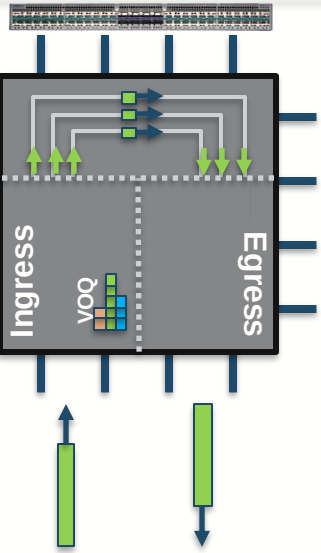
- 分散型のフォワーディングではLCにあるスイッチングチップ同士で転送を行う。
- イーサネットフレームに内部ヘッダーをつけ転送するものもあれば、内部ではセル化して転送するものもある

パケットプロセッサー

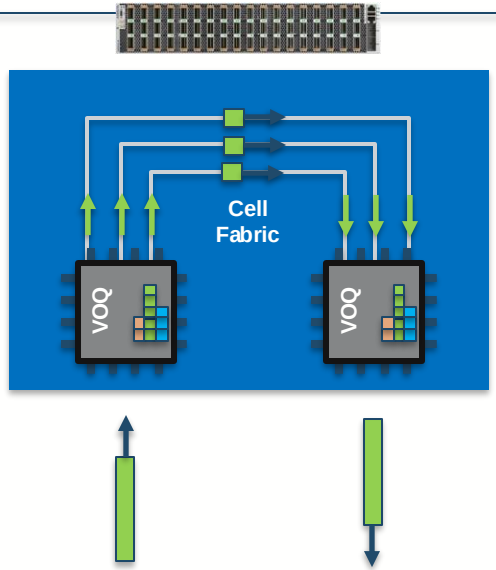


- スイッチングのパケットプロセッサーはパケットを受け取りFIB(FEC)より出力先を決定する(入力受信パケットプロセッサー)
- セル化してファブリックインターフェースに送る(入力送信パケットプロセッサー)
- セルを組み立ててフレームに戻す(出力受信パケットプロセッサー)
- 出力ヘッダーに書き換える(出力送信パケットプロセッサー)

