
# Data Center Networking: Rip van Winkle Edition

Dinesh Dutt, Chief Scientist

July 11, 2016

# Why Should You Care About This Talk ?

Data centers are experiencing a shift in operations

Traditional silos between network, compute and storage are breaking down

How are changes in networking fueling this transition ?

If networking is no longer the purview of CCIE/JNIE etc., isn't it time to get reacquainted ?

# Je m'appelle Dinesh Dutt (@ddcumulus)

Chief Scientist at Cumulus

Ex-Cisco Fellow

A key architect of many of Cisco's products from Cat6k to MDS to Nexus family of switches, including many Cisco initiatives

Co-author of VxLAN and TRILL drafts

Filed for over 40 patents

# Agenda

Session 1: The Winter of our Discontent

Session 2: Software, including networks

Session 3: Transition effects: today & tomorrow

# Disclaimers

The focus is on data centers

This is a technology/industry talk

Opinions are my own, and they will be highlighted vs well established facts

# Prologue: The Applications Did It

# Evolution of Applications & Their Networks

## Mainframe Era

- Monolithic apps
- Skinny, custom interconnects
- Proproprietary protocols

## Client-Server Era

- Simple distributed apps
- Mostly N-S traffic
- Standardized interconnects (Eth) & protocols (TCP/IP)
- Very oversubscribed network

## Cloud Era

- Complex distributed apps
- Lots of E-W traffic
- High scale apps
  - Lots of endpoints
- High, affordable capacity networks

# Three High Level Communication Problems For Distributed Apps

How the various pieces communicate with each other ?

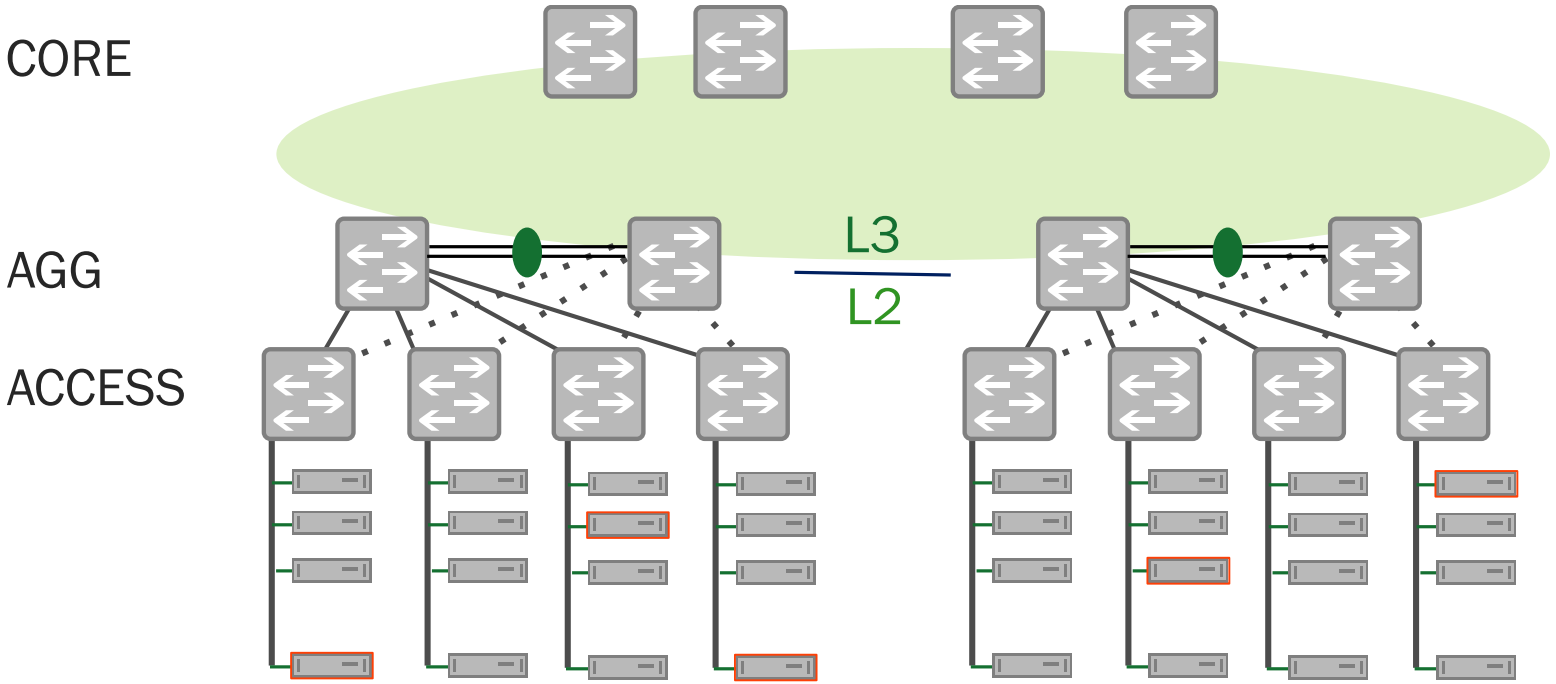
Cluster discovery & membership

Cluster member health check

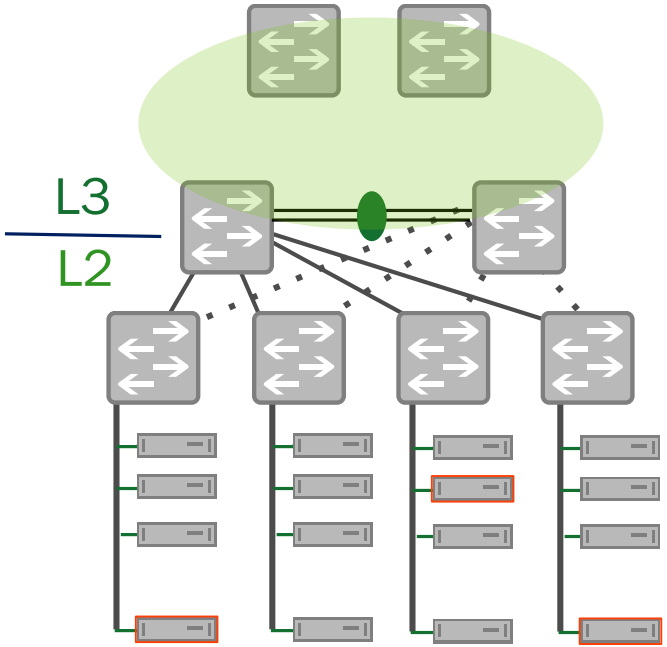
# Session 1:

# The Winter Of Our Discontent

# Understanding the Status Quo



# Characteristics Of The Network Design



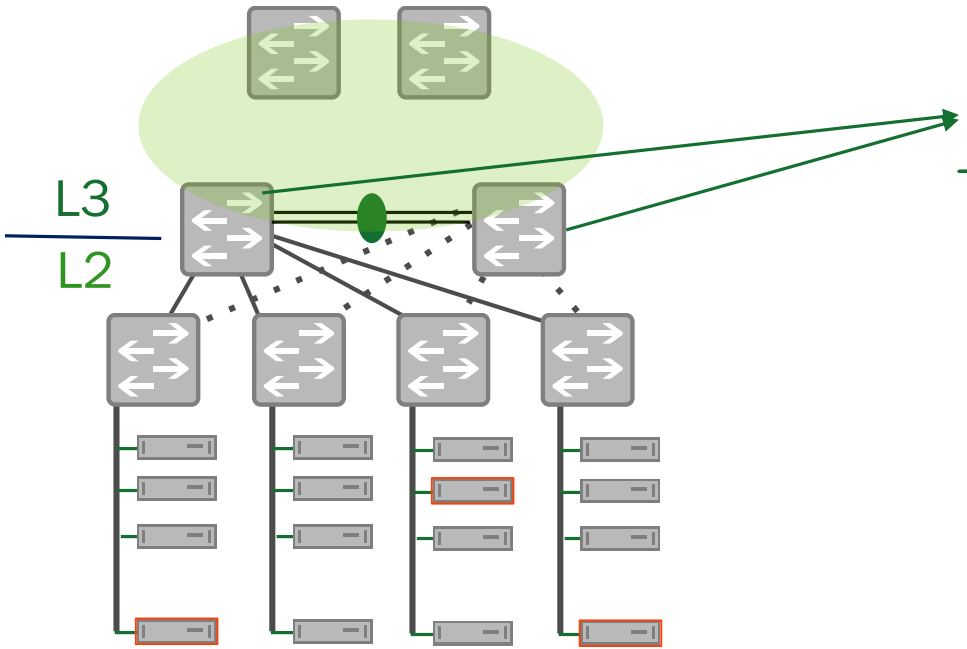

## Suited to N-S traffic pattern

## Inflexible, Unscalable design

## Brittle L2 protocol alphabet soup

- Too many protocols
- Many vendor-specific

# Characteristics Of The Network Design (2)




- Expensive Godboxes
- Coarse grained failure domain
  - Adds to complexity due to desire to avoid failure
- Complex Failure modes

# Application Design Before The Advent of Modern DC Applications

## Applications reliability rested on networks

- Led to more complex software on network boxes

## Applications relied on L2 characteristics

- Broadcast for cluster discovery
- Multicast for heartbeat
- IP subnet assumptions

## Expensive boxes meant skinny links

- Coupled with inefficient L2 pattern meant more complex solutions such as QoS

# Manageability

## Network boxes were managed manually

## SNMP was the predominant monitoring protocol for networks

- Its inefficiency meant network monitoring happened at a far coarser timescale than applications
- This implied network problems were harder to diagnose

## Upgrades were entire images, not specific apps

- Meant adding features or bug fixes required expensive requalification

# What Changed ?

Server Virtualization

Applications

Scale

# Effect Of Server Virtualization on Network

## Changed Scale

- Number of endpoints

## Changed agility of application deployment

- VMs could be spun up and down far faster than servers could be added or removed

## Made it harder for the network to track VM action

- Virtual link up/down were no longer easily visible to network devices

# Changing Applications

Search

Big Data

Clouds

Other Web 2.0 Applications

# Characteristics of Application Change on Network

## Traffic became predominantly E-W, not N-S

## Applications took on resiliency instead of relying on network

- Network resilience was communication resilience, not box resilience

## Applications used modern service discovery

- DNS for discovering nodes
- Cluster protocols no longer relied on broadcast/multicast (eg. Gossip)

## Applications became far more scalable than before

# Implications of Scale

On Complexity

On Manageability

On Reliability

On Agility

On Flexibility

# Impedance Mismatch Between Network & Application

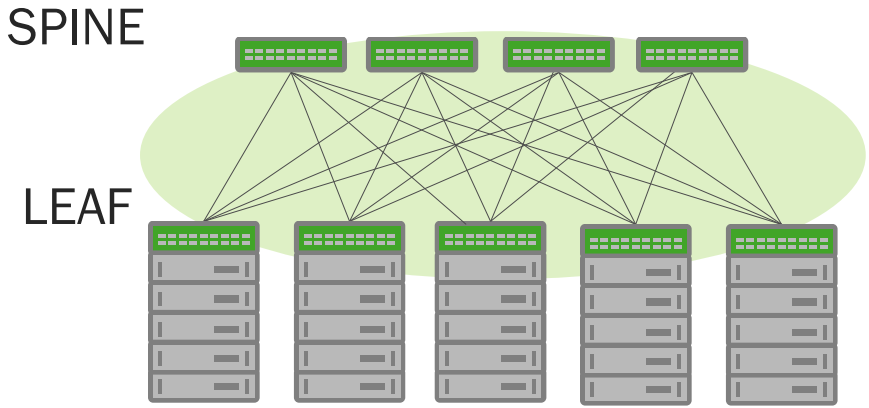

Network's Function is to serve the application needs

Existing Network design is a bad fit for the modern DC application

Image credit: http://bestandworstever.blogspot.com/2012/07/best-lane-ending-sign-ever.html

