Experiences with BGP in Large Scale Data Centers:

Teaching an old protocol new tricks

Presented by: Edet Nkposong, Tim LaBerge, 北島直紀

Global Networking Services Team, Global Foundation Services, Microsoft Corporation

Microsoft

Agenda

- ネットワーク設計要件 [Network design requirements]
- プロトコル選定：BGP vs IGP [Protocol selection: BGP vs IGP]
- ルーティング設計詳細 [Details of Routing Design]
- なぜBGP SDNなのか？ [Motivation for BGP SDN]
- BGP SDNコントローラの設計 [Design of BGP SDN Controller]
- BGP SDNロードマップ [The roadmap for BGP SDN]

ネットワーク設計要件[Design Requirements]

データセンターの規模[Scale of the data-center network:]

- 10万台超のサーバ per クラスタ[100K+ bare metal servers]
- 3千台超のネットワーク機器 per クラスタ [Over 3K network switches per DC]

稼働するアプリケーション[Applications]

- Map/Reduce: Social Media, Web Index and Targeted Advertising
- Public and Private Cloud Computing: Elastic Compute and Storage
- Real-Time Analytics: Low latency computing leveraging distributed memory across discrete nodes. /[クラウドコンピューティングやMapReduce等など]

まとめると[Key outcome:]

- East-Westトラフィックの増大が大容量の二分割(bisectional)帯域幅を必要とする。即ちボトルネックが生じないこと。 [East ←→ West traffic profile drives need for large bisectional bandwidth.]

要件を設計に落とし込むと
[Translating Requirement to Design]

トポロジー条件[Network Topology Criteria]

East-Westトラフィックをノンブロッキングにさばくこと
[East <-> West Traffic Profile with no over-subscription]

- 投資の最小化[Minimize Capex and Opex]
  - 安価なコモディティスイッチの利用
    [Cheap commodity switches]
  - 低消費電力[Low power consumption]

同種のコンポーネントを使うこと
[Use Homogeneous Components]

- スイッチ、Optics、ファイバーなど
  [Switches, Optics, Fiber etc]
- 運用の複雑さを軽減
  [Minimize operational complexity]
- 単価の最小化
  [Minimize unit cost]

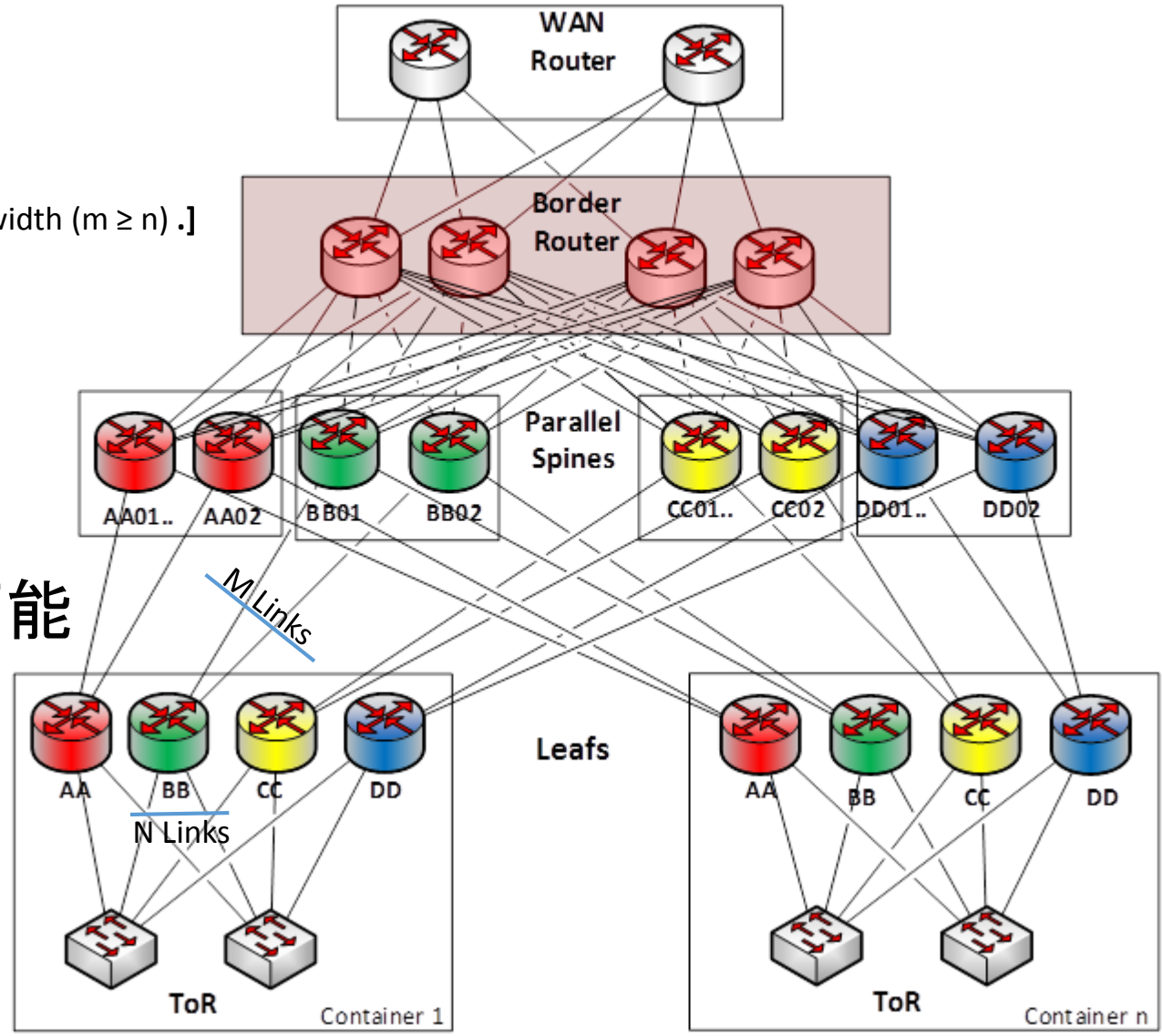
プロトコル条件[Network Protocol Criteria]

- 標準ベース[Standards Based]
- 制御プレーンがスケールし安定していること[Control Plane Scaling and Stability]
- CPU, TCAM等のリソース消費が少なく、予測可能であること
  [Minimize resource consumption e.g. CPU, TCAM usage - predictable and low]
- Layer 2ドメインのサイズを小さくすること
  [Minimize the size of the L2 failure domain]
- Layer 3 with equal-cost multipathing (ECMP)
- プログラマブルであること[Programmable]
  - 拡張性と自動化を担保
    [Extensible and easy to automate]

ネットワーク設計:トポロジー編[Network Design: Topology]

- 三層折返しCLOS型[3-Stage Folded CLOS.]
- フル二分割帯域幅(m ≥ n) [Full bisection bandwidth (m ≥ n) .]
- 水平展開が容易
  （スケールアウト vs スケールアップ）
  [Horizontal Scaling (scale-out vs. scale-up)]
- ECMPが自然に実装できる
  [Natural ECMP link load-balancing.]
- 高密度コモディティスイッチで実装可能

  [Viable with dense commodity hardware.]
  - 巨大な仮想スイッチを多数の
    コンポーネントで作り出す。
    [Build large “virtual” boxes out of small components]



ネットワーク設計：プロトコル編
[Network Design: Protocol]

プロトコルへの要件[Network Protocol Requirements]

- 冗長性が高いこと[Resilience and fault containment]
  - Closトポロジーは多数のリンクを必要とするため、リンク障害頻度高。この影響を抑えたい。[CLOS has high link count, link failure is common, so limit fault propagation on link failure.]
- 制御プレーンの安定化[Control Plane Stability]
  - ネットワーク機器は千台規模、リンク数は一万本のオーダ
    [Consider number of network devices, total number of links etc.]
  - 制御プレーンのステート数を減らす[Minimize amount of control plane state.]
  - リンク障害や再起動が制御プレーンの資源を食い尽くす状況を回避
    [Minimize churn at startup and upon link failure.]
- Traffic Engineering
  - DCの中ではECMPが簡単に使える。→TEの重要度は低い
    [Heavy use of ECMP makes TE in DC not as important as in the WAN.]
  - 一方、障害時にトラフィックをうまく迂回させるメカニズムは必要
    [However we still want to “drain” devices and respond to imbalances]

なぜBGPなのか？　IGPじゃなくて。
[Why BGP and not IGP?]

- プロトコルとしてシンプルである[Simpler protocol design compared to IGPs]
  - ステートを複製する仕組みの観点において[Mostly in terms of state replication process]
  - 優れたベンダー相互運用性[Better vendor interoperability]
  - より少ないState-machine,データ構造など[Fewer state-machines, data-structures, etc]
- 障害対応もよりシンプル[Troubleshooting BGP is simpler]
  - Paths propagated over link
  - ASPATH is easy to understand.
  - Easy to correlate sent & received state
- ECMPを自然に実装できる [ECMP is natural with BGP]
  - リンクステートプロトコルとの比較において[Unique as compared to link-state protocols]
  - 粒度の高いポリシーを入れやすい（後述）[Very helpful to implement granular policies]
  - Unequal-cost Anycastのload-balancingがやりやすい[Use for unequal-cost Anycast load-balancing solution]

なぜBGPなのか？　IGPじゃなくて。（続き）

[Why BGP and not IGP? (cont.)]

- イベントの伝搬がより制約的[Event propagation is more constrained in BGP]
  - イベントのfloodingが懸念される事象に対しても高い安定性が期待できる
    [More stability due to reduced event “flooding” domains]
  - 例：BGP UPDATEをASNを使って制御できる等
    [E.g. can control BGP UPDATE using BGP ASNs to stop info from looping back]
  - これはディスタンスベクトル型の特性
    [Generally is a result of distance-vector protocol nature]
- 設定が複雑化するのでは？[Configuration complexity for BGP?]
  - 自動生成するので問題なし　[Not a problem with automated configuration generation]
  - データセンター内部はWANに比べ静的だからできる　[Especially in static environments such as data-center]
- 収束性はどうか？[What about convergence properties?]
  - ポリシーはできるだけ単純にしている[Simple BGP policy and route selection helps.]
  - ベストパス選択アルゴリズムがうまく機能する[Best path is simply shortest path (respecting AS_PATH)]
  - 収束は最悪でも数秒、ほとんどの場合1秒以下
    [Worst case convergence is a few seconds, most cases less than a second.]