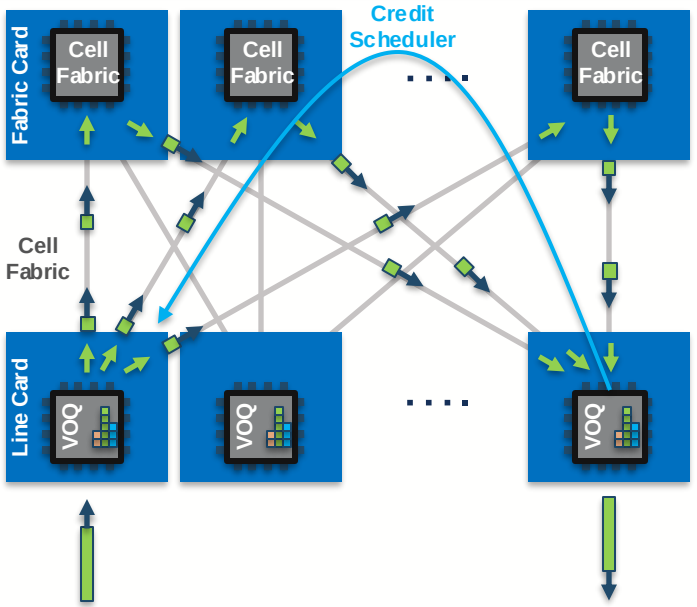
Broadcom DNX/Jerichoを使ったシステム例



シングルチップ固定型



マルチチップ固定型

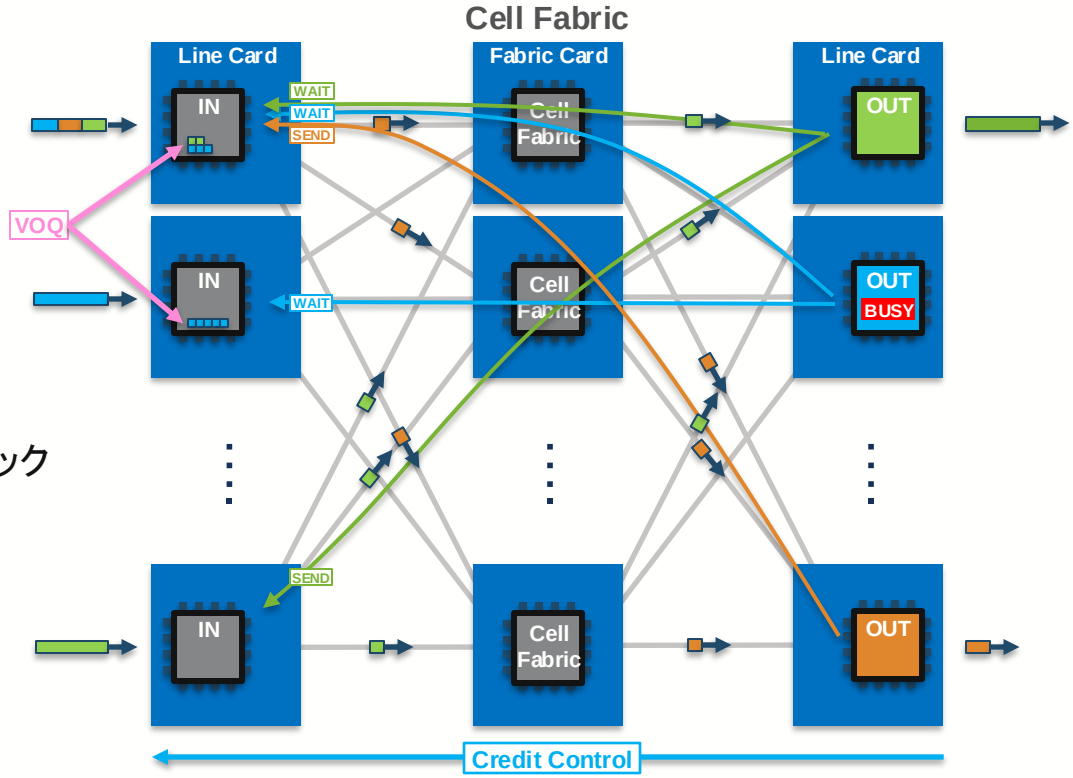


モジュラーシステム

固定システムとモジュラーシステムは共通のVOQスケジューリングを利用する

VOQ＆セル・ファブリックアーキテクチャー

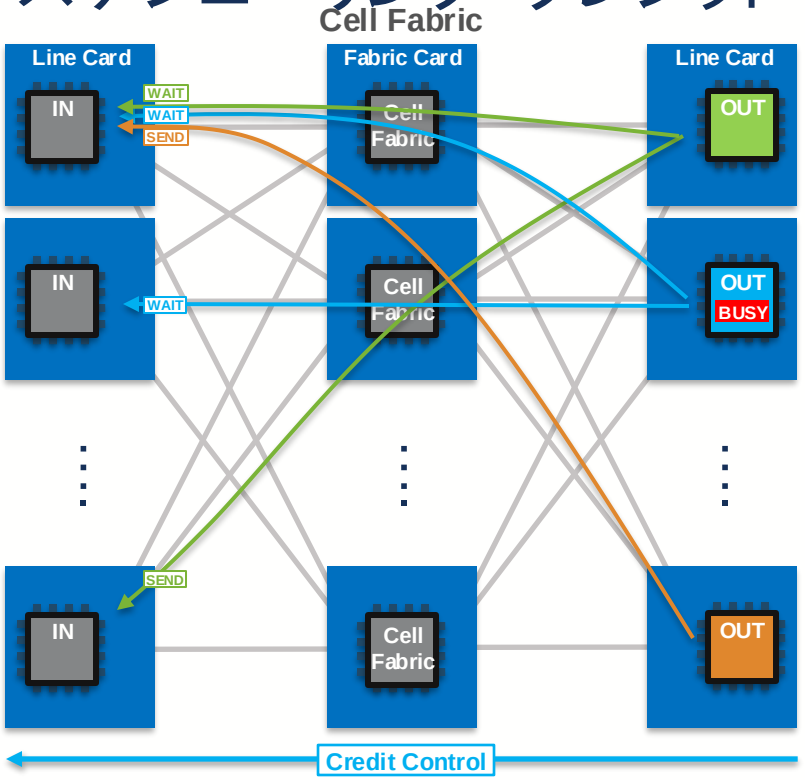
- イングレス・パケット・バッファによるインキャストの防止
  - チップあたり大容量バッファ（8GB）
  - LCの追加に応じてリニアにスケールする
  - 増加するEgressのオーバーサブスクリプションを排除
- バッファを仮想キューに分割
  - 出力ポートごとTCごとのVOQ
  - シングルキューのHOLBの解消
  - 公平なシステムワイドQoSを可能にする
- 出力ポート別にスケジューリングされたトラフィック
  - TXが利用可能な場合のみファブリックに送信される
  - Busy時に分散VOQに保持される
  - ファブリックの内側では衝突しない
- セル・ベースファブリック
  - 100%効率的なセル・スプレー
  - ハッシュ問題なし
  - インターフェース・スピードの不一致を分離