# Morphology of the New DC Network

# A New Blueprint For Networks: Clos Network

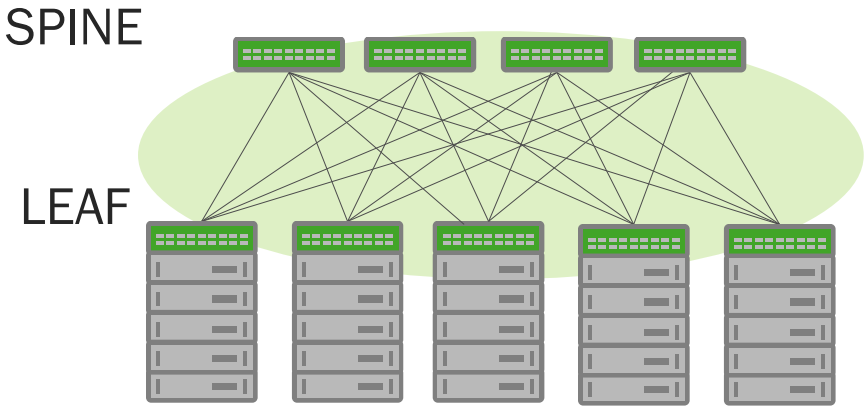



Invented by Charles Clos in 1953

How to build ever larger telephony networks without building ever larger telephony switches

http://en.wikipedia.org/wiki/Clos_network

# Characteristics Of Clos Network




Well matched for E-W traffic pattern

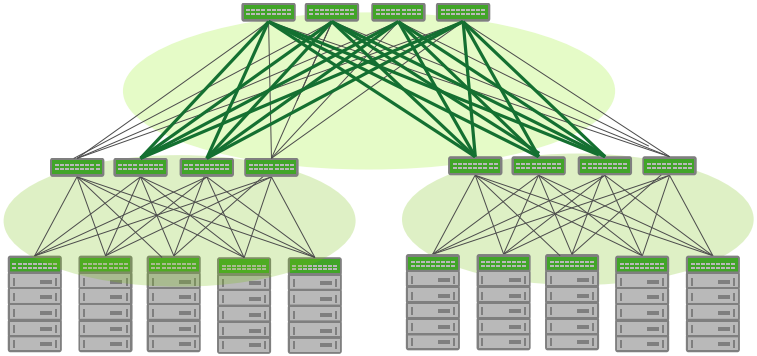
Scalable network topology

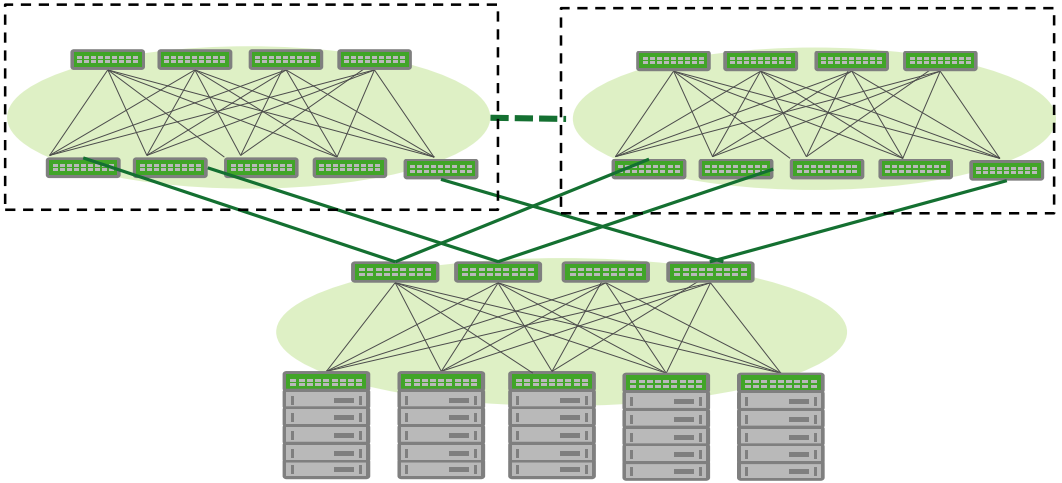
Fine grained failure domain

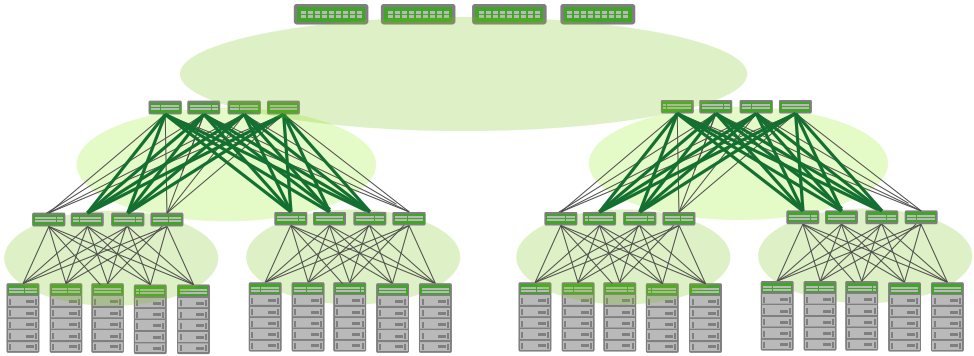
Predictable latency

Ability to build higher capacity networks

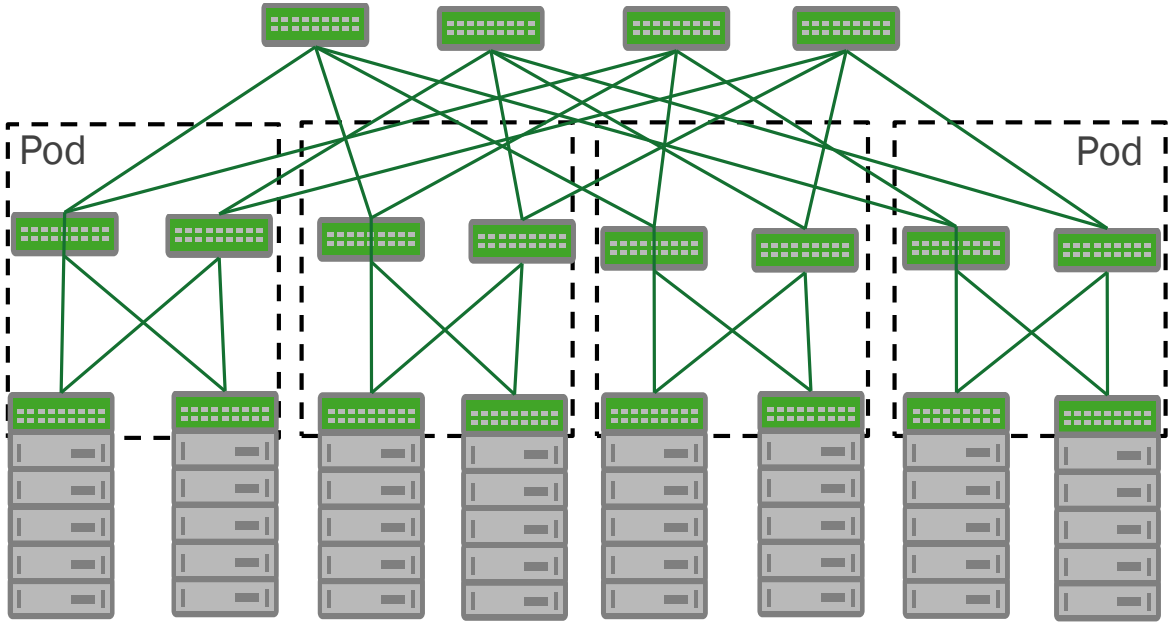
# Paganini Variations: Scalable Network Design



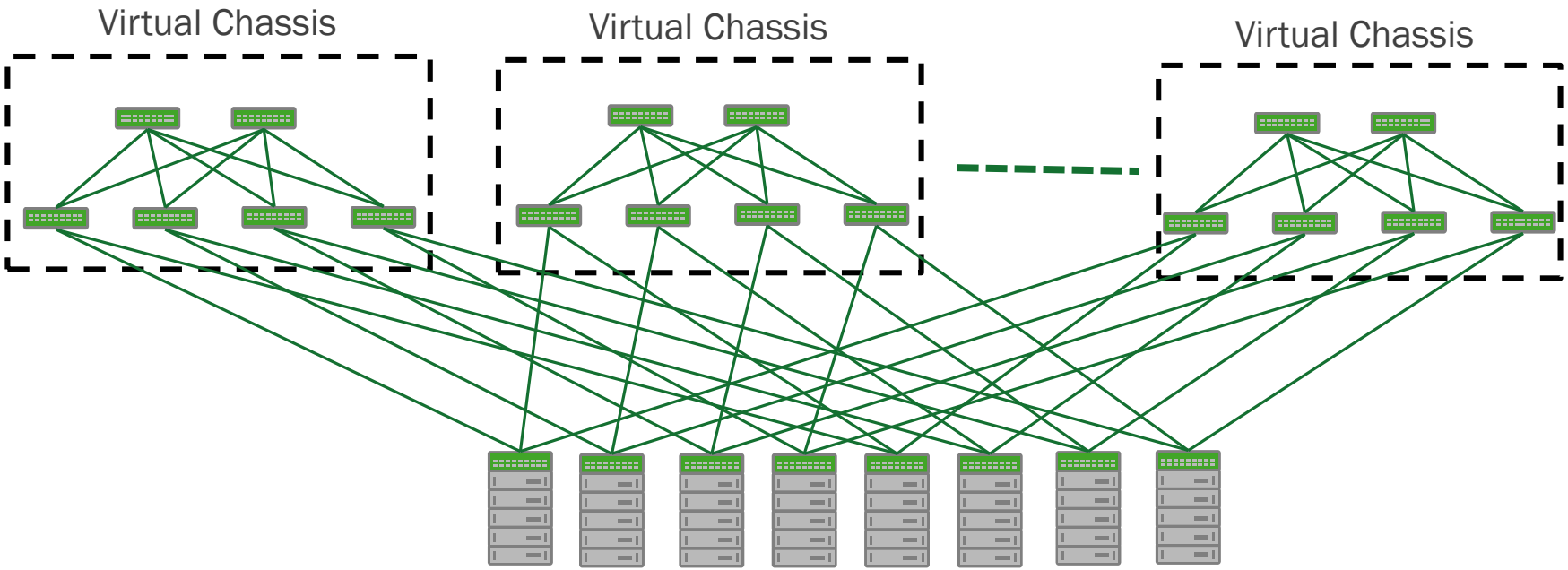




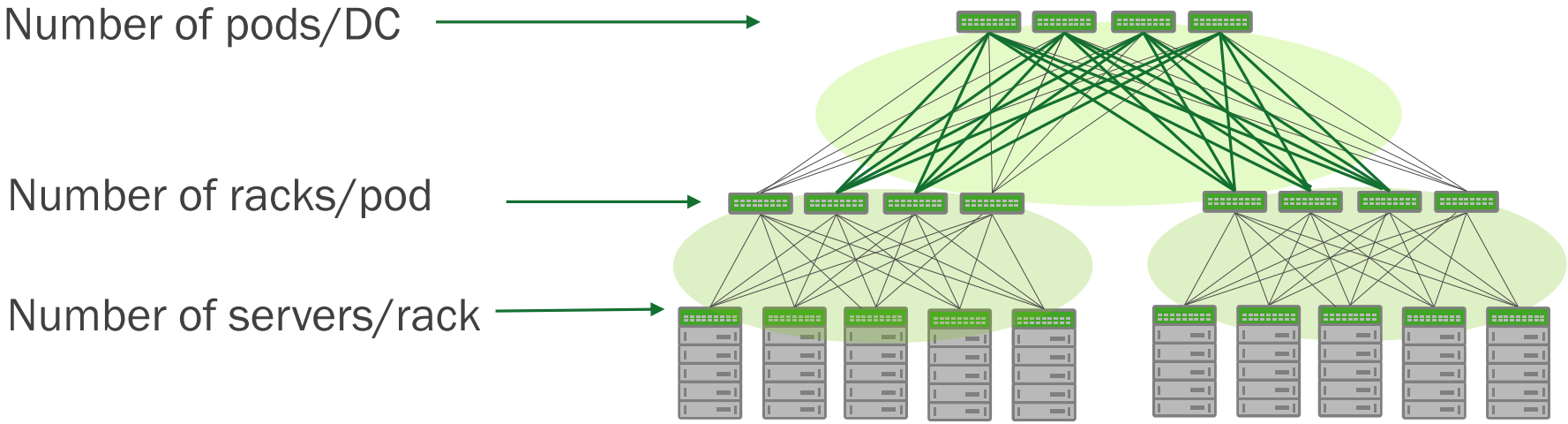
# Pod Based Design



# Virtual Chassis Based Design



# Port Math From A Different Perspective

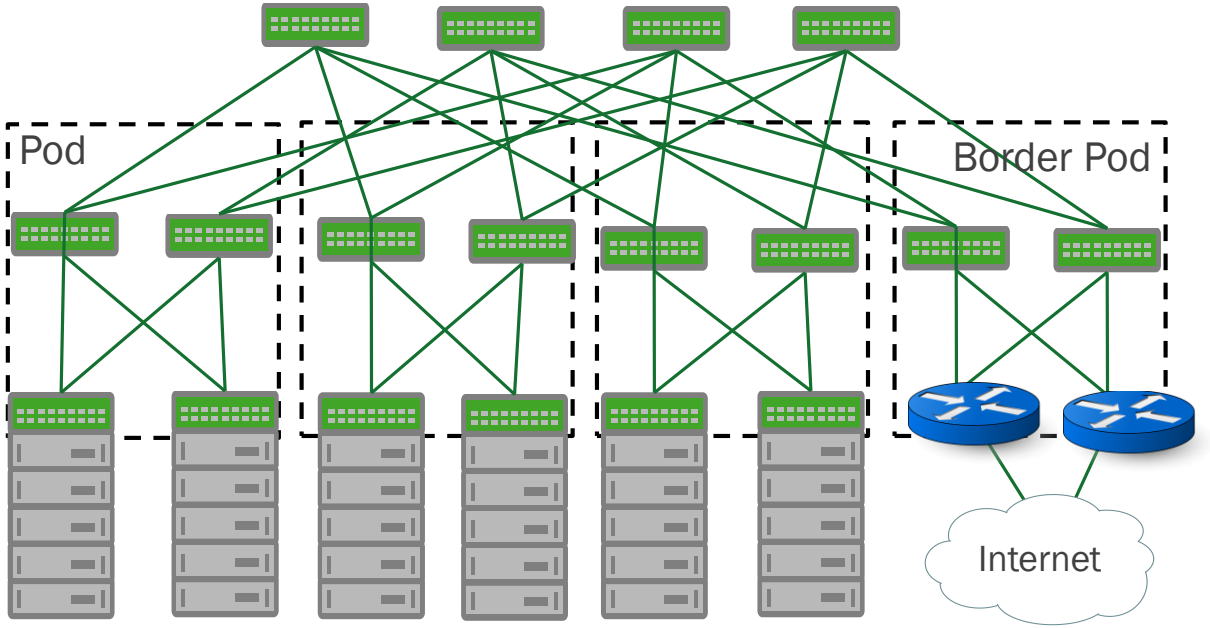



**Number of spines and ISL link speed is a function of failure characteristics, cost and cabling simplification**

# Calculating Number of Servers: Some Concrete Numbers

| | 2-tier | 3-tier |
|---|---|---|
| 40 servers/rack: 10GE server interconnect | | |
| Trident2 (40GE switch interconnects)<br>Oversubscription: 2.5 (with 4 spines) | 1280 (40*32) | 20K(40*32*16) |
| Trident2 (108 port 10GE switch interconnect)<br>Oversubscription: Controllable (2.5 with 16 spines or none with 40 spines) | 4320 (40*108) | 227K (40*108*54) |