ラボ試験から得た教訓 [Validating Protocol Assumptions]

OSPFとBGPの比較試験を実施（詳しくは最後のスライド）

[We simulated PoC tests using OSPF and BGP, details at end of Deck.]

- 付記：いくつかの問題はベンダー固有でもあり、リンクステートプロトコルだけで実装は可能。ベンダー毎チューニングがかなり必要。[Note: some issues were vendor specific ☺ Link-state protocols could be implemented **properly**!, but requires tuning.]

- OSPFの場合、LSDBが多量のベストパスになりえない経路で満たされる。
  [Idea is that LSDB has many “inefficient” non-best paths.]

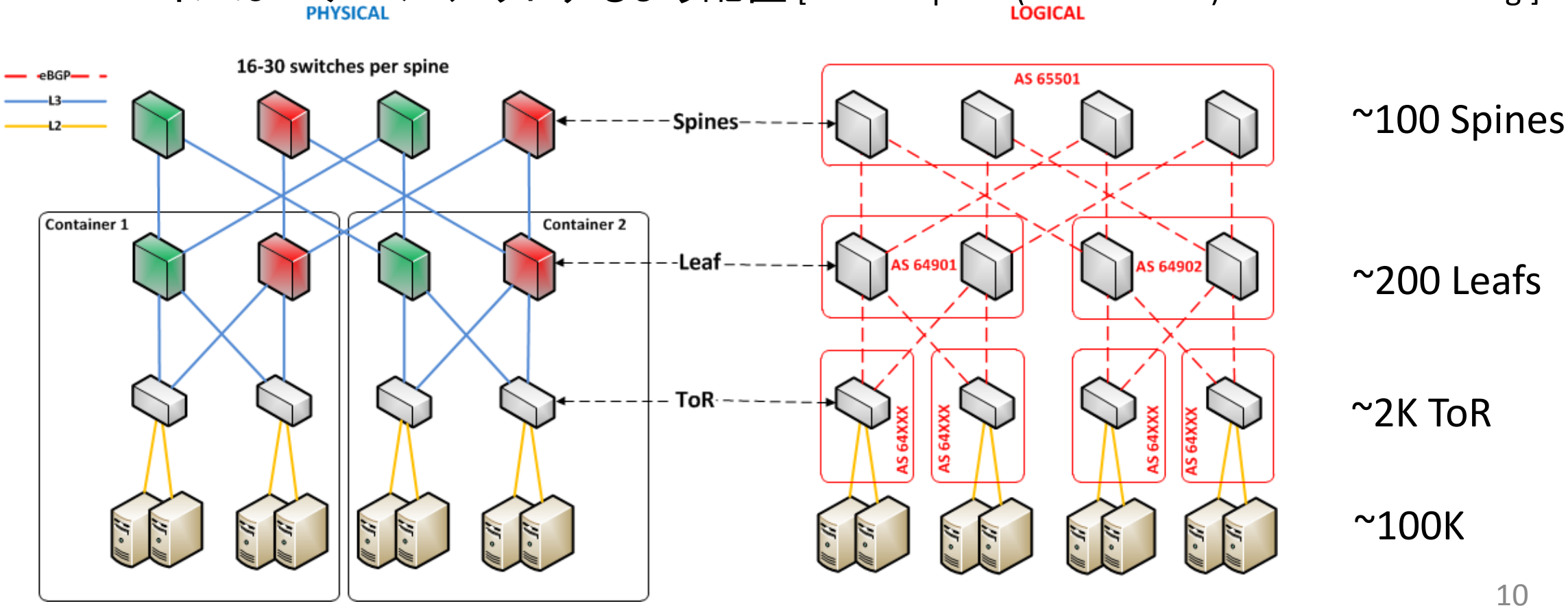
- リンク障害時にそれらの経路がベストになり、FIBにInstallされる。
  [On link failure, these “inefficient” non-best paths become best paths and are installed in FIB.]

- 結果、FIB容量が限界に達し、ゲームオーバー。
  [This results in a surge in FIB utilization---Game Over.]

- BGPなら原理的にこれを回避可能。[With BGP, ASPATH keeps only “useful” paths---no surge.]

ルーティング設計[Routing Design]

- デバイス同士は単一論理リンクのみ。全階層eBGP。IGPなし。
  [Single logical link between devices, eBGP all the way down to the ToR.]
- ToRごとに別AS番号を割当。そのASはコンテナごとに再利用。
  [Separate BGP ASN per ToR, ToR ASN's reused between containers.]
- スパインはスケールアウトするよう配置 [Parallel spines (Green vs Red) for horizontal scaling.]



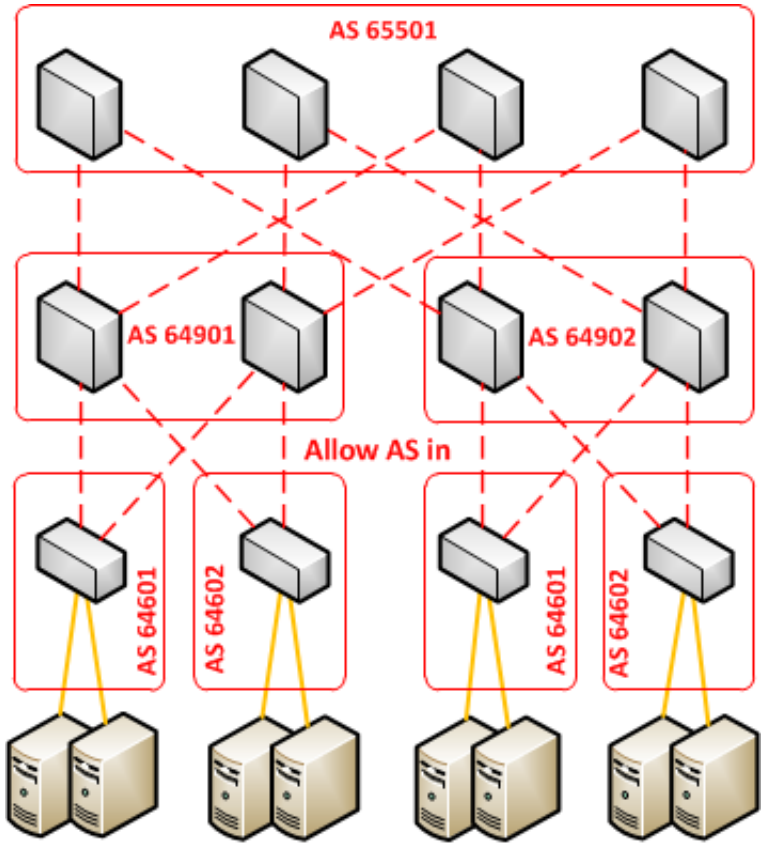
BGP設計に関して[BGP Routing Design Specifics]

- BGP AS_PATH Multipath Relaxの使用
  - AS-PATHが異なっていてもECMPできるように
    [For ECMP even if AS_PATH doesn't match.]
  - AS-PATH長が同じならそれでいい
    [Sufficient to have the same AS_PATH length]
- 2-octet private BGP ASN'sを使用
  - WANとの境界でAS番号を隠すのが容易
    [Simplifies path hiding at WAN edge (**remove private AS**)]
  - WAN境界でのフィルタリングが書きやすい
    [Simplifies route-filtering at WAN edge (single regex).]
  - しかし1022個しか使えない[But we only have 1022 Private ASN's…]
- 4-octet ASを使いたいがベンダーによる
  [4-octet ASNs would work, but not widely supported]

BGP設計に関して: “Allow AS In”の使用

[BGP Specifics: Allow AS In]

- AS番号の1022個制限をどうするか？
  [This is a numbering problem: the amount of BGP 16-bit private ASN’s is limited]
- 対策 : ToRではPrivate AS番号を再利用する.[Solution: reuse Private ASNs on the ToRs.]
- “*Allow AS in*” on ToR eBGP sessions.
- ToRのAS番号付けはコンテナかクラスター毎にまとめる。
  [ToR numbering is local per container/cluster.]
- *ベンダー側の実装依存だが。。*
  [Requires vendor support, but feature is easy to implement]

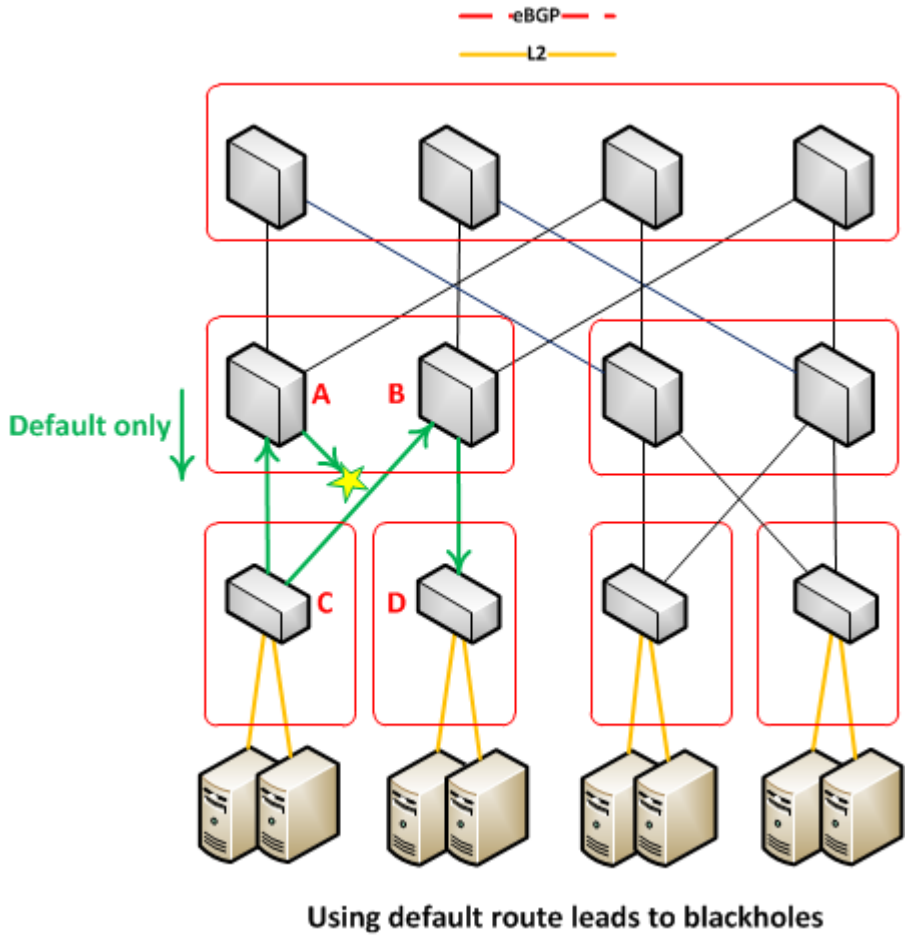


AllowAS In configured on ToR eBGP sessions to the Leafs

デフォルトルートと経路集約の問題

[Default Routing and Summarization]