独自のアーキテクチャで数千ポートまで拡張 – ボトルネックを解消

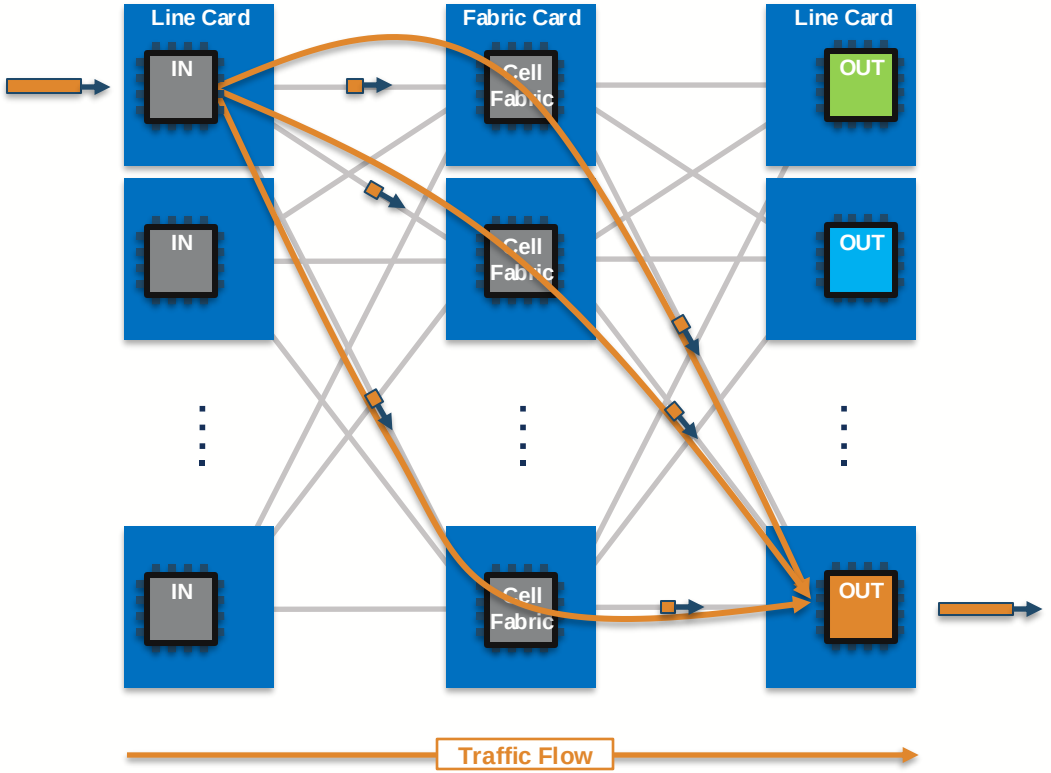
VOQおよびセル・ファブリック – スケジューリング・クレジット・コントロール

- ファブリックを横切るユニキャストトラフィックは、Egress PPによってスケジューリング
- 各Egress PPはIngress PPにトラフィックを受信できることを通知
- Egressスケジューリングは、ファブリックまたはEgress PPでの衝突を防ぐ
- ビジー状態の場合、送信クレジットの不足によりIngress PPのバックプレッシャーになる
- トラフィックは小さい出力バッファで衝突するのではなく、入力側でバッファする



