SDN: situação do mercado e próximos movimentos

Prof. Christian Esteve Rothenberg
FEEC/UNICAMP

27 de agosto de 2014

Disclaimer/Warning

- Ack the credits for most of the content
- Wake Up! Lots of content ahead! (especially for a 35min talk)

Agenda

SDN

- An evolving paradigm
- Understanding Different Models
- Hybrid Deployments

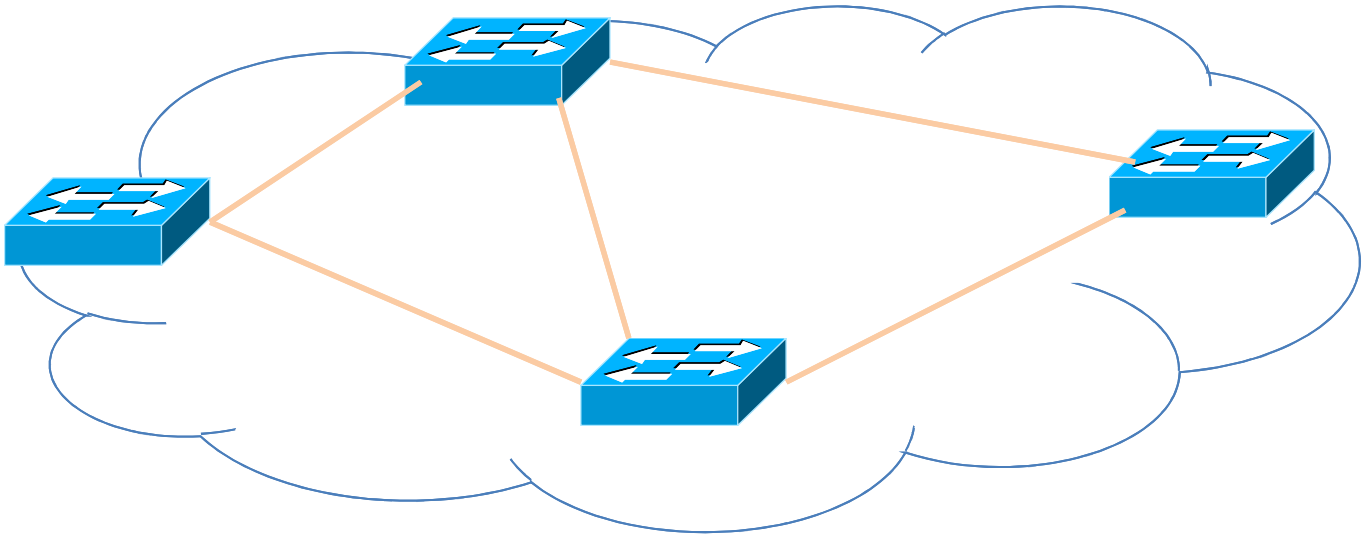
OpenFlow Future

- Challenges: Protocol versions and Model
- Work at ONF, industry and academia

Rethinking the "Division of Labor"

Traditional Computer Networks

Data plane:
Packet streaming



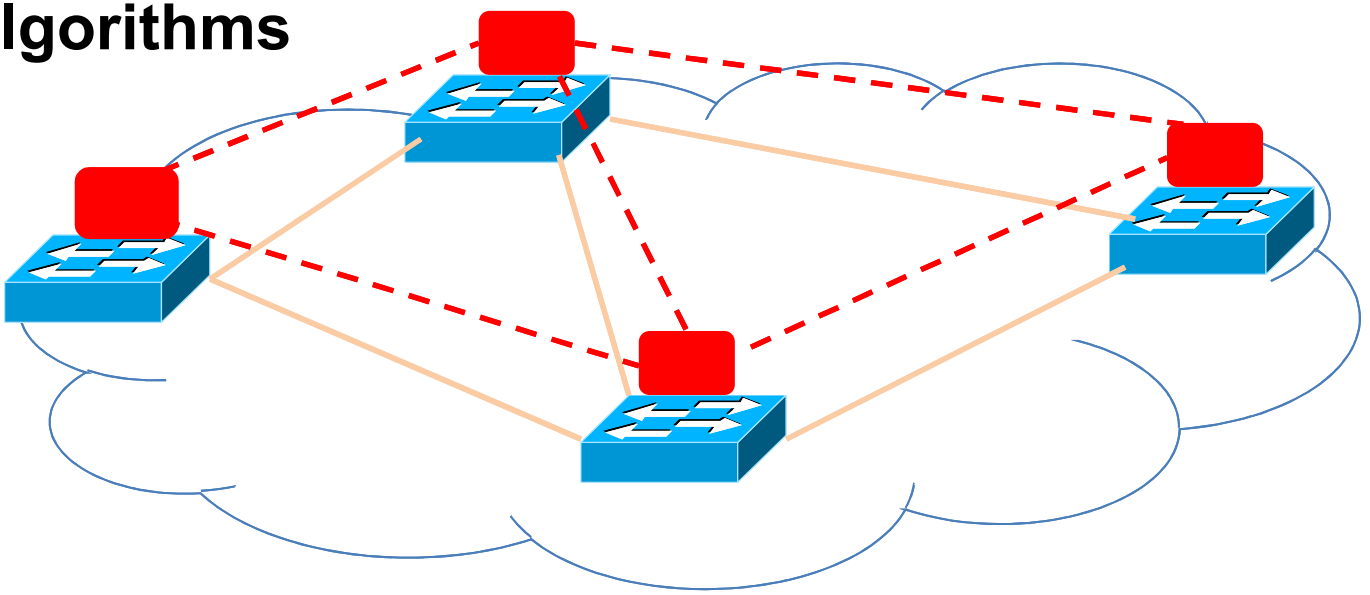
Forward, filter, buffer, mark, rate-limit, and measure packets

Source: Adapted from J. Rexford

Rethinking the "Division of Labor"

Traditional Computer Networks

Control plane:
Distributed algorithms



Track topology changes, compute routes, install forwarding rules

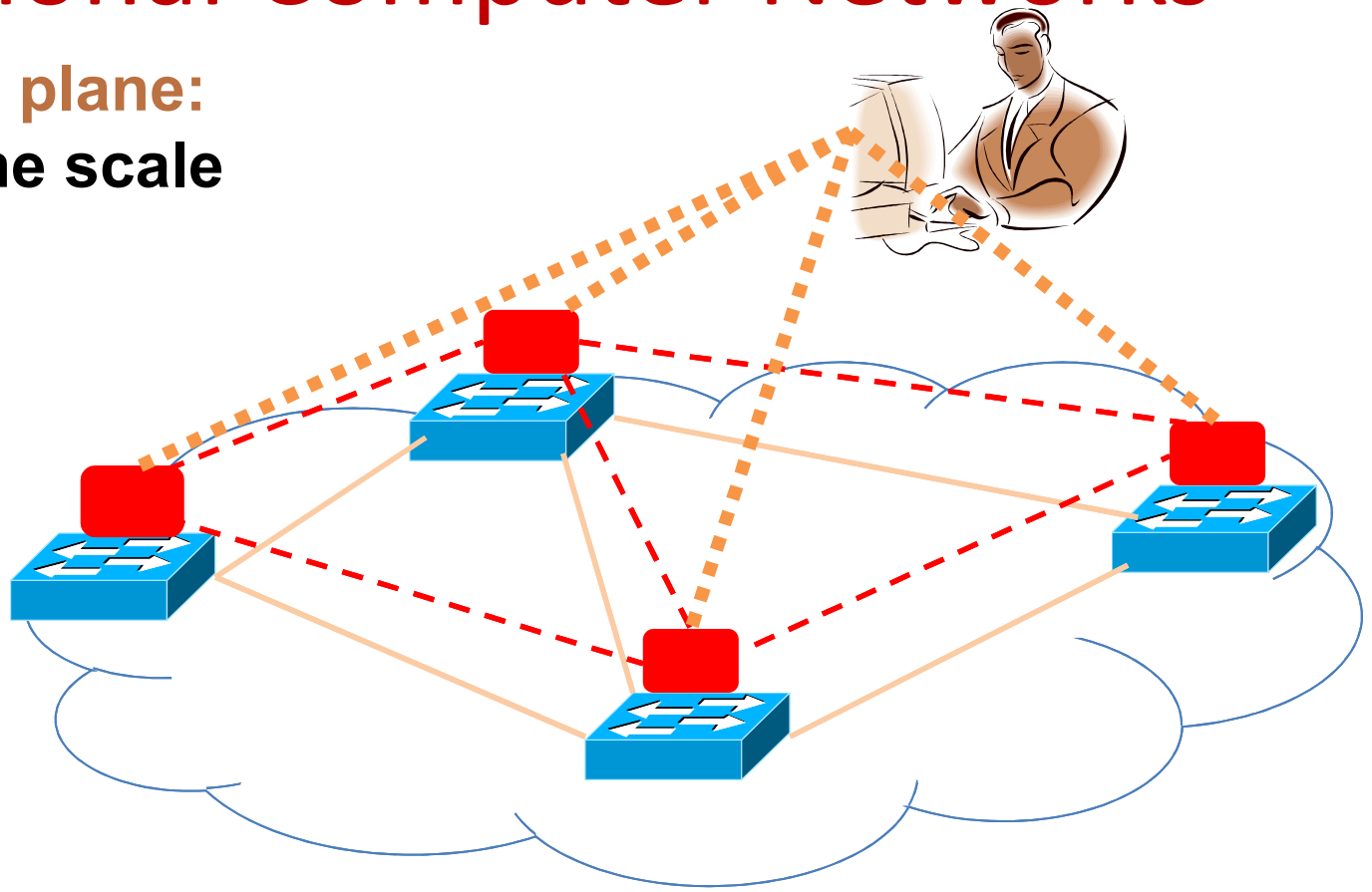
Source: Adapted from J. Rexford

Rethinking the "Division of Labor"

Traditional Computer Networks

Management plane:

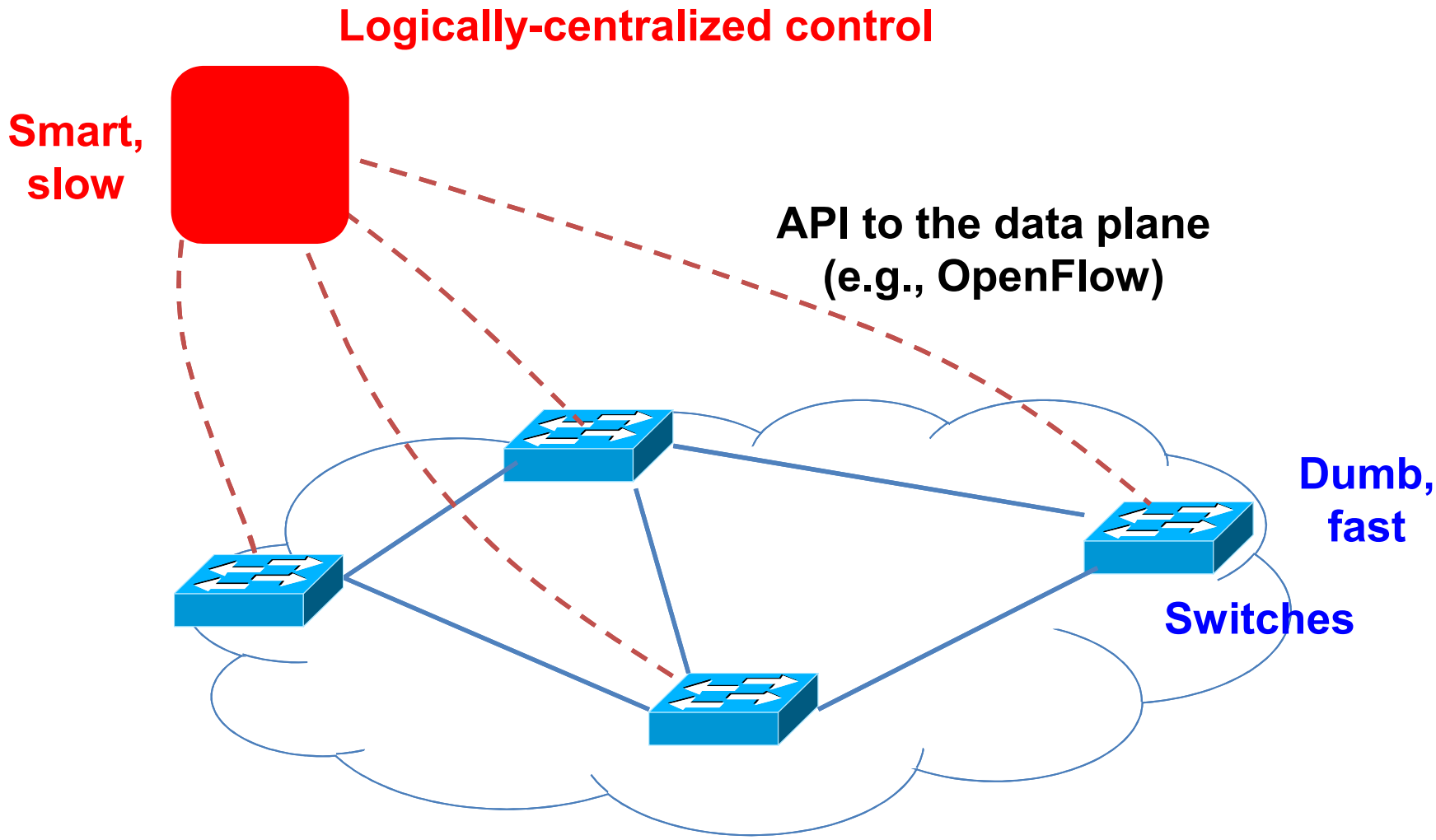
Human time scale



Collect measurements and configure the equipment

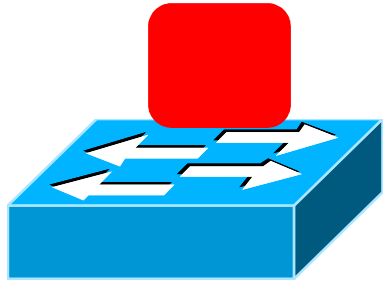
Source: Adapted from J. Rexford

Software Defined Networking (SDN)



Death to the Control Plane!?

- Simpler management
  - No need to “invert” control-plane operations
- Faster pace of innovation
  - Less dependence on vendors and standards
- Easier interoperability
  - Compatibility only in “wire” protocols?
- Simpler, cheaper equipment
  - Minimal software



Source: Adapted from J. Rexford

(Promised) Implications of SDN

Separation of Control/Data Plane

- Today, routers implement both
  - They forward packets
  - And run the control plane software
- SDN networks
  - Data plane implemented by switches
    - Switches act on local forwarding state
  - Control plane implemented by controllers
    - All forwarding state computed by SDN platform
- This is a technical change, with broad implications

Source: Adapted from S. Shenker

Old Economic Model

Bought HW/SW from single vendor

- Closed control SW, proprietary forwarding HW

HW deployment needs to be all from one vendor

"Vendor lock-in"

New Economic Model

Can buy HW from anyone (theoretically)

- HW becomes interchangeable, if it supports OpenFlow

Can buy SW from anyone

- Runs on controllers, so doesn't need to run on switch

SDN platform sold separately from ctrl apps?

