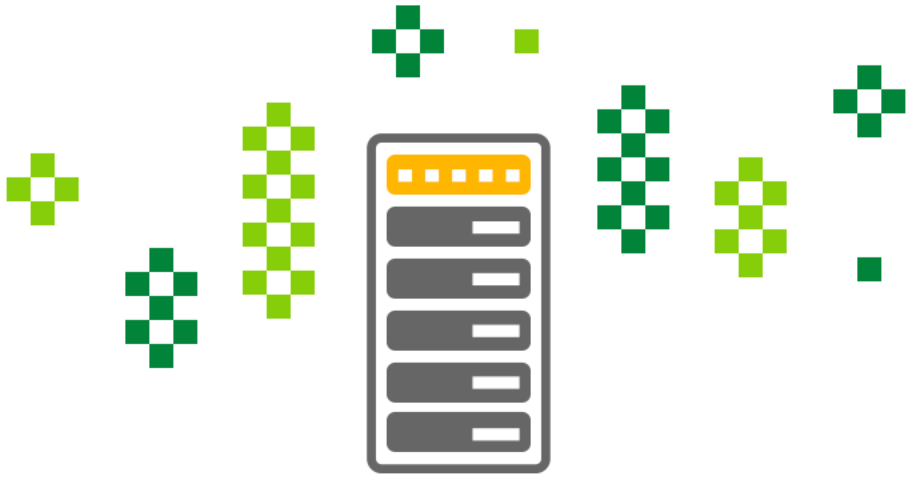


Over coming traditional network limitations with open source

cumulus networks

Modern datacenters adoption of new technologies

- application demands push protocol scale
- physical (100Gbps distance, speed of light)
- administrative (layer 8) resources

"SDN", OpenFlow, network virtualization, configuration management, and many other efforts are throwing a wrench against traditional networking practices

Traditional networking

Typical Network Operating System (switch and/or router)

- Structured as “black box”
  - `CLI != API`
- Closed development model
  - Diagnostics “under the hood” difficult to see
- Complicated management tool chains
  - SNMP MIB’s… hell
  - Screen scraping… regex’s change on OS version
  - Arcane / low adoption scripting languages
- Not geared for rapid spin-up / spin-down of resources

Modern datacenter network roots

- IP-based networks
  - Limited adoption - large scale L2, InfiniBand, ATM
- Configuration management / automation
  - Monitoring
  - Policy enforcement
  - Rapid spin-up / spin-down
- New breed of applications
  - East-West vs. North-South flows