- Default route利用はデータセンター外部への到達用途のみとする[Default route for external destinations only.]
- More specificは隠さないこと。[Don't hide more specific prefixes.]
- さもないとリンク障害がブラックホールを引き起こす[O.W. Route Black-Holing on link failure!]
- If D advertises a prefix P, then some of the traffic from C to P will follow default to A. If the link AD fails, this traffic is black-holed.
- If A and B send P to C, then A withdraws P when link AD fails, so C receives P only from B, so all traffic will take the link CB.
- **サーバLANの経路集約には同じ問題がある**
  [Similarly for summarization of server subnets]



Using default route leads to blackholes

運用上の問題 [Operational Issues with BGP]

- ベンダー毎の実装機能の相違
  [Lack of Consistent feature support:]
  - Not all vendors support everything you need, e.g.:
  - BGP Add-Path
  - 32-bit ASNs
  - AS_PATH multipath relax
- 相互運用性問題[Interoperability issues:]
  - CoPPとCPU queuingを一緒に使ったケース
    [Especially when coupled with CoPP and CPU queuing (Smaller L2 domains helps--less dhcp]
  - 小さなミスマッチが大障害を引き起こす！
    [Small mismatches may result in large outages!]

運用上の問題[Operational Issues with BGP]

- 予期しないデフォルト動作[Unexpected ‘default behavior’]
  - 最古のパスがbestになるとか。
    [E.g. selecting best-path using ‘oldest path’]
  - 隣のルータにAS-PATH Multipath Relaxがなかった場合とか。
    [Combined with lack of as-path multipath relax on neighbors…]
- ハッシュの再利用によるトラフィックの偏り
  [Traffic polarization due to hash function reuse]
  - しょっちゅう起こる[This is not a BGP problem but you see it all the time]
- 過度にAggressiveなタイマー値を設定した場合
  - セッションフラップがCPUを浪費
    [Overly aggressive timers – session flaps on heavy CPU load]
- RIB/FIB不整合 [inconsistencies]
  - This is not a BGP problem but it is consistently seen in all implementations

The Next Step: BGP SDN for Data-Centers

Tim LaBerge, Edet Nkposong, 北島直紀

Microsoft

SDNユースケース [SDN Use Cases for Data-Center]

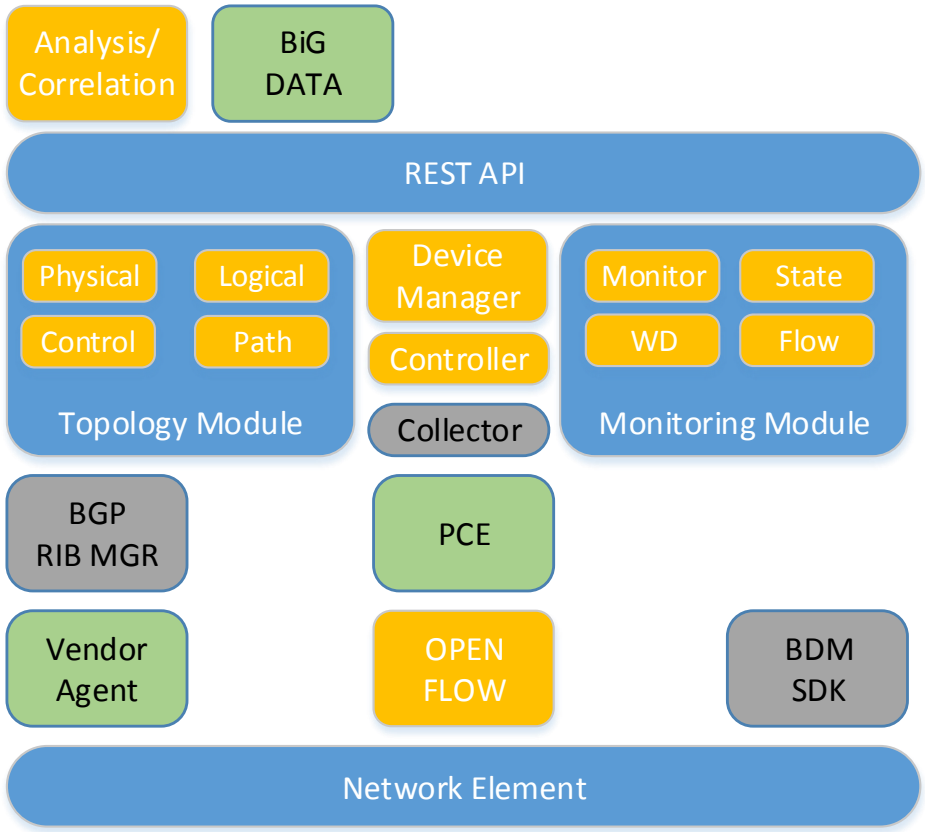
- ECMP Anycast経路の注入 [Injecting ECMP Anycast prefixes]
  - 実装済み（スライド最後のReferenceを参照） [Already implemented(see references.)]
  - ロードバランスに使用 [Used for load-balancing in the network]
  - 最小限のBGPスピーカで事足りるようにしたい [Uses a “minimal” BGP speaker to inject routes.]
- リンクやデバイスを流れるトラフィックのOn/Offに利用
  [Moving Traffic On/Off of Links/Devices]
  - Gracefulリロードやメンテナンス自動化
    [Graceful reload and automated maintenance.]
  - 障害時の原因デバイスの孤立化 [Isolating network equipment experiencing grey failures.]
- ECMPバランス比率の変更 [Changing ECMP traffic proportions]
  - 重み付きトラフィック負荷分散 [Unequal-cost load distribution in the network]
  - 例：リンク障害時の最適な負荷リバランス（後述）
    [E.g. to compensate for various link failures and re-balance traffic. (Network is symmetric but traffic may not be.]

BGP SDN Controller

- Focus is the DC – controllers scale within DC, partition by cluster, region and then global sync

- Controller Design Considerations
  - Logical vs Literal
  - Scale - Clustering
  - High Availability
  - Latency between controller and network element

- Components of a Controller
  - Topology discovery
  - Path Computation
  - Monitoring and Network State Discovery
  - REST API



Controller is a component of a Typical Software Orchestration Stack

BGP SDN Controllerの基礎 [BGP SDN Controller Foundations]

- **OpenFlowと比較して　[Why BGP vs OpenFlow]**
  - 新たなプロトコルは不要 [No new protocol.]
  - 新たなシリコンも不要 [No new silicon.]
  - 新たなOSもSDKも不要 [No new OS or SDK bits.]
  - コントローラはどっちにしろ必要 [Still need a controller.]

- **実装方法**
  [Have "literal" SDN, software generates graphs that define physical, logical, and control planes.]
  - 基底状態を定義[Graph define the ideal ground state, used for config generation.]
  - 現在の状況をリアルタイムに入手[Need the current state in real time.]
  - 変更したい内容を計算[Need to compute new desired state.]
  - 変更したい内容をForwarding Stateに注入[Need to inject desired forwarding state.]

- **RIB経由でネットワークをプログラムする [Programming forwarding via the RIB]**
  - BGPを通じてトポロジー情報を入手[Topology discovery via BGP listener (link state discovery).]
  - BGPセッション経由のPrefix注入によりルーティングを変える
    [RIB manipulation via BGP speaker (injection of more preferred prefixes.]

