Experiences with BGP in Large Scale Data Centers:
Teaching an old protocol new tricks

Presented by: Edet Nkposong, Tim LaBerge, Naoki Kitajima

Global Networking Services Team, Global Foundation Services, Microsoft Corporation

Microsoft

Agenda

- Network design requirements
- Protocol selection: BGP vs IGP
- Details of Routing Design
- Motivation for BGP SDN
- Design of BGP SDN Controller
- The roadmap for BGP SDN

Design Requirements

Scale of the data-center network:

- 100K+ bare metal servers
- Over 3K network switches per DC

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.

Key outcome:

East ←→ West traffic profile drives need for large bisectional bandwidth.

Translating Requirement to Design

Network Topology Criteria

Support East <-> West Traffic Profile with no over-subscription

- Minimize Capex and Opex
  - Cheap commodity switches
  - Low power consumption

Use Homogenous Components

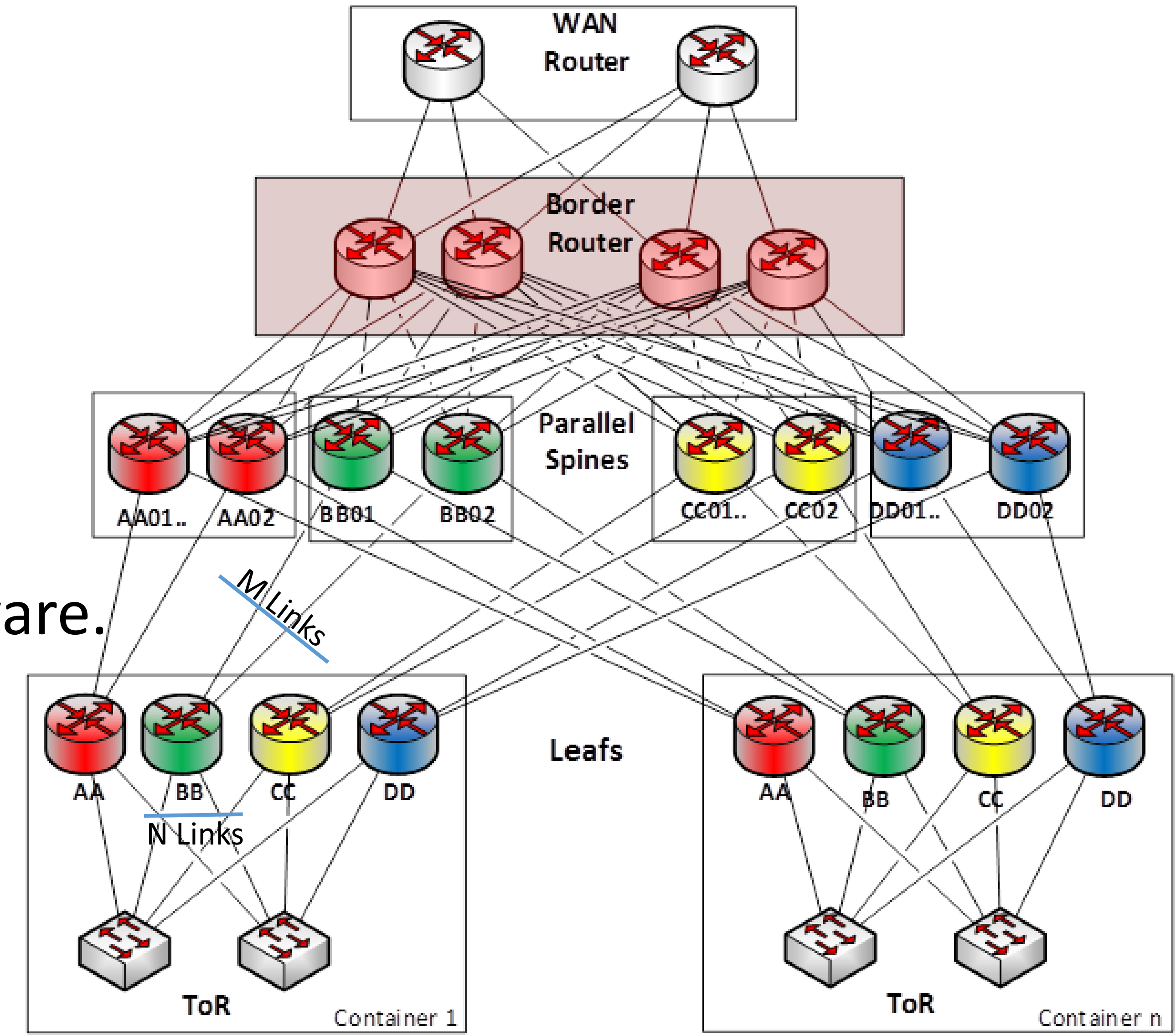
- Switches, Optics, Fiber etc
- Minimize operational complexity
- Minimize unit costs

Network Protocol Criteria

- Standards Based
- Control Plane Scaling and Stability
- Minimize resource consumption e.g. CPU, TCAM usage - predictable and low
- Minimize the size of the L2 failure domain
- Layer3 equal-cost multipathing (ECMP)
- Programmable
  - Extensible and easy to automate

Network Design: Topology

- 3-Stage Folded CLOS.
- Full bisection bandwidth (m ≥ n) .
- Horizontal Scaling (scale-out vs. scale-up)
- Natural ECMP link load-balancing.
- Viable with dense commodity hardware.
  - Build large “virtual” boxes out of small components



Network Design: Protocol

Network Protocol Requirements

- Resilience and fault containment
  - CLOS has high link count, link failure is common, so limit fault propagation on link failure.
- Control Plane Stability
  - Consider number of network devices, total number of links etc.
  - Minimize amount of control plane state.
  - Minimize churn at startup and upon link failure.
- Traffic Engineering
  - Heavy use of ECMP makes TE in DC not as important as in the WAN.
  - However we still want to “drain” devices and respond to imbalances

Why BGP and not IGP?

- Simpler protocol design compared to IGPs
  - Mostly in terms of state replication process
  - Fewer state-machines, data-structures, etc
  - Better vendor interoperability
- Troubleshooting BGP is simpler
  - Paths propagated over link
  - AS PATH is easy to understand.
  - Easy to correlate sent & received state
- ECMP is natural with BGP
  - Unique as compared to link-state protocols
  - Very helpful to implement granular policies
  - Use for unequal-cost Anycast load-balancing solution

Why BGP and not IGP? (cont.)

- Event propagation is more constrained in BGP
  - More stability due to reduced event “flooding” domains
  - E.g. can control BGP UPDATE using BGP ASNs to stop info from looping back
  - Generally is a result of distance-vector protocol nature
- Configuration complexity for BGP?
  - Not a problem with automated configuration generation. Especially in static environments such as data-center
- What about convergence properties?
  - Simple BGP policy and route selection helps.
  - Best path is simply shortest path (respecting AP_PATH).
  - Worst case convergence is a few seconds, most cases less than a second