- Would require stable and open platform interface
- Currently a debate within ONF….
- Much less lock in (we hope)

Role of OpenFlow

- **Architecturally: boring**
  - Just a switch specification….zzzzzz
- **Economically: hugely important**
  - Could help break HW vendor lock-in

Source: Adapted from S. Shenker

Changes the Testing Model

- Unit test hardware
  - The forwarding abstraction is well specified
  - Completely separate from control plane
- Use simulator to analyze large-scale control planes
  - External events are input to simulation
  - Can do regression testing
- Much wider testing coverage than is possible today
  - Today use physical testbeds, limited in size/scope

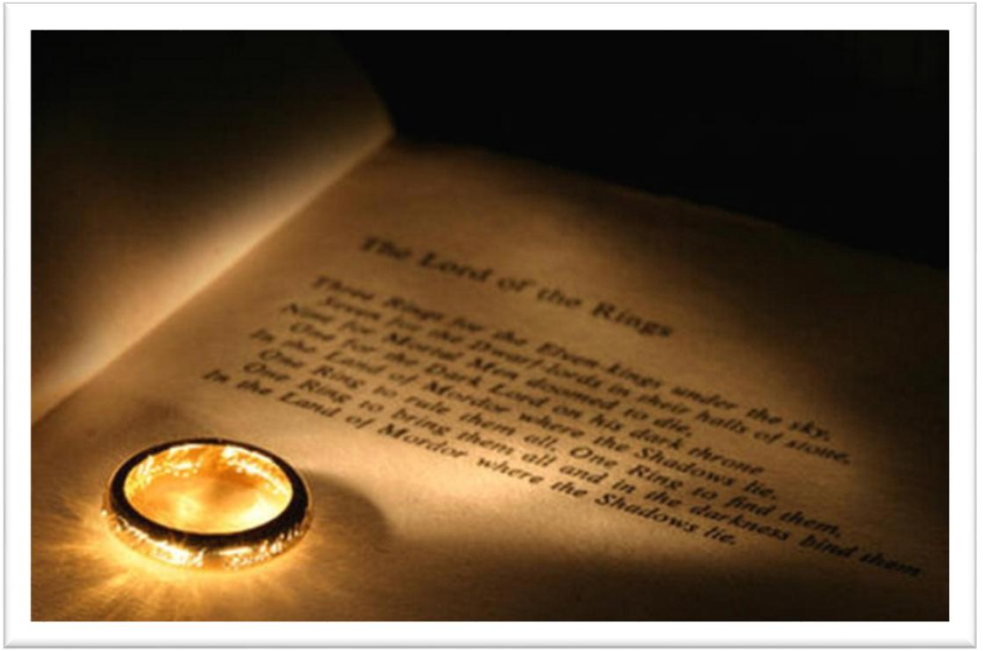
Source: Adapted from S. Shenker

Which Then Changes Deployment…

- Before SDN, upgrades were scary and rare events
  - Code was not well tested (because it was hard to test)
  - And could only come from hardware vendor
- With SDN, upgrades can be easy and frequent
  - Much better testing
  - And the software can come from anyone
- Rate of innovation will be much greater!
  - Old: innovation mainly to get customers to buy new HW
  - SDN: innovation to make customers happier!

Source: Adapted from S. Shenker

One SDN controller to rule them all, with a discovery app to find them,
One SDN controller to tell them all, on which switchport to bind them.
In the Data Center, where the packets fly.



One SDN to rule them all

Actually not, different reasonable models and approaches to SDN are being pursued

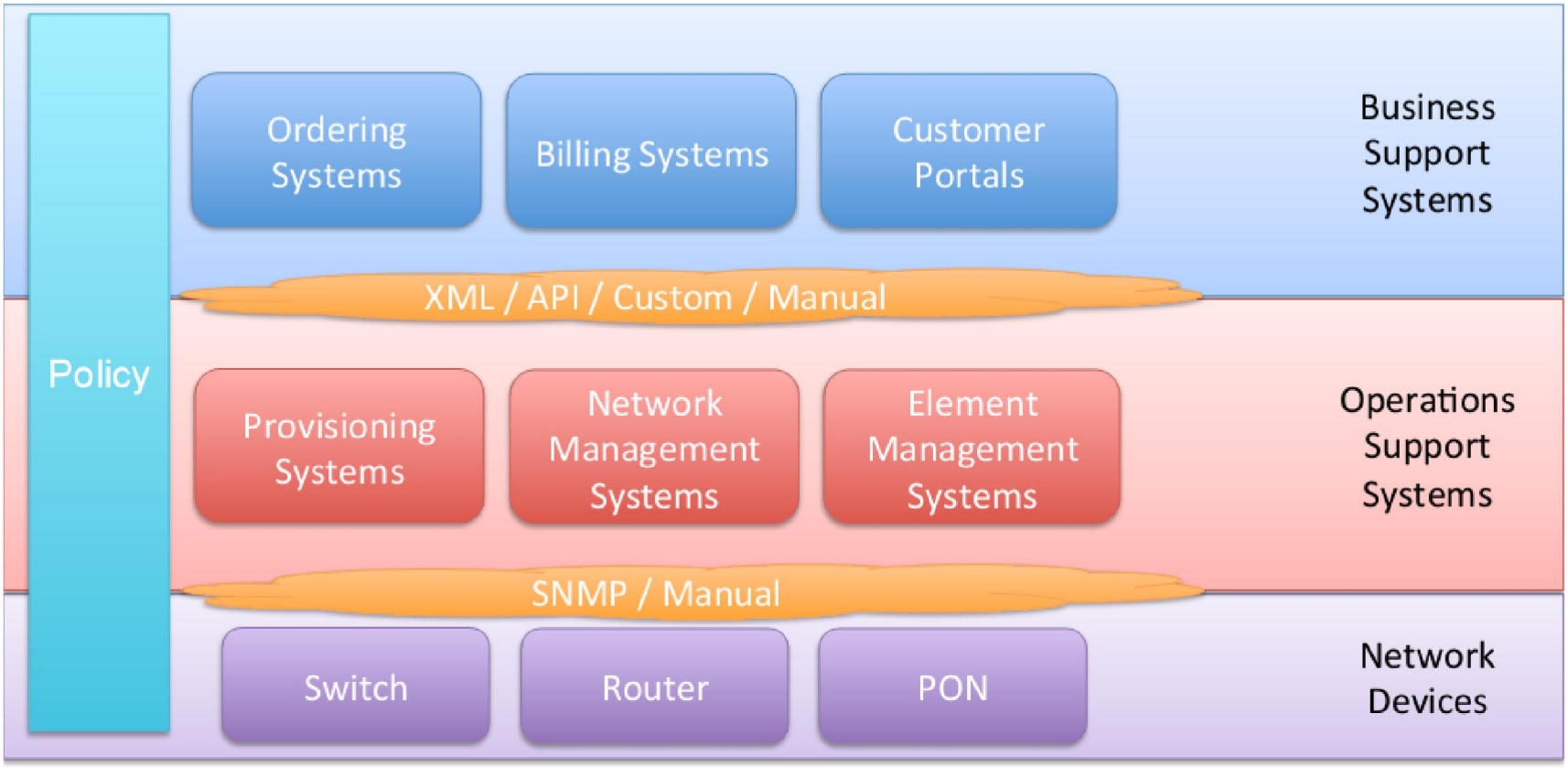
Source Poem: http://dovernetworks.com/?p=83

SDN Models

SDN can be considered in terms of different models:

- **Yesterday's SDN:** Automation and some level or device/network programmability using (vendor- and platform-specific) CLI commands and APIs (for legacy management protocols and systems) on a per-device basis to indirectly affect the network state
- **Canonical/Open SDN:** A **Networking/Operating/System** that oversees the network data plane and hosts a number of "control programs" that implement networking services (e.g. OpenFlow model). Split / Decoupled control plane!
- **Broker SDN:** **An API-driven (software-driven / SDN-augmented) hybrid approach where a broker** interacts which applications to affect the network so that apps are more effective, efficient and/or offer better user experience
- **Proactive / Declarative SDN: Compiler** that translates a high-level language in which an operator defines what they want from the network and compiles it into data plane instructions and configuration
- **Overlay SDN:** An approach where the **network edge** is programmed by an SDN controller to manage **tunnels between hypervisors and/or ToR switches.** Underlay network control plane untouched.

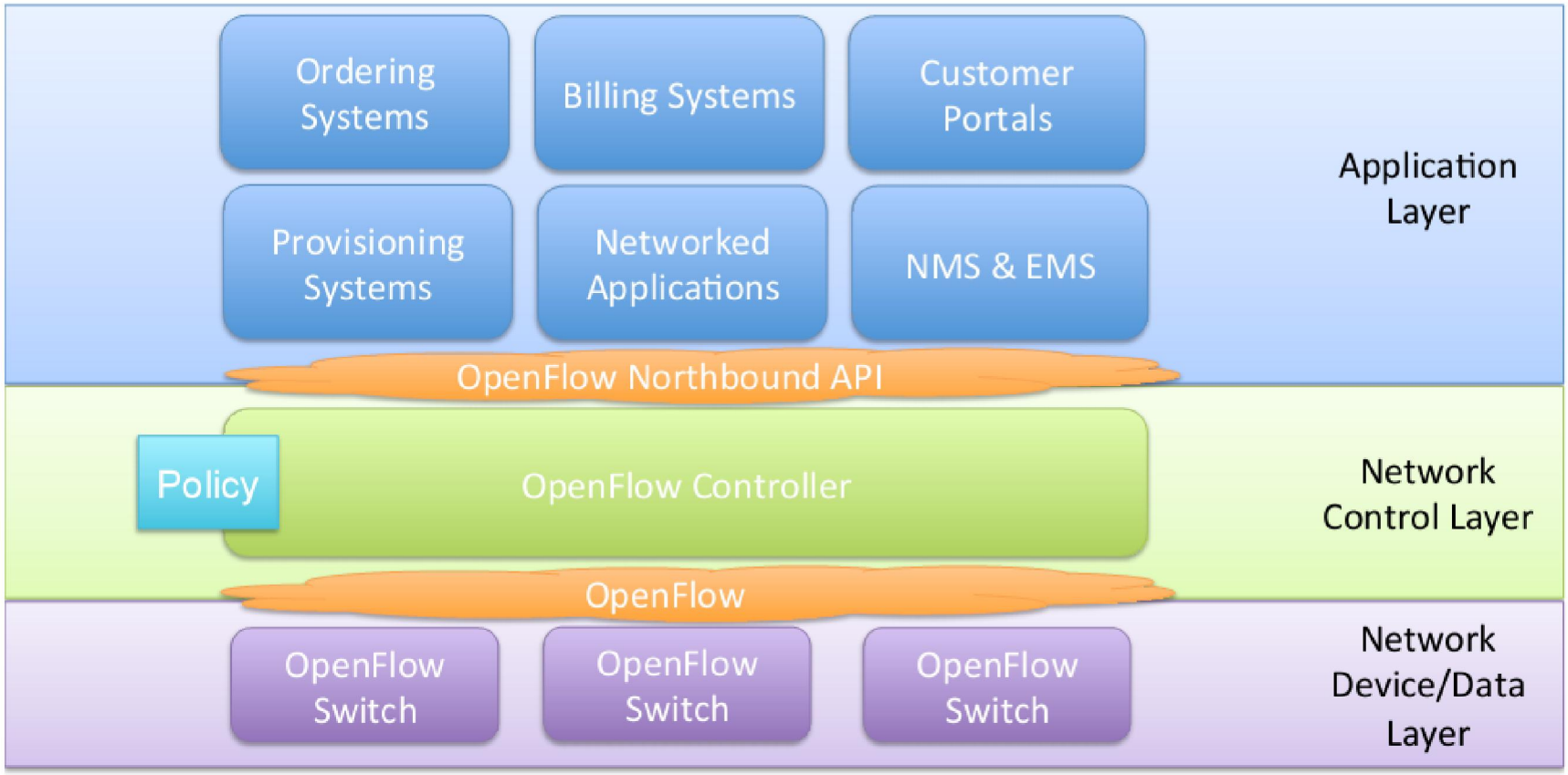
Yesterday´s “SDN”



Source: Chris Grundemann

Canonical / Open SDN with Network OS

e.g. the OpenFlow model



Source: Chris Grundemann

OpenFlow : Not the Only SDN Tool

Tool/Standard	Functionality
OpenFlow (ONF)	FIB/TCAM manipulation
NETCONF (IETF)	Configuration management
OF-Config	OpenFlow switch configuration management (YANG schema)
Internet Routing System (IRS, IETF non-WG)	Routing table interaction/manipulation