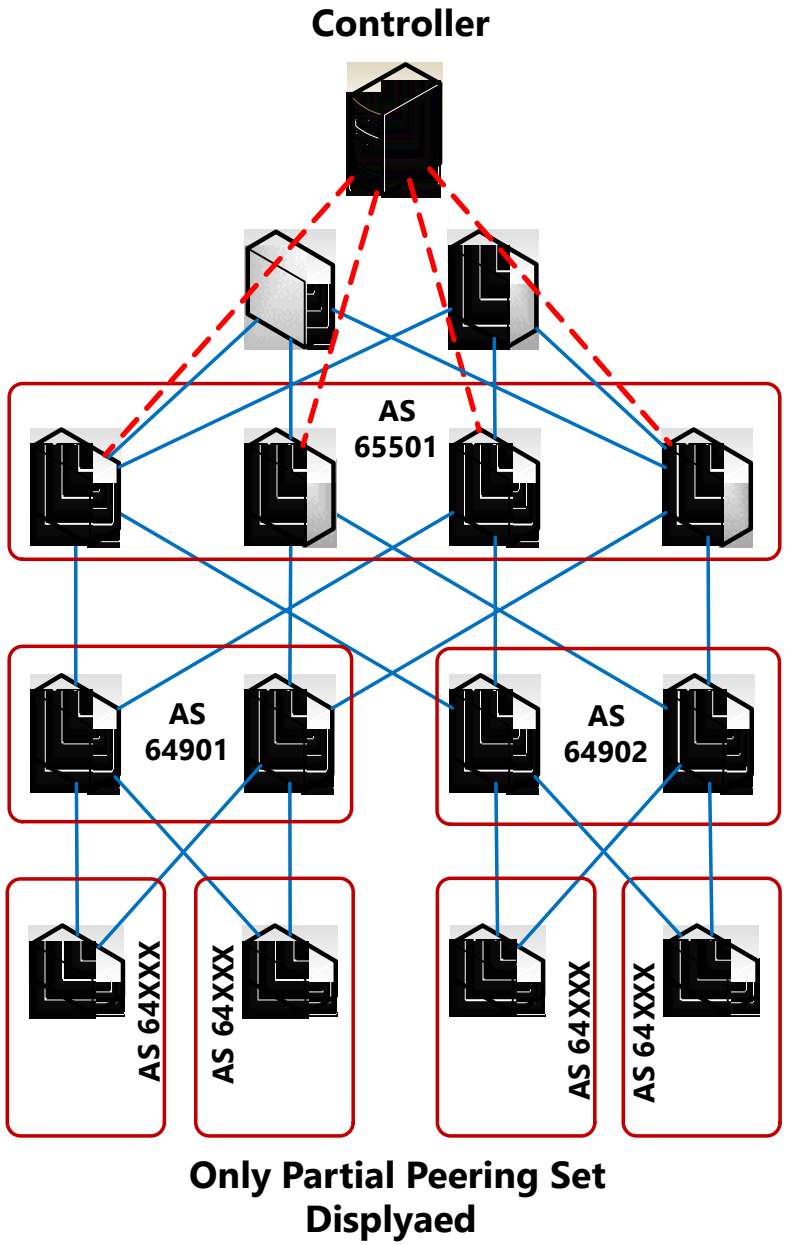
Network Setup

Device Configuration

- コントローラとのピア(passive)
  [Template to peer with the central controller (passive listening)]
- コントローラ経由の経路を優先
  [Policy to **prefer** routes injected from controller]
- ある種の経路だけをコントローラへ広報（後述）[Policy to announce only **certain** routes to the controller]

Peering to the Controller

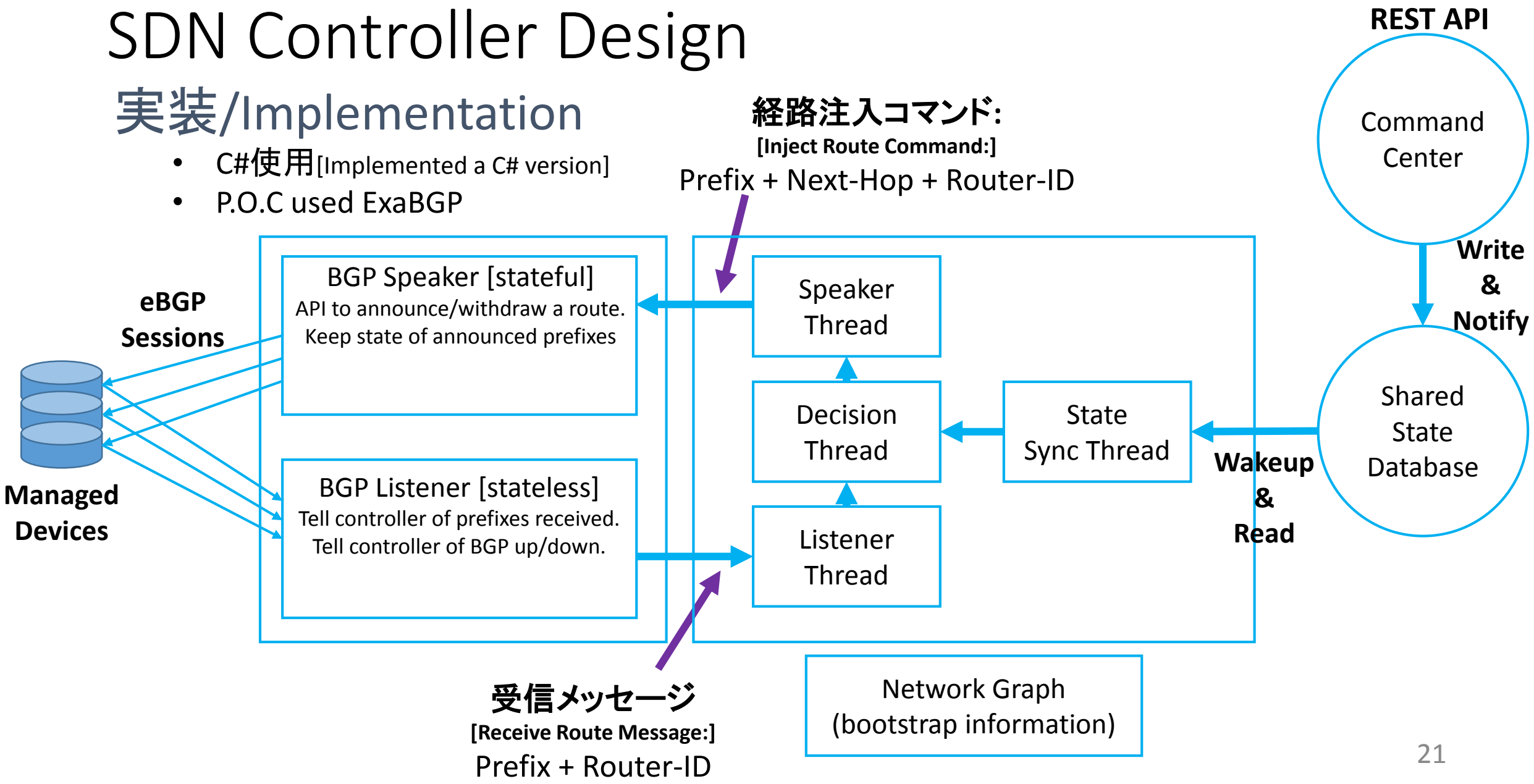
- 全デバイスとピア(multihop)
  [Multi-hop peering with all devices.]
- ピアが安定している必要あり
  [Key requirement: path resiliency]
- CLOSは非常に密結合だからネットワーク分断は極めて起こりにくい
  [CLOS has very rich path set, network partition is very unlikely.]



SDN Controller Design

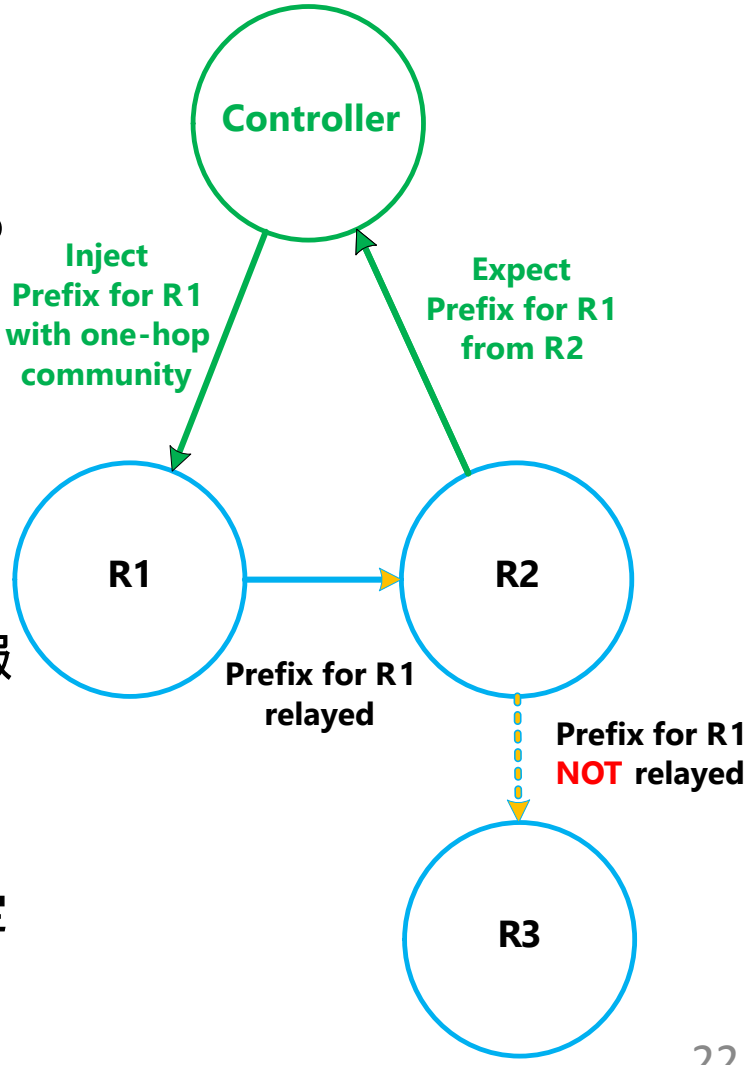
実装/Implementation

- C#使用[Implemented a C# version]
- P.O.C used ExaBGP



ネットワークグラフの生成[Building Network Link State]

- 目的：実ネットワークのリンク状態を学習
  [Goal: Build Link-State of Live Network]
  - ある種のControl Plane Pingを使う
    [Use a special form of "**control plane ping**"]
  - BGPセッション状態がリンクの健全性に拠っていることを利用する
    [Rely on the fact that BGP session reflects "link health"]
  - デバイス間のBGPセッションは一つとする
    [Assumes single BGP session b/w two devices]
- 動作[How it works]
  - デバイスごとに固有の/32を生成。例えばR1用。
    [Create a /32 prefix for every device, e.g. R1.]
  - その/32をR1に注入
    [Inject prefix into device R1]
  - この/32はR1に直結の全デバイス[R2…Rn]からコントローラに広報されてくるはず
    [Expect to hear this prefix via all devices R2…Rn, directly connected to R1]
  - 広報されてきたなら、R1-R2リンクはupと認識。
    [If heard, declare link between R1 – R2 as up]



コミュニティタグとポリシーで、固有/32がワンホップ先から再広報されないよう設定
[Community tagging + policy ensures prefix only leaks "one hop" from point of injection, but is reflected to the controller.]