Validating Protocol Assumptions

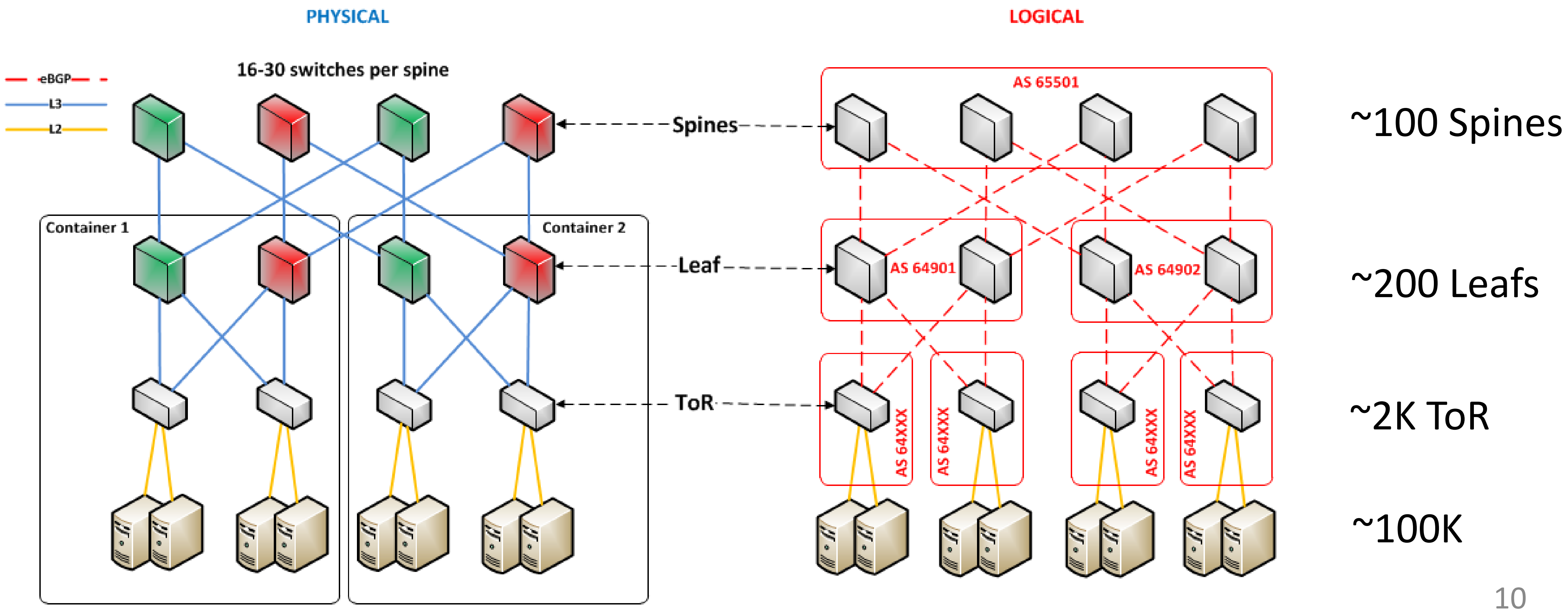
Lessons from Route Surge PoC Tests:

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

- Note: some issues were vendor specific ☺ Link-state protocols could be implemented **properly**!, but requires tuning.
- Idea is that LSDB has many “inefficient” non-best paths.
- On startup or link failure, these “inefficient” non-best paths become best paths and are installed in the FIB.
- This results in a surge in FIB utilization---Game Over.
- With BGP, ASPATH keeps only “useful” paths---no surge.

Routing Design

- Single logical link between devices, eBGP all the way down to the ToR.
- Separate BGP ASN per ToR, ToR ASN's reused between containers.
- Parallel spines (Green vs Red) for horizontal scaling.

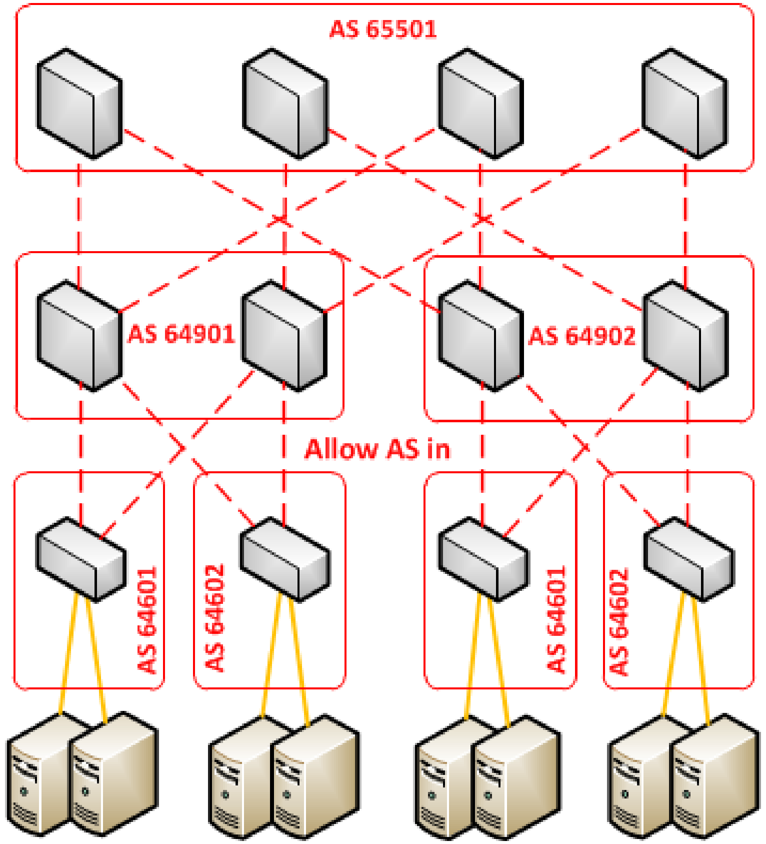


BGP Routing Design Specifics

- BGP AS_PATH Multipath Relax
  - For ECMP even if AS_PATH doesn't match.
  - Sufficient to have the same AS_PATH length
- We use 2-octet private BGP ASN's
  - Simplifies path hiding at WAN edge (**remove private AS**)
  - Simplifies route-filtering at WAN edge (single regex).
  - But we only have 1022 Private ASN's…
- 4-octet ASNs would work, but not widely supported

BGP Specifics: Allow AS In

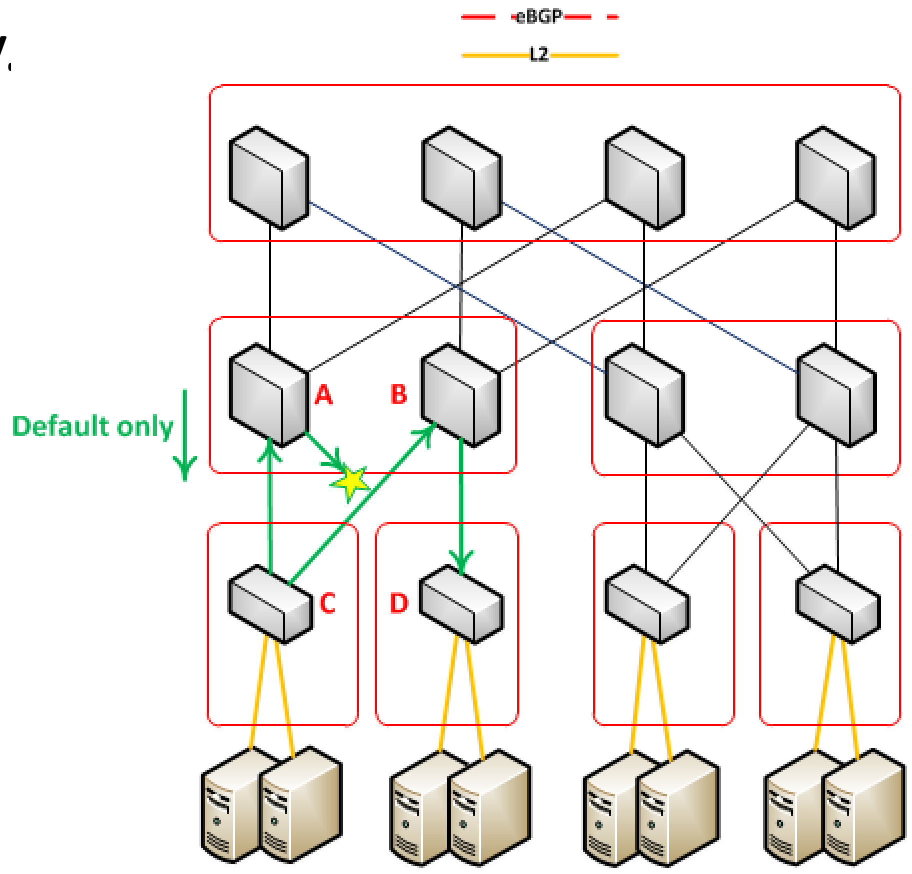
- This is a numbering problem: the amount of BGP 16-bit private ASN's is limited
- Solution: reuse Private ASNs on the ToRs.
- "*Allow AS in*" on ToR eBGP sessions.
- ToR numbering is local per container/cluster.
- *Requires vendor support, but feature is easy to implement*



AllowAS In configured on ToR eBGP sessions to the Leafs

Default Routing and Summarization

- Default route for external destinations only.
- Don't hide server subnets.
- 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.
- Similarly for summarization of server subnets.