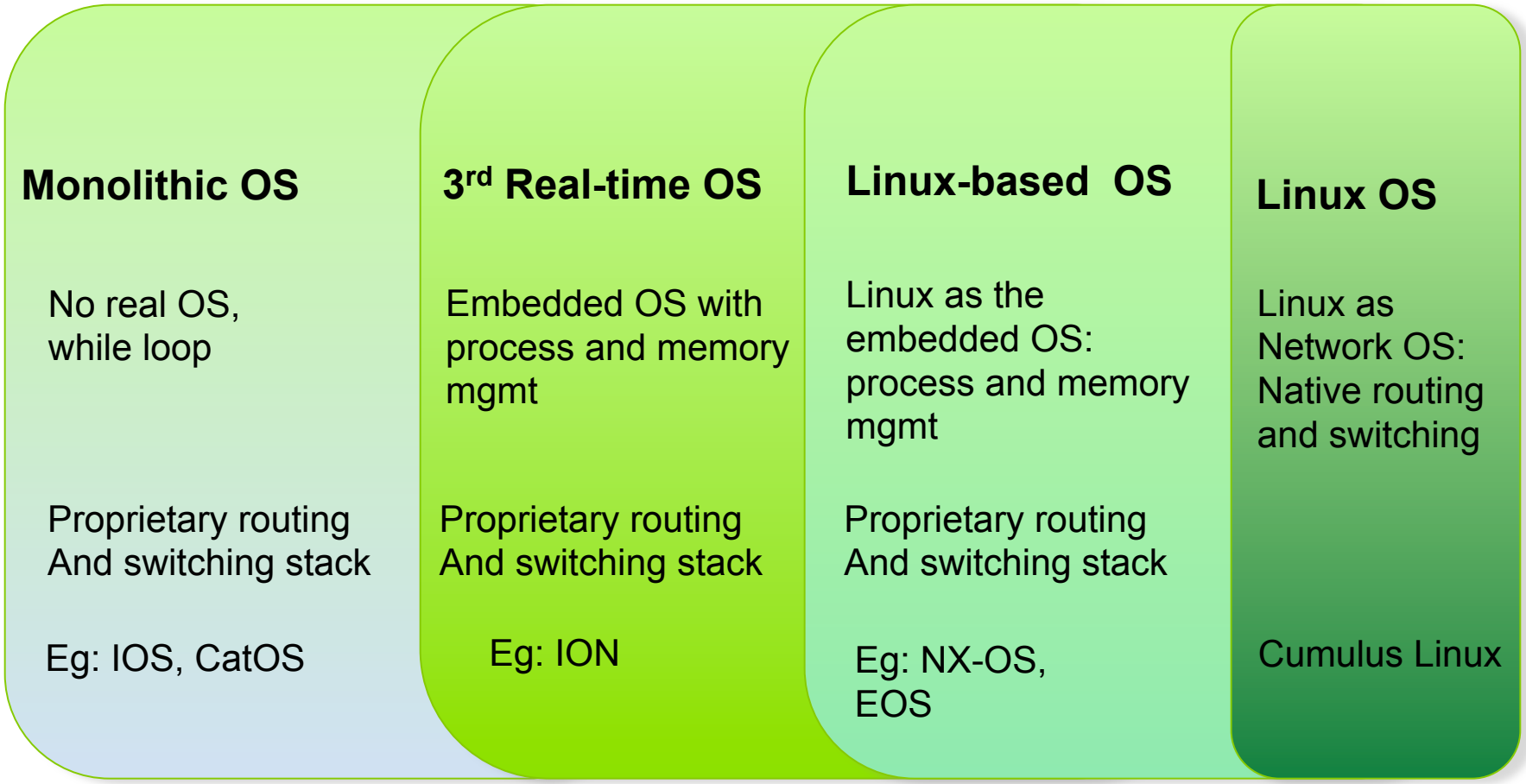
Linux?

- Dominate server platform
  - Well established ecosystem of distributions, best practices, knowledge
  - Open well documented API, large selection of language interpreters
  - Excellent networking support - IPv6, NAT's, QoS, accounting
- Vibrant community which fuels rapid innovation
- Heavy automation frameworks
  - Open nature has facilitated huge management tool-chain progress

In other words…

GNU/Linux is a great fit as the OS for *not just* servers but also routers and switches in the modern data center

Operating System Evolution



What advantages does this provide?

Open Source L2 & L3

Routing

- Quagga (many forks), BIRD, Xorp
  - OSPF unnumbered
  - BGP next-hop self
  - Looking glass

Bridging

- Kernel STP, MSTPD (BDPU Guard, Bridge Assurance)

Discovery

- LLDAP, Open-LLDP, LLDPD (many implement CDP, FDP, etc.)

L8 Management

- Traditional tools
  - **TCL** – limited adoption
  - **XSLT** – single vendor, mostly supplied tools
  - **Except** – Rancid base, very popular

- "DevOps" tools have major adoption
  - Cfengine, Puppet, Chef, Ansible
  - Salt, Trigger, … literally new tools every quarter
  - Large diverse communities (conferences, books, professional services)
  - Nirvana = same tool chain for compute AND network environments

- "NetDevOps" re-born again
  - NetDev abstraction layer in puppet, chef, & ansible
  - Possible "SDN" pill which CCIE's can appreciate?