ルーティング動作の上書き[Overriding Routing Decisions]

- フォワーディングDBの操作[Populating Forwarding Databases]

コントローラは全てのサーバLANとデバイスを認識
[The controller knows of all server subnets and devices]

コントローラはSPF計算を行い。。:[The controller runs SPF and]

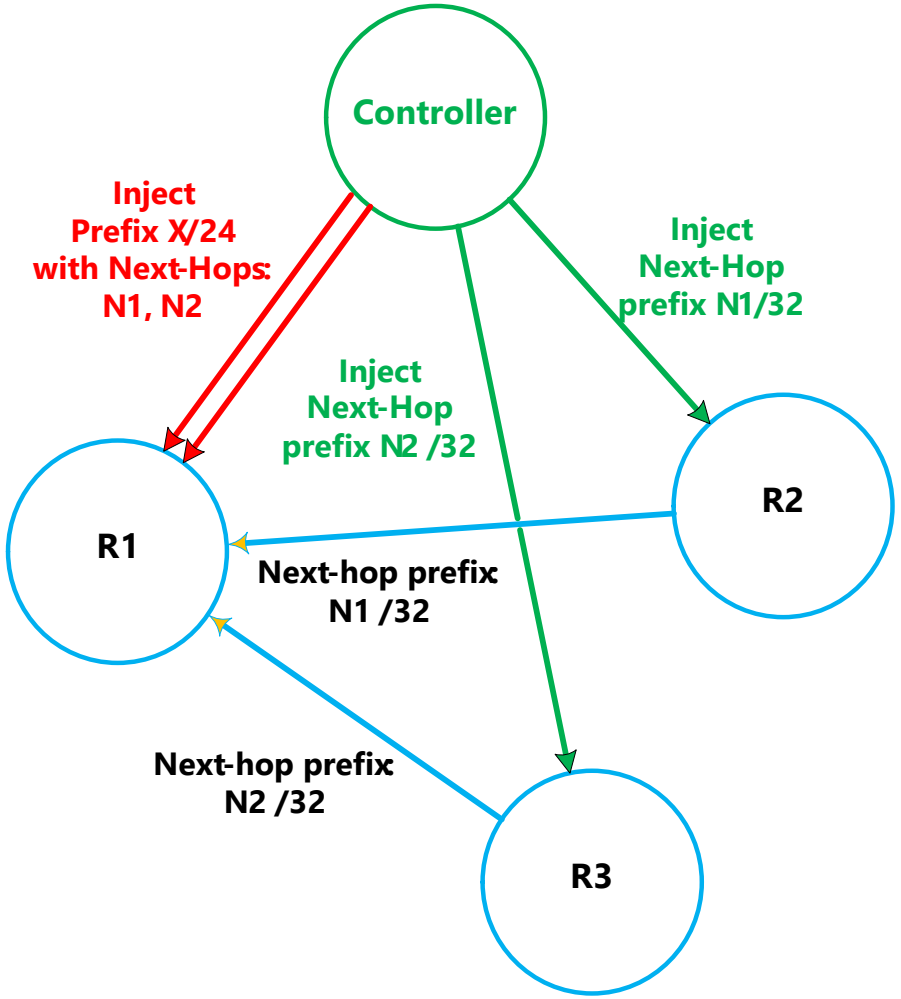
- 全デバイス毎、全Prefixについてネクストホップを算出
  [Computes next hops for every prefix at every device]
- 計算結果を静的なネットワークグラフで算出した値と比較
  [Check if this is different from “static network graph” decisions]
- 差違だけをルータに送信[Only push the “deltas”]
- プレフィックスは”third party” next-hop付きで送信する(next slide)
  [Prefixes are pushed with “**third party**” next-hops (next slide) and a better metric.]

- ポイント[Key observations]
  - コントローラがトポロジーを完全に把握していること[Controller has full view of the topology]
  - デフォルトのルーティング動作と変わらない場合は、何もしないこと[Zero delta if no differentce from “default” routing behavior]
  - コントローラがトラフィック迂回のためリンク障害を宣言できること。実際にはupでも。
    [Controller may declare a link down to re-route traffic.]
  - コントローラ障害の場合は、BGPのデフォルト動作に戻れる。
    [Seamless fallback to default BGP routing in the case of controller failure.]

ルーティング動作の上書き（続き）
[Overriding Routing Decisions cont.]

- Next-hopの扱い[What about next-hops?]
  - 注入される経路はthird-party N-Hを持つ。
    [Injected routes have third-party next-hop]
  - そのNHはBGPテーブルを参照して解決。
    [Those need to be resolved via BGP]
  - **そのNH自体も注入する。**
    [Next-hops have to be injected as well!]
  - 固有のNH/32をデバイス毎に生成。
    [A next-hop /32 is created for every device]
  - グラフ生成に使ったのと同じ”one hop”属性を持たせる。[Same “**one hop**” BGP community used]

- ECMP経路の注入について[Injecting ECMP Routes]
  - BGPセッション毎にone pathだけ追加できる。
    [By default only one path allowed per BGP session]
  - よってAdd-Pathか複数セッションが必要。
    [Need either Add-Path or multiple peering sessions]
  - ECMP上限がセッションの数で決まるのはマズい。
    [Worst case: # sessions = ECMP fan-out]
  - Add-Path **Receive-Only** would help!



ルーティング動作の上書き（続き）
[Overriding Routing Decisions cont.]

- REST APIでネットワークの状態を上書き
  [Simple REST to manipulate network state “overrides”]
- API一覧 [Supported calls:]
  - Linkのshutdown/un-shutdown (Logically)
  - Deviceのshutdown/un-shutdown (Logically)
  - Prefixを特定のnext-hopとともに広報 [Announce a prefix with next-hop set via a device]
  - 現在のLink/Deviceの状態を取得 [Read current state of the down links/devices]

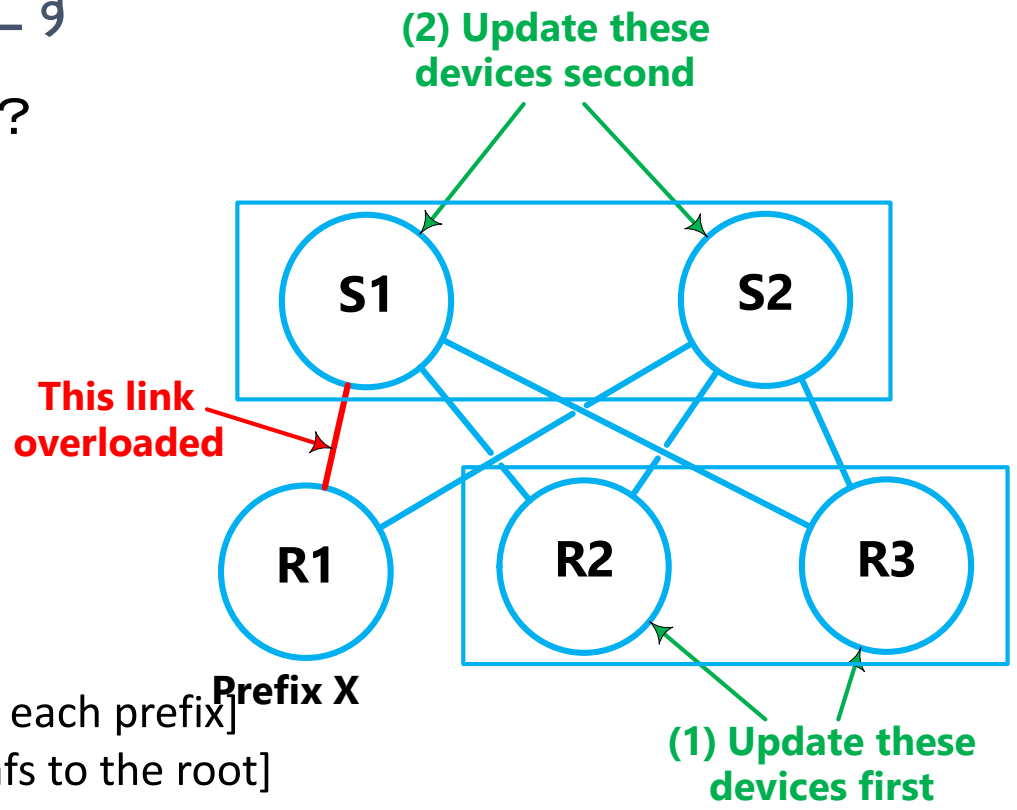
PUT http://<controller>/state/link/up=R1,R2&down=R3,R4

- ステートデータベースは必須[This requires a state database]
  - コントローラが再起動しても維持される必要あり [State is **persistent** across controller reboots]
  - 複数コントローラ間で共有される必要あり [State is **shared** across multiple controllers]

順序付きFIBプログラミング [Ordered FIB Programming]

- 分散プログラミングモデルは問題を引き起こす
  [Distributed programming poses issues]
  デバイス上のBGP RIBをランダムに更新したらどうなる？
  [If updating BGP RIB's on devices in random order…]
  …RIB/FIBテーブルの同期がズレてマイクロループが発生しうる。
  […RIB/FIB tables could go out of sync
  Micro-loops problem!]

- 中央集権的に動作する必要あり
  [Central controller helps]
  リンクの状態を変える場合は、都度：
  [For every link state change]
  - 影響を受けるPrefixを特定 [Find prefixes affected]
  - Prefix毎にreverse-SPF計算実施 [Build reverse-SPF for each prefix]
  - 枝葉から値に向かって更新する [Update from the leafs to the root]
  - コントローラが更新順序を制御 [Controller sequences the updates]
  - つまり爆縮するということ。爆発じゃなくて。[Implode, no explode]

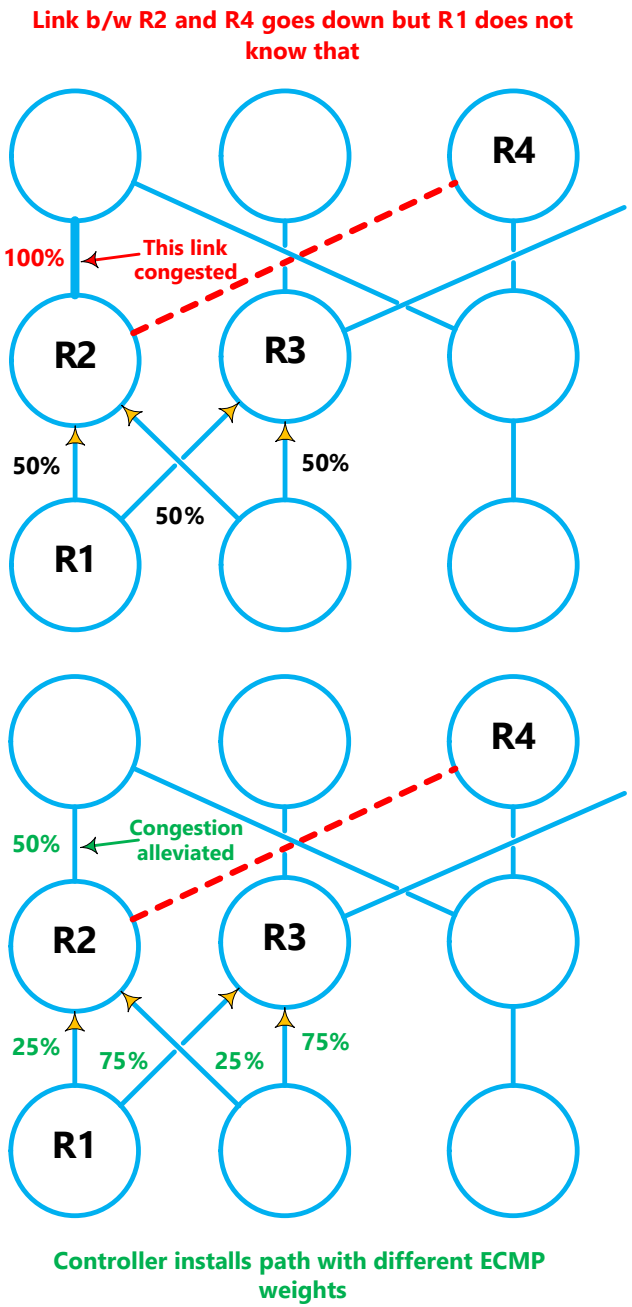


Note: assumes FIB programming is fast!