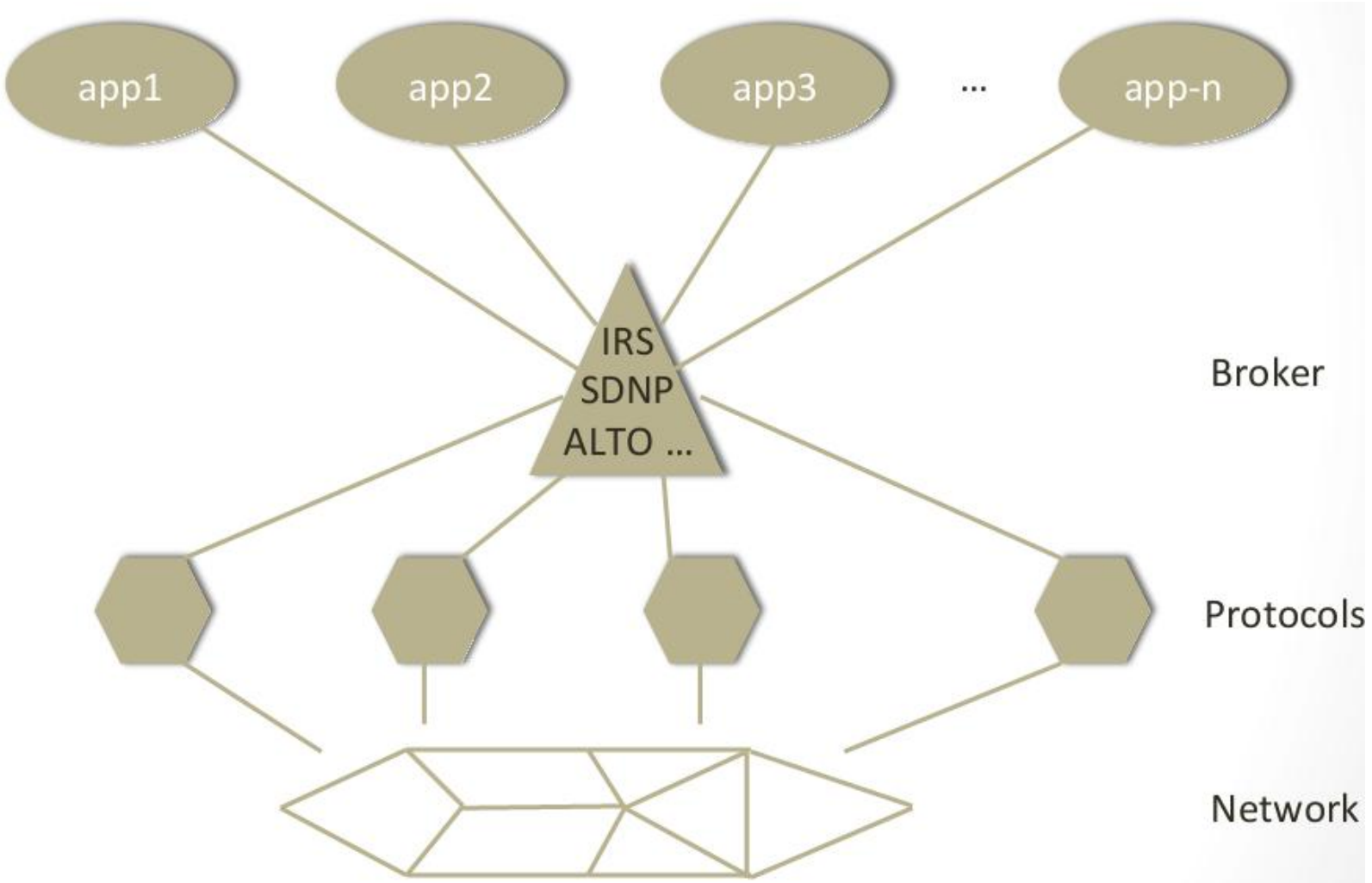
Vendor APIs

- Cisco: Open Networking Environment (ONE), EEM (Tcl), Python scripts)
- Juniper: Junos XML API and SLAX (human-readable XSLT)
- Arista EOS: XMPP, Linux scripting (including Python and Perl)
- Dell Force10: Open Automation Framework (Perl, Python, NetBSD shell)
- F5: iRules (Tcl-based scripts)

Source: I. Pepelnjak

Broker / API-based SDN:

A SW-driven / SDN-augmented / Hybrid approach



Source: K. Kompella, slides-85-sdnrg-2.pdf

Broker / API-based SDN

Example: I2RS (Interface to the Routing System)

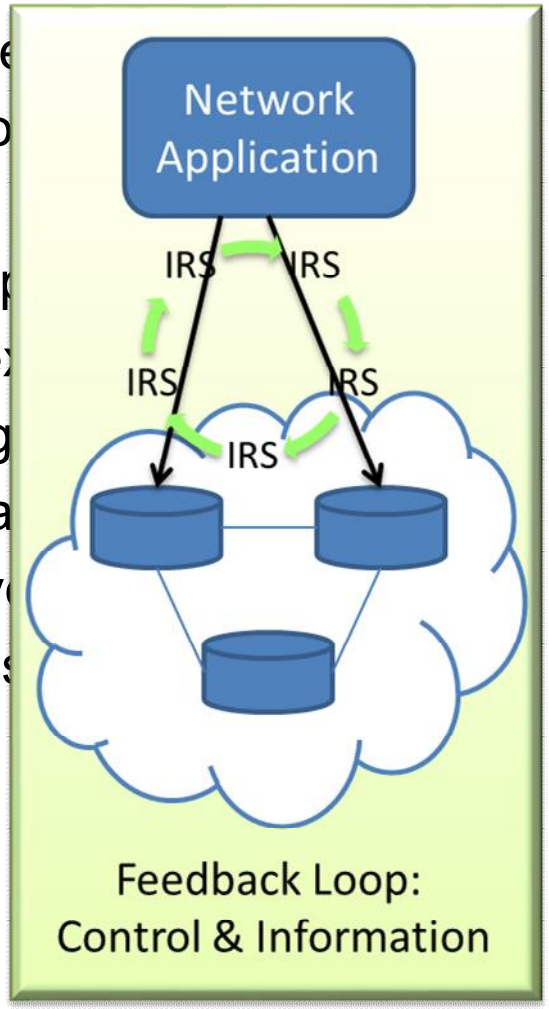
Additional SDN Function

Applications ne

- Augment ro
- Policy
- Flow and ap
- Time and ex

With knowledg

- Topology (a
- Network ev
- Traffic meas
- Etc.



Advanced SDN Use Cases

Programming the Routing Information Base

- For example, adding static routes

Setting routing policy

- Control how the FIB is built

Other router policies

- Modify BGP import/export policies

Topology extraction

- Pull routing information (including SRLGs) from network

Topology management

- Create virtual links by making connections ir lower layers

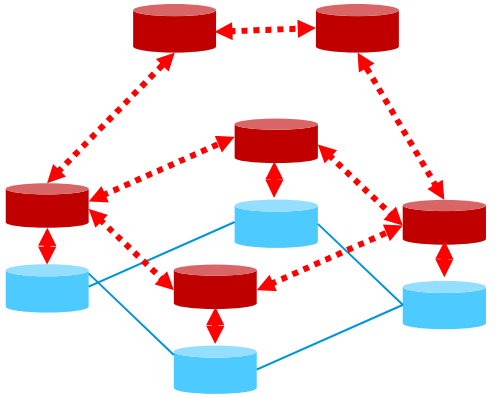
Service management

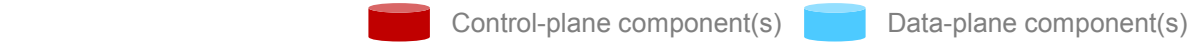
- Request LSPs, connections, pseudowires
- Bandwidth scheduling
- "Set up a VPN"

Source: Adrian Farrel

Broker / API-based SDN
Software-driven Networks

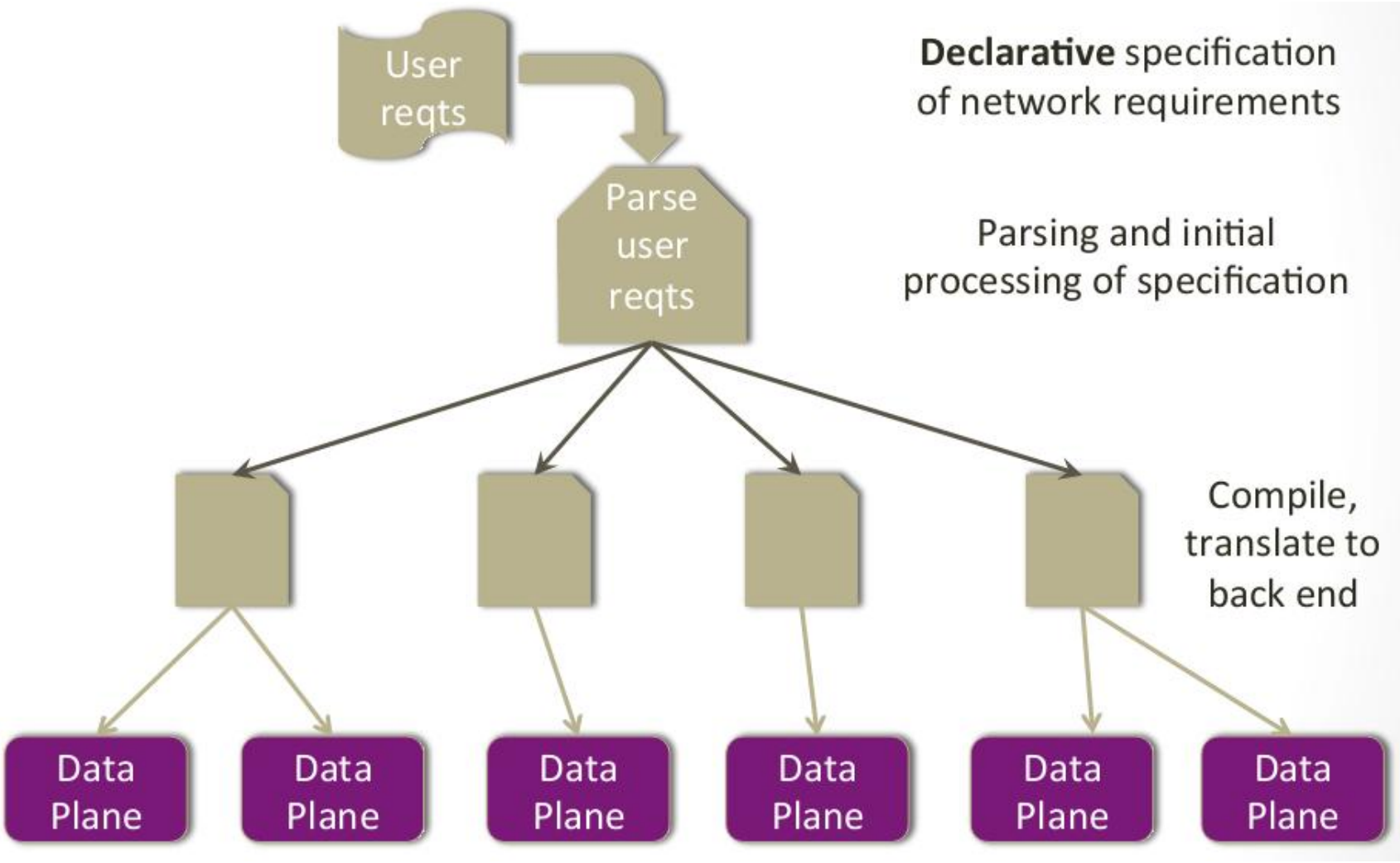
- More than play on words :
  - some in the industry refer to SDN as *software-driven networks*, as opposed to software-defined networks.
- Rather than viewing the network as being comprised of logically centralized control planes with brainless network devices,
  - one views the world as more of a hybrid of the old and the new
- Hybrid approach
  - some portion of networks operated by a logically centralized controller,
  - other parts would be run by the more traditional distributed control plane