# Connectivity To Outside World

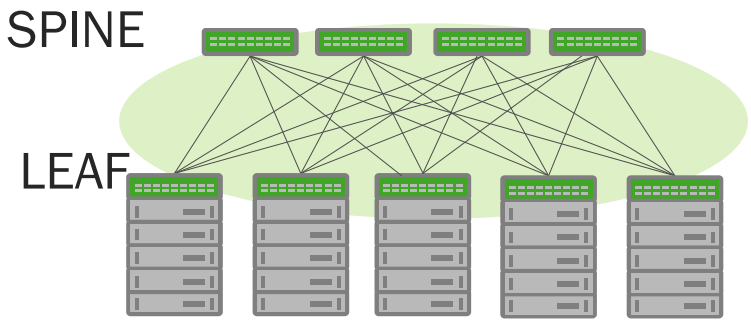


# Coda for the Godboxes

- Multiple components trying to appear as one
- Many internal, hidden, troublesome protocols & services
- They are overcomplicated, create lock-in, and aren't re-usable

- Identical components, individually deployed => simplified inventory
- Open, extensible, automatable, monitor-able protocols & services
- More to manage, but each is simple, replaceable and re-usable

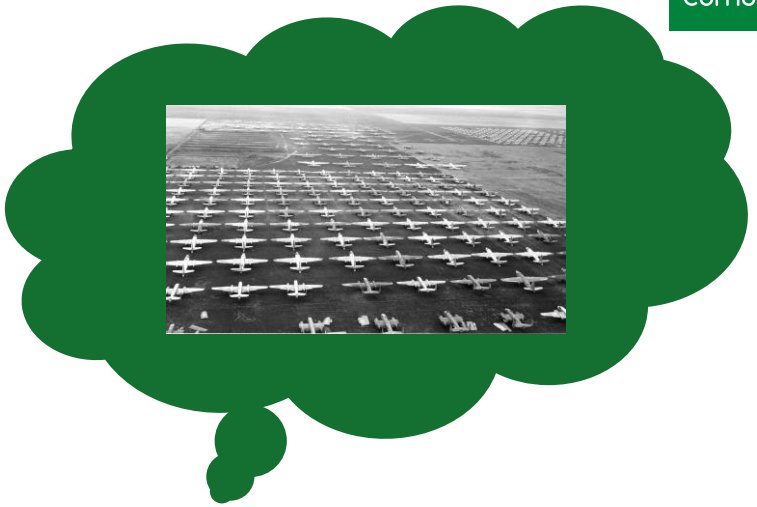
# Rise of Merchant Silicon




+

BROADCOM®

=



amazon web services™

...

Photo courtesy: https://www.flickr.com/photos/34076827@N00/

# Rise of ODM

Traditional network vendors contracted ODMs to build boxes based on merchant silicon for these webscale companies

Companies like Quanta, DNI, Edgecore were building boxes for these large network vendors

- Sold after huge markup by network vendors

Webscale companies contract to buy boxes directly from these ODMs

Arista's whole business rests on using merchant silicon

# Traditional Networking With Merchant Silicon

cumulus

OEM
(Includes OS)

ARISTA

CISCO™

Hardware
Manufacturer

DNI
A Delta Group Company

SUPERMICRO

Celestica™

Quanta

Edgecore

Silicon

BROADCOM

# But The Challenge Remains…

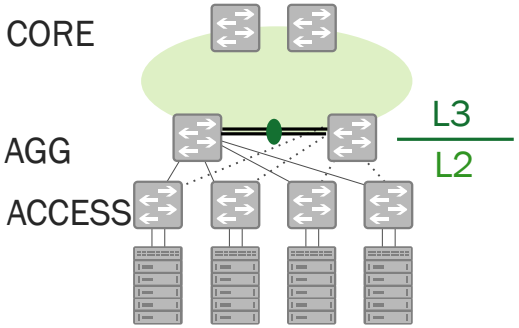
## The rest of the industry now wants to follow these webscale companies

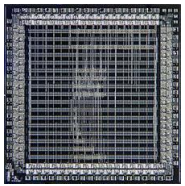
- The rise of the private cloud
- Adoption of applications such as Hadoop into traditional enterprises

## ODMs had surplus boxes, but had no channel to sell them in, or OS to sell it with

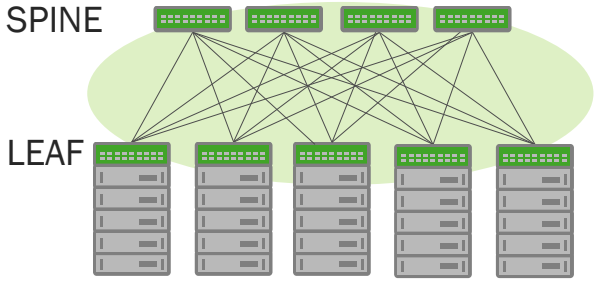
- Traditional network vendors retain their power

# Summing up Session 1



Custom ASIC




Merchant Silicon

BROADCOM.

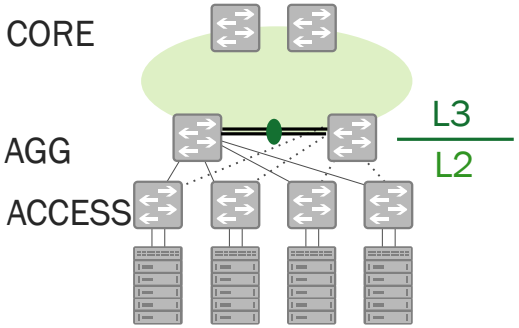
intel

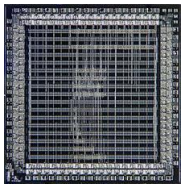
Mellanox TECHNOLOGIES

# Session 2:

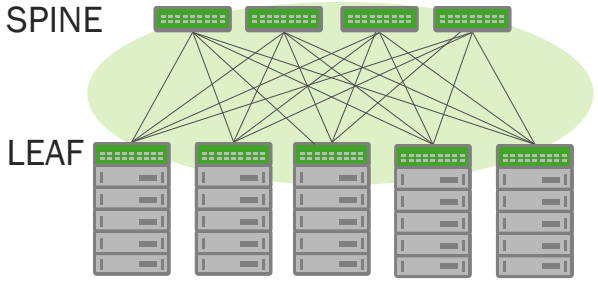
# The Rise of The Modern DC Network

# What We Learned in Session 1



Custom ASIC




Merchant Silicon

BROADCOM.

intel

Mellanox TECHNOLOGIES

# But…

Long lag between operator demand and vendor supply

Boxes completely closed to innovation

Merchant silicon available, but no way to consume it except via boxes from traditional network vendors

Low powered CPU on ODM boxes

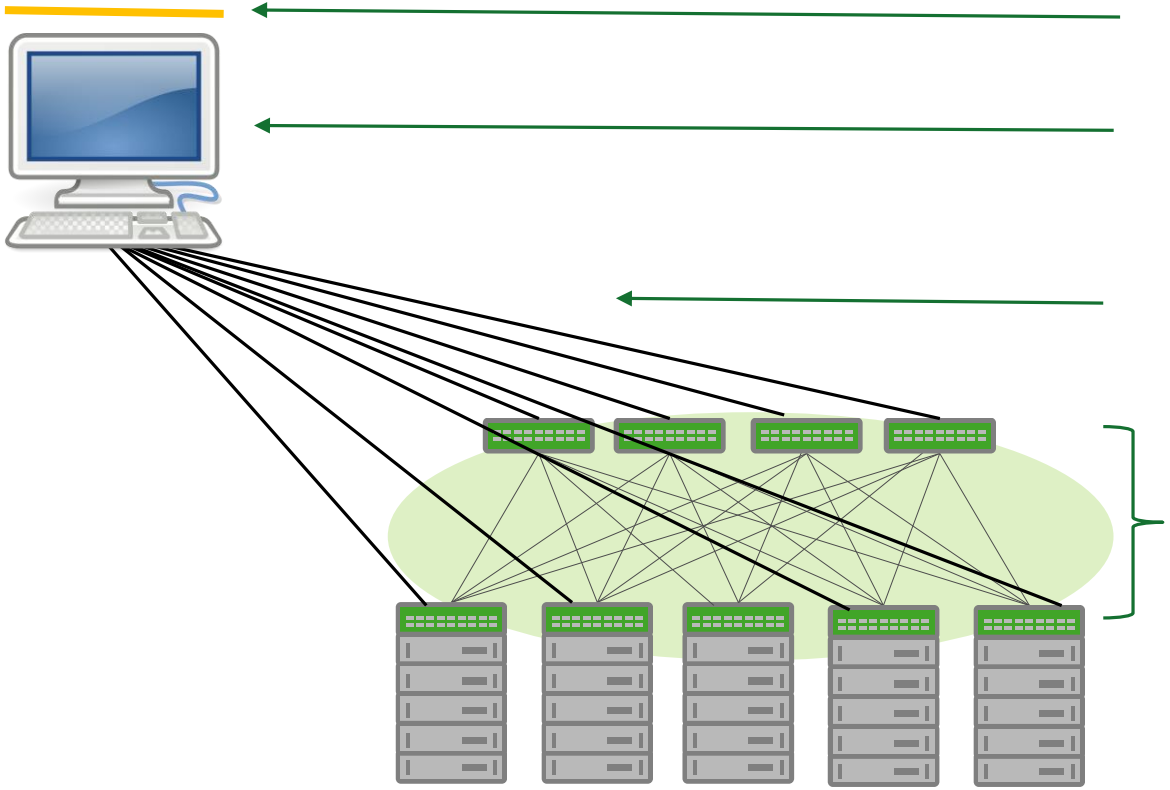
Hardware-driven features

# Clean slate approach to redo networking

## Main goals:

- Decouple feature velocity from hardware
- Make hardware flexible & simple
- Improve manageability
- Lower barrier to entry in network innovation
- Control Costs

# Round 1: Openflow



Northbound API for apps

Centralized Controller

Openflow, a new protocol to manage switches

Switches with relatively dumb software, mostly OF agent and HW programming

# Issues with OpenFlow

Overly flexibly model that prevents pragmatic ASIC implementations

Central controller seen as another form of vendor lockin

Central controller failure story not fleshed out

Perception that it's the wrong tool for the problem

# Customer Story To Illustrate Openflow's Shortcoming

# Round 2: Learning From The Server World

## Servers were:

- Disaggregated

  Hardware and software from different vendors
- Open source and proprietary options for software

## Network Devices were:

- Single vendor for hardware and software
- Proprietary only

# Disaggregation: Filling in The Missing Pieces

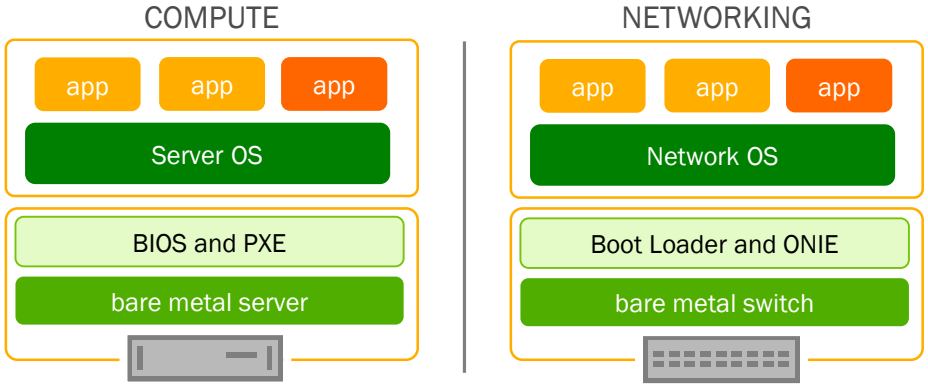
| | Compute | Network |
|---|---|---|
| Bootloader | BIOS | - |
| OS Installer | PXE | ONIE |
| OS | Linux, Windows, OS X, BSD | ?? |
| Applications | Open & Closed, Multiple vendors | ?? |

# Bare Metal Switch Provisioning

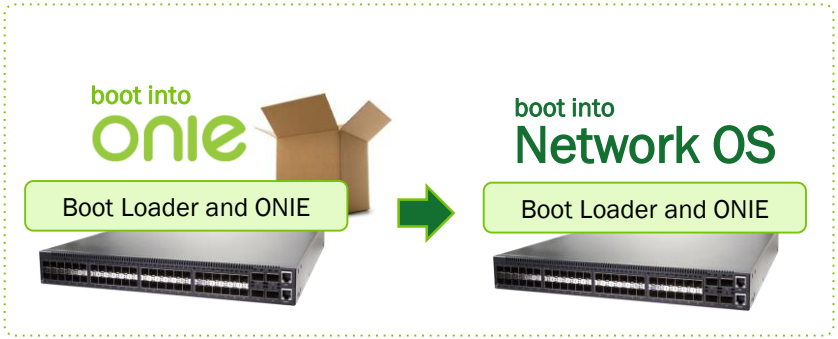
## Framework

- Hardware preloaded with ONIE
- Use ONIE to load OS onto bare metal switch
- Zero Touch Provisioning to load configuration

*Similar to installing a server OS using PXE*




*ONIE looks for and installs network OS image*



# OCP (Open Compute Project): http://opencompute.org/about/

Facebook started OCP in 2011 to standardize open compute hardware (and some software)

- With the backing of Intel, Goldman Sachs and Rackspace

Today just about every ODM and system integrator is a member

Has various groups for storage, compute, network, rack etc.