Roadmap

Traffic Engineering

- ECMPは非常に素朴[ECMP Routing is oblivious]
  リンク障害がトラフィック不均衡を引き起こす
  [Failures may cause traffic imbalances]
  例えば [This includes:]
  - 物理障害 [Physical failures]
  - 論理的に孤立化させた場合も [Logical link/device overloading]
- コントローラの動作 [Central controller helps]
  - トラフィック分布を再計算 [Compute new traffic distribution]
  - ECMPの重みを算出 [Program weighted ECMP]
  - BGP Link Bandwidthでシグナリング
    [Signal using **BGP Link Bandwidth]**
  - ベンダー実装はそれほど進んでいない
    [Not implemented by most vendors] ☹



Traffic Engineering (cont.)

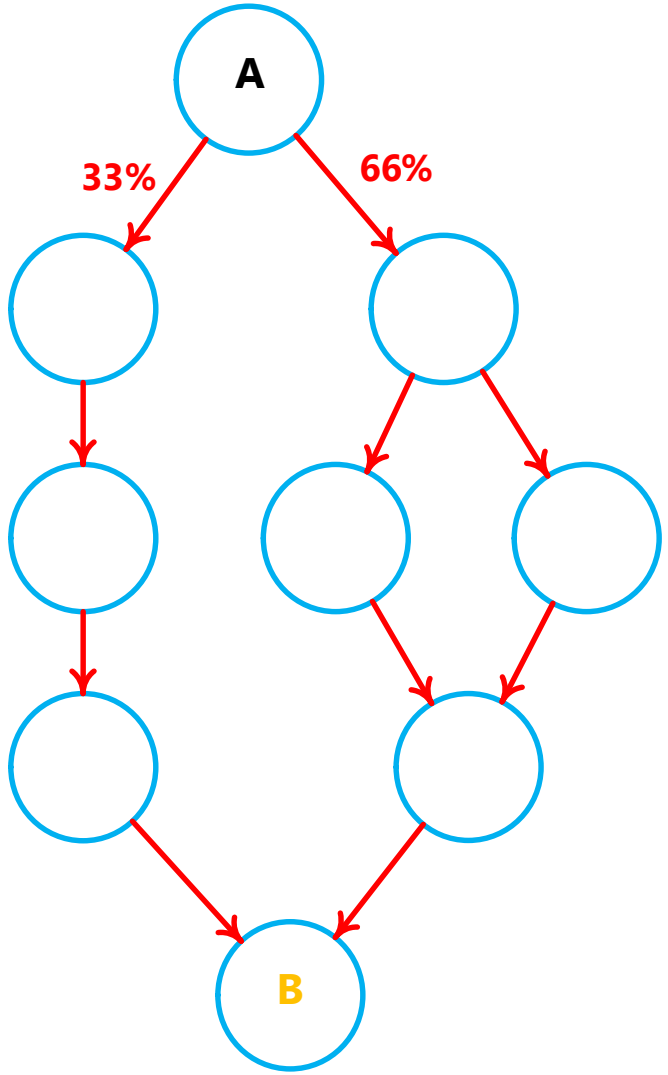
- 実装 [Implementation]

  情報収集 [Requires knowing]
  - トラフィックマトリクス(TM) [traffic matrix (TM)]
  - トポロジーと容量 [Network topology and capacities]

  線形計画問題の解をもとめる [Solves Linear Programming problem]

  ECMP重みを算出 [Computes ECMP weights]
  - Prefix毎 [For every prefix]
  - Hop毎 [At every hop]

    得られたTMに対する最適化[Optimal for a given TM]
- すなわち [This has implications]
  - リンクup/down毎に再プログラムを実施。
    [Link state change causes reprogramming]
  - より多くのStateが注入される事になる。
    [More state pushed down to the network]



ベンダーへの要請[Ask to the vendors!]

- 重み付きECMPのサポート [Weighted ECMP in DC switches]
  - ハード的制約はないはず [Most common HW platforms **can do it** (e.g. Broadcom)]
  - BGPによるシグナリング機構の実装もそう難しくなかろう
    [Signaling via BGP does not look complicated either]
  - *Note: ハードウェアリソース的な制約はあるかも*
    [Note: Has implications on hardware resource usage]

- コンシステントなHashing機構 [Consistent Hashing]
  - 重み付きECMPとうまく動くように [Goes well with weighted ECMP]
  - Well defined in RFC 2992

- Add-Path: Receive Only
  - 非標準（ヤレヤレ）[Not a standard (sigh)]
  - この機能はとても必要 [We really like receive-only functionality]

Conclusions

学んだこと [What we learned]

- BGP SDNは現実解 [BGP SDN is Practical]
  - 新たなファームウェアもAPIも不要 [Does not require new firmware, silicon, or API's]
  - BGPの拡張が少しあればそれで十分動く [Some BGP extensions are nice to have.]
- BGP SDNはシンプルで安定 [BGP SDN is Simple and Stable]
  - BGP codeは枯れている [BGP Code is tends to be mature.]
  - 通常のルーティングへの切り戻しが容易 [Easy to roll-back to default BGP routing]
- BGP SDNは効率的 [BGP SDN is Efficient]
  - 現状の問題を解決可能。将来的な適用範囲も広い。
    [Solves our current problems and allows solving more]
- MPLS抜きでもTE可能 [TE is possible without MPLS]

Questions?

Contacts:

Edet Nkposong - edetn@microsoft.com

Tim LaBerge - Tim.LaBerge@microsoft.com

北島直紀 - naokikit@microsoft.com

References

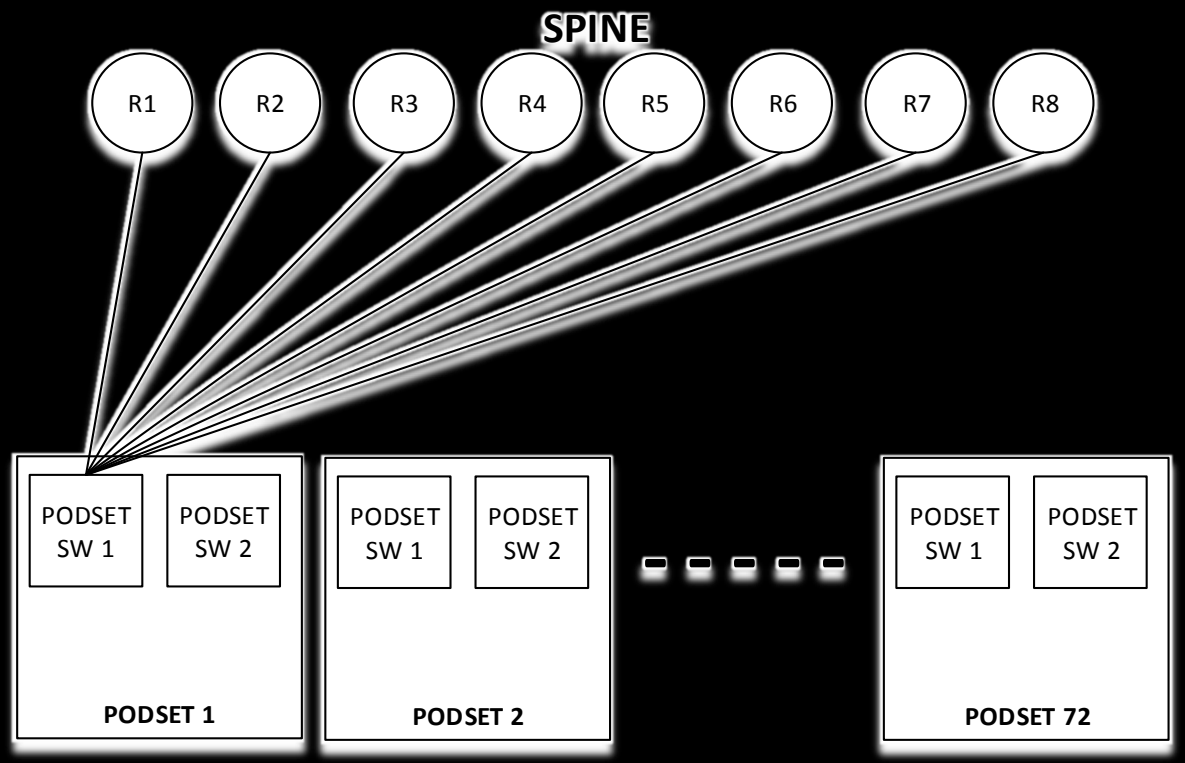
- BGP Routing for Data-Centers
  http://datatracker.ietf.org/doc/draft-lapukhov-bgp-routing-large-dc/
- ExaBGP
  http://code.google.com/p/exabgp/
- BGP Link-Bandwidth
  http://datatracker.ietf.org/doc/draft-ietf-idr-link-bandwidth/
- BGP SDN
  http://datatracker.ietf.org/doc/draft-lapukhov-bgp-sdn/