VOQおよびセル・ファブリック – トラフィック転送

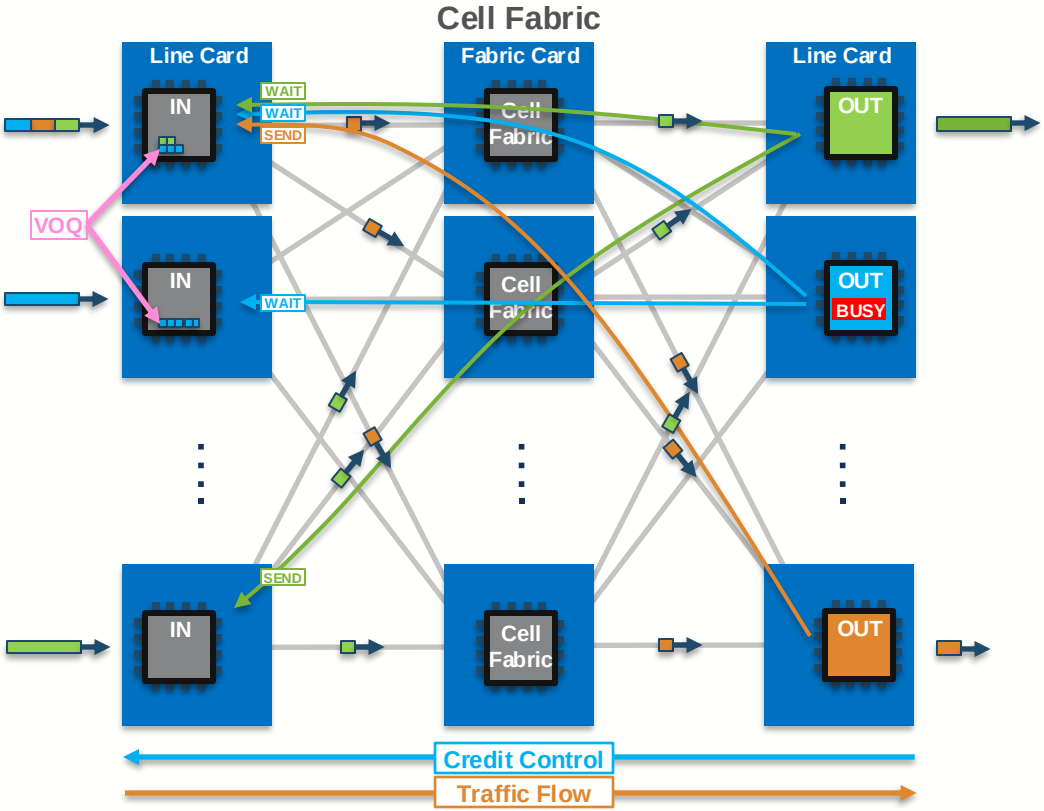
- Ingress PPがパケットをセルにスライスする（IP/Ethernetではない）
- セルはFE全体に散布されるため、非常に効率的な負荷分散が行われる
- ファブリックはオーバープロビジョニングで障害に対応
- パケットの再組み立てと最終書き込みは、Egress PPで実行される。



セル・スプレーがFabricのハッシュ偏りとエレファント・フローを解消

VOQおよびセル・ファブリック – Virtual Outputキューイング

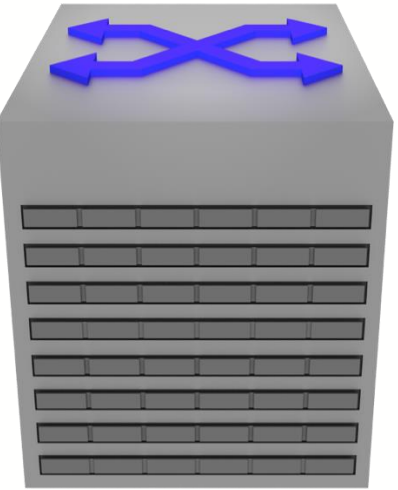
- イングレスPPは、オンチップおよびオフチップ・バッファを使用してVirtual Output Queueを実装
- バッファスペースは、トラフィッククラスごと、出力ポートごとにユニークなキューに分割
- すべてのIngress PPはローカルVOQを持つ – システムごとに何千ものキューを持つ
- PPを追加するごとに、キューやバッファリングが追加
- 帯域が利用可能になるまで、ビジー状態の宛先のトラフィックをIngress PPの宛先VOQに保持
- ビジーでない宛先のトラフィックは通常通り流れる



VOQはバッファリングのリニアなスケーリングを提供し、HOLBを排除

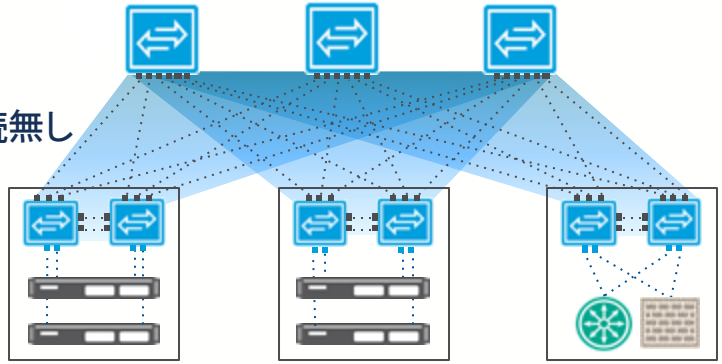
コントロールプレーン/制御部

コントロールプレーンの集中化(OpenFlow)