Monitoring

- **Traditional tools**
  - SNMP – Where can I get a copy of the MIB?
  - MRTG
  - Cacti
- **Newer tools** (again, compute folks learned long ago SNMP was a fail)
  - CollectD
  - Diamond
  - Graphite
  - Sensu
- **Deploy agents directly on the network devices, pushing stats and state, instead of polling**

Cumulus Networks contributions

- **ONIE**
  - Open Source boot loader for network devices
- **Prescriptive Topology Daemon**
  - Data centre cable verification using LLDP
- **Quagga**
  - Actively submitting patches, major bugfixes
- **MSTP**
  - Bridge assurance, various bugfixes

ONIE - Open Network Install Environment

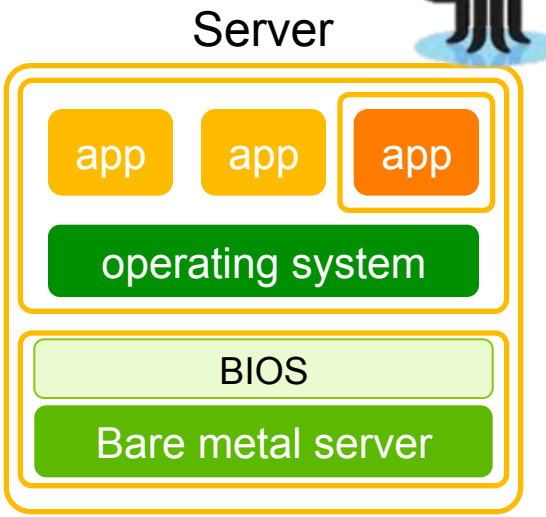
Problem:

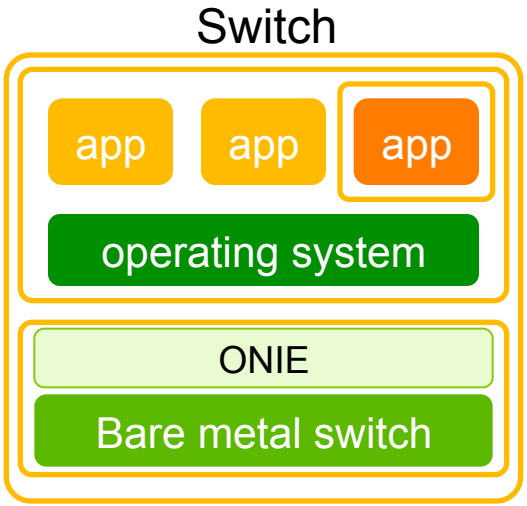
•Switches need the equivalent of a boot loader to allow disaggregation of hardware from operating system

Solution: ONIE, installer environment to address open hardware ecosystem

•A small, Linux based operating system that comes pre-installed on bare metal switches

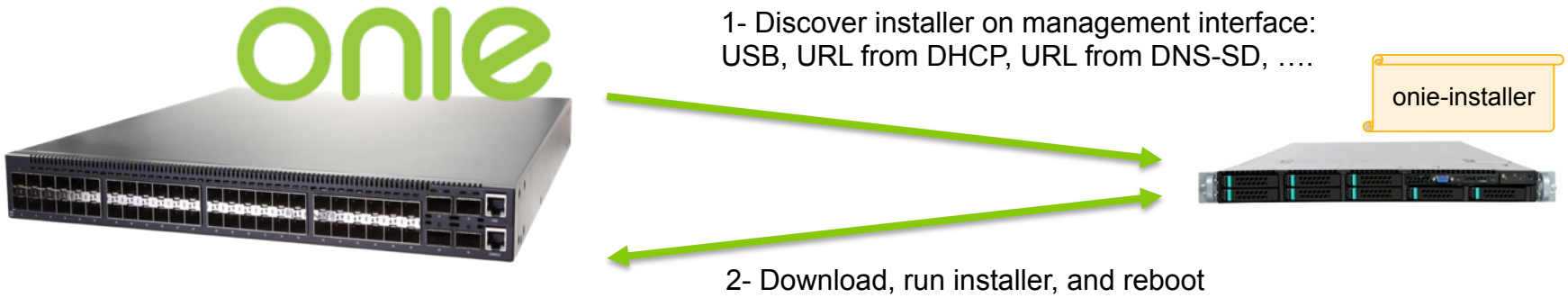
•Provides an environment for network OS installers (Network operating system neutral)