Cumulus Networks contributed ONIE to OCP to foster disaggregation

All Open Networking boxes today MUST support ONIE to be OCP certified

# Disaggregated Networking

Hardware Manufacturer


Silicon



# Network OS Evolution

## Monolithic OS

- No real OS
- while loop
- Proprietary routing and switching stack

**Examples:**

- IOS, CatOS

## Third party real time OS

- Embedded OS with process and memory management
- Proprietary routing and switching stack

**Example:**

- ION, iCOS/Fastpath

## Linux-based OS

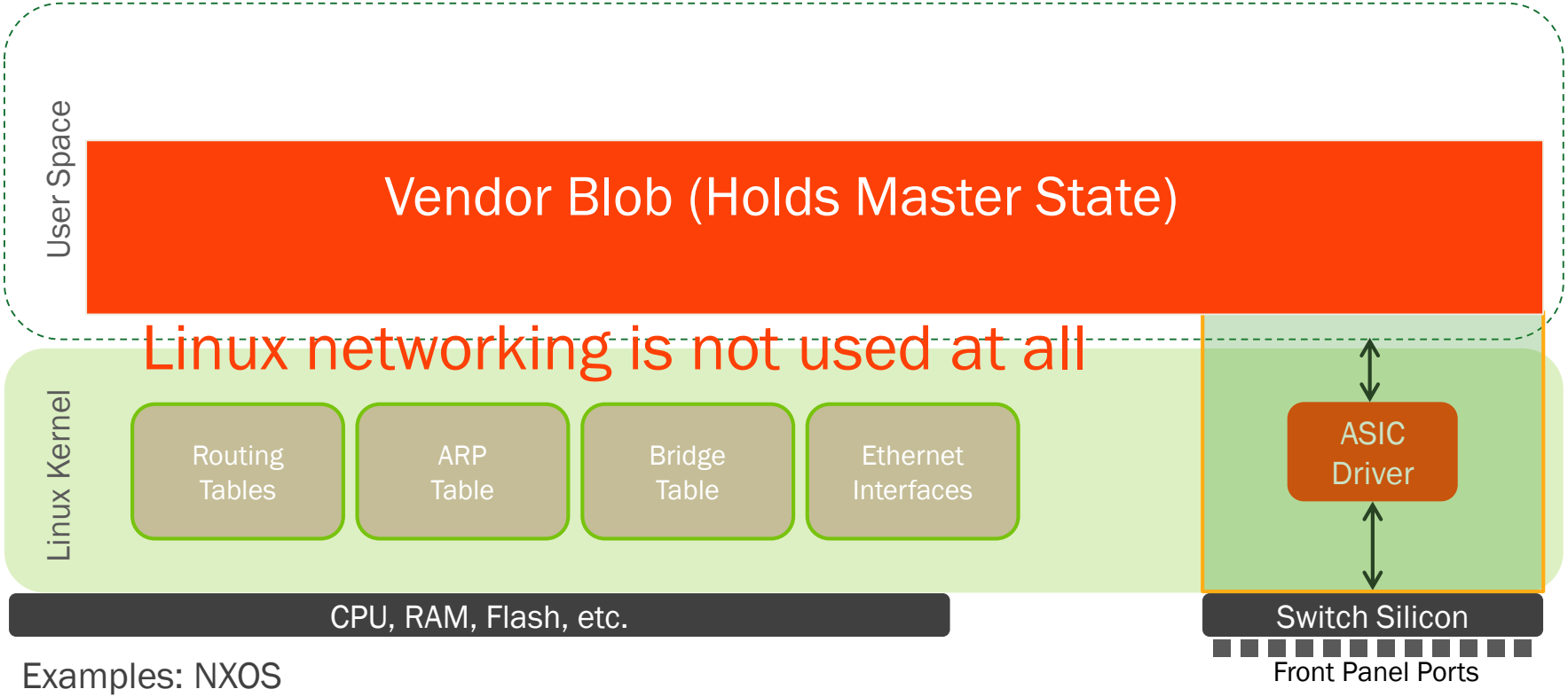
- Linux as embedded OS with process and memory management
- Proprietary routing and switching stack

**Examples:**

- NX-OS, EOS

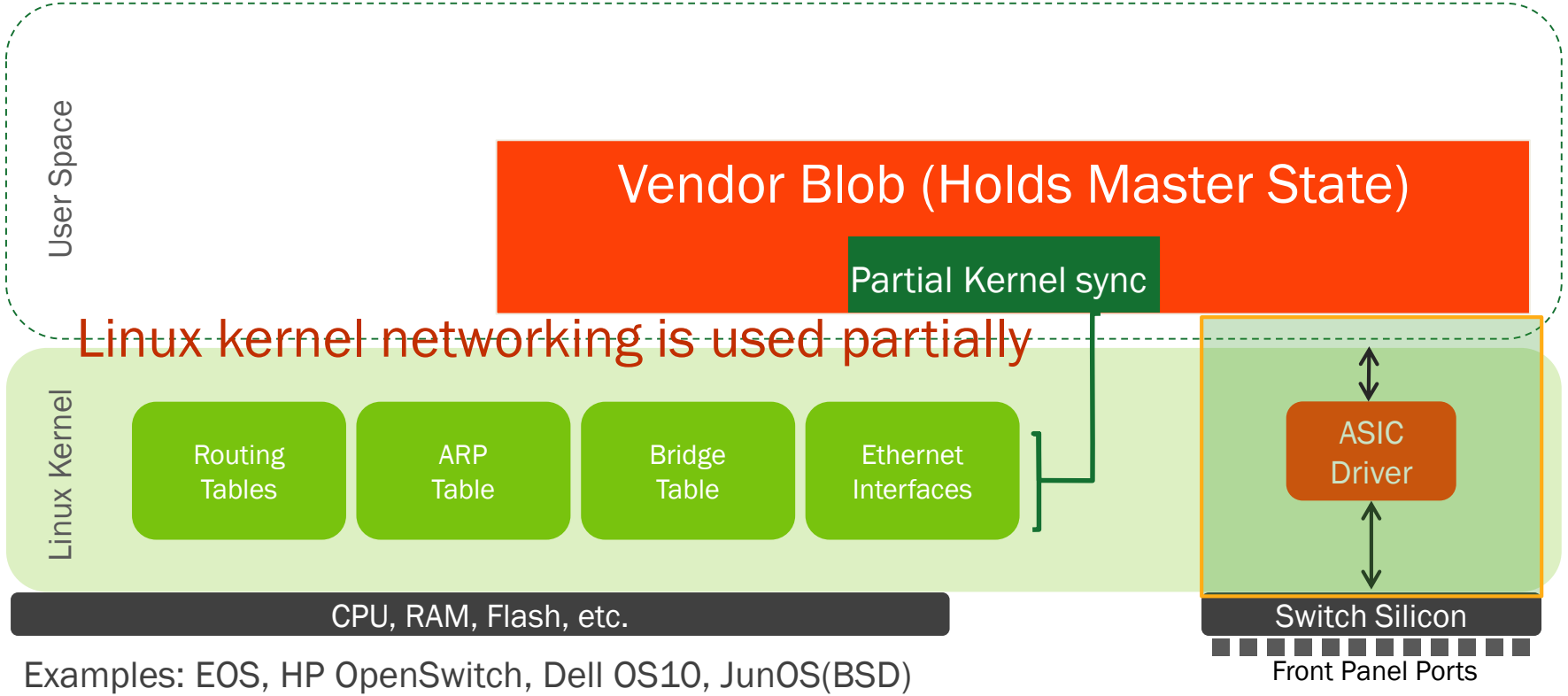
??

# Linux As a NOS: Version 1



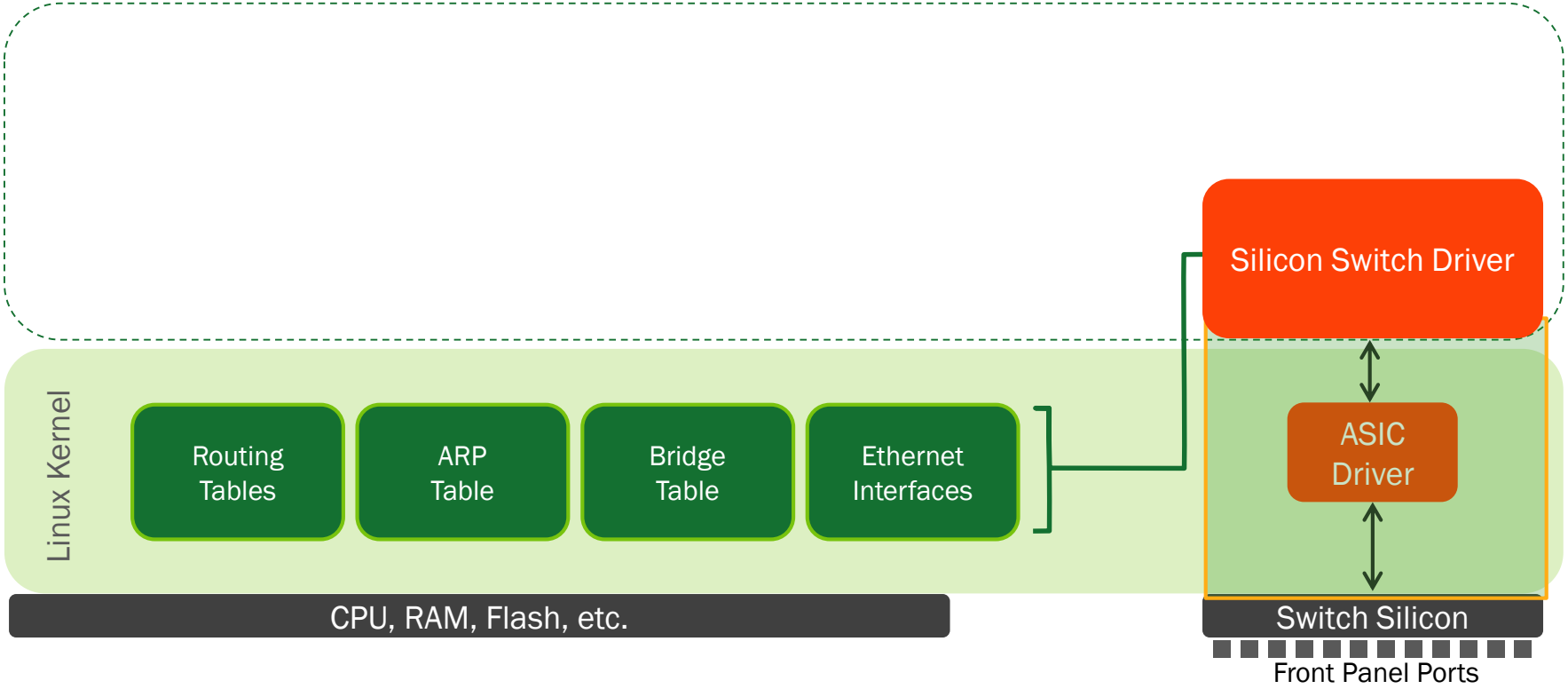

Examples: NXOS

# Linux As a NOS: Version 2



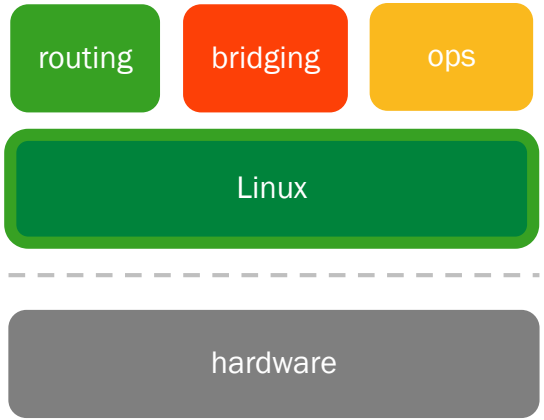

Examples: EOS, HP OpenSwitch, Dell OS10, JunOS(BSD)

# Linux As A NOS: Version 3



# Linux as NOS

- Treat as a server, performs like a traditional switch/router
  - Hardware accelerate networking forwarding path (a.k.a driver)
  - ifconfig, ethtool, isc-dhcp … apt-get
- Consistent tooling across compute & networking
  - CLI is usually bash, no walled garden
  - Use your favorite automation suite
- Choice on HW & SW suppliers
  - Same as bare metal computing is today
  - Applies to costly optics & cabling too!



cumulus

## Networking in host OS has active, community-led development

- Host driven due to VMs and Containers
- Network driven (MPLS, VRF etc.) due to open networking

## The kernel defines the API and the behavior

- Not vendor-specific

# The Next Logical Step in Software Evolution

## Evolution of the network OS

### Monolithic OS

- No real OS
- while loop
- Proprietary routing and switching stack

**Examples:**

- IOS, CatOS

### Third party real time OS

- Embedded OS with process and memory management
- Proprietary routing and switching stack

**Example:**

- ION, iCOS/Fastpath

### Linux-based OS

- Linux as embedded OS with process and memory management
- Proprietary routing and switching stack