Proactive / Declarative SDN

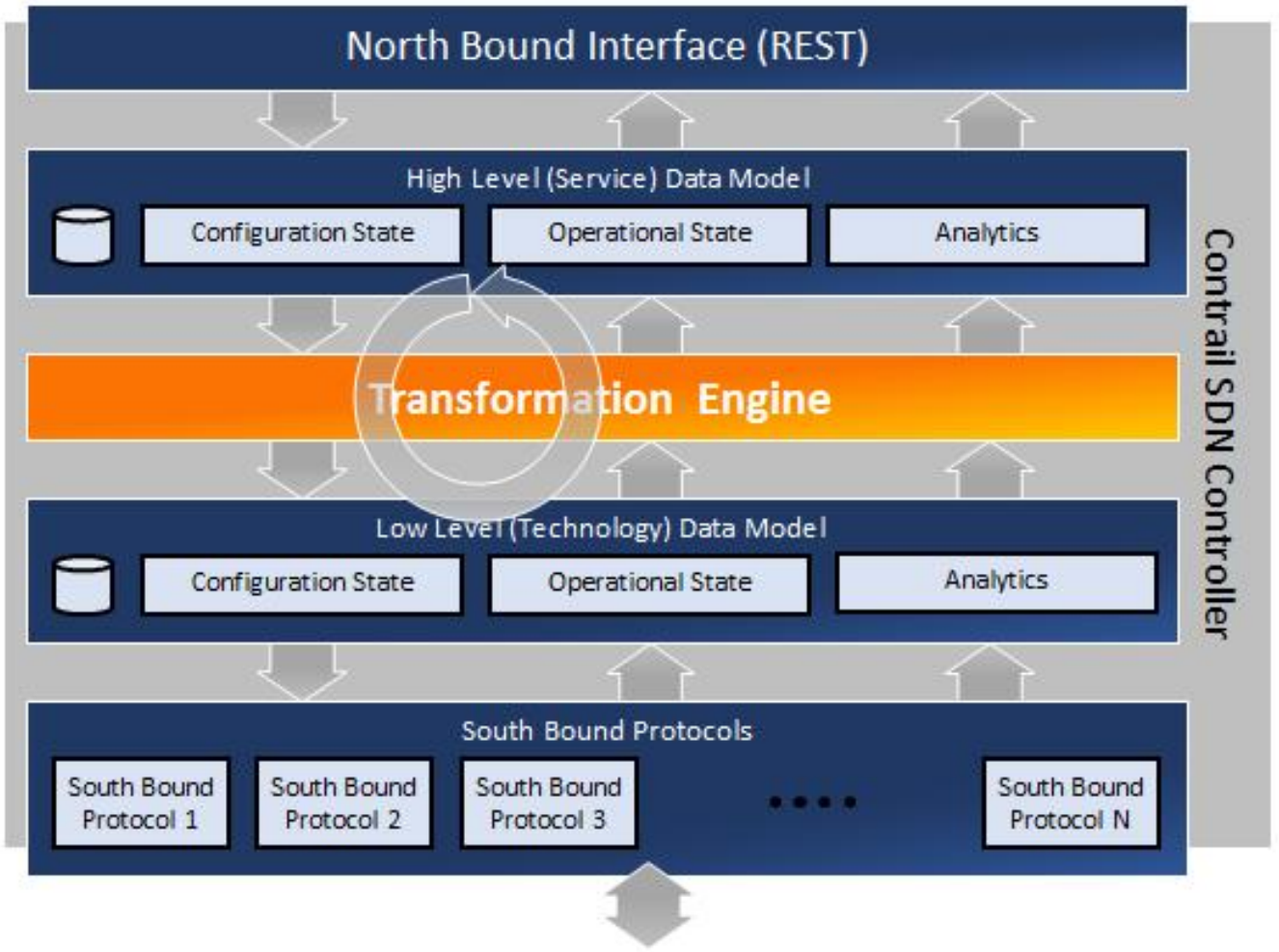
Model : Compiler



Source: K. Kompella, slides-85-sdnrg-2.pdf

Compiler Model: Example

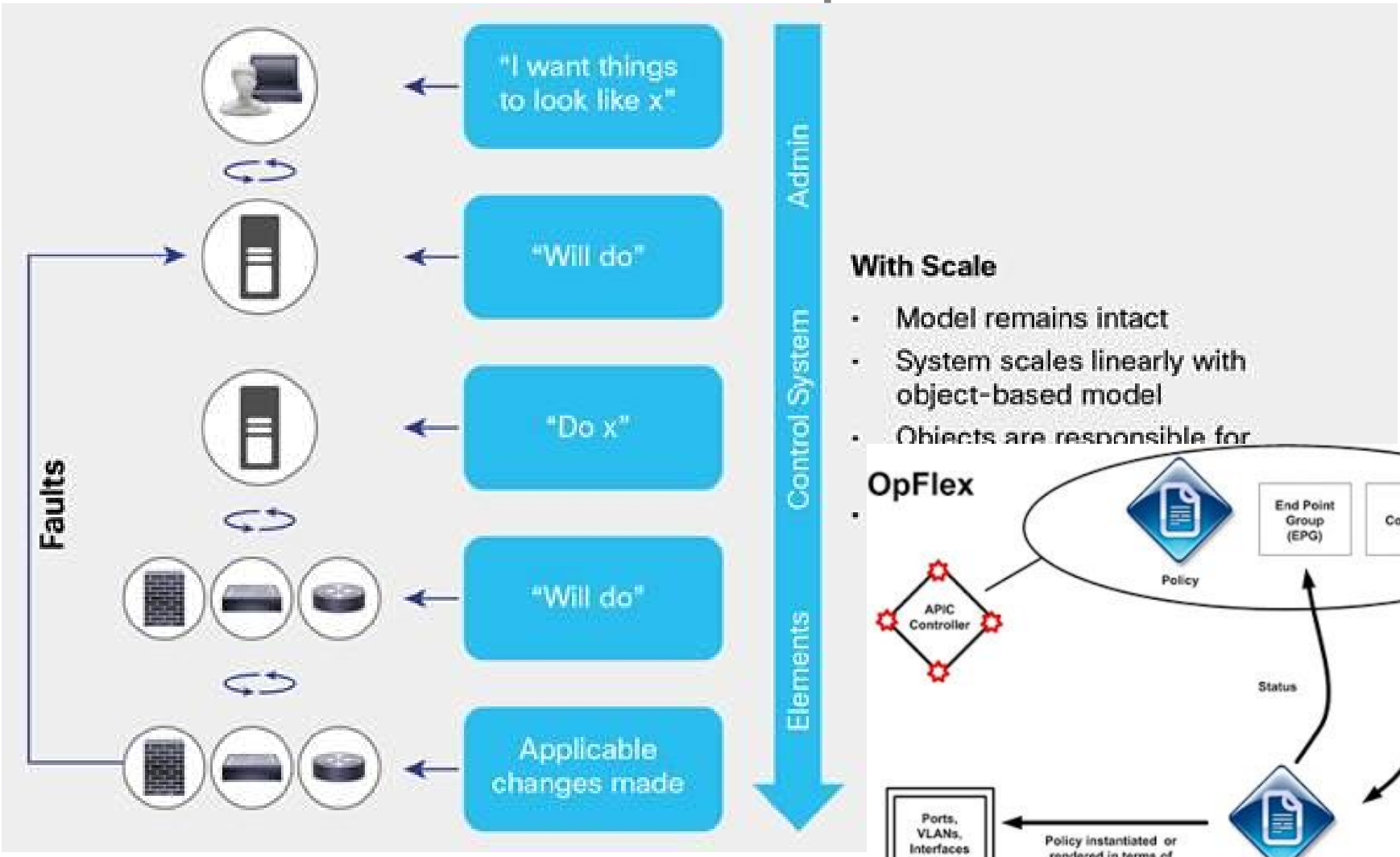
Juniper Centralized Configuration Management



Source: http://opencontrail.org/the-importance-of-abstraction-the-concept-of-sdn-as-a-compiler/