Using default route leads to blackholes

Operational Issues with BGP

- Lack of Consistent feature support:
  - Not all vendors support everything you need.
  - BGP Add-Path
  - 32-bit ASNs
  - AS_PATH multipath relax
- Interoperability issues:
  - Especially when coupled with CoPP and CPU queuing (Smaller L2 domains helps---less dhcp)
  - Small mismatches may result in large outages!

Operational Issues with BGP

- Unexpected ‘default behavior’
  - E.g. selecting best-path using ‘oldest path’
  - Combined with lack of as-path multipath relax on neighbors…
- Traffic polarization due to hash function reuse
  - This is not a BGP problem but you see it all the time
- Overly aggressive timers – session flaps on heavy CPU load
- RIB/FIB 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, Naoki Kitajima

Microsoft

SDN Use Cases for Data-Center

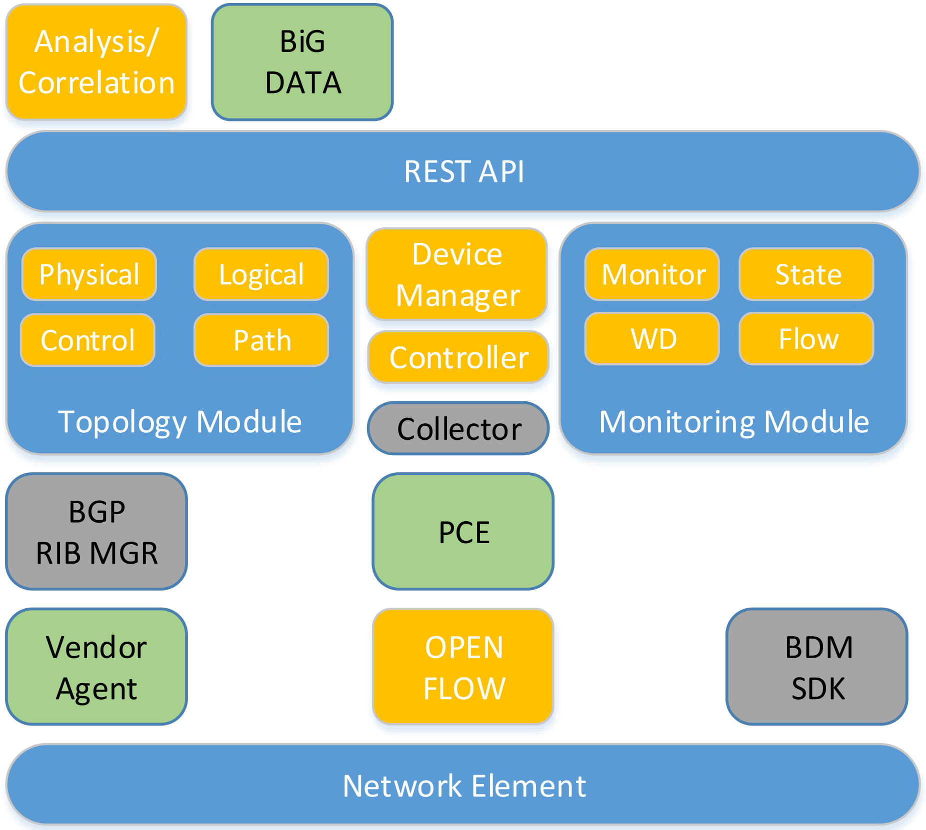
- Injecting ECMP Anycast prefixes
  - Already implemented (see references).
  - Used for software load-balancing in the network.
  - Uses a "minimal" BGP speaker to inject routes.
- Moving Traffic On/Off of Links/Devices
  - Graceful reload and automated maintenance.
  - Isolating network equipment experiencing grey failures.
- Changing ECMP traffic proportions
  - Unequal-cost load distribution in the network
  - 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 Foundations

- **Why BGP vs OpenFlow**
  - No new protocol.
  - No new silicon.
  - No new OS or SDK bits.
  - Still need a controller.

- **Have “literal” SDN, software generates graphs that define physical, logical, and control planes.**
  - Graphs define the ideal ground state, used for config generation.
  - Need the current state in real time.
  - Need to compute new desired state.
  - Need to inject desired forwarding state.

- **Programming forwarding via the RIB**
  - Topology discovery via BGP listener (link state discovery).
  - RIB manipulation via BGP speaker (injection of more preferred prefixes).

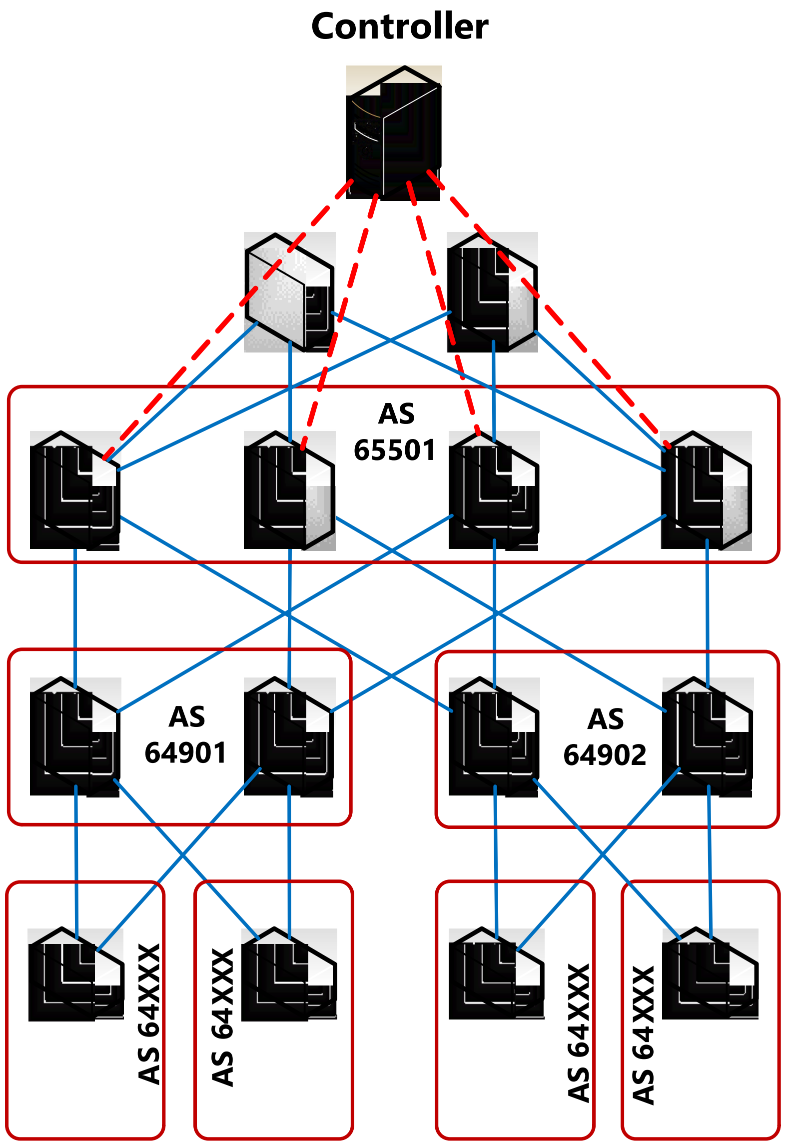
Network Setup

Device Configuration

- Templates to peer with the central controller (passive listening)
- Policy to **prefer** routes injected from controller
- Policy to announce only **certain** routes to the controller

Peering to the Controller

- Multi-hop peering with all devices.
- Key requirement: path resiliency
- CLOS has very rich path set, network partition is very unlikely.