**Examples:**

- NX-OS, EOS

### Linux OS

- Linux as network OS
- Native routing and switching
- Open and proven

**Example:**

- Cumulus Linux

# Disaggregation: Not Just For Startups, By Startups

cumulus

## Juniper announced plans to support ONIE and whitebox switching

- Albeit with just their whitebox

## Cisco announced support for ONIE

- And withdrew it from public support

## Arista has said it can support ONIE

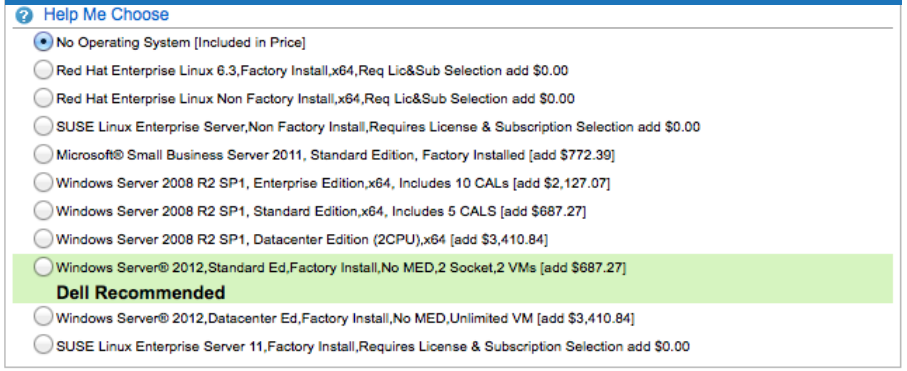
## Juniper announced its considering selling JunOS separately from its hardware

## And so on…

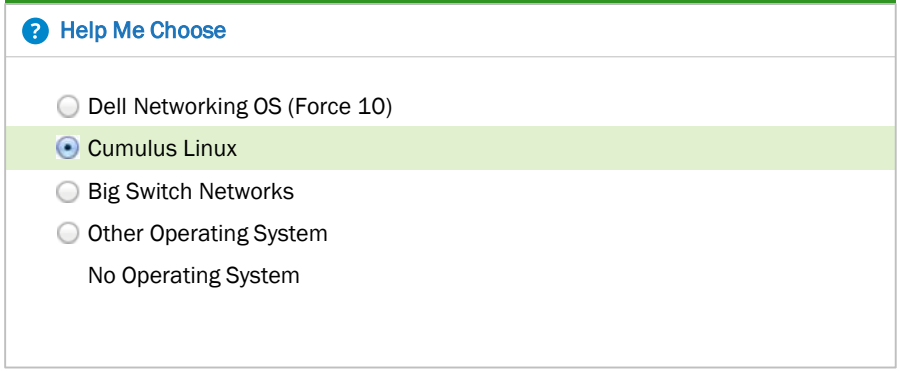
# Dell Takes the First Step

## Networking (catching up)

How do I make my IT infrastructure live up to its true potential?




How do I make my network live up to its true potential?



# Typical ODM Box Configuration

CPUs are now x86 based or ARM based

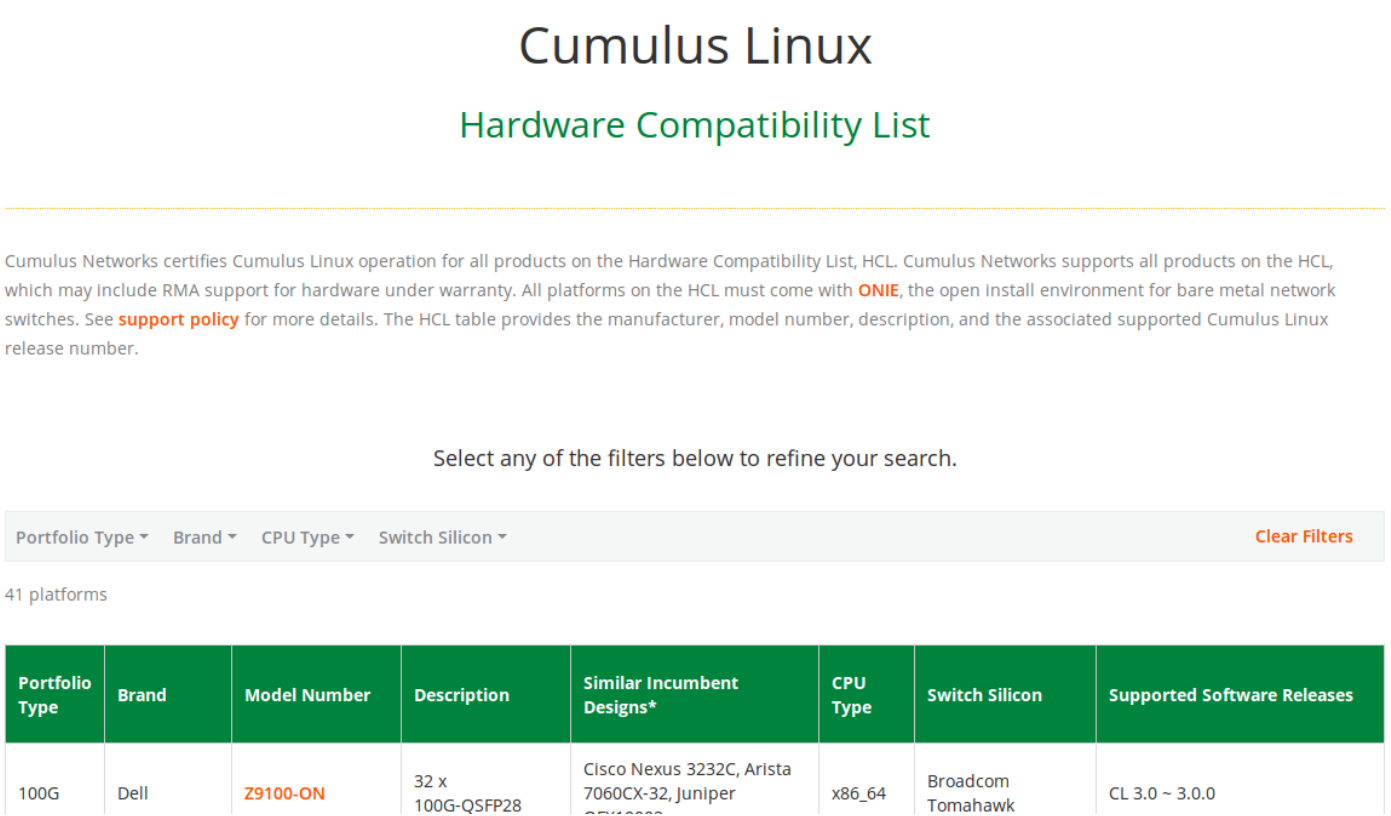
ARM exclusively used only for low end (1G) boxes

Typical Configurations:

32x100GE, 32x40GE, 48x10GE+6x40GE, 48x10GE+2x100GE etc.

No different from what traditional vendors boxes

# One Place to Find Usable Whitebox Switches



## Cumulus Linux

### Hardware Compatibility List

Cumulus Networks certifies Cumulus Linux operation for all products on the Hardware Compatibility List, HCL. Cumulus Networks supports all products on the HCL, which may include RMA support for hardware under warranty. All platforms on the HCL must come with **ONIE**, the open install environment for bare metal network switches. See **support policy** for more details. The HCL table provides the manufacturer, model number, description, and the associated supported Cumulus Linux release number.

Select any of the filters below to refine your search.

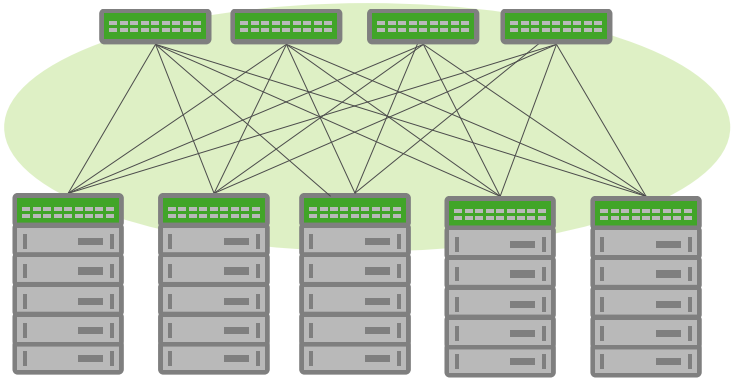
Portfolio Type ▾ Brand ▾ CPU Type ▾ Switch Silicon ▾ Clear Filters

41 platforms

| Portfolio Type | Brand | Model Number | Description | Similar Incumbent Designs* | CPU Type | Switch Silicon | Supported Software Releases |
|---|---|---|---|---|---|---|---|
| 100G | Dell | Z9100-ON | 32 x 100G-QSFP28 | Cisco Nexus 3232C, Arista 7060CX-32, Juniper | x86_64 | Broadcom Tomahawk | CL 3.0 ~ 3.0.0 |

# Tackling the Routing Beast

# CLOS Forces Change in Fundamental Building Block



L2 -> L3

Traffic distribution needs to use all links, not some

Efficient link utilization requires fine-grained traffic distribution

L2 learning model doesn't scale

ECMP falls out of IP routing naturally

Mature, sophisticated technology

# Simplifications Wrought By L3

## Fewer protocols to configure

- Single routing protocol vs many L2 protocols
- No FHRP required
- Standard, inter-operable protocols

## Fewer protocols to troubleshoot

## You can traceroute across the network!

cumulus

## Many more boxes to manage

## Key insight:

- Managing **even** 15 boxes can be painful if done manually
- With automation, managing tens of boxes is no different than hundreds of boxes

# Simplify L3 Networking: Simplify Configuration

## IP is the base technology (sometimes MPLS)

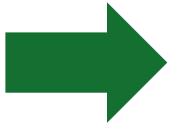
- Primary challenge is that its considered hard to configure

  Understand IP address and subnets

- Easy to configure does not mean “single point of management”

# The Work Before Automation

commons.wikimedia.org



https://www.flickr.com/photos/rubbermaid/

# Simple Patterns

## Exploit order and regularity of network

- Same ports across all boxes connected to uplink ports
- Same host connected to pair of leaves on same port on both leaves

## From this order and regularity emerge simple patterns

## Patterns can be automated

# Eliminate the Unessential

High bandwidth networks eliminate a lot of the complexity around configuring and managing QoS

- With disaggregated networking, this is very affordable

Eliminating L2, except maybe from the rack, eliminates a lot of complexity by getting rid of a lot of protocols and their configuration

# Principles of Simplifying Configuration

## Cookie cutter configuration a.k.a substitutability

- As little node-specific variation as possible

  Nothing more than a single IP address, node name, for example

- As little duplication of information as possible

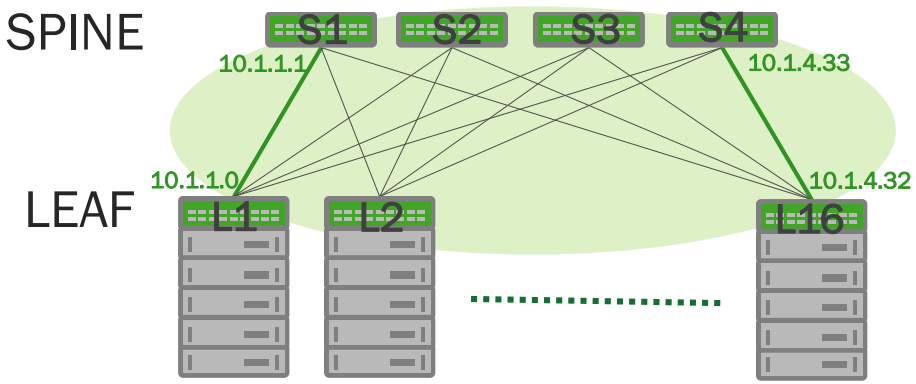
  Specifying IP addresses of interfaces AND in OSPF/BGP network statements

- As much configuration as necessary, not more

## If its simple, its automatable

- Why do boring stuff

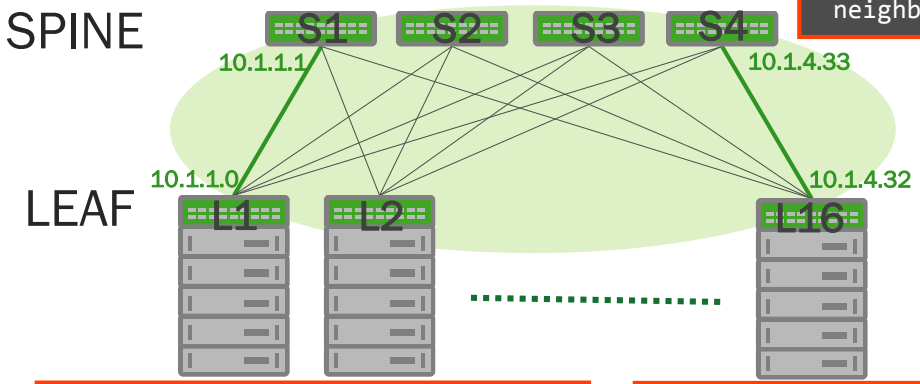
# Traditional Routing Configuration




Every interface is assigned an IP address

Each end of the link SHOULD belong to the same subnet

# Traditional BGP Configuration




S1

```
router bgp 65000
 bgp log-neighbor-changes
 bgp router-id 10.0.0.17
 !
 neighbor 10.1.1.0 remote-as 64501
 neighbor 10.1.1.2 remote-as 64502
 …
 neighbor 10.1.1.32 remote-as 64517
```

S4

```
router bgp 65000
 bgp log-neighbor-changes
 bgp router-id 10.0.0.20
 !
 neighbor 10.1.4.0 remote-as 64501
 neighbor 10.1.4.2 remote-as 64502
 …
 neighbor 10.1.4.32 remote-as 65534
```