- BGP is the Better IGP http://www.nanog.org/meetings/nanog55/presentations/Monday/Lapukhov.pdf
- BGP for SDN in the Data Center
  http://www.nanog.org/sites/default/files/wed.general.brainslug.lapukhov.20.pdf
- Autopilot: Automatic Data Center Management http://research.microsoft.com/pubs/64604/osr2007.pdf
- Ananta: Cloud Scale Loadbalancing http://research.microsoft.com/en-us/people/chakim/slb-sigcomm2013.pdf

Simulation Tests --- IGP vs BGP Protocol Selection

OSPF – Route Surge Test

- Test bed that emulates 72 PODSETs
- Each PODSET comprises 2 switches
- Objective – study system and route table behavior when control plane is operating in a state that mimics production



Test Bed

- 4 Spine switches
- 144 VRFs created on a router – each VRF = 1x podset switch
  - Each VRF has 8 logical interfaces (2 to each spine)
  - This emulates the 8–way required by the podset switch
- 3 physical podset switches
- Each podset carries 6 server–side IP Subnets

Test Bed

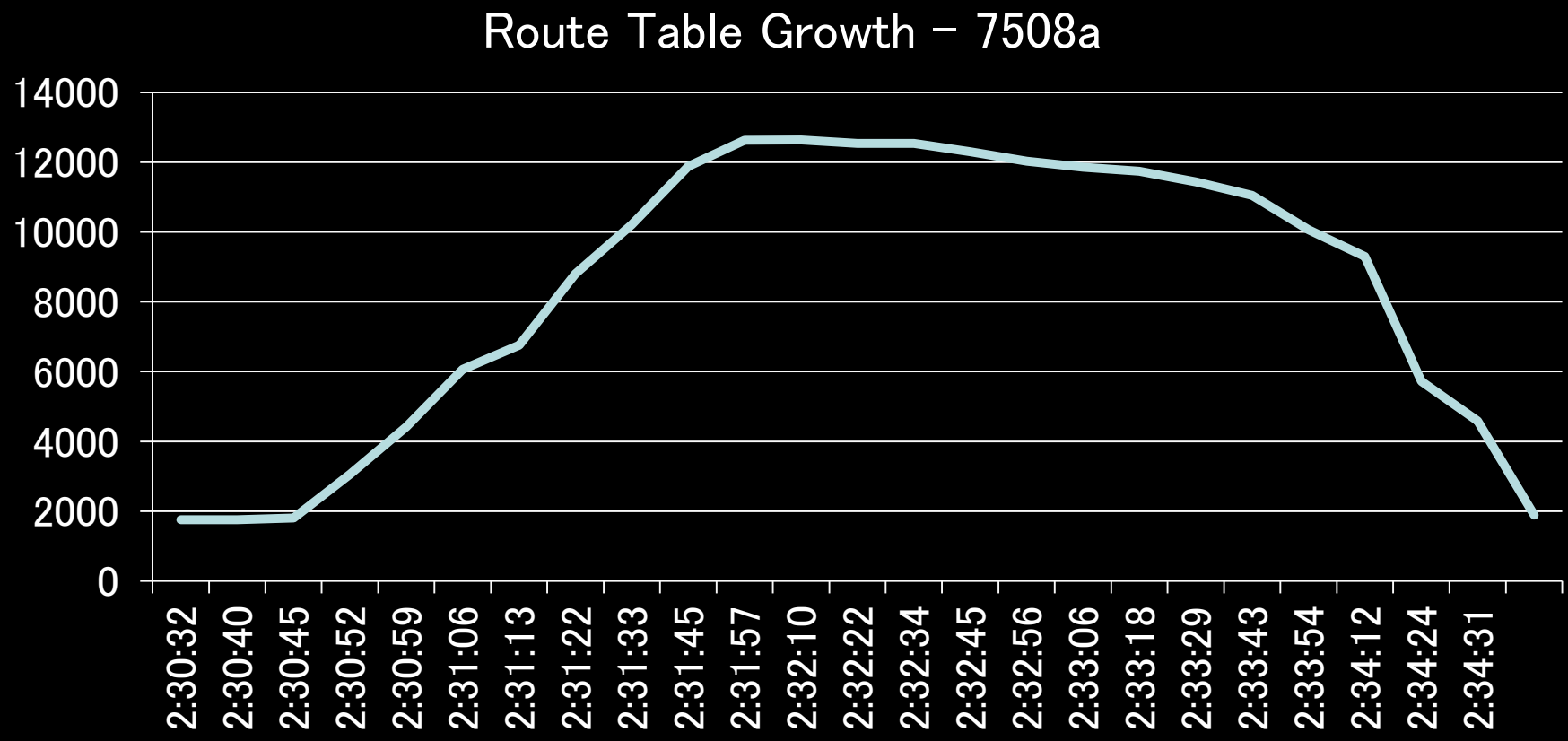
- Route table calculations
  - Expected OSPF state
    - 144 x 2 x 4 = 1152 links for infrastructure
    - 144 x 6 = 864 server routes (although these will be 4-way since we have brought everything into 4 spines (instead of 8)
    - Some loopback addresses and routes from the real podset switches
    - We expect ~ (144 x 2 x 4) + (144 x 6) – 144 = 1872 routes

- Initial testing proved that the platform can sustain this scale (control and forwarding plane) – document name

- What happens when we shake things up ?

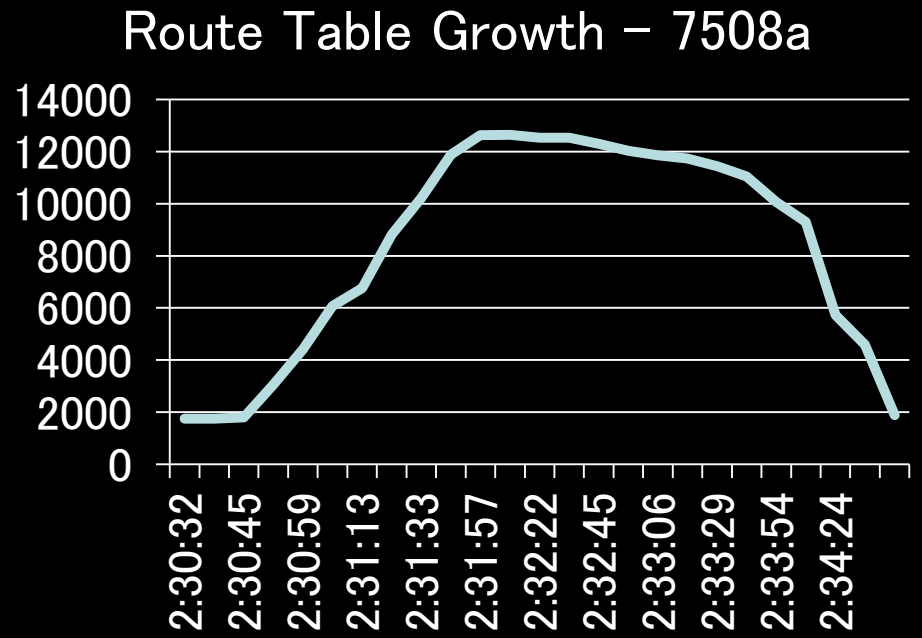
OSPF Surge Test

- Effect of bringing up 72 podset (144 OSPF neighbors) all at once



OSPF Surge Test

- Why the surge ?
  - As adjacencies come up, the spine learns about routes through other podset switches
  - Given that we have 144 podset switches, we expect to see 144-way routes although only 16-way routes are accepted



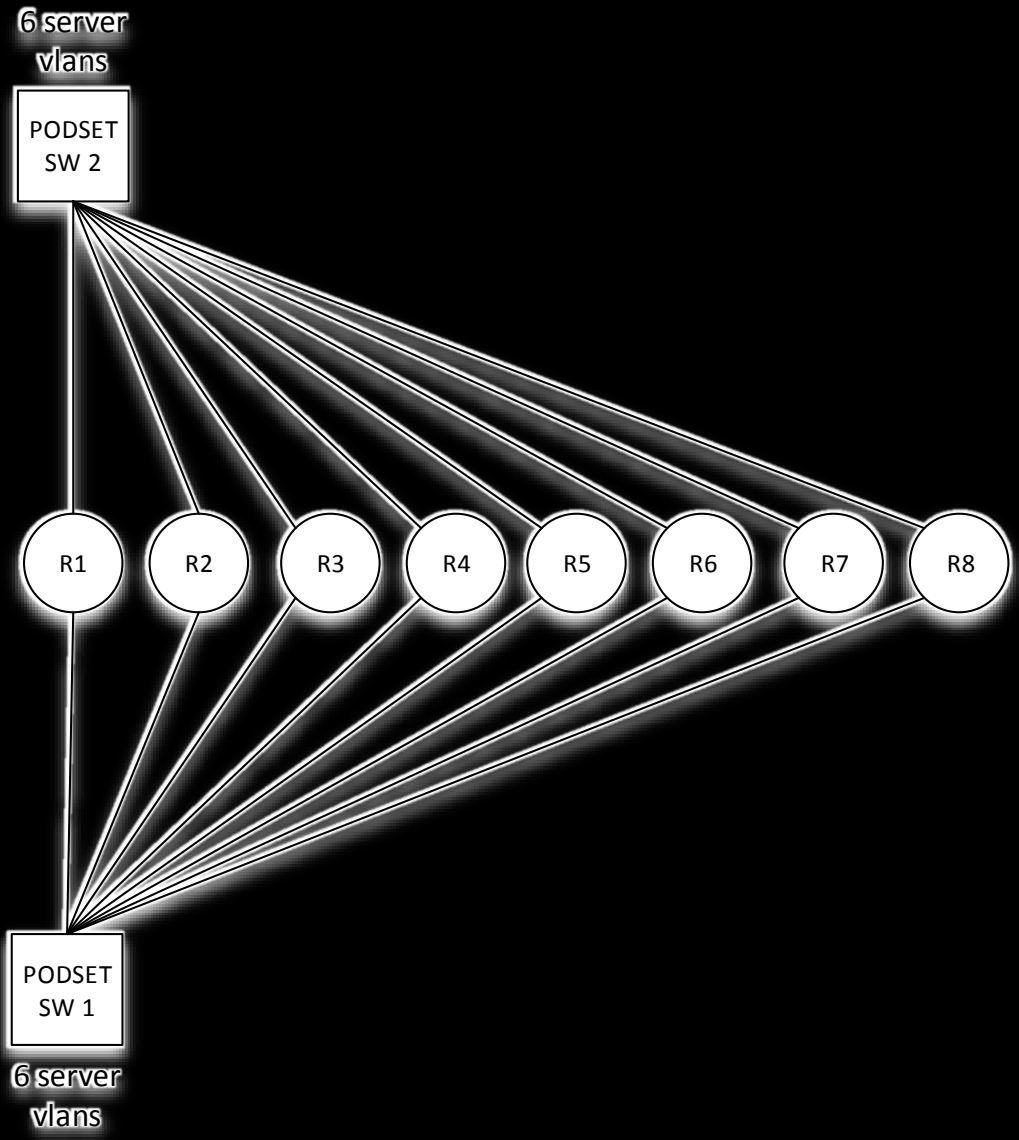
- Sample route

```
O    192.0.5.188/30 [110/21] via 192.0.1.33
                             via 192.0.2.57
                             via 192.0.0.1
                             via 192.0.11.249
                             via 192.0.0.185
                             via 192.0.0.201
                             via 192.0.2.25
                             via 192.0.1.49
                             via 192.0.0.241
                             via 192.0.11.225
                             via 192.0.1.165
                             via 192.0.0.5
                             via 192.0.12.53
                             via 192.0.1.221
                             via 192.0.1.149
                             via 192.0.0.149
```