ONIE - Open Network Install Environment

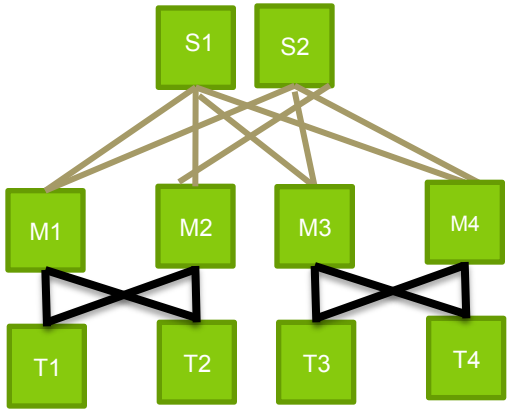
- **Zero-touch install of operating system on industry-standard gear running the Open Network Install Environment (ONIE)**
  - Industry standard gear comes with ONIE
  - ONIE provides the installer environment for auto-installation of network operating system
  - ONIE discovers the operating system through USB, DHCP,..., and Cumulus Linux gets downloaded and installed on the system



Prescriptive Topology Module

- **Goal: Operational simplicity, reduced Operator errors**
  - Verify connectivity per cabling plan
  - Bring up routing adjacency only if cabling test passes
  - Selective actions on link up
- **How? Network topology specified via DOT language and distributed to all nodes**
  - Each node determines its relevant information
  - Use LLDP to verify connectivity
  - Logs errors
  - Daemon executes a set of scripts on topology pass and a different set of scripts on topology fail

Topology

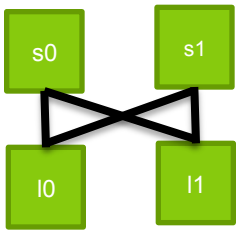


Topology graph:

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

Prescriptive Topology Module

Topology




```
digraph G {

    //spine0's connections
    spine0:swp1 -> leaf0:swp1;
    spine0:swp2 -> leaf1:swp1;

    //spine1's connections
    spine1:swp1 -> leaf0:swp2;
    spine1:swp2 -> leaf1:swp2;

    //leaf0's connections
    leaf0:swp1 -> spine0:swp1;
    leaf0:swp2 -> spine1:swp1;

    //leaf1's connections
    leaf1:swp1 -> spine0:swp2;
    leaf1:swp2 -> spine1:swp2;
}
```

Prescriptive Topology Module

- Written in C and Python
- Communicates with LLDPD (based on *https://github.com/vincentbernat/lldpd*)

```
cumulus@S1:~# ptmctl
--------------------------------------------------------------------
Port   Status Expected Nbr         Observed Nbr         Last Updated
--------------------------------------------------------------------
swp1   pass   M1:swp3              M1:swp3              17h:39m:21s
swp2   pass   M2:swp3              M2:swp3              17h:39m:21s
swp3   pass   M3:swp3              M3:swp3              17h:39m:21s
swp4   pass   M4:swp3              M4:swp3              17h:39m:21s
Swp5   fail   M5:swp3              M4:swp4              17h:39m:21s

cumulus@S1:~#
```

Prescriptive Topology Module

Fork me on GitHub

Interoperability

- Any device running an LLDP daemon
- Routing adjacencies can be brought by the device running PTM.