L1

```
router bgp 64501
 bgp log-neighbor-changes
 bgp router-id 10.0.0.1
 !
 neighbor 10.1.1.1 remote-as 65000
 neighbor 10.1.2.1 remote-as 65000
 neighbor 10.1.3.1 remote-as 65000
 neighbor 10.1.4.1 remote-as 65000
```

L2

```
router bgp 64502
 bgp log-neighbor-changes
 bgp router-id 10.0.0.2
 !
 neighbor 10.1.1.3 remote-as 65000
 neighbor 10.1.2.3 remote-as 65000
 neighbor 10.1.3.3 remote-as 65000
 neighbor 10.1.4.3 remote-as 65000
```

L16

```
router bgp 64516
 bgp log-neighbor-changes
 bgp router-id 10.0.0.16
 !
 neighbor 10.1.1.33 remote-as 65000
 neighbor 10.1.2.33 remote-as 65000
 neighbor 10.1.3.33 remote-as 65000
 neighbor 10.1.4.33 remote-as 65000
```

# Whats Really Involved in Configuring Routing ?

## Three basic parts

- Who do I communicate with (neighbor, peer etc.)
- What do I tell them (IP prefixes usually)
- Tuning the conversation (timers, various protocol specific knobs)

## But first, who am I ?

- Router ID

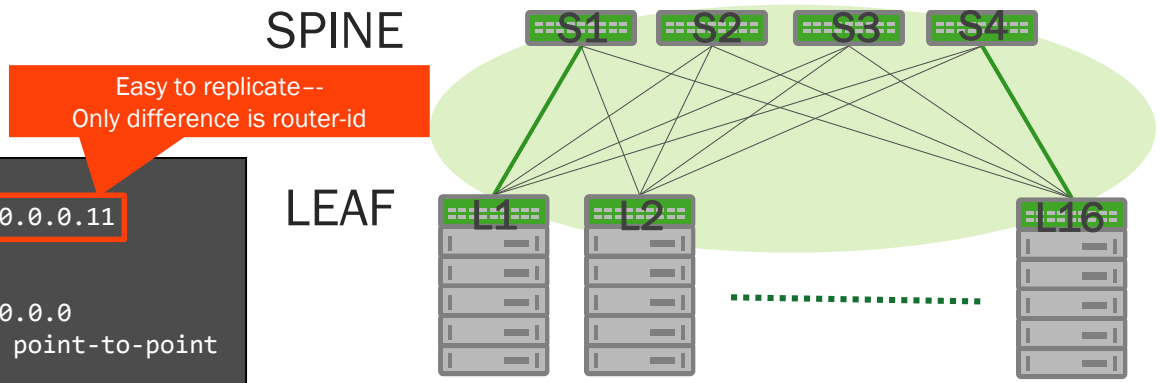
```
router ospf
  router-id 0.0.0.1
 interface swp1
   ip ospf area 0.0.0.0
   ip ospf network point-to-point
 interface swp2
    ip ospf area 0.0.0.0
    ip ospf network point-to-point

router bgp 65535
   bgp router-id 0.0.0.7
   neighbor 1.2.3.4 remote-as 65534
   neighbor 1.2.3.4 activate
   redistribute connected
```

# Simplifying Routing: Unnumbered Interfaces

## Interswitch link addresses are typically never propagated

- Reduces FIB size
- Reduces attack vector since only single reachable address
  As opposed to as many addresses as there are links
- See RFC 7404 for more details
- IETF, other network vendors also advocating the use of LLA:
  https://blog.apnic.net/2016/02/16/change-of-paradigm-with-ipv6-no-global-addresses-on-router-interfaces/

# OSPF Unnumbered




Easy to replicate--
Only difference is router-id

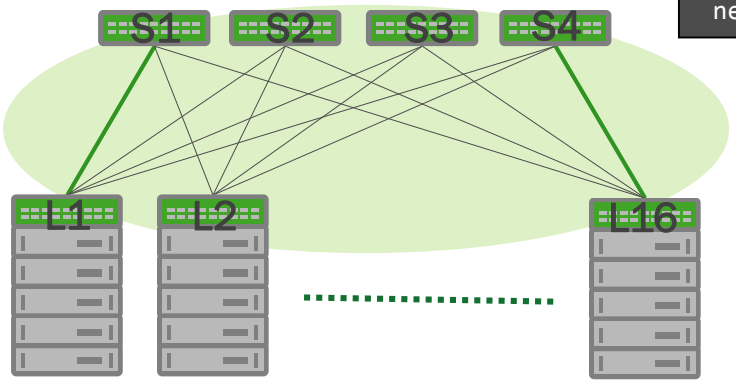
```
router ospf
ospf router-id 10.0.0.11
!
interface swp1
 ip ospf area 0.0.0.0
 ip ospf network point-to-point
!
interface swp2
 ip ospf area 0.0.0.0
 ip ospf network point-to-point
!
interface swp3
 ip ospf area 0.0.0.0
 ip ospf network point-to-point
!
interface swp4
 ip ospf area 0.0.0.0
 ip ospf network point-to-point
```

- Same configuration across L1-L16
- Same configuration across S1-S4, except they have swp1-16
- Only difference between boxes is router-id
- Old technology, part of base OSPF RFC

# BGP Unnumbered Configuration

SPINE

LEAF




S1

```
router bgp 65000
 bgp log-neighbor-changes
 bgp router-id 10.0.0.17
 !
 neighbor swp1 remote-as  64501
 neighbor swp2 remote-as  64502
 …
 neighbor swp16 remote-as 64516
```

S4

```
router bgp 65000
 bgp log-neighbor-changes
 bgp router-id 10.0.0.20
 !
 neighbor swp1 remote-as  64501
 neighbor swp2 remote-as  64502
 …
 neighbor swp16 remote-as 64516
```

L1

```
router bgp 64501
 bgp log-neighbor-changes
 bgp router-id 10.0.0.1
 !
 neighbor swp1 remote-as 65000
 neighbor swp2 remote-as 65000
 neighbor swp3 remote-as 65000
 neighbor swp4 remote-as 65000
```

L2

```
router bgp 64502
 bgp log-neighbor-changes
 bgp router-id 10.0.0.2
 !
 neighbor swp1 remote-as 65000
 neighbor swp2 remote-as 65000
 neighbor swp3 remote-as 65000
 neighbor swp4 remote-as 65000
```

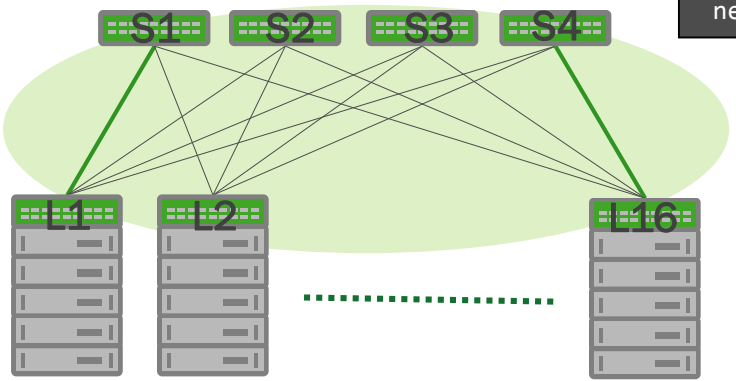
L16

```
router bgp 64516
 bgp log-neighbor-changes
 bgp router-id 10.0.0.16
 !
 neighbor swp1 remote-as 65000
 neighbor swp2 remote-as 65000
 neighbor swp3 remote-as 65000
 neighbor swp4 remote-as 65000
```

# Removing the Need For Specifying Specific Remote AS

SPINE

LEAF




S1

```
router bgp 65000
 bgp log-neighbor-changes
 bgp router-id 10.0.0.17
 !
 neighbor swp1 remote-as external
 neighbor swp2 remote-as external
 …
 neighbor swp16 remote-as external
```

S4

```
router bgp 65000
 bgp log-neighbor-changes
 bgp router-id 10.0.0.20
 !
 neighbor swp1 remote-as external
 neighbor swp2 remote-as external
 …
 neighbor swp16 remote-as external
```

L1

```
router bgp 64501
 bgp log-neighbor-changes
 bgp router-id 10.0.0.1
 !
 neighbor swp1 remote-as external
 neighbor swp2 remote-as external
 neighbor swp3 remote-as external
 neighbor swp4 remote-as external
```

L2

```
router bgp 64502
 bgp log-neighbor-changes
 bgp router-id 10.0.0.2
 !
 neighbor swp1 remote-as external
 neighbor swp2 remote-as external
 neighbor swp3 remote-as external
 neighbor swp4 remote-as external
```

L16

```
router bgp 64516
 bgp log-neighbor-changes
 bgp router-id 10.0.0.16
 !
 neighbor swp1 remote-as external
 neighbor swp2 remote-as external
 neighbor swp3 remote-as external
 neighbor swp4 remote-as external
```

# Savings In IPv4 Address Utilization

## Case 1

| | Spine | Leaf | Total |
|---|---|---|---|
| Cumulus | 4 | 16 | **20** |
| Traditional BGP | 4+ 4*16= 68 | 16+ 16*4= 80 | **148** |

## Case 2

| | Spine | Leaf | Total |
|---|---|---|---|
| Cumulus | 16 | 96 | **112** |
| Traditional BGP | 16 + 16*96 = 1552 | 96 + 96*16 = 1632 | **3184** |

# Automation Benefit: # Variables Used in Ansible Playbook

Case 1

| | Spine | Leaf | Total |
|---|---|---|---|
| Cumulus | 1 + 1 (loopback subnet + spine ASN) | 1 (Leaf ASN base, same loopback subnet) | 3 |
| Traditional BGP | 4+(4*16)+1 = 69<br>(Router IDs +<br>Total switches*TOR IPv4 +<br>ASN) | 16+(16*4) +16 = 96<br>(Router IDs +<br>Total switches*uplink IPv4 +<br>ASN) | 165 |

Case 2

| | Spine | Leaf | Total |
|---|---|---|---|
| Cumulus | 1 + 1 (loopback subnet + spine ASN) | 1 (Leaf ASN base, same loopback subnet) | 3 |
| Traditional BGP | 16+(16*96)+1 = 1552<br>(Router IDs +<br>Total switches*TOR IPv4 +<br>ASN) | 96+(96*16) +96 = 1728<br>(Router IDs +<br>Total switches*uplink IPv4 +<br>ASN) | 3280 |

# Typical BGP Configuration On Leaf With 2 Spines

```
log file /var/log/quagga/quagga.log
log timestamp precision 6
interface swp1
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 5
interface swp2
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 5
router bgp 65000
  bgp router-id 10.1.1.10
  bgp log-neighbor-changes
  bgp bestpath as-path multipath-relax no-as-set
  bgp network import-check
  maximum-paths 64
  redistribute connected route-map LOCAL_ROUTES
  neighbor ISL peer-group
  neighbor ISL advertisement-interval 0
  neighbor ISL timers connect 10
  neighbor 10.1.1.1 remote-as 65001
  neighbor 10.1.1.1 peer-group ISL
  neighbor 10.1.1.1 remote-as 65002
  neighbor 10.2.1.1 peer-group ISL
!
! Advertise all local rts except mgmt.
!
route-map LOCAL_ROUTES deny 10
   match interface eth0
route-map LOCAL_ROUTES permit 20
!
! Set src of outgoing packets to loopback's
!
route-map SETSRC permit 10
   set src 10.1.1.1
!
ip protocol bgp route-map SETSRC
```

# The Simplified Version with Cumulus Quagga

```
log file /var/log/quagga/quagga.log
log timestamp precision 6
router bgp 65000
  bgp router-id 10.1.1.1
  bgp bestpath as-path multipath-relax
  neighbor swp1 interface remote-as external
  neighbor swp2 interface remote-as external
  redistribute connected
```

# How BGP Unnumbered Works

Not really unnumbered: Uses IPv6 Link local address for BGP Sessions

Uses IPv6 Router Advertisement to learn neighbor's link local address

Uses RFC 5549 to support advertising IPv4 addresses over IPv6 session

Works on Servers and Routers

# RFC 5549 In Action

```
cumulus@tor-11$ ip route
6.0.0.9 via 169.254.0.1 dev swp3  proto zebra  src 6.0.0.10  metric 20 onlink
6.0.0.11  proto zebra  src 6.0.0.10  metric 20
        nexthop via 169.254.0.1  dev swp1 weight 1 onlink
        nexthop via 169.254.0.1  dev swp2 weight 1 onlink
        nexthop via 169.254.0.1  dev swp3 weight 1 onlink
6.0.0.12  proto zebra  src 6.0.0.10  metric 20
        nexthop via 169.254.0.1  dev swp1 weight 1 onlink
        nexthop via 169.254.0.1  dev swp2 weight 1 onlink
        nexthop via 169.254.0.1  dev swp3 weight 1 onlink

cumulus@tor-11$ ip neighbor
fe80::202:ff:fe00:d dev swp2 lladdr 00:02:00:00:00:0d router REACHABLE
fe80::202:ff:fe00:13 dev swp3 lladdr 00:02:00:00:00:13 router REACHABLE
fe80::202:ff:fe00:7 dev swp1 lladdr 00:02:00:00:00:07 router REACHABLE
192.168.0.3 dev eth0 lladdr 52:55:c0:a8:00:03 STALE
192.168.0.2 dev eth0 lladdr 52:55:c0:a8:00:02 REACHABLE
169.254.0.1 dev swp2 lladdr 00:02:00:00:00:0d PERMANENT
169.254.0.1 dev swp1 lladdr 00:02:00:00:00:07 PERMANENT
169.254.0.1 dev swp3 lladdr 00:02:00:00:00:13 PERMANENT
```