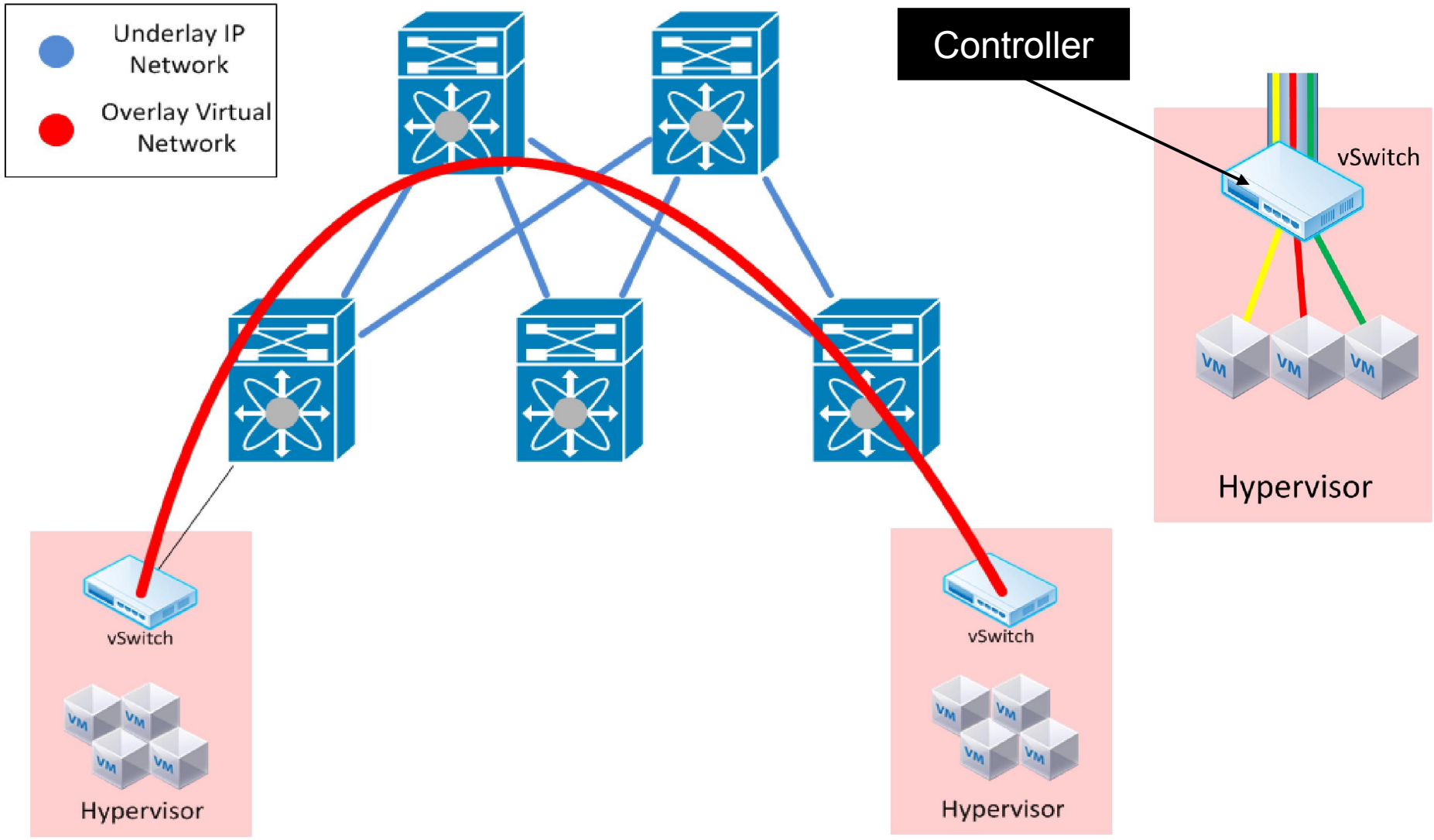
Compiler Model: Example

Cisco OpFlex



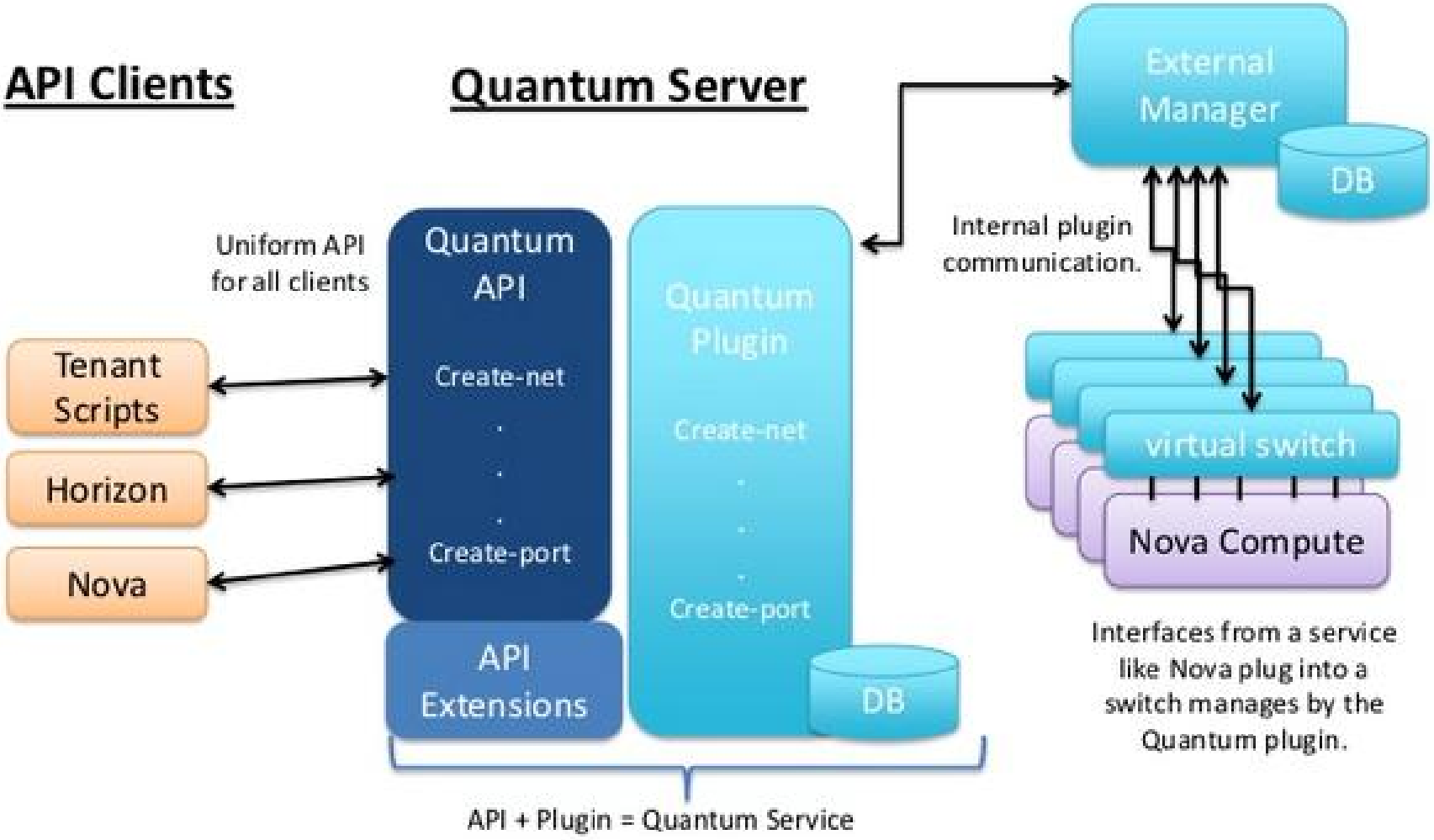
Overlay SDN

Controller-Defined Edge Tunnels

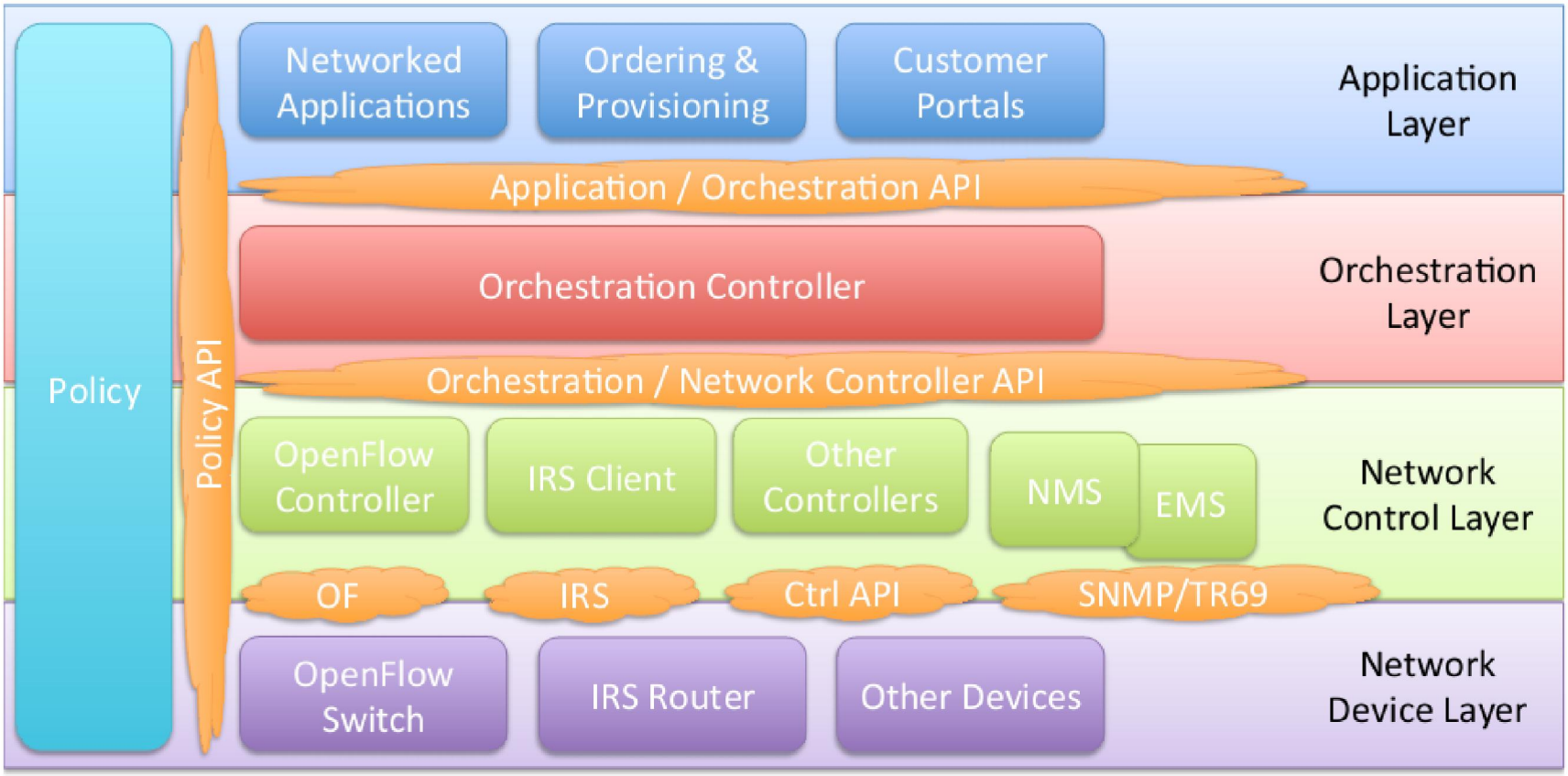


Overlay SDN

Example Integration with OpenStack

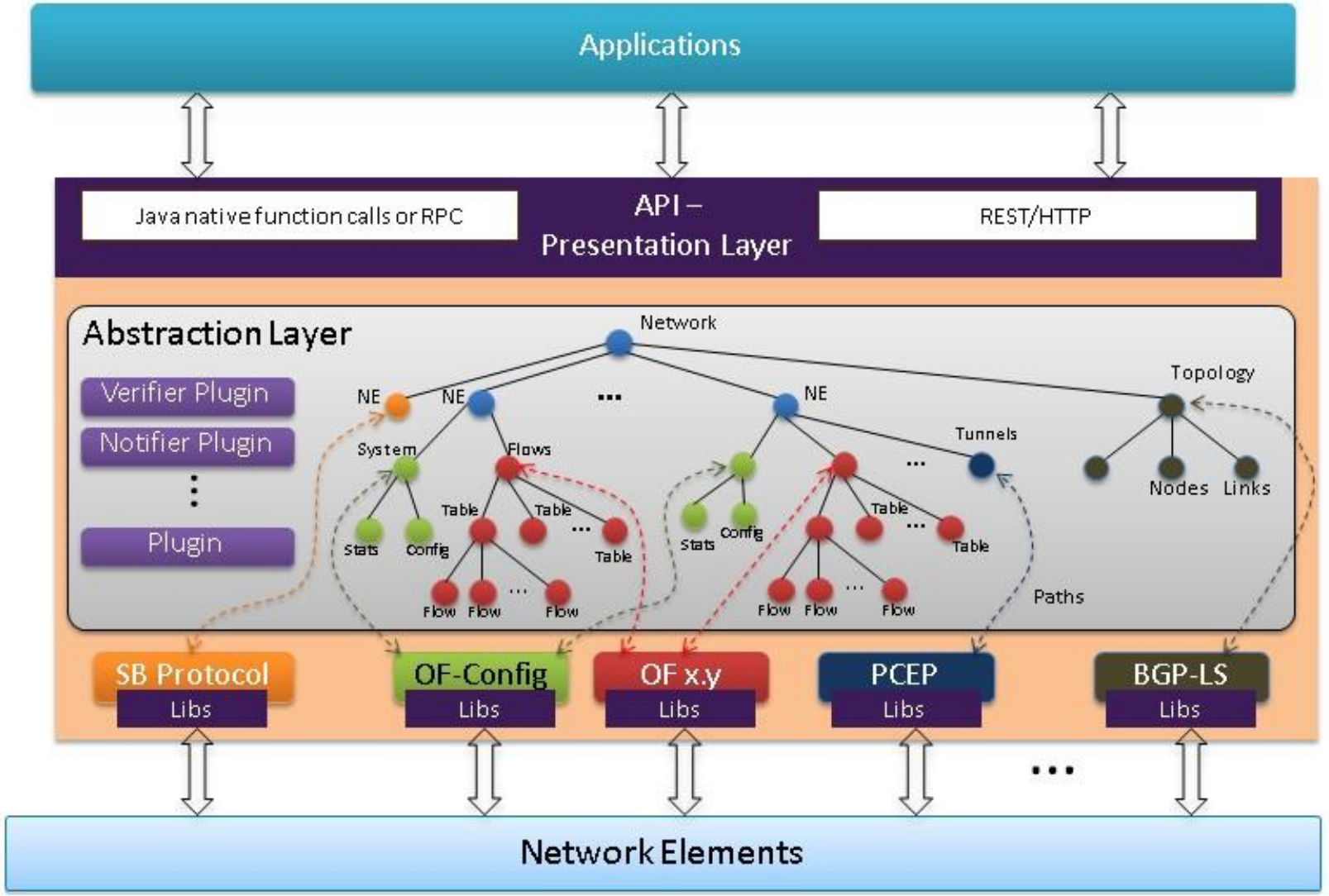


Evolving SDN Ecosystem



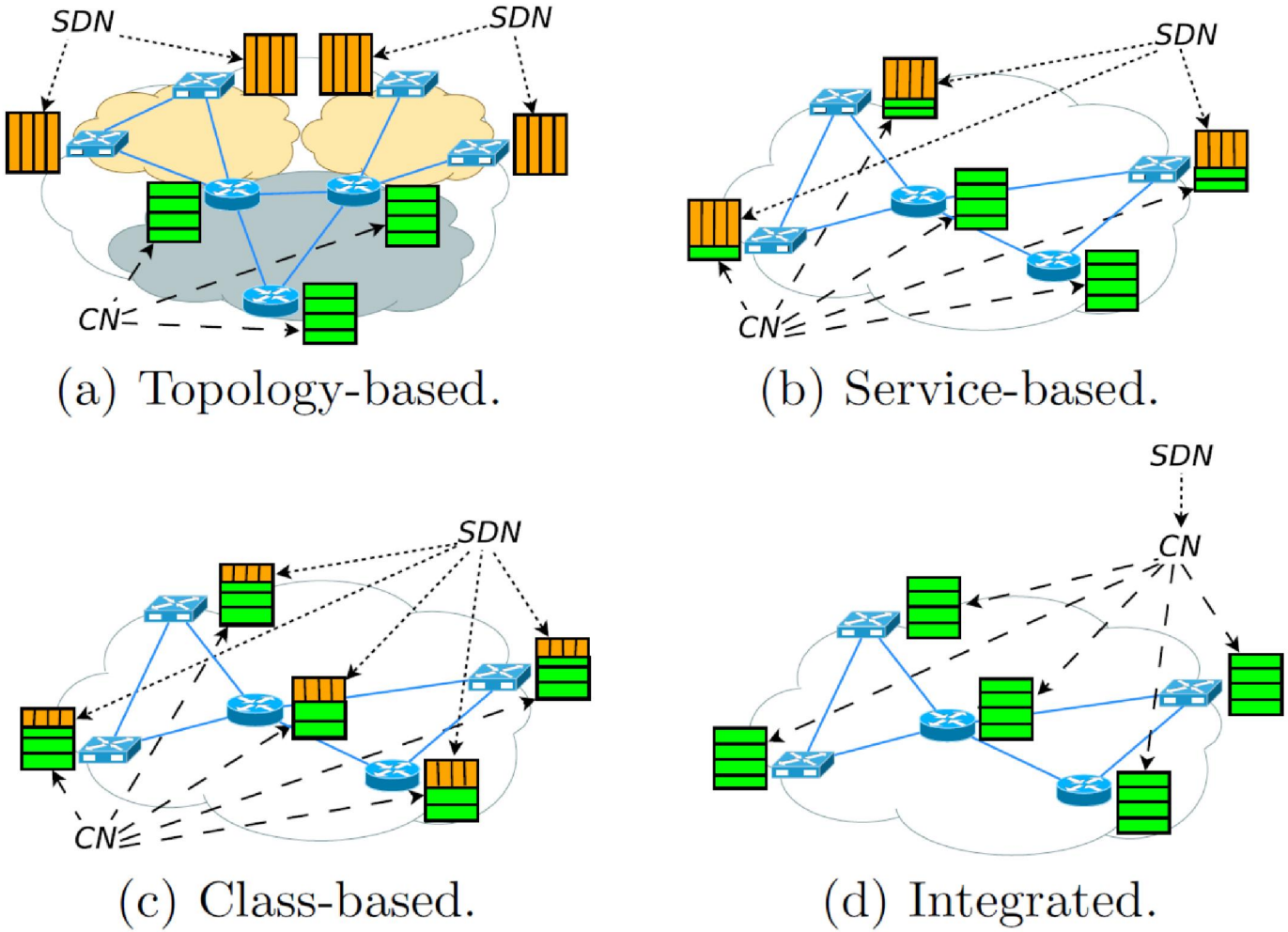
Source: Chris Grundemann

OpenDaylight Controller Platform



Source: opendaylight.org

Hybrid Networking in SDN



Source: S. Vissicchio et al. Opportunities and Research Challenges of Hybrid Software Defined Networks. In ACM Computer Communication Review, 44(2), April 2014.

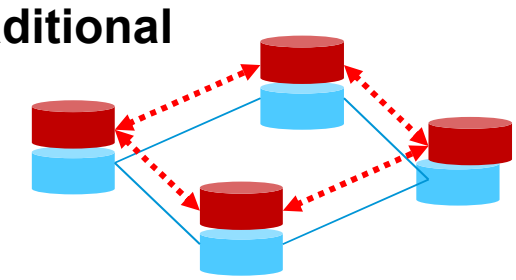
Trade-offs of Hybrid Networking in SDN

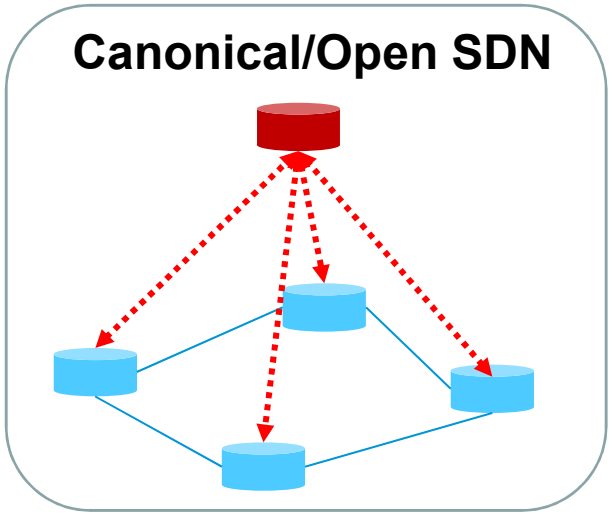
	CN	TB hSDN	SB hSDN	CB hSDN	Integrated	SDN
non IP-based forwarding	hard, complex configuration	programmable in SDN zones	programmable for SDN services	programmable for SDN traffic	very hard (e.g., BGP FlowSpec)	programmable
traffic steering, middleboxing	hard (e.g., box replication)	programmable in SDN zones	programmable for SDN services	programmable for SDN traffic	programmable by the controller	programmable
scalability and robustness	by CN protocols	by CN protocols in CN zones	by CN protocols for CN services	by CN protocols for CN traffic	possibly, by CN protocols	SDN controller concern
required custom software	none	controllers of SDN zones	controllers for SDN services	controllers for SDN flows	SDN controller	SDN controller
upgrade costs (hw, sw, expert)	none	partial, progressive renovation	partial, progressive renovation	partial, progressive renovation	none	global renovation
paradigm interaction	none	control-plane collaboration	data-plane visibility	control-plane coordination	control-plane integration	none

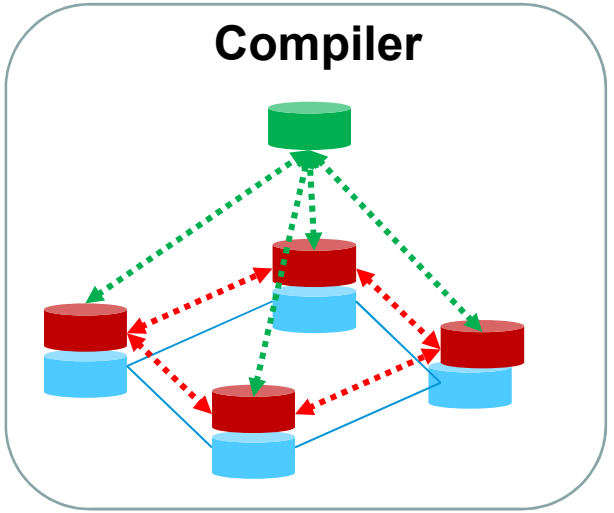
- **Tradeoff analysis suggests that the combination of centralized and distributed paradigms can provide mutual benefits.**
- Future work is needed to devise techniques and interaction mechanisms that maximize such benefits while limiting the added complexity of the paradigm coexistence.
- **Combination of hybrid models:** A wider range of tradeoffs can be obtained by combining hybrid models together.