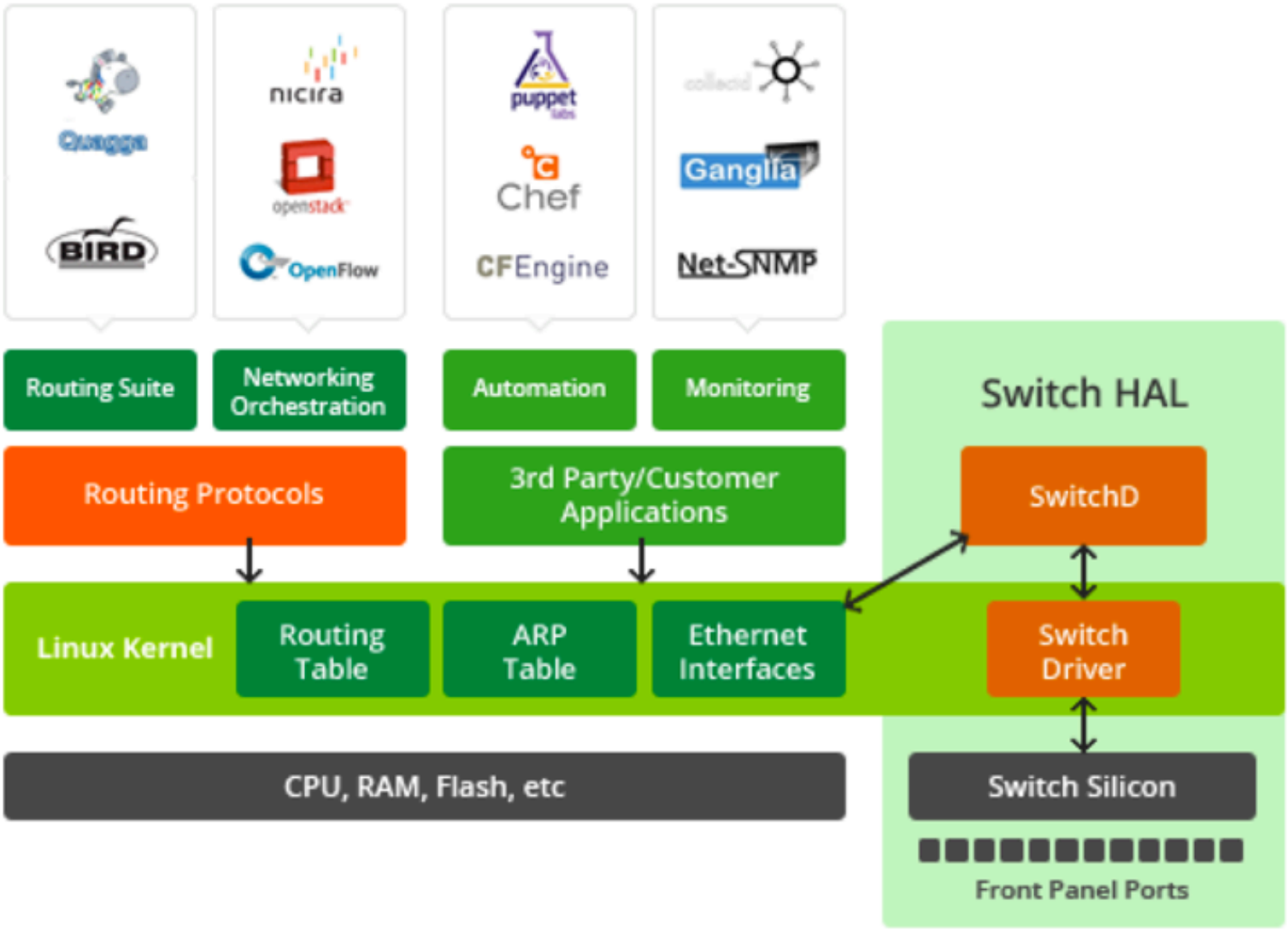
```
digraph G {;
      S1:swp1 -> S2:swp1;
      S1:swp2 -> S2:swp2;
      S1:swp3 -> "procurve1.lab":21;
      S1:swp4 -> "procurve1.lab":22;
      S1:swp5 -> "cisco1.lab":"GigabitEthernet0/1";
      S1:swp6 -> "jmx480":"xe-0/0/0";
      S1:swp7 -> webserver1:eth0;
      S1:swp8 -> webserver1:eth1;
}
```

```
cumulus@S1:~# ptmctl
--------------------------------------------------------------------
Port   Status Expected Nbr          Observed Nbr          Last Updated
--------------------------------------------------------------------
swp1   pass   S2:swp1               S2:swp1               17m: 2s
swp2   pass   S2:swp2               S2:swp2               17m: 2s
swp3   pass   procurve1.lab:21      procurve1.lab:21      17m: 10s
swp4   pass   procurve1.lab:22      procurve1.lab:22      17m: 10s
swp5   pass   cisco1.lab:GigabitEthernet0/1 cisco1lab:GigabitEthernet0/1
swp6   pass   jmx480.lab:xe-0/0/0    jmx480.lab:xe-0/0/0    17m: 1s
swp7   pass   webserver1:eth0       webserver1:eth0       17m: 3s
swp8   pass   webserver1:eth1       webserver1:eth1       17m: 3s
```

What are we missing?

Hardware acceleration of the networking forwarding path

One way of hardware accelerating



Thank you!

nat@cumulusnetworks.com | @natmorris

cumulus networks