コントローラー
X86サーバー上に展開
2台構成でクラスター構成が可能

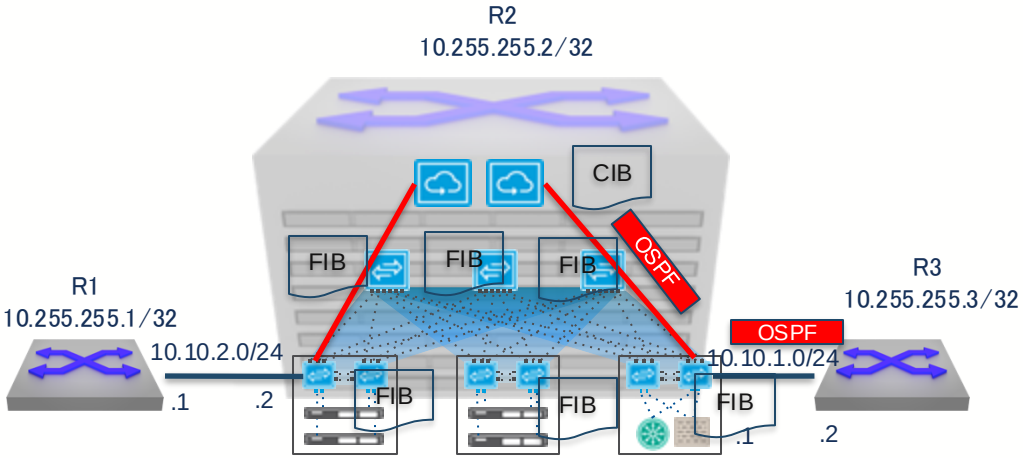
Spineスイッチ
Fabricのバックボーン
Fabricへのリンクのみで外部接続無し
各リーフの接続が可能



Leafスイッチ
Fabricの入口/出口となる
外部接続を担当(サーバー/ルータ/ファイヤーウォールなど)
各Spineへの接続は必要

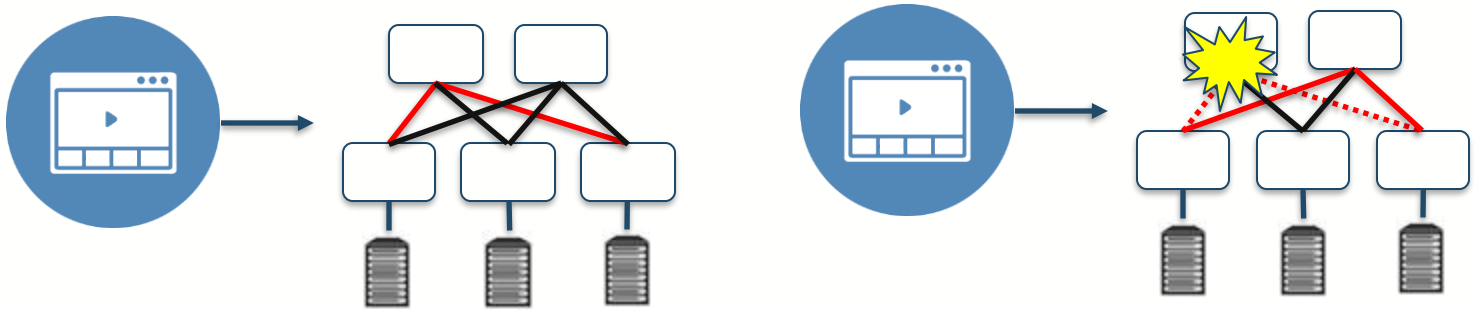
- OpenFlowでは各スイッチをラインカードの様に動作させコントローラーでOSPF/BGPなどの計算を行うことが可能

OpenFlowルータのイメージ



- 外部のルータは特になにも意識しない
- OSPFやBGPのコントロールパケットはコントローラーに送り届けられる
- コントローラーではL2/L3の情報をまとめ、それぞれのスイッチにFIBとして送る
- データパケットはFabricの最適な経路を通って通信される