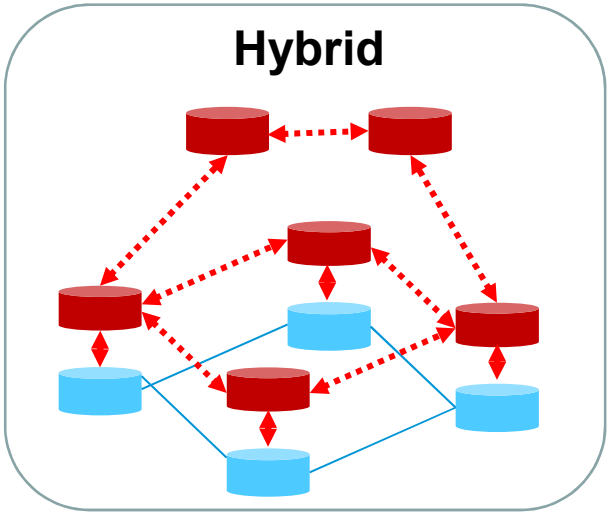
Source: S. Vissicchio et al., CCR 14

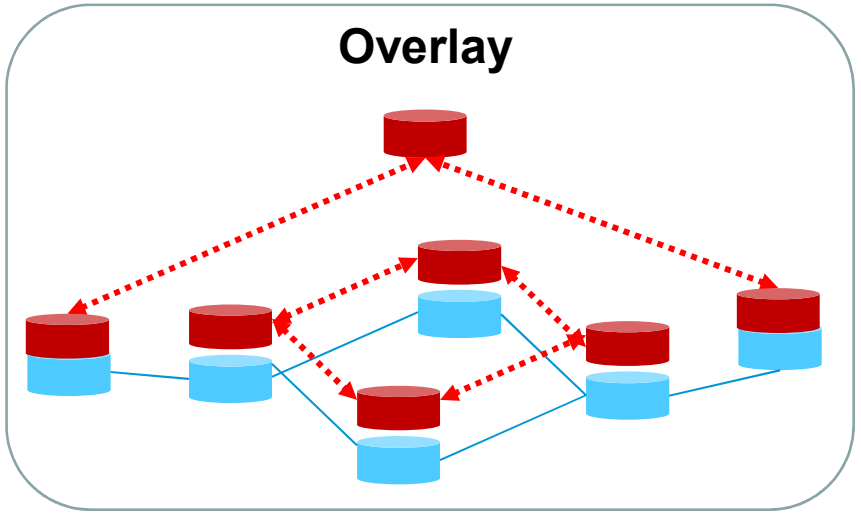
Comparing Models



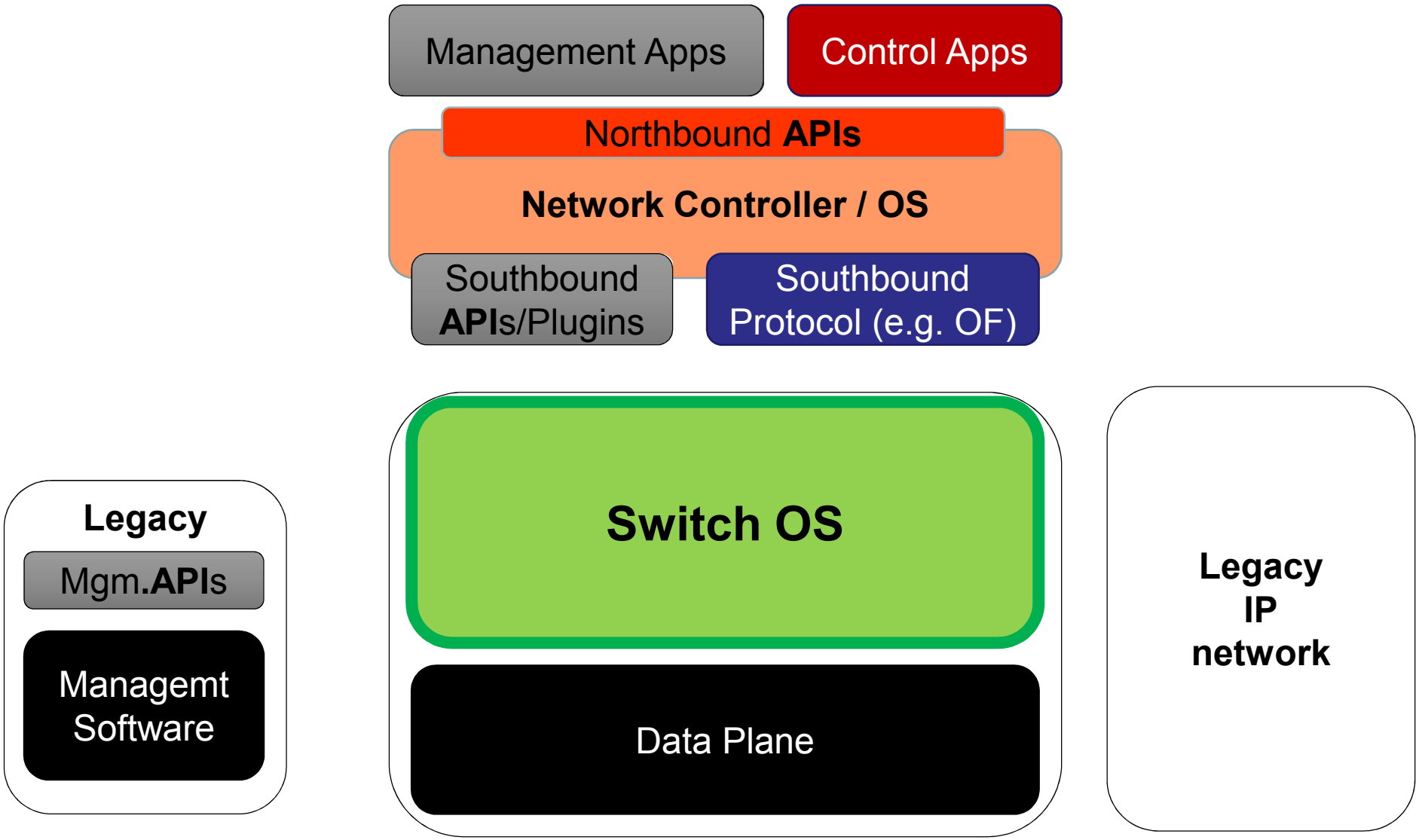








Comparing different SDN Models



Yet another model emerging:
Bare Metal SDN

WHAT'S INSIDE A SWITCH?

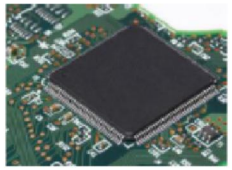
Application

Network OS

Hardware Driver

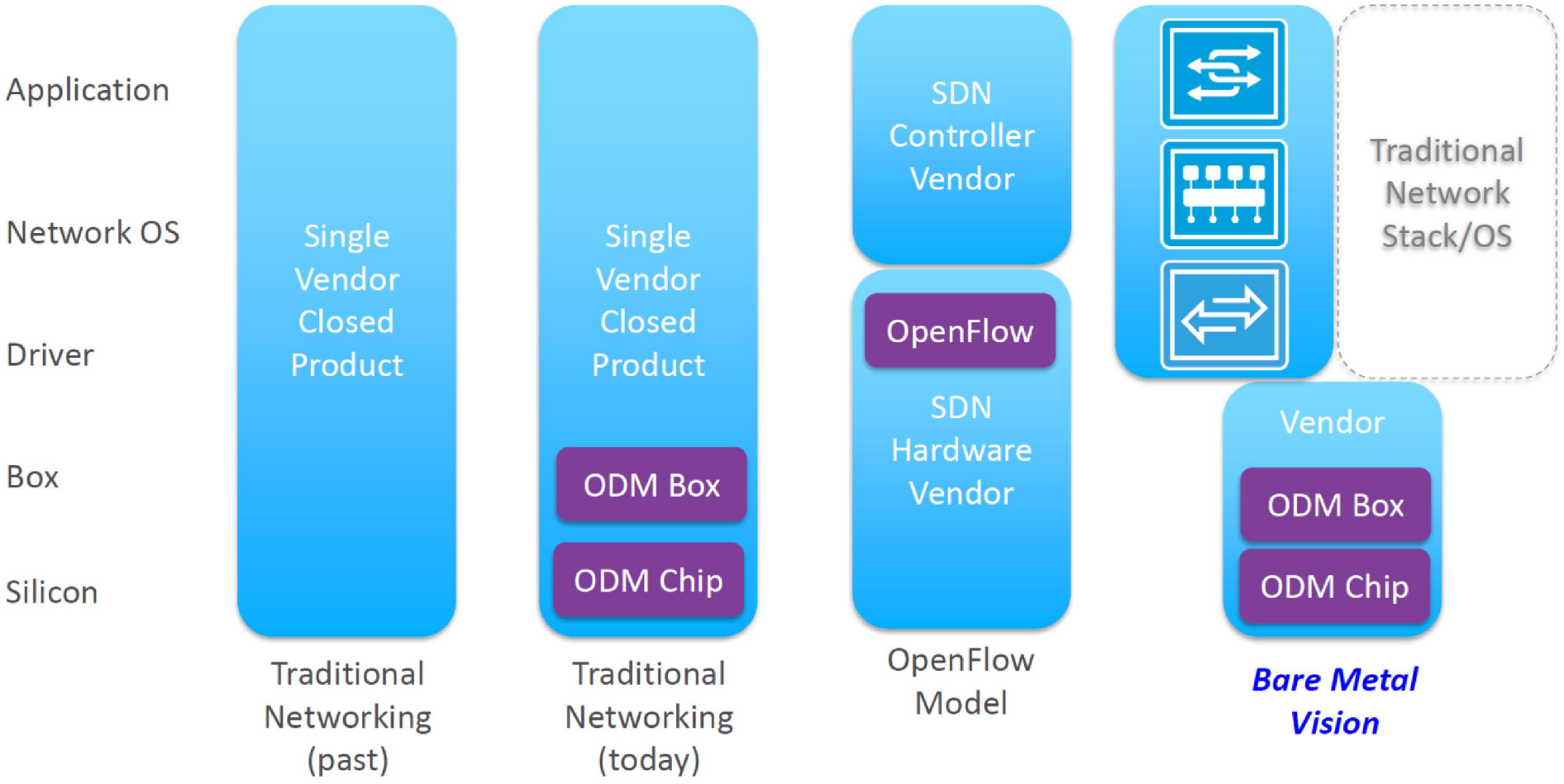
Box

Silicon

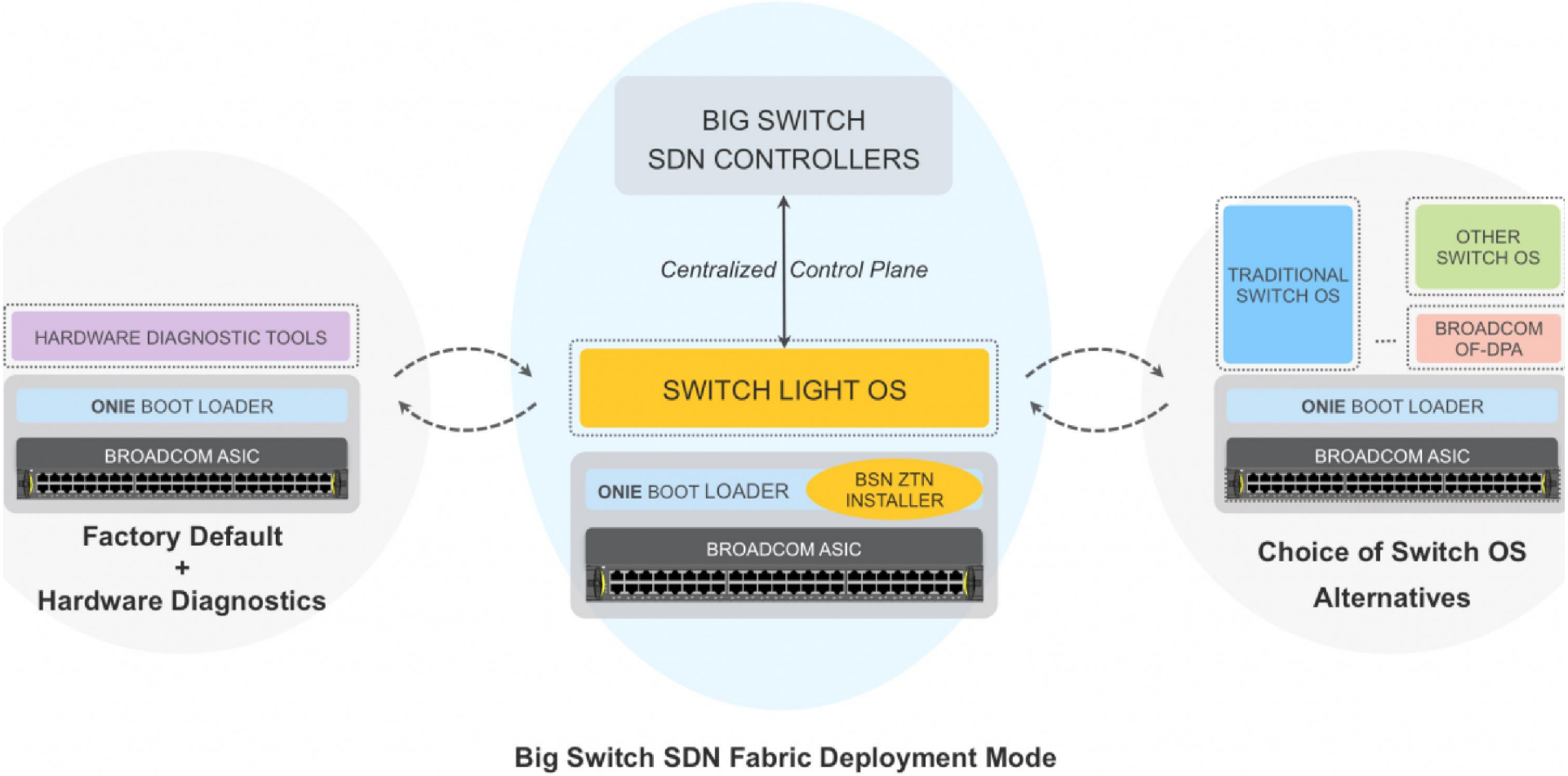


Source: Rob Sherwood, Big Switch Networks

COMPONENT ECOSYSTEM & BARE METAL SDN



Bare Metal Switches: Choice of Switch OS

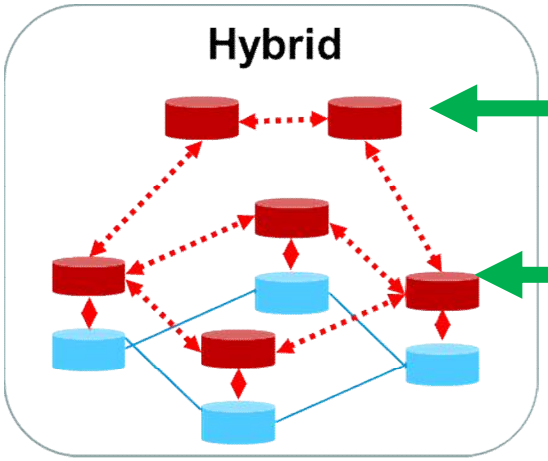


Big Switch SDN Fabric Deployment Mode

Source: Rob Sherwood, Big Switch Networks

Bare Metal Switching and Programming

Blows Up the SDN Blob!



Out-of-the-box

Programmable Control App

In-the-box

app app app

operating system

hardware

Single Vendor Blob

app app app

operating system

hardware

Multi Vendor Ecosystem

route monitor

operating system

hardware

Openflow Agent

Minimal OS

hardware

Source: Adapted from Shrijeet Mukherjee, Cumulus Networks

“OpenFlow” Future

From a protocol to a model perspective

A view on ONF evolving attitude[1]

Early attitude in the ONF: Me! Me!

Revolution!