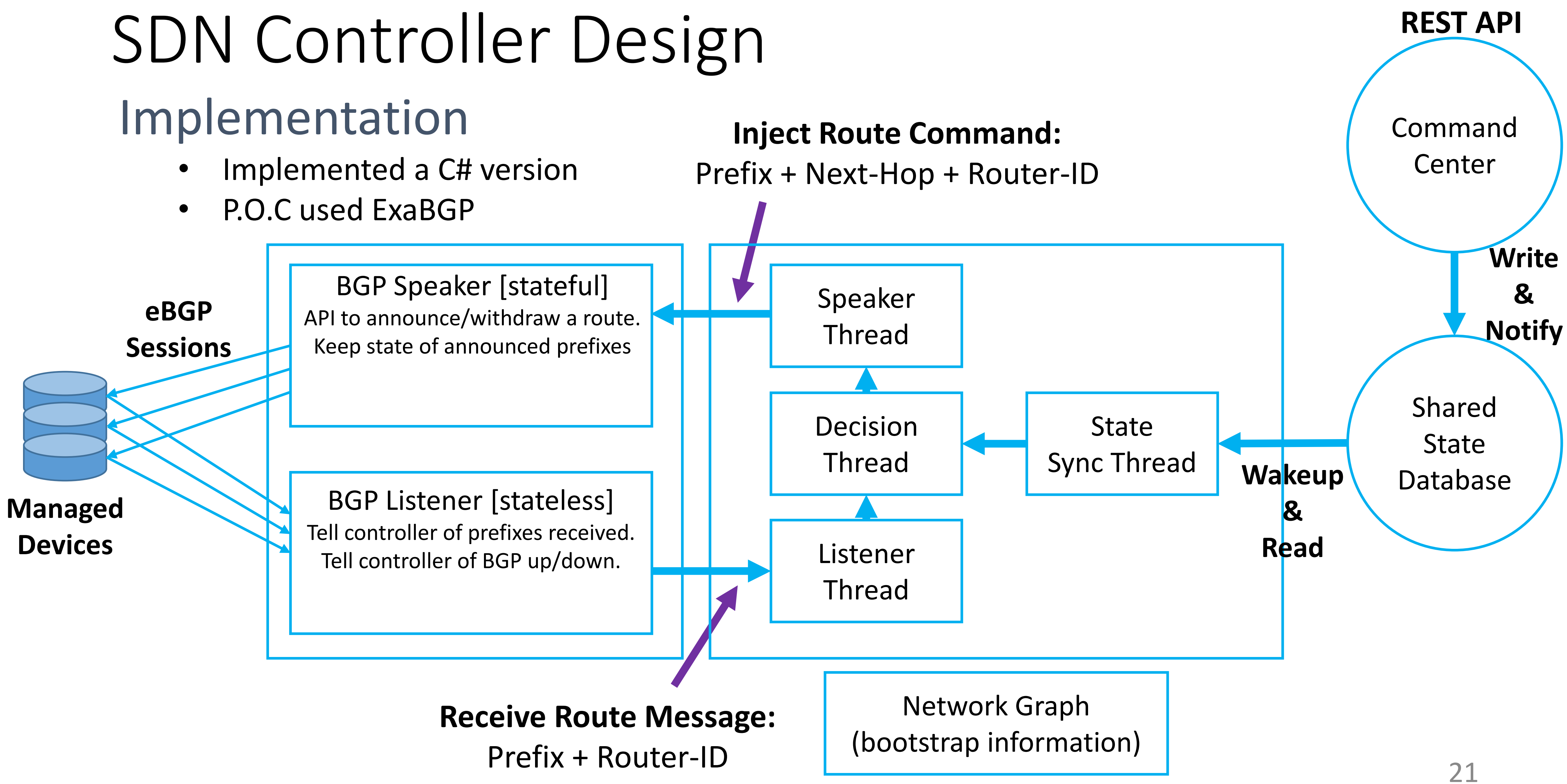


Only Partial Peering Set Displyaed

SDN Controller Design

Implementation

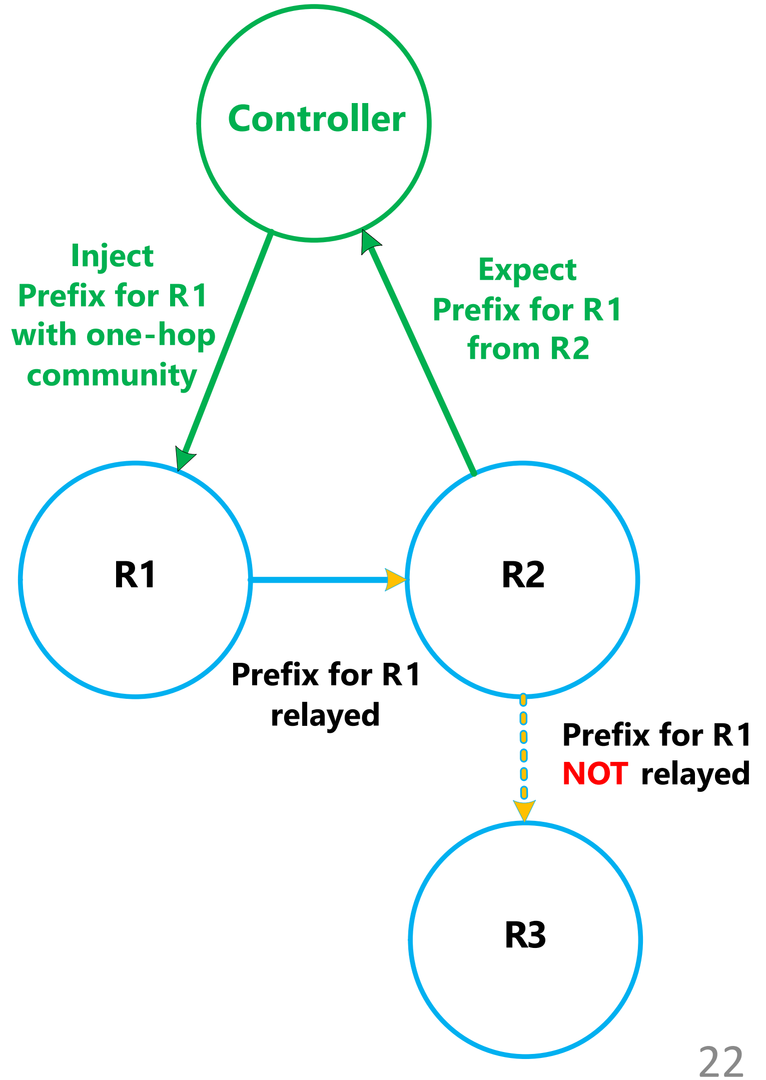
- Implemented a C# version
- P.O.C used ExaBGP



Building Network Link State

- Goal: Build Link-State of Live Network
  - Use a special form of "**control plane ping**"
  - Rely on the fact that BGP session reflects "link health"
  - Assumes single BGP session b/w two devices
- How it works
  - Create a /32 prefix for every device, e.g. R1.
  - Inject prefix into device R1.
  - Expect to hear this prefix via all devices R2…Rn directly connected to R1.
  - If heard, declare link R1 --- R2 as up.

Community tagging + policy ensures prefix only leaks "one hop" from point of injection, but is reflected to the controller.



Overriding Routing Decisions

- Populating Forwarding Databases

The controller knows of all server subnets and devices.