InfiniBand Subnet Manager



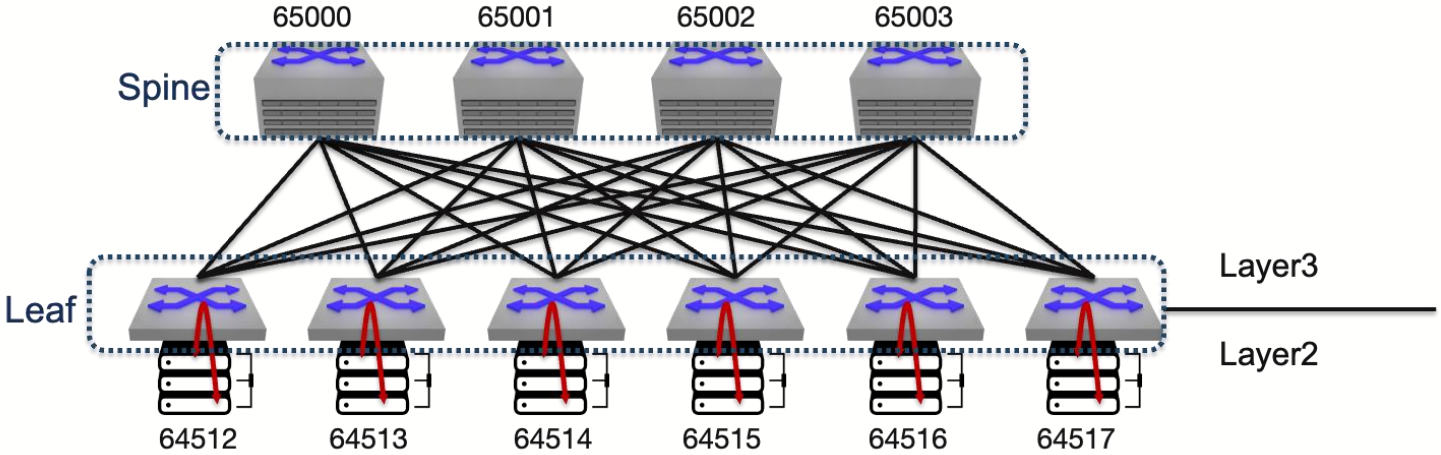
- InfiniBandの世界でもコントロールプレーンの中央化のアプローチを行っている
- Subnet Managerはサーバーもしくはスイッチ上で動作することが出来る
- SMは物理トポロジーを見つけ、独自のルーティングアルゴリズムで計算したベストパスを各スイッチにプログラムする
- 障害時には自動的に検知し、新しいパスをプログラムする

Ultra Ethernet

特徴	InfiniBand	Ethernet/RoCEv2	Ultra Ethernet
プライマリRFMインターフェース	IB Verbs	IB Verbs	libfabric https://github.com/ofiwg/libfab
スケーラビリティのあるコントロールプレーン			
複数経路のパケットスプレイ			
フロー制御	Credit-base	PFC/ECN	Dynamic Multi-Path
スケジュール化されたFabric			
E2Eドロップ再生			
トランスポート暗号化			
マルチベンダーエコシステム			