- "My feature, my feature!", and "More flex! Optical! Wireless! …! "
- Fully Programmable Dataplane!
- Protocol independent generic commands! (Byte offset, bit mask, etc)
- "New chips will come and it will all be good!" – But…

Growing attitude: We need this stuff to work

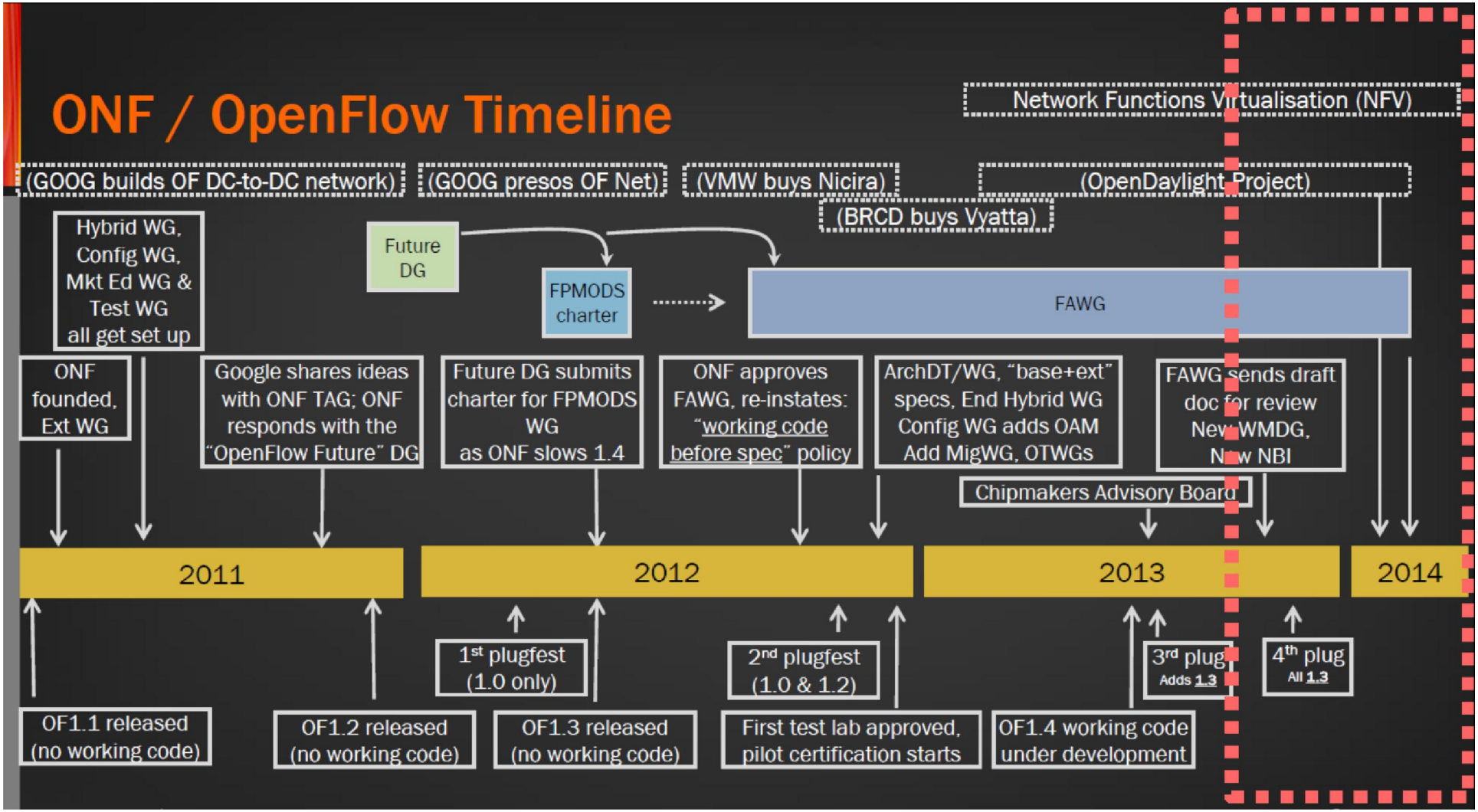
- Now, lots of OF1.3 capable boxes (Nov'13 plugfest)
  - don't work together that well…
  - how do I code (or test) using optional features?

Reality strikes!

- Then the CAB formed (responsibility: nudge chipmakers!)
  - Chip Advisory Board explained how to get chipmakers to make new chips (== biz case)

Business case (still) dominates!

[1] Adapted from: Curt Beckmann, Brocade, CURT'S ONF UPDATE, NFD7



Slide courtesy: Curt Beckmann, Brocade

In the Beginning…

- OpenFlow was simple
- A single rule table
  - Priority, pattern, actions, counters, timeouts
- Matching on any of 12 fields, e.g.,
  - MAC addresses
  - IP addresses
  - Transport protocol
  - Transport port numbers

Over the Past Five Years…

- Proliferation of header fields

Version	Date	# Headers
OF 1.0	Dec 2009	12
OF 1.1	Feb 2011	15
OF 1.2	Dec 2011	36
OF 1.3	Jun 2012	40
OF 1.4	Oct 2013	41

- Multiple stages of heterogeneous tables
- Still not enough
  (e.g., VXLAN, NVGRE, STT, …)

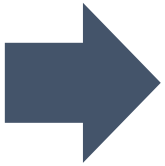
Source: P4, http://arxiv.org/abs/1312.1719

Device / OF Option Alignment

	ASIC / Merchant Si	FPGA	NPU	GP Server
OFS1.0	Doable, limited func, many products	Doable, limited func, many products	Doable, limited func, some products	Doable, limited func, some products
OFS1.0+ext	"Depends" on the extension! Some products	Should work, but still depends. Some products	Doable, better functionality, products?	Doable, better functionality, OpenVSwitch
Single Table OFS1.3	Not too hard, better func, some prods in development	Not too hard, better func, some prods in dev	(See below)	(See below)
Multi-Table OFS1.3	Very hard[2]	Very hard[2]	Doable, offers very flexible DP forwarding	Doable, offers very flexible DP forwarding

Source: Curt Beckmann, Brocade

Mapping low level instructions when pipelines differ



```
Table0  00  Count ← 16
        01  Prod ← 0
        02  Bit ← 1
        03  If (RegX & Bit == 0) goto 05
Table1  04  Prod += RegY
        05  Bit <<= 1
        06  RegY <<= 1
Table2  07  Count -= 1
        08  If (Count != 0) goto 03
        [Or: If (Bit != 0) or (RegY != 0) ]

Table3
```

```
00  Prod ← RegX * RegY
```

A smart compiler can see it's a "multiply"

As long as it can see the complete set of code

But if the code is in scattered in time?

If we ask the compiler to do the translation piecemeal, it becomes impossible

Similarly, mapping multi-table OF to legacy ASICs is tricky or worse… if we must do it **all** at run-time

But we actually don't have to do it ALL at run-time

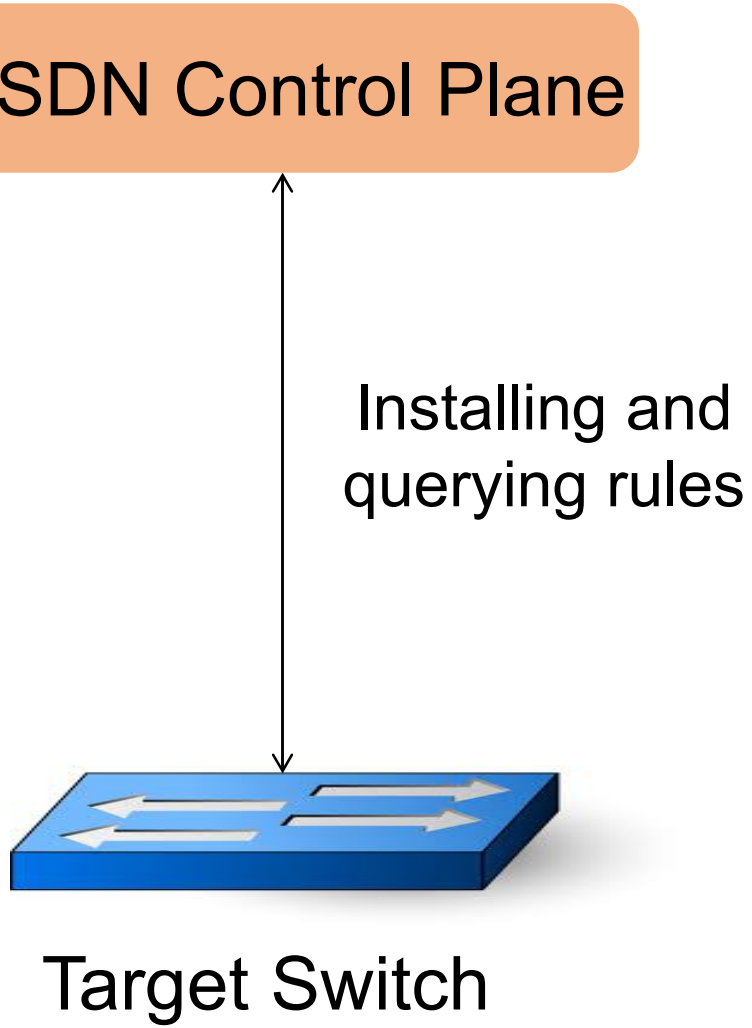
Source: Curt Beckmann, Interoperable OpenFlow with NDMs and TTPs

Future SDN Switches

- Where does it stop?!?
  - Simplicity would be nice, but it just isn't practical
- Configurable packet parser
  - Not tied to a specific header format
- Flexible match+action tables
  - Multiple tables (in series and/or parallel)
  - Able to match on all defined fields
- General packet-processing primitives
  - Copy, add, remove, and modify
  - For both header fields and meta-data

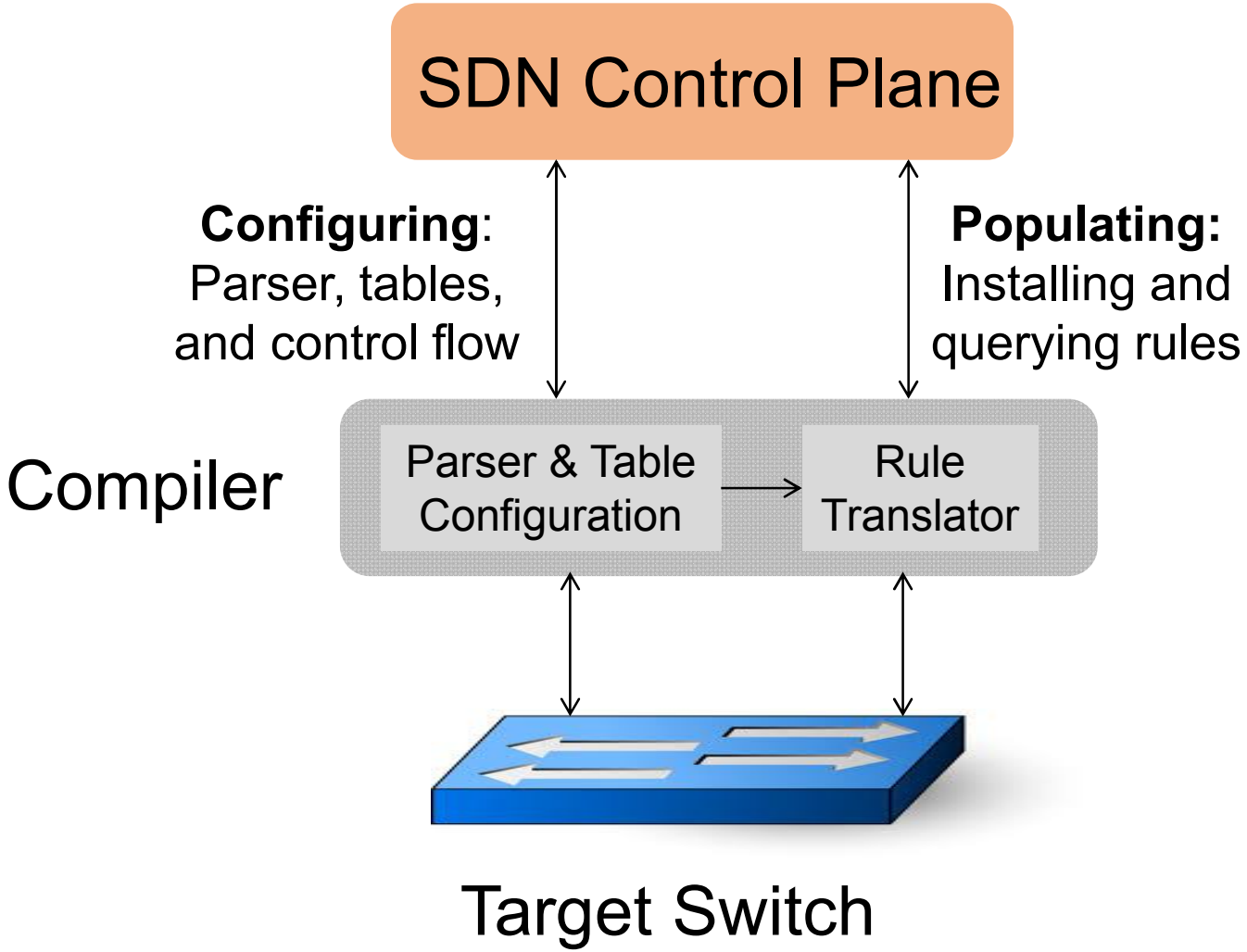
Source: P4, http://arxiv.org/abs/1312.1719

“Classic” OpenFlow (1.x)



Source: P4, http://arxiv.org/abs/1312.1719

“OpenFlow 2.0”



Source: P4, http://arxiv.org/abs/1312.1719

Three Goals

1. Protocol independence
   - Configure a packet parser
   - Define a set of typed match+action tables
2. Target independence
   - Program without knowledge of switch details
   - Rely on compiler to configure the target switch
3. Reconfigurability
   - Change parsing and processing in the field

Source: P4, http://arxiv.org/abs/1312.1719

P4 Language

Programming Protocol-Independent Packet Processing

P4 Compiler

- Parser
  - Programmable parser: translate to state machine
  - Fixed parser: verify the description is consistent
- Control program
  - Target-independent: table graph of dependencies
  - Target-dependent: mapping to switch resources
- Rule translation
  - Verify that rules agree with the (logical) table types
  - Translate the rules to the physical tables

Source: Programming Protocol-Independent Packet Processors, http://arxiv.org/abs/1312.1719

Ongoing efforts towards an alternate OF+

- OpenFlow 1.x
  - Vendor-agnostic API. But, only for fixed-function switches
- An alternate future?
  - Protocol independence
  - Target independence
  - Reconfigurability in the field
- P4 language: a straw-man proposal
  - To trigger discussion and debate. Much, much more work to do!
- Related Work
  - Abstract forwarding model for OpenFlow
  - Kangaroo programmable parser
  - NOSIX portability layer for OpenFlow
  - Protocol-oblivious forwarding (POF) by Huawei
  - OpFlex by Cisco ?
  - **Table Type Patterns in ONF FAWG**

ONF FAWG work on Table Type Patterns (TTP)

Defining Datapath Models in advance

- "Datapath Model" must be detailed, unambiguous
  - Must spell out matches and actions allowed in each table
    - So no "pipeline surprises" at run time
- Apps will have different needs…no single DP model will work
- So, a range of Datapath Models
  - Powerful platforms might support more than one model
  - Some apps may work on more than one model
  - Models need not be specified by ONF, others can do it too
- App and switch must agree on same model
  - A multi-vendor ecosystem means sharing → common language
  - "Agree" means synching up… "negotiation"
  - "Negotiable Datapath Model" → NDM
  - Must evolve over time as OF evolves