The controller runs SPF and

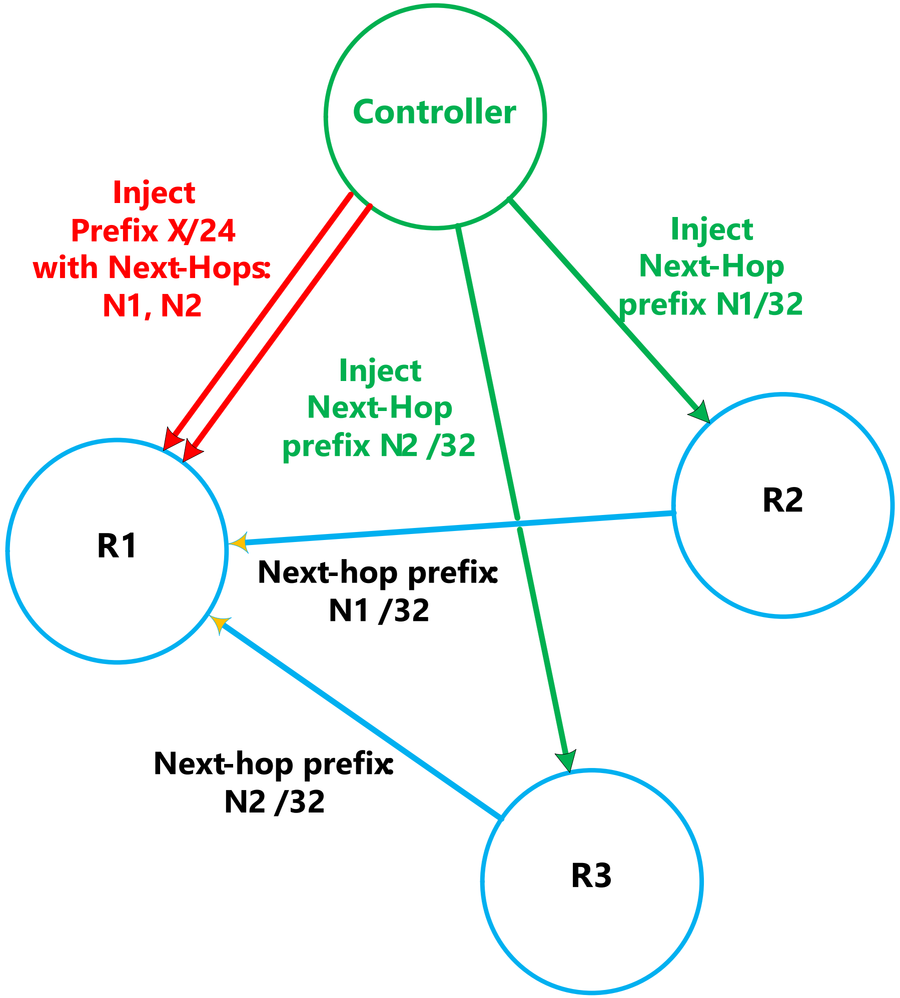
  - Computes next hops for every server subnet at every device
  - Checks if this is different from "static network graph" decisions
  - Only pushes the "deltas"
  - These prefixes are pushed with "**third party**" next-hops (next slide) and a better metric.

- Key observations
  - Controller has full view of the topology
  - Zero delta if no difference from "default" routing behavior
  - Controller may declare a link down to re-route traffic…
  - Seamless fallback to default BGP routing in the case of controller failure.

Overriding Routing Decisions cont.

- What about next-hops?
  - Injected routes have third-party next-hop
  - Those need to be resolved via BGP
  - **Next-hops have to be injected as well!**
  - A next-hop /32 is created for every device
  - Same "**one hop**" BGP community used

- Injecting ECMP Routes
  - By default only one path allowed per BGP session
  - Need either Add-Path or multiple peering sessions
  - Worst case: # sessions = ECMP fan-out
  - Add-Path **Receive-Only** would help!



Overriding Routing Decisions cont.

- Simple REST to manipulate network state “overrides”
- Supported calls:
  - Logically shutdown/un-shutdown a link
  - Logically shutdown/un-shutdown a device
  - Announce a prefix with next-hop set via a device
  - Read current state of the down links/devices

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

- This requires a state database
  - State is **persistent** across controller reboots
  - State is **shared** across multiple controllers

Ordered FIB Programming

- Distributed programming poses issues

  If updating BGP RIB's on devices in random order…

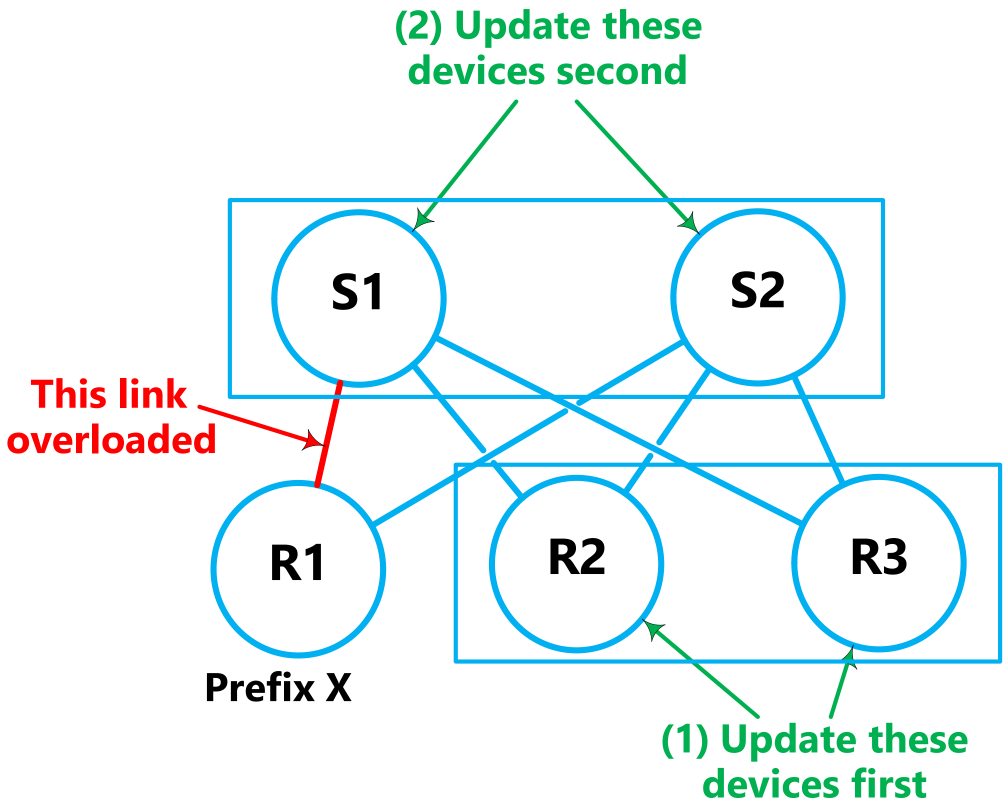
  …RIB/FIB tables could go out of sync

  Micro-loops problem!

- Central controller helps

  For every link state change

  - Find prefixes affected
  - Build reverse-SPF for each prefix
  - Update from the leafs to the root
  - Controller sequences the updates
  - Implode, not explode.



Note: assumes FIB programming is fast!

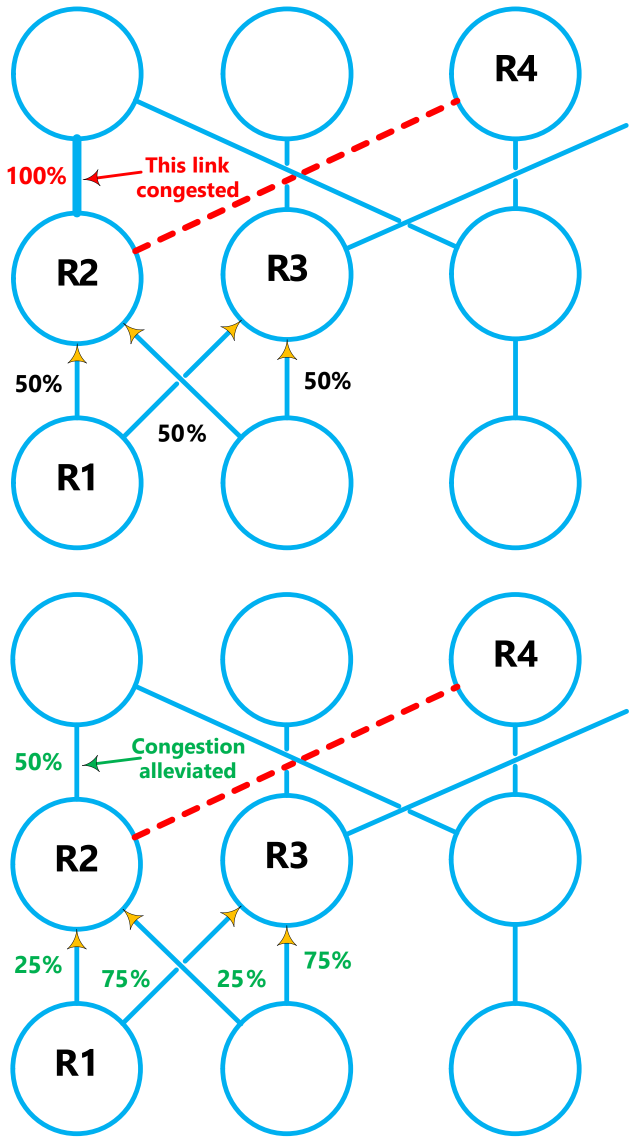
Roadmap

Traffic Engineering

- ECMP Routing is oblivious

  Failures may cause traffic imbalances
  This includes:
  - Physical failures
  - Logical link/device overloading
- Central controller helps
  - Compute new traffic distribution
  - Program weighted ECMP
  - Signal using **BGP Link Bandwidth**
  - Not implemented by most vendors ☹