- UECでは現在のイーサネット同様なスケラビリティのあるコントロールプレーンが求められる
- つまり分散型のコントロールプレーン

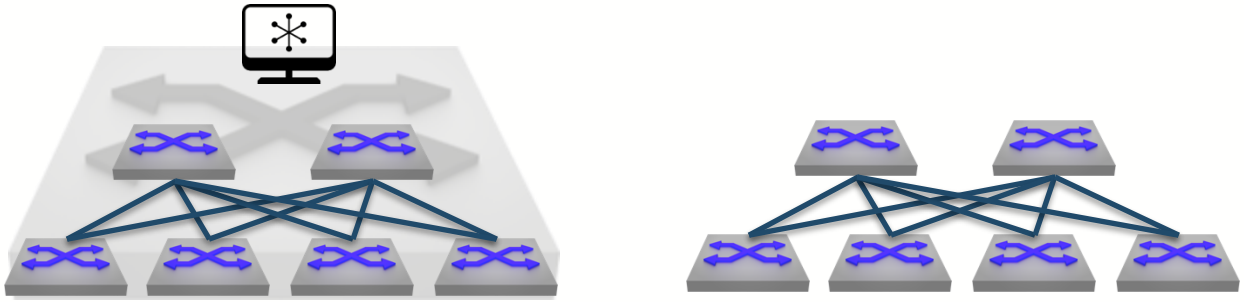
改めてイーサネット上のルーティング



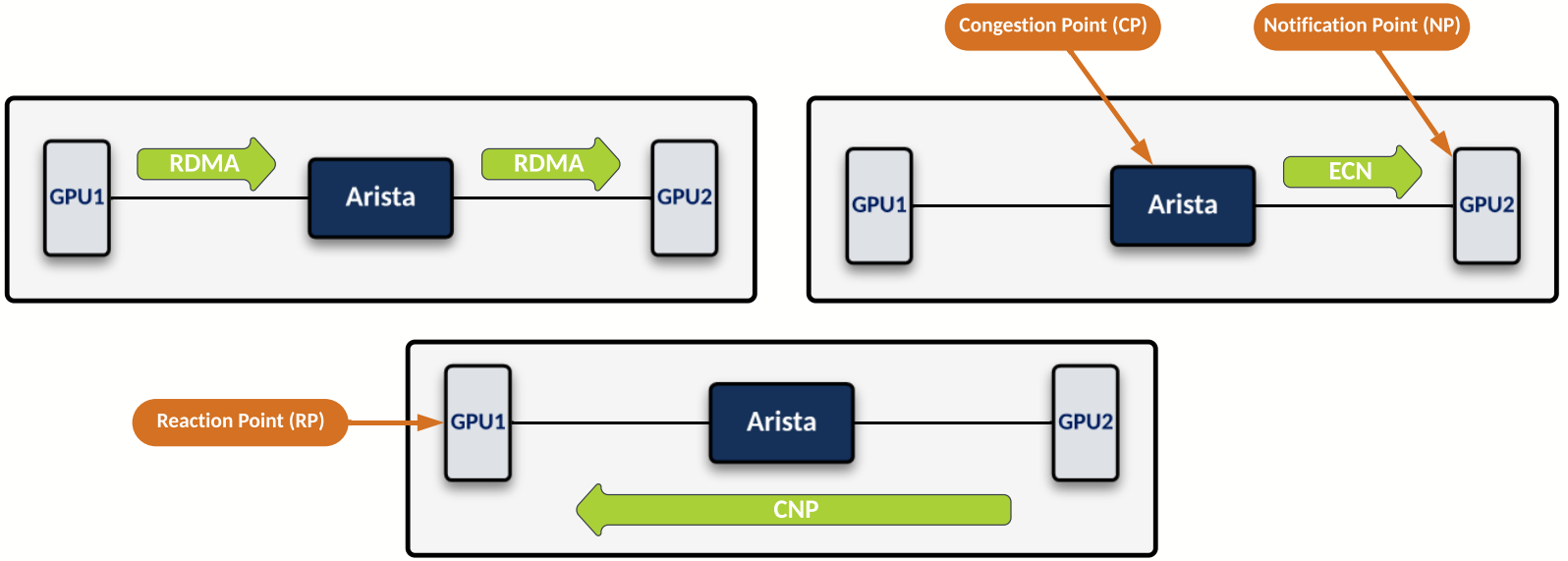
- InfiniBandとは異なり、イーサネットのルーティング・プロトコルはパス計算をネットワーク・ノードに分散する。
- このようなアプローチにより、ルーティングコントロールプレーンにおける単一障害点が排除される。
- ルーティングの決定を分散することで、ネットワークの回復力も向上。ポートがダウンした場合、転送テーブルを再プログラムするために計算や指示を待つのではなく、ローカルスイッチが即座に新しいパスを決定することが出来る。
- 単一システムの障害でネットワーク全体のトラフィックが停止することが無い

集中か分散か？

- UECでは現在のイーサネット同様なスケラビリティのあるコントロールプレーンが求められる
- つまり分散型のコントロールプレーン
- しかしデータプレーンはGPUおよびスイッチで協調して動く必要がある

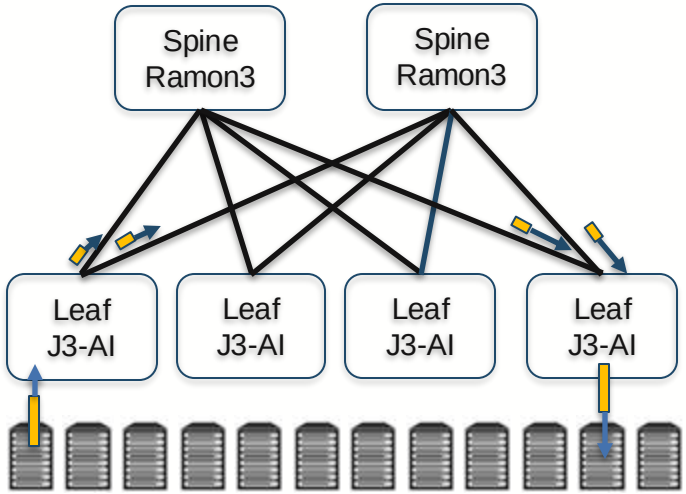


DCQCN/Data Center Quantized Congestion Notification



- Congestion Point(CP):ECNをマークしたスイッチ
- Notification Point(NP):ECNをマークしたパケットを受け取って、CNPを送信する受信GPUのNIC
- Reaction Point(RP):NPからCNPを受け取り、バッファオーバーフローさせないようにレートを調整