- ✓ Use of IPv4 Link Local Address to make up NextHop
- ✓ Use of Onlink attribute of Linux

- ✓ ARP entries populated on basis of MAC address learnt from RA

# Traceroute with Unnumbered Interfaces

```
cumulus@tor-11$ ip -br addr
lo               UNKNOWN  127.0.0.1/8 6.0.0.10/32 ::1/128
eth0             UP       192.168.0.15/24 fe80::201:ff:fe00:a00/64
swp1             UP       fe80::202:ff:fe00:19/64
swp2             UP       fe80::202:ff:fe00:1a/64
swp3             UP       fe80::202:ff:fe00:1b/64
swp4             UP
br1              UP       20.0.10.1/24 fe80::202:ff:fe00:1c/64
cumulus@tor-11$ traceroute 6.0.0.13
traceroute to 6.0.0.13 (6.0.0.13), 30 hops max, 60 byte packets
 1  6.0.0.7 (6.0.0.7)  0.324 ms  0.259 ms  0.197 ms
 2  6.0.0.13 (6.0.0.13)  0.601 ms  0.747 ms  0.742 ms
cumulus@tor-11$
```

No interface IP addresses
Only LLA

Shows a successful traceroute, from one leaf to another, via a spine

# Note Well: Traceroute with Multipath

## Basic Linux traceroute supports multipath

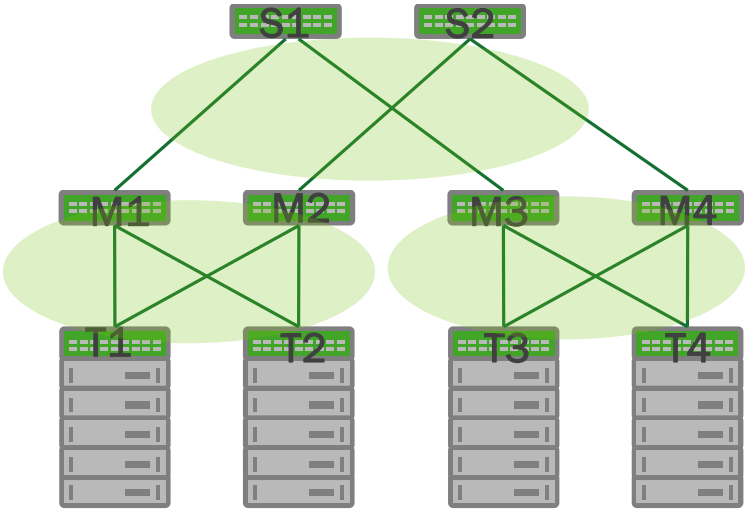
```
cumulus@host-11$ traceroute -q 6 -n 6.0.0.15
traceroute to 6.0.0.15 (6.0.0.15), 30 hops max, 60 byte packets
 1  20.0.10.1  0.594 ms  0.506 ms  0.493 ms  0.570 ms  0.542 ms  0.535 ms
 2  6.0.0.8  0.479 ms  0.466 ms  6.0.0.7  0.558 ms  6.0.0.9  0.481 ms  6.0.0.8  0.440 ms  6.0.0.9  0.425 ms
 3  6.0.0.15  0.739 ms  0.625 ms  0.721 ms  0.612 ms  0.571 ms  0.555 ms
cumulus@host-11$
```

## Other tools such as CAIDA's scamper provide nicer outputs:

```
cumulus@host-11$ scamper -i 6.0.0.15 -c "tracelb"
tracelb from 20.0.10.253 to 6.0.0.15, 2 nodes, 1 links, 18 probes, 95%
20.0.10.1 -> (6.0.0.7, 6.0.0.8, 6.0.0.9) -> 6.0.0.15
cumulus@host-11$
```

# PTM: Tackle Cabling Complexity of CLOS

- Verify cabling correctness
- Define expected topology using DOT language
- Verify connectivity per topology plan using LLDP
- Take dynamically defined actions based on mis/match of expected & actual




Topology Graph

```
Graph G {
S1:p1 – M1:p3;
S1:p2 – M3:p3;
S2:p1 – M2:p3;
S2:p2 – M4:p3;
S2:p3 – M3:p4;
M1:p1 – T1:p1;
M1:p2 – T2:p1;
...
M4:p2 – T4:p2;
}
```

# Network Virtualization: Building Block for Clouds

# Network Virtualization: The Need For a New Network Model

## VLAN was the old way to virtualize networks

- L2 concept
- Baked deep into control and data plane of network equipment

## VLAN is neither scalable nor agile

- 12 bit number is too small
- Control protocol scalability forced careful, slow moving network changes to add/remove VLANs

## Rise of server virtualization drove the need for a new network virtualization model

# VxLAN, NvGRE: New Network Virtualization Concepts

## Network virtualization typically handles at endpoints (compute nodes)

- Since VNIC spin up/down is not detectable by network devices

## VxLAN is an IP encapsulation that carries a L2 payload

- UDP-based
- Works well on both routed and bridged networks

## NvGRE is similar, but used only within Microsoft

# Summing Up Session 2

L2

L3


app app app
operating system
hardware

app app app
operating system
hardware

# Session 3: Ecosystem & Futures

# The Story So Far

## Disaggregation has begun in networking

- Every indicator you can look at points to this

## Network OS is a toss-up between Linux and Linux-based models

- Why should you care which model is picked ?

## But what about apps and the ecosystem ?

- How the answer above affects the answer to this

# A Long, Long Time Ago…

Networking
And
Computing
Diverged

Photo Courtesy of https://www.flickr.com/photos/joethorn/

# So problems are solved twice

# Different Models Between Compute & Network

| | Compute | Networking |
|---|---|---|
| Hardware Packaging | Disaggregated HW & SW | Monolithic |
| Software Packaging | Multiple: deb, rpm, binary | Binary, proprietary packaging |
| Management | Automated | Manual |
| Admin Toolchain | Rich, diverse, open | Stodgy, limited, proprietary |
| Monitoring | Near realtime, extensible | Minutes, vendor-specific |
| | | |
| | | |

# Radically Different {Open} Development Processes

```
 * internal xml or external
 * external is needed when running in static mode
 *
 * @var boolean
 */
define('PSI_INTERNAL_XML', false);

if (version_compare("5.2", PHP_VERSION, ">")) {
    die("PHP 5.2 or greater is required!!!");
}
if (!extension_loaded("pcre")) {
    die("phpSysInfo requires the pcre extension to php in order to work
        properly.");
}

require_once APP_ROOT.'/includes/autoloader.inc.php';

// Load configuration
require_once APP_ROOT.'/config.php';

if (!defined('PSI_CONFIG_FILE') || !defined('PSI_DEBUG')) {
    $tpl = new Template("/templates/html/error_config.html");
    echo $tpl->fetch();
    die();
```

VS

https://pixabay.com/en/code-php-web-development-583795/

https://www.theguardian.com/sustainable-business/2015/apr/30/tradeshift-eliminates-friction-in-nhs-bureaucracy-using-e-billing

# Does This Difference in Process Matter ?

cumulus

"We hope this will let the machine learning community—everyone from academic researchers, to engineers, to hobbyists—exchange ideas much more quickly, *through working code rather than just research papers*."

- From Google's blog about Tensorflow
  (https://googleblog.blogspot.com/2015/11/tensorflow-smarter-machine-learning-for.html)

"Rough Consensus, Working Code"

- IETF Founding Maxim

# Structured I/O

vs

# Myth of the Uniform Data Model

# And The Admins Were Left With…

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;
- Robert Frost

Photo Courtesy of https://www.flickr.com/photos/joethorn/

# And The Consequence Is..

https://www.flickr.com/photos/docsearls/


https://www.flickr.com/photos/davegray/


https://www.flickr.com/photos/pikerslanefarm/

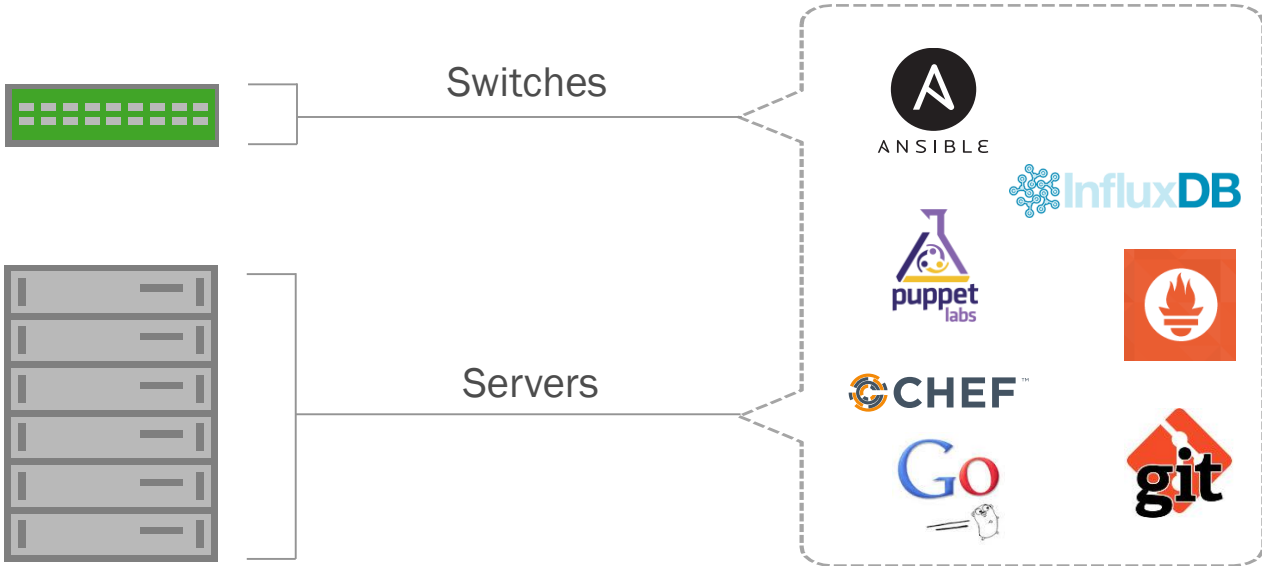
# Implications of Unifying Compute & Networking

# Breakdown the Silo'd Model Of Operations


Picture courtesy: http://www.interiordecoratorcourse.com/3-things-you-should-check-before-tearing-down-a-wall/

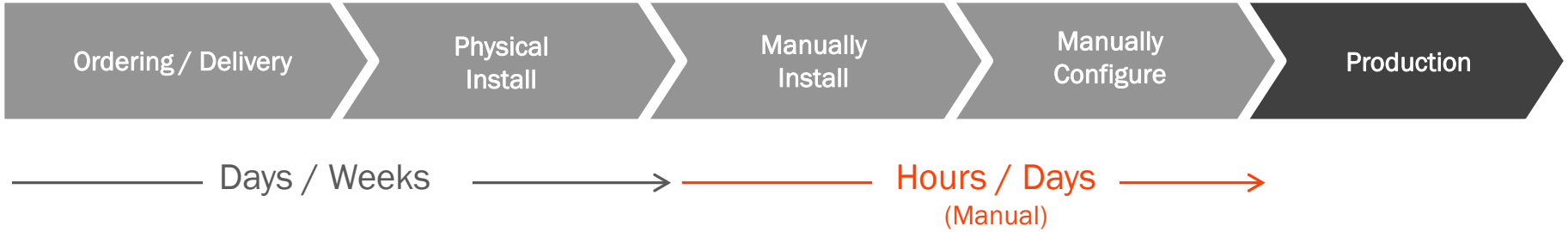
# Converged Administration

## Same automation tools and languages for managing servers are now available for the network

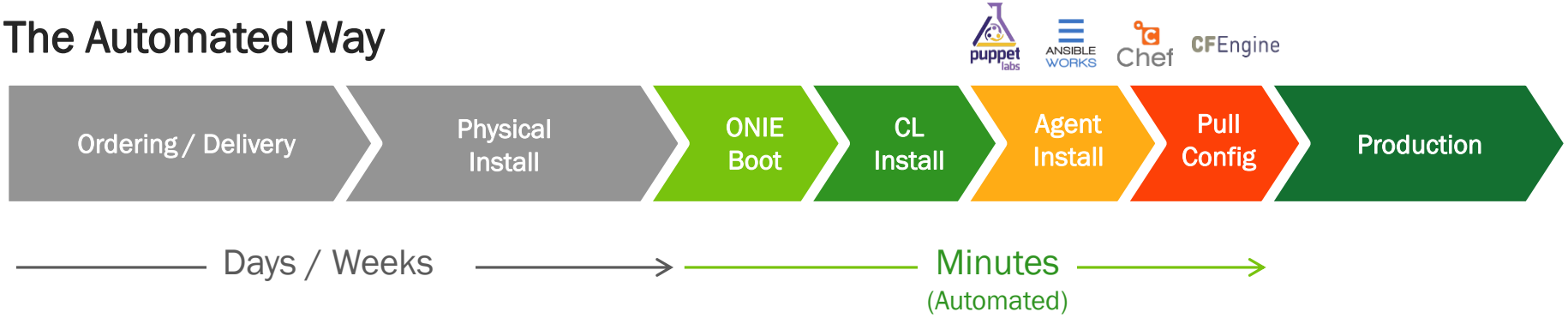


# Provisioning: From Days to Minutes

## The Traditional Way




## The Automated Way



# Another Example: Validate Networks Before Deploying Them

## Using Vagrant, Ansible (or your favorite configuration tool) and Cumulus VX to build a data center on your laptop

- Validate configuration via Serverspec or Behave
- Make changes and see the effect before deploying
- Use Prometheus or Influxdb or collectd/ganglia to monitor
- Use ELK or Splunk to analyze logs, query the past etc.
- Interesting possibilities for troubleshooting

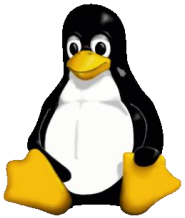
And then use the same configuration and validation and troubleshooting to deploy your production network

## Linux Implementation

- Kernel v4.3 and forward

## Developed by Cumulus Networks for Linux

- Consistent API across all Linux Devices - switches and hosts



# Linux OS Distributions (June 2016)

| Distribution | Kernel Version | VRF status | Capabilities |
|---|---|---|---|
| **Cumulus Linux 3.0** | 4.1 | enabled | All + Mgmt VRF |
| **Debian - stretch** | 4.6 | enabled | IPv4, IPv6 global, “VRF All” TCP sockets |
| **Ubuntu 16.04** | 4.4 | enabled | IPv4, IPv6 global |
| **Fedora 24 (May beta)** | 4.5 | disabled | none |
| **Fedora 23** | 4.5 | disabled | none |

# MPLS in Linux Kernel

MPLS considered too complex to administer

In reality, it can be simplified to be almost routing

Avoids the plethora of encapsulation protocols still being designed/developed

- Because MPLS encap cannot be started on hosts

WIP in the Linux kernel

# Routing From The Host

Many popular open source routing suites available now

Instead of using bridging and VLANs, can you use routing and VRFs ?