Link b/w R2 and R4 goes down but R1 does not know that



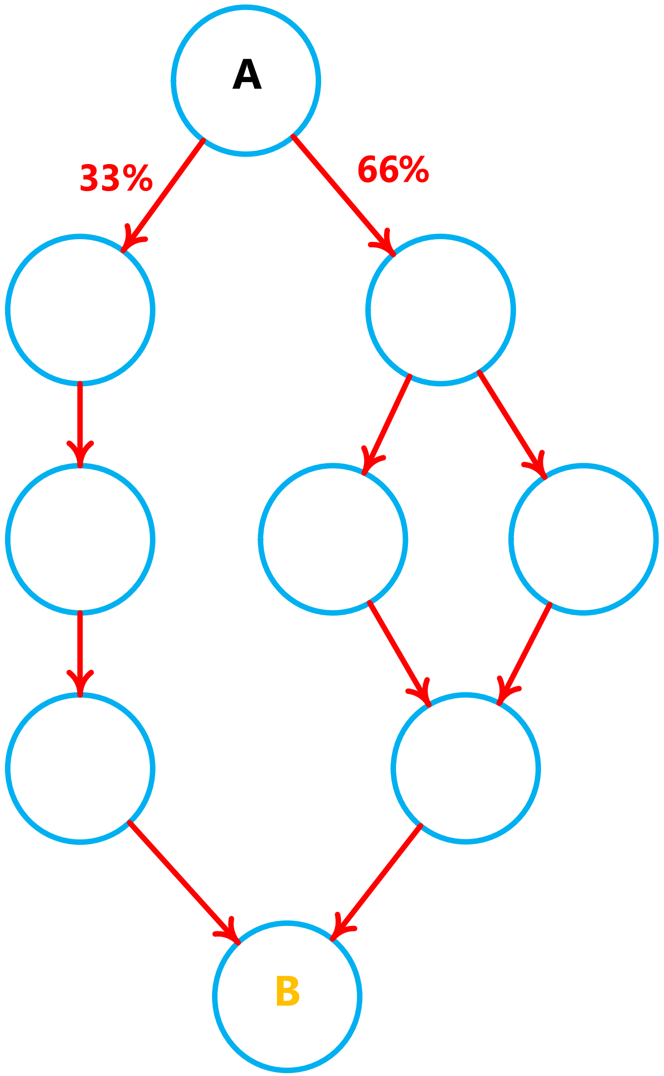
Controller installs path with different ECMP weights

Traffic Engineering (cont.)

- Implementation
  Requires knowing
  - traffic matrix (TM)
  - Network topology and capacities

  Solves Linear Programming problem
  Computes ECMP weights
  - For every prefix
  - At every hop

  Optimal for a given TM
- This has implications
  - Link state change causes reprogramming
  - More state pushed down to the network



Ask to the vendors!

- Weighted ECMP in DC switches
  - Most common HW platforms **can do it** (e.g. Broadcom)
  - Signaling via BGP does not look complicated either
  - *Note: Has implications on hardware resource usage*

- Consistent Hashing
  - Goes well with weighted ECMP
  - Well defined in RFC 2992

- Add-Path: Receive Only
  - Not a standard (sigh)
  - We really like receive-only functionality

Conclusions

What we learned

- BGP SDN is Practical
  - Does not require new firmware, silicon, or API's.
  - Some BGP extensions are nice to have.
- BGP SDN is Simple and Stable
  - BGP Code is tends to be mature .
  - Easy to roll-back to default BGP routing.
- BGP SDN is Efficient
  - Solves our current problems and allows solving more.
- TE is possible without MPLS

Questions?

Contacts:

Edet Nkposong - edetn@microsoft.com
Tim LaBerge - Tim.LaBerge@microsoft.com
Naoki Kitajima - 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/

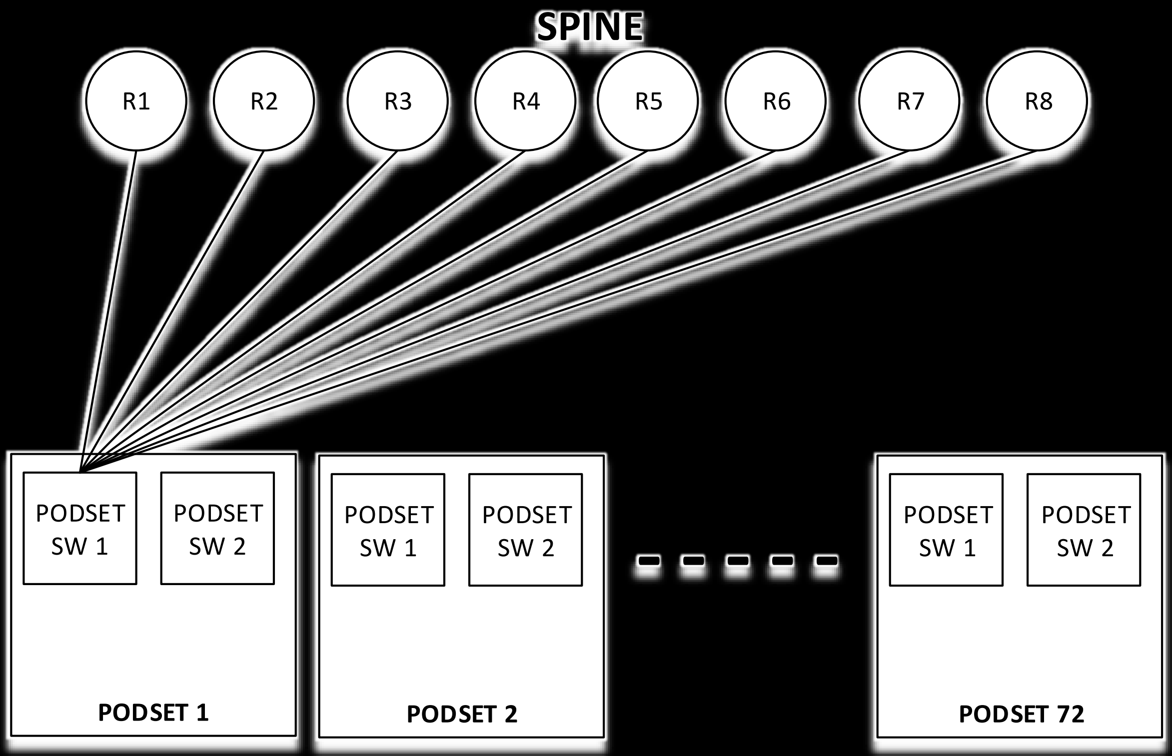
- BGP is the Better IGP http://www.nanog.org/meetings/nanog55/presentations/Monday/Lapukhov.pdf
- BGP for SDN in the Data Center
  http://www.nanog.org/sites/default/files/wed.general.brainslug.lapukhov.20.pdf
- Autopilot: Automatic Data Center Management http://research.microsoft.com/pubs/64604/osr2007.pdf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