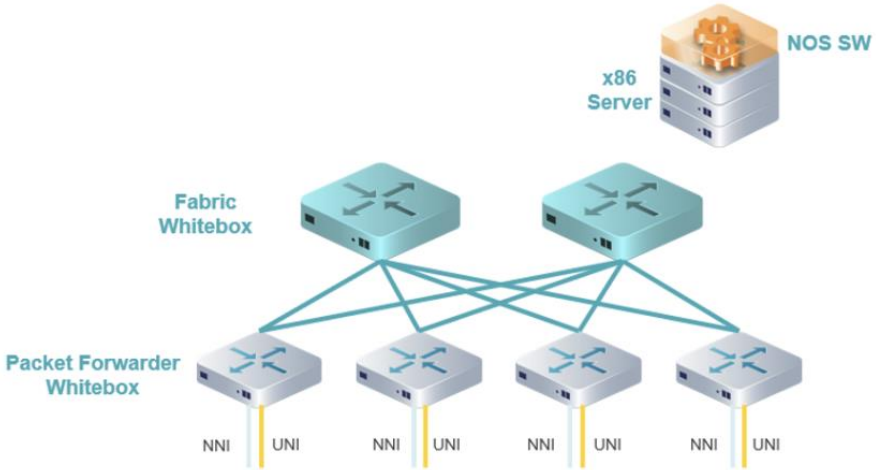
ブロードコムが提示したAIイーサネットファブリックのアーキテクチャー



Broadcom Jericho3 AI Ethernet Fabric
https://youtu.be/fMKq0Y0jPHk?si=lwyoQBTvqTSIXLp6

- Fabricカードに使用してたRamonをAI Fabric専用Spine箱に使用
- 従来のスイッチングチップはLeafに
- モジュラーシステムで使用してたアーキテクチャーをネットワークワイドに

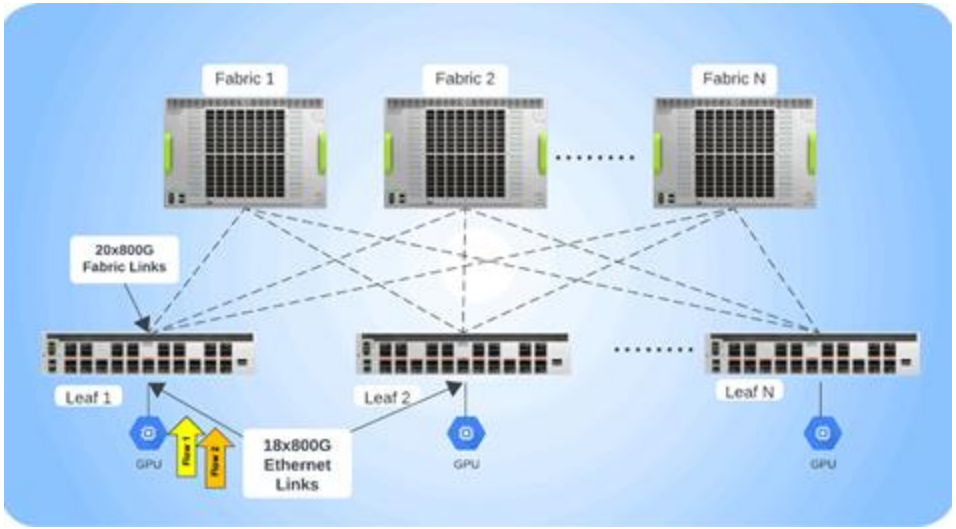
TIF(Telecom Infra Project)
DDBR (Disaggregated Distributed Backbone Router)



https://telecominfraproject.com/wp-content/uploads/ddbr.png

- TIFの提唱するDBR
- コントロールプレーンやデータプレーンを完全に分離する
- 集中型コントロールプレーンと分散型アーキテクチャー

Distributed Etherlink Switch



- Leaf/Spineそれぞれ独立してOSが動作
- スケジューラーのみがモジュラーシステムの様に協調して動作
- GPUやアクセレーターから見ると一台の様に動作する
- 分散フォワーディング・分散コントロール

まとめ

- ネットワーク(機器)の分散か集中かに関して考えてみた
- 個人的な感想は以下の通り
  - マネージメントプレーン
    - 分散で良いけど、集中管理が出来るべき
  - データプレーン
    - 分散が良いけど、協調して動くべき
  - コントロールプレーン
    - スケールや耐障害性のことを考えれば分散であるべき

Thank You

arista.com