- Route table reveals that we can have 16-way routes for any destination including infrastructure routes
- This is highly undesirable but completely expected and normal

OSPF Surge Test

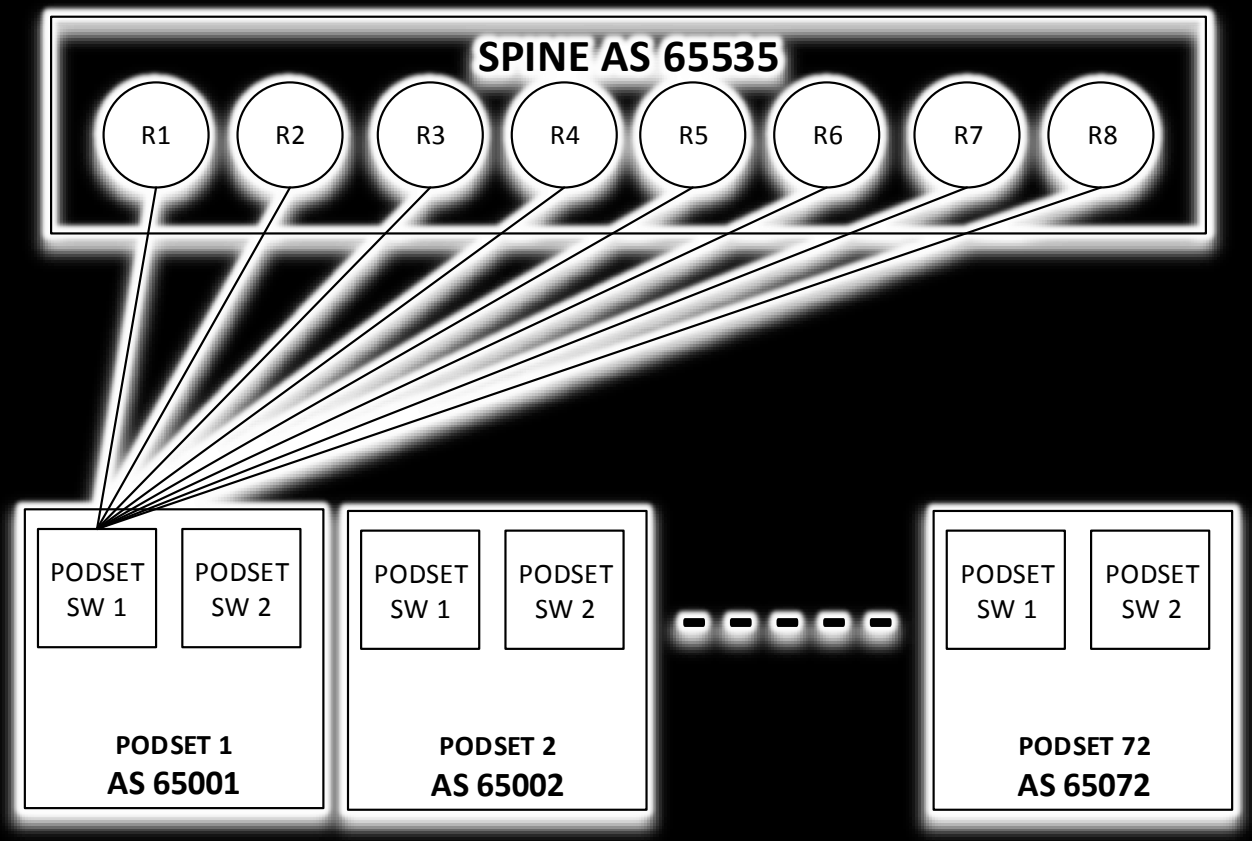
- Instead of installing a 2-way towards the podset switch, the spine ends-up installing a 16-way for podset switches that are disconnected
- If a podset switch-spine link is disabled, the spine will learn about this particular podset switches IP subnets via other shims
  - Unnecessary 16-way routes
- For every disabled podset switch-spine link, the spine will install a 16-way route through other podset switches
- The surge was enough to fill the FIB (same timeline as graph on slide 12)




```
2011-02-16T02:33:32.160872+00:00 sat-a75ag-poc-1a SandCell: %SAND-3-
ROUTING_OVERFLOW: Software is unable to fit all the routes in hardware
due to lack of fec entries. All routed traffic is being dropped.
```

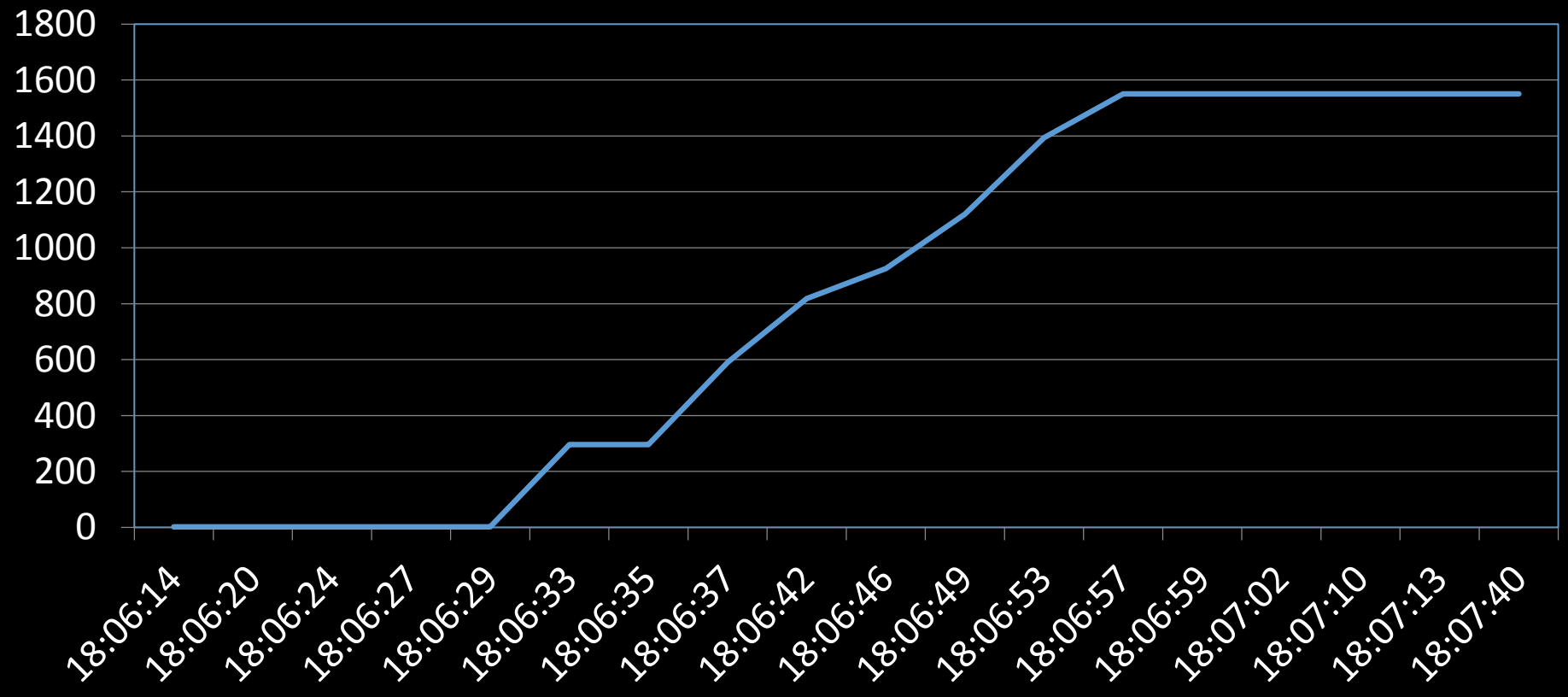
BGP Surge Test

- BGP design
  - Spine AS 65535
  - PODSET AS starting at 65001, 65002 etc



BGP Surge Test

- Effect of bringing up 72 PODSETs (144 BGP neighbors) all at once



OSPF vs BGP Surge Test – Summary

- With the proposed design, OSPF exposed a potential surge issue (commodity switches have smaller TCAM limits) – could be solved by specific vendor tweaks – non standard.

- Network needs to be able to handle the surge and any additional 16-way routes due to disconnected spine-podset switch links
  - Protocol enhancements required
  - Prevent infrastructure routes from appearing as 16-way.

- BGP advantages
  - Very deterministic behavior
  - Protocol design takes care of eliminating the surge effect (i.e. spine won’t learn routes with its own AS)
  - ECMP supported and routes are labeled by the container they came from (AS #) – beautiful !