Backup Slides

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

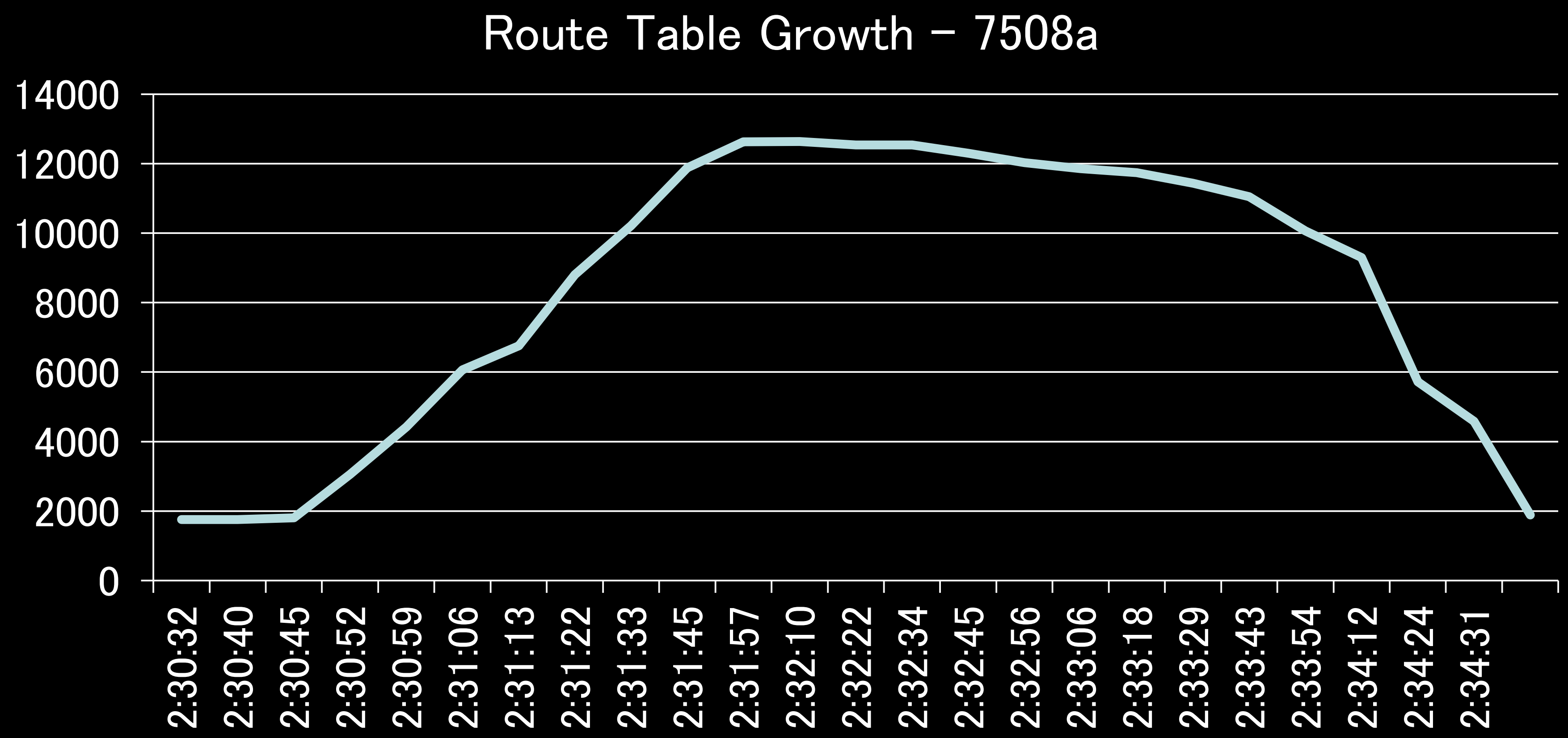
- Route table calculations
  - Expected OSPF state
    - 144 x 2 x 4 = 1152 links for infrastructure
    - 144 x 6 = 864 server routes (although these will be 4-way since we have brought everything into 4 spines (instead of 8)
    - Some loopback addresses and routes from the real podset switches
  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

- What happens when we shake things up ?

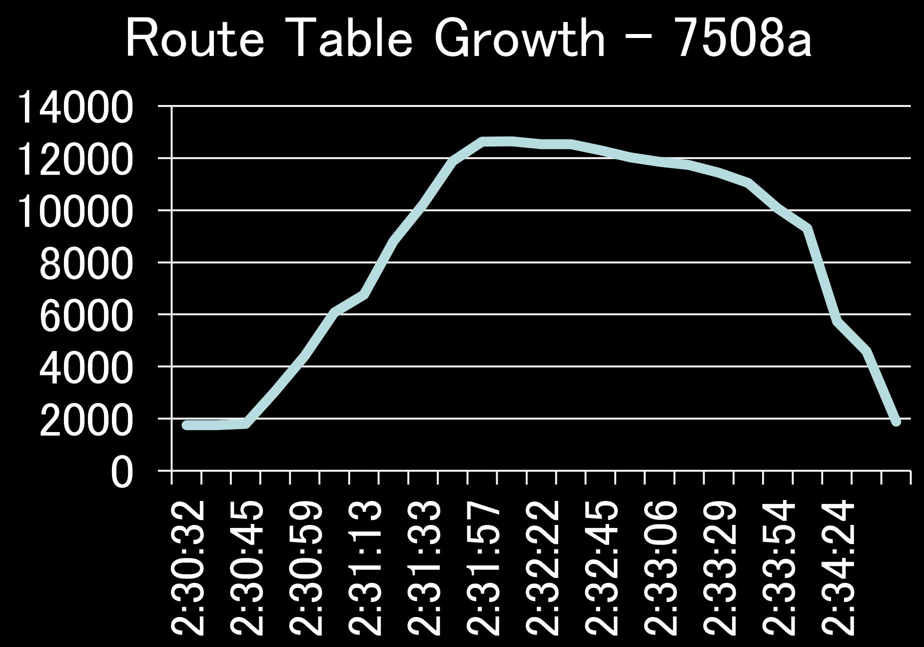
OSPF Surge Test

- 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

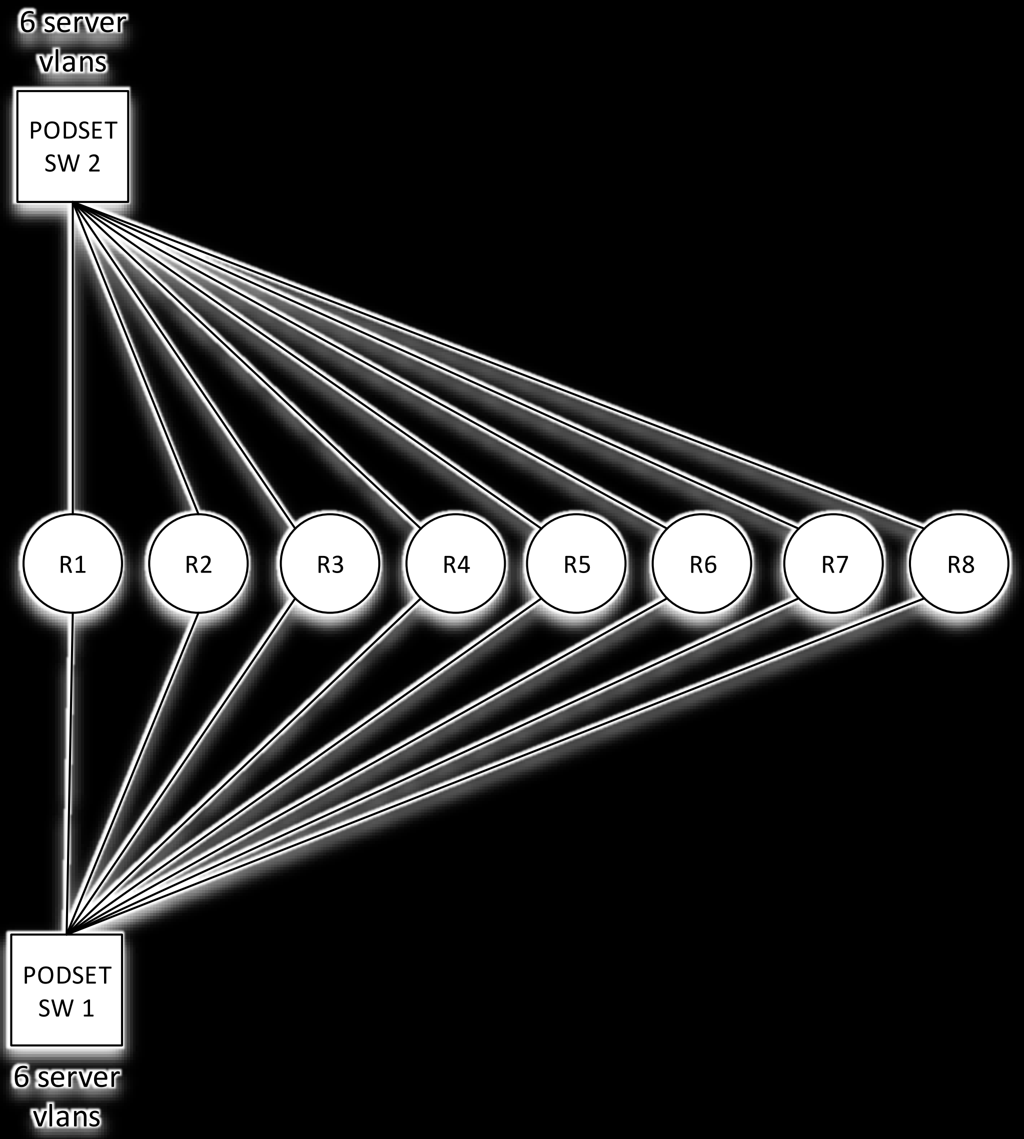
OSPF Surge Test

- 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 podset switches
  - Unnecessary 16-way routes