# Typical BGP Configuration

```
log file /var/log/quagga/quagga.log
log timestamp precision 6
interface swp1
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 5
interface swp2
  no ipv6 nd suppress-ra
  ipv6 nd ra-interval 5
router bgp 65000
  bgp router-id 10.1.1.10
  bgp log-neighbor-changes
  bgp bestpath as-path multipath-relax no-as-set
  bgp network import-check
  maximum-paths 64
  redistribute connected route-map LOCAL_ROUTES
  neighbor ISL peer-group
  neighbor ISL advertisement-interval 0
  neighbor ISL timers connect 10
  neighbor 10.1.1.1 remote-as 65001
  neighbor 10.1.1.1 peer-group ISL
  neighbor 10.1.1.1 remote-as 65002
  neighbor 10.2.1.1 peer-group ISL
```

```
!
! Advertise all local rts except mgmt.
!
route-map LOCAL_ROUTES deny 10
   match interface eth0
route-map LOCAL_ROUTES permit 20
!
! Set src of outgoing packets to loopback's
!
route-map SETSRC permit 10
   set src 10.1.1.1
!
ip protocol bgp route-map SETSRC
```

# The Simplified Version with Cumulus Quagga

```
log file /var/log/quagga/quagga.log
log timestamp precision 6
router bgp 65000
  bgp router-id 10.1.1.1
  neighbor swp1 interface remote-as external
  neighbor swp2 interface remote-as external
  redistribute connected
```

# Another Example: Docker Networking

# Switch As a Platform, Not a Blackbox

## Open network up to innovation again

- This was a key reason why IP overcame competing technologies

## Network operating system as an enabler, not gatekeeper

- Support “no assembly required” networking
- Allow customization if customer/partner desire
- Enable rich ecosystem

# And The Promise...

# con·sil·i·ence

kənˈsilēəns/Submit
noun
agreement between the approaches to a topic of different academic subjects, especially science and the humanities.

# Summing Up: Session 3

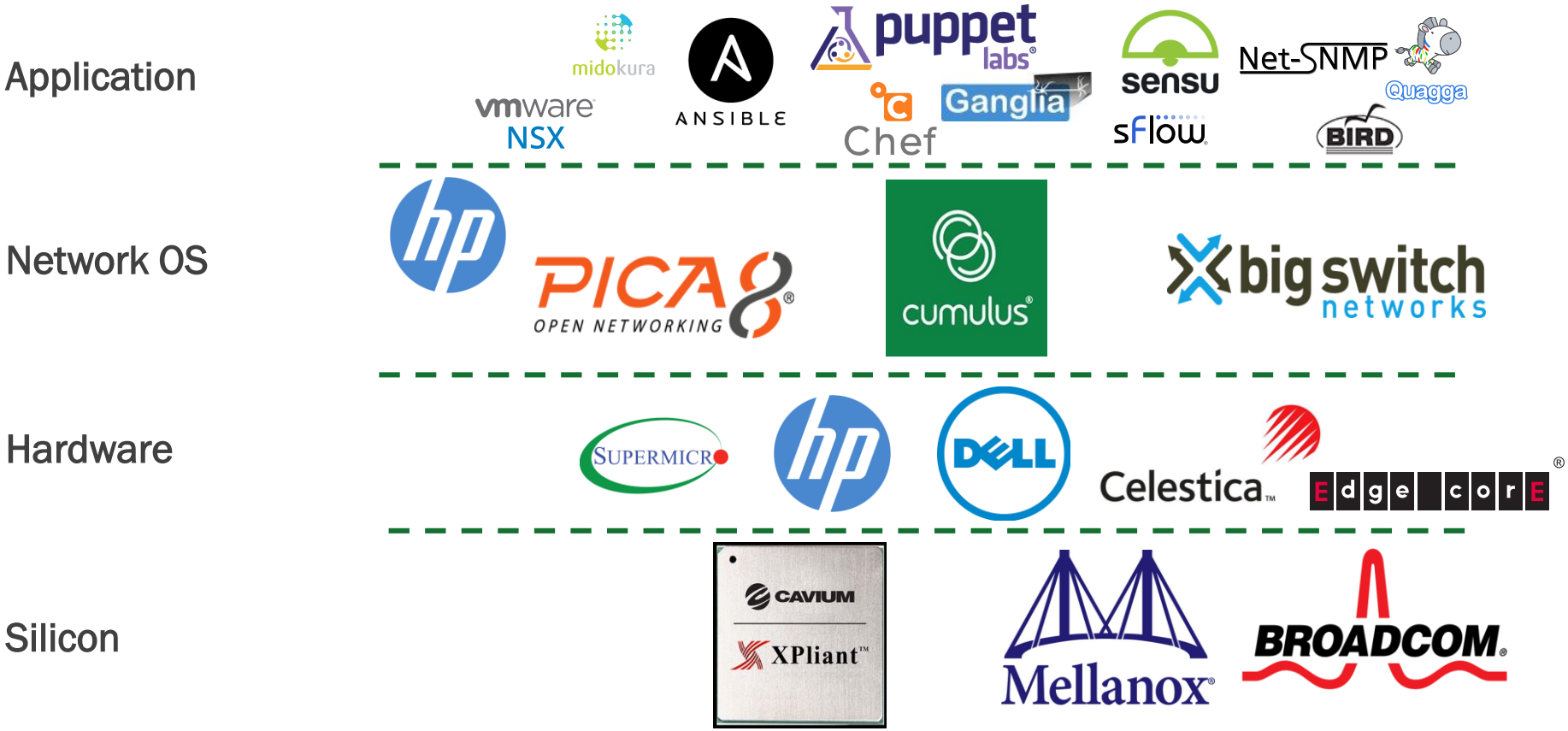
Consilience is a new possibility in the DC

Break down the silos between network, compute and storage

DevOps/SRE, Open Networking fueling this transition

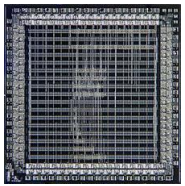
Consilience has the promise to fuel innovation again

# Bare Metal Switching Ecosystem: {small} Sample



# Key Transformations Underway in Networking Industry

app app app
operating system
hardware



Custom
ASIC

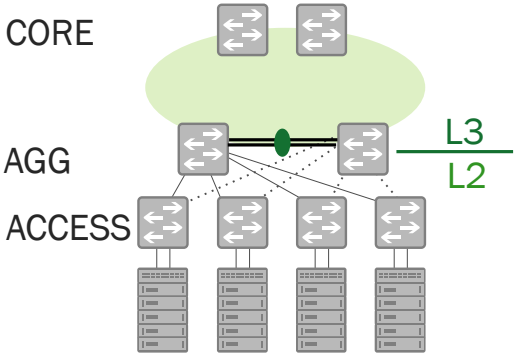
app app app
operating system
hardware

Merchant
Silicon

BROADCOM. intel Mellanox TECHNOLOGIES

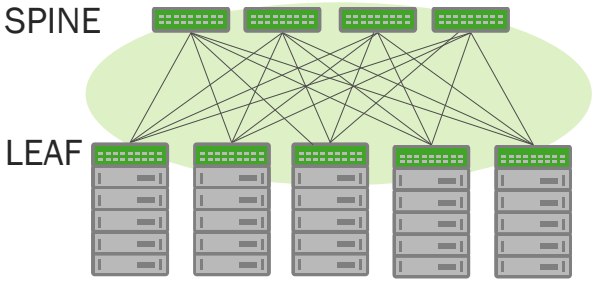
# Key Transformations in Networking Operations

L2

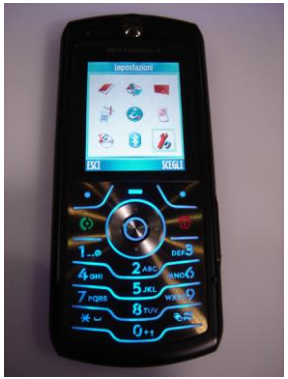


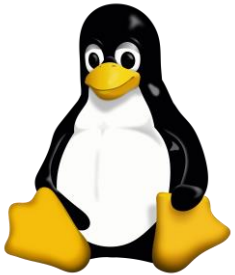


L3



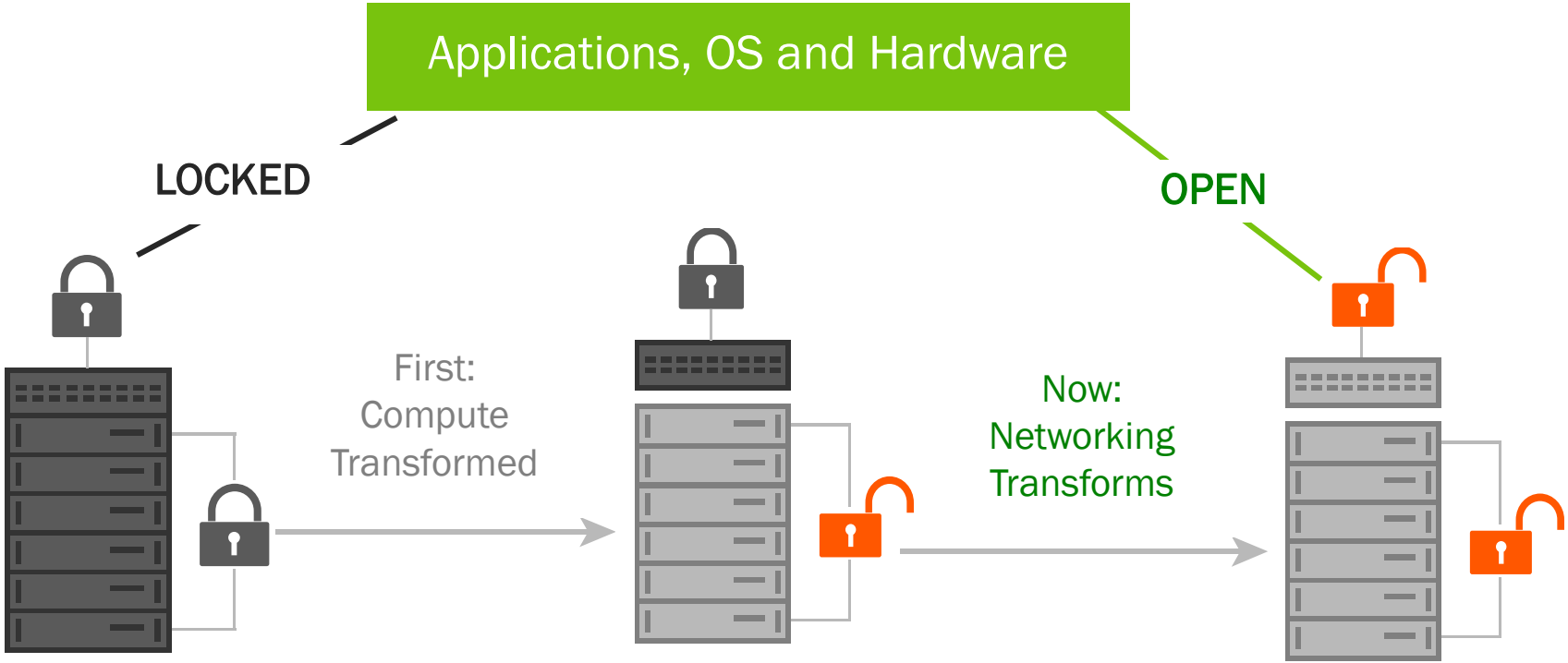
# Key Transformations Still in Progress





```c
/* This line basically imports the "stdio" header file, part of
 * the standard library. It provides input and output functionality
 * to the program.
 */
#include <stdio.h>

/*
 * Function (method) declaration. This outputs "Hello, world" to
 * standard output when invoked.
 */
void sayHello() {
    // printf() in C outputs the specified text (with optional
    // formatting options) when invoked.
    printf("Hello, world!");
}

/*
 * This is a "main function". The compiled program will run the code
 * defined here.
 */
void main() {
    // Invoke the sayHello() function.
    sayHello();
}
```

# Transforming: Compute, now Networking



# Thank You!