- For every disabled podset switch-spine link, the spine will install a 16-way route through other podset switches

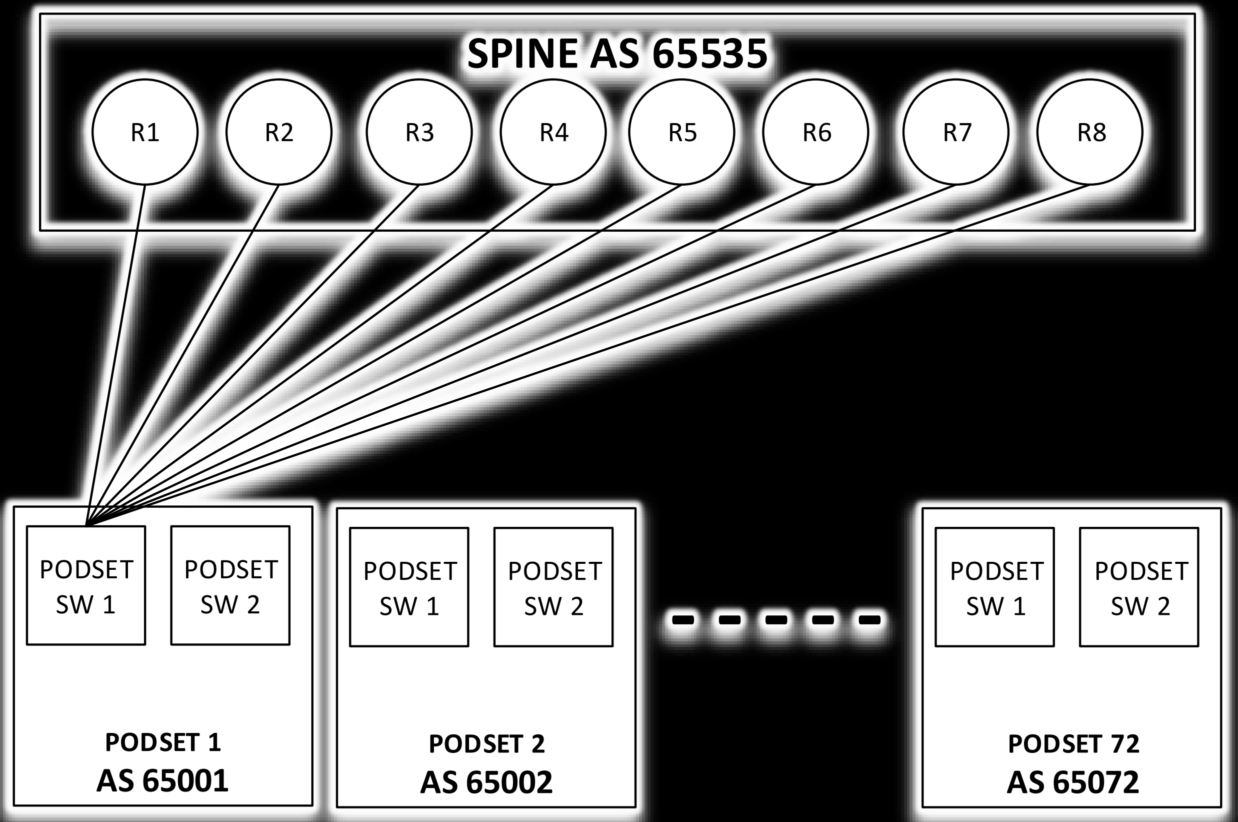
- The surge was enough to fill the FIB (same timeline as graph on slide 12)




```
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.
```

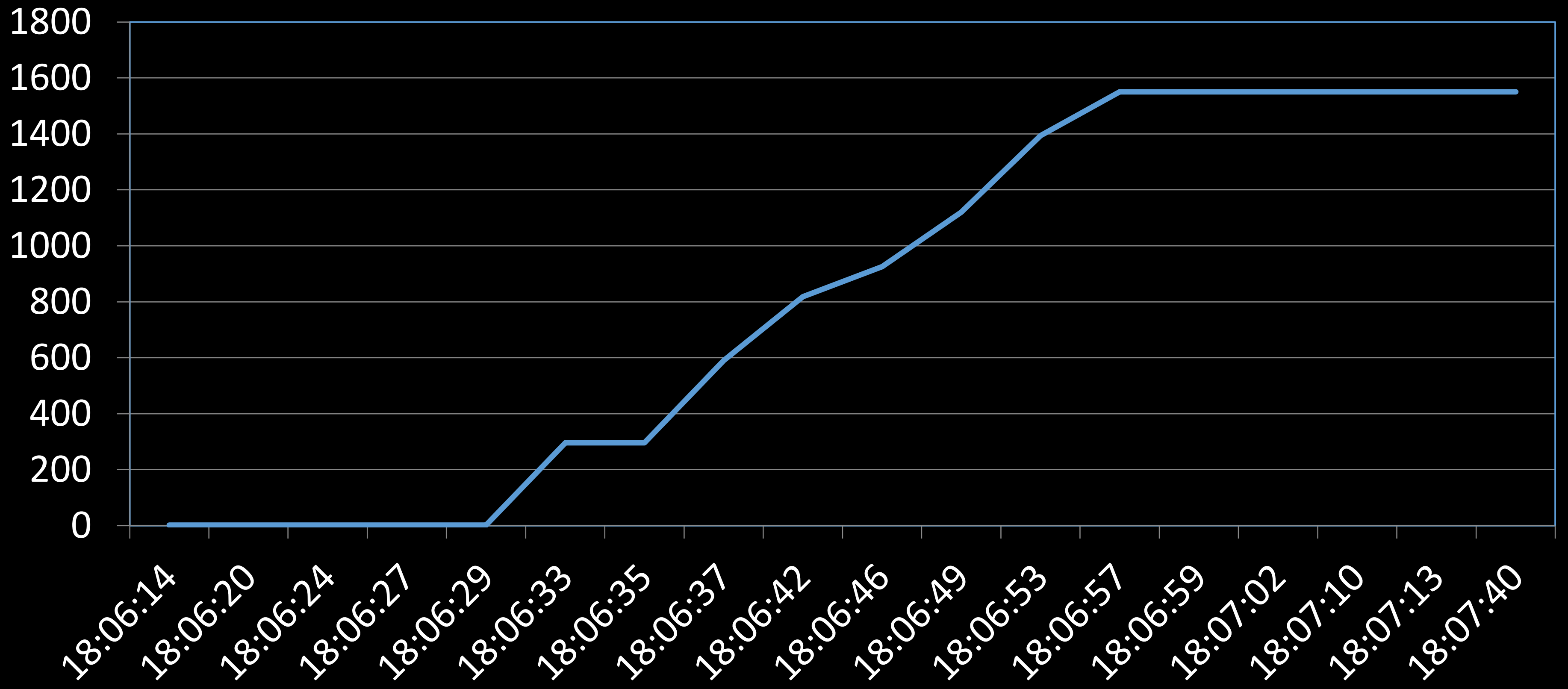
BGP Surge Test

- 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 podset they came from (AS #) – beautiful !