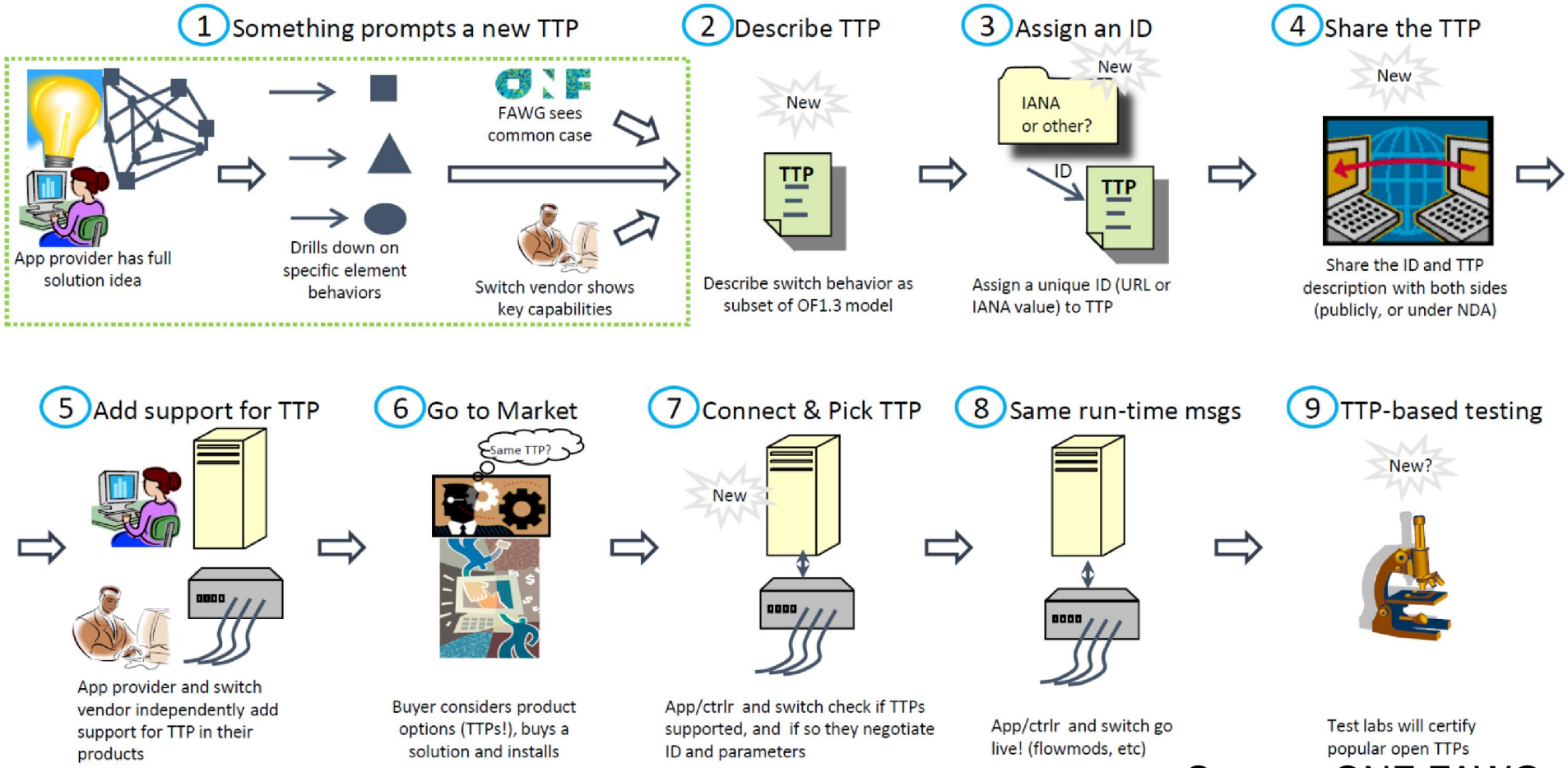
Source: ONF FAWG

How TTPs Can Help

- TTPs are “Table Type Patterns” that market participants can define
  - TTPs are 1st gen of “Negotiable Datapath Models” (NDMs)
- TTPs = “pre-baked pipelines” specific switch funcs in OF1.x terms
  - With TTPs, pipelines can be mapped before run-time
  - Switches, controllers become deterministic (as they need to be)
    - Once TTP is agreed, Controller uses only TTP messages, Switch supports all TTP messages, All messages are valid OF1.x messages
- TTP Examples:
  - “VID Mapping L2 Switch”, “VXLAN Gateway”, “NVGRE Gateway”, “v4 Router w Ingress ACL”, “v6 Router w Egress ACL”, “MPLS Edge & Core Router”
- TTPs help sort out interoperability
  - Product sheets list supported TTPs, clarifies what works with what

Source: ONF FAWG

TTP “Lifecycle”`



Source: ONF FAWG

TTP Benefits

- Ease of development within a context of diversity
- Done such that interoperability is deterministic
- Interoperability visible to market participants
- No logjams required by “standardized profiles”
- Framework is for products that are “TTP aware”
  - Key for determinism when multiple flow tables needed
  - But TTPs also turned out quite useful for single tables!
    - TTPs can serve as precise test profiles
    - Can resolve the “optional feature” challenge
    - Visible to market participants

Source: ONF FAWG

SDN asks (at least) three major questions

Where the control plane resides "Distributed vs Centralized" ?

How does the Control Plane talk to the Data Plane ?

How are Control and Data Planes programmed ?

Source: T. Nadeu, slides-85-sdnrg-5.pptx

SDN asks (at least) three major questions

Where the control plane resides "Distributed vs Centralized" ?

1

- **What state belongs in distributed protocols?**
- **What state must stay local to switches?**
- **What state should be centralized?**

•What are the effects of each on:

- state synchronization overhead
- total control plane overhead
- system stability and resiliency
- efficiency in resource use
- control loop tightness

Source: E. Crabbe, slides-85-sdnrg-7.pdf

SDN asks (at least) three major questions

How does the Control Plane talk to the Data Plane ?

2

- **Prop. IPC**
- **OpenFlow (with or w/extensions)**
- **Open Source south-bound protocols**
- **Via SDN controller broker and south-bound plug-ins**
- **Other standardized protocols**

•What are the effects of each on:

- Interoperability, Evolvability, Performance
- Vendor Lock-in

SDN asks (at least) three major questions

How are Control and Data Planes programmed ?

3

- **Levels of Abstraction**
- **Open APIs**
- **Standardized Protocols**

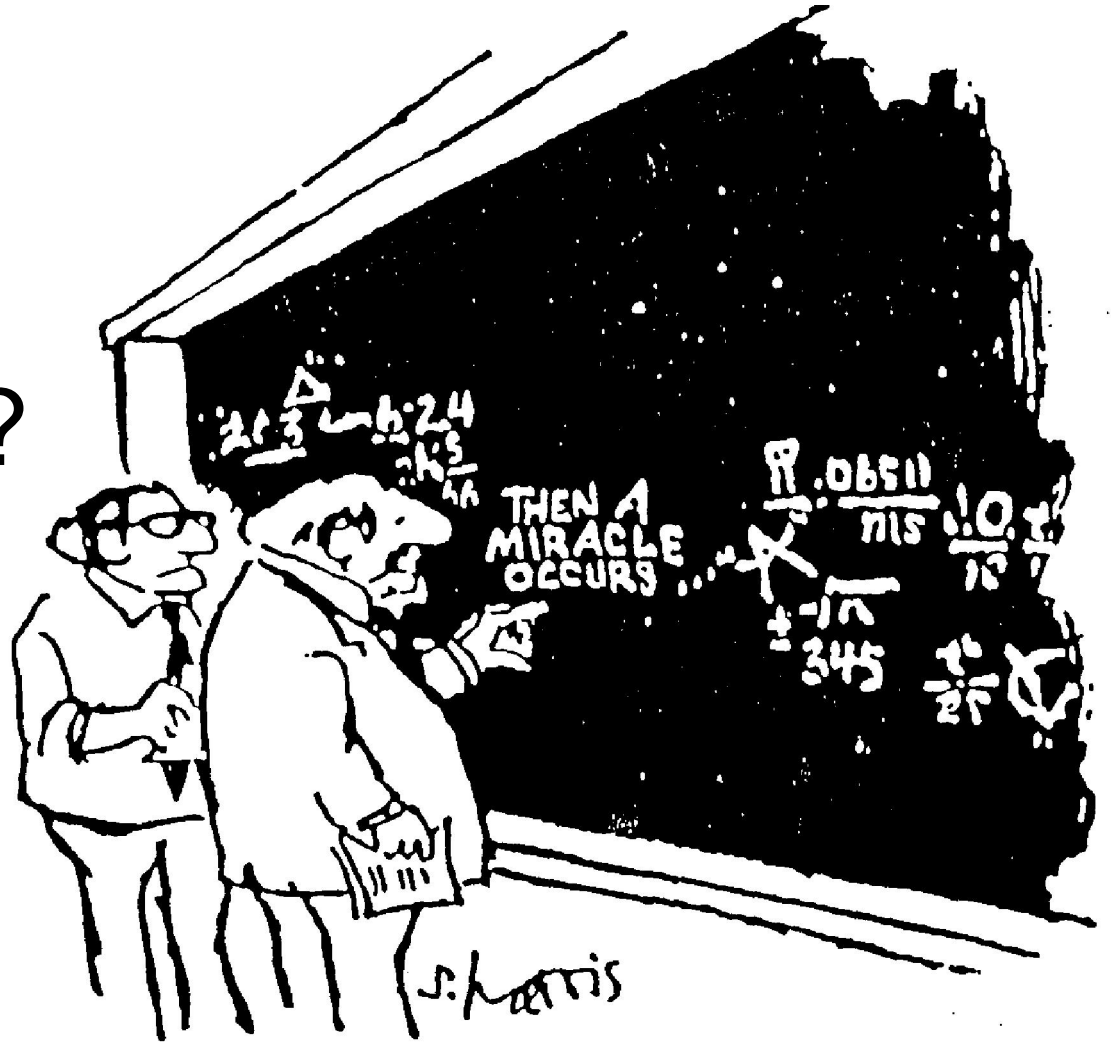
- What are the effects of each on:
  - Data plane flexibility
  - Integration with legacy
  - Interoperability (CP / DP)
  - Vendor lock-in

Source: E. Crabbe, slides-85-sdnrg-7.pdf

Concluding thoughts on SDN

- Remember: SDN is not a protocol (OpenFlow yes);
  - SDN is an operational and programming architecture.
- SDN starts a new dialogue about network programmability, control models, the modernization of application interfaces to the network, and true openness around these things.
- From device-centric HW-constrained networking to network-wide service-oriented SW-defined networking
  - SDN is a new approach to the current world of networking, but it is still networking.
- Vendor Lock-in : It is about features, be it SW or HW
- Cost discussion : May be shifted from HW to SW / services

Obrigado! (mais) Perguntas?



Further reading: "**Software-Defined Networking: A Comprehensive Survey**"
http://arxiv.org/abs/1406.0440

Contributions welcome:
https://github.com/SDN-Survey/latex/wiki

BACKUP

SDN: a Fundamental Step Forward

- or just a new whip to beat vendors with?

What makes SDN attractive?

- The idea that a network is more than the sum of its parts
  - I.e., take a network-wide view rather than a box-centric view
- The idea that creating network services can be a *science* rather than a set of *hacks on hacks on hacks*
  - Especially hacks that vary by box, by vendor and by OS version
- The idea that there should be a *discipline* and *methodology* to service *correctness*
  - Rather than testing (and more testing), declaring victory, only to fail in the real world because of some unanticipated interaction

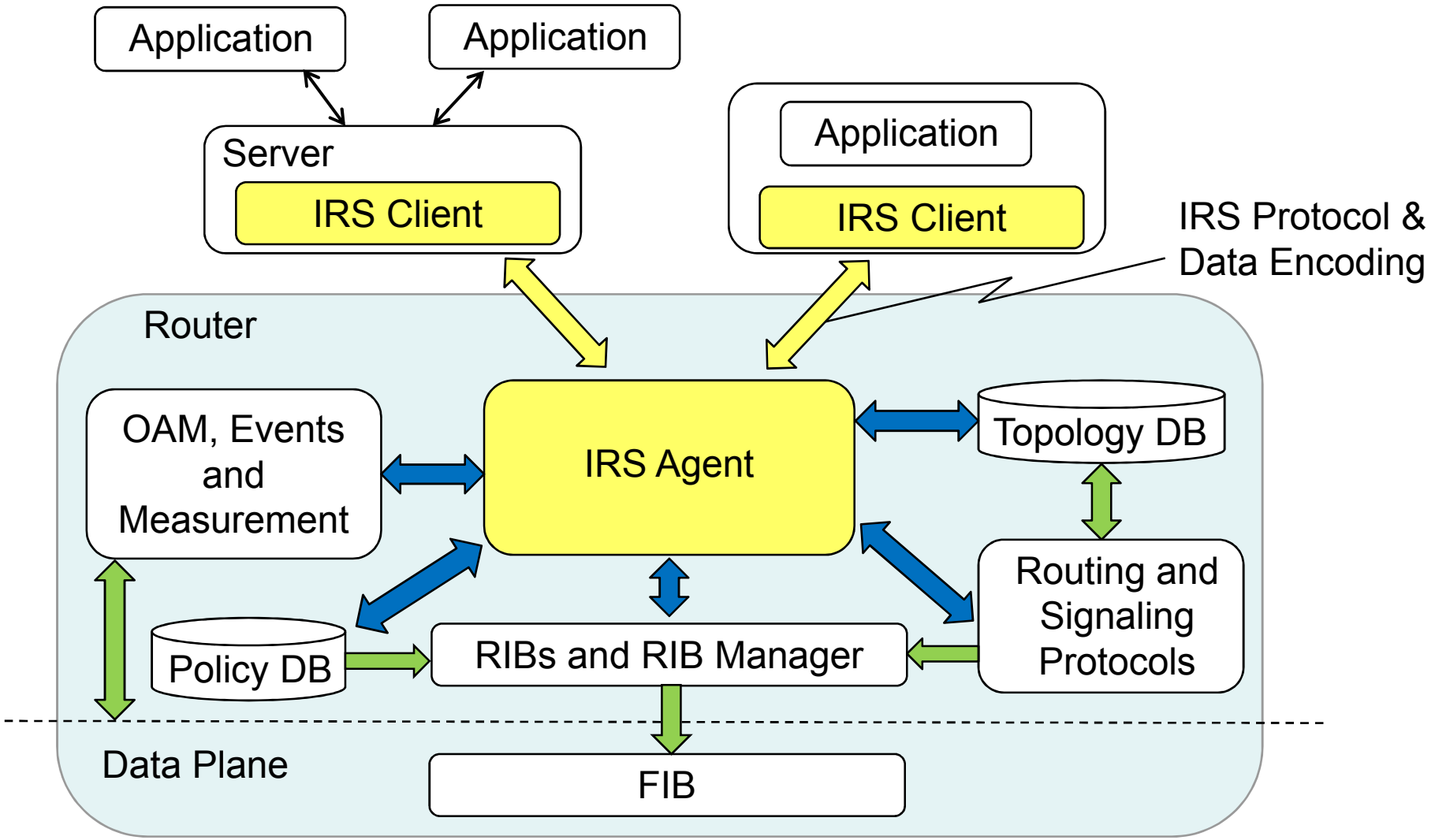
Source: K. Kompella, slides-85-sdnrg-2.pdf

SDN is a real step

1. *if* SDN gives us an **abstraction** of the network
2. *if*, through this abstraction, we have a means of **reasoning** about the network and network services
3. *if* SDN offers a means of **verifying** correct operation of the network or of a service
4. *if* SDN offers a means of **predicting** service interaction
5. Finally, *if* SDN offers a means of setting (conceptual) asserts by which we can get **early warning** that something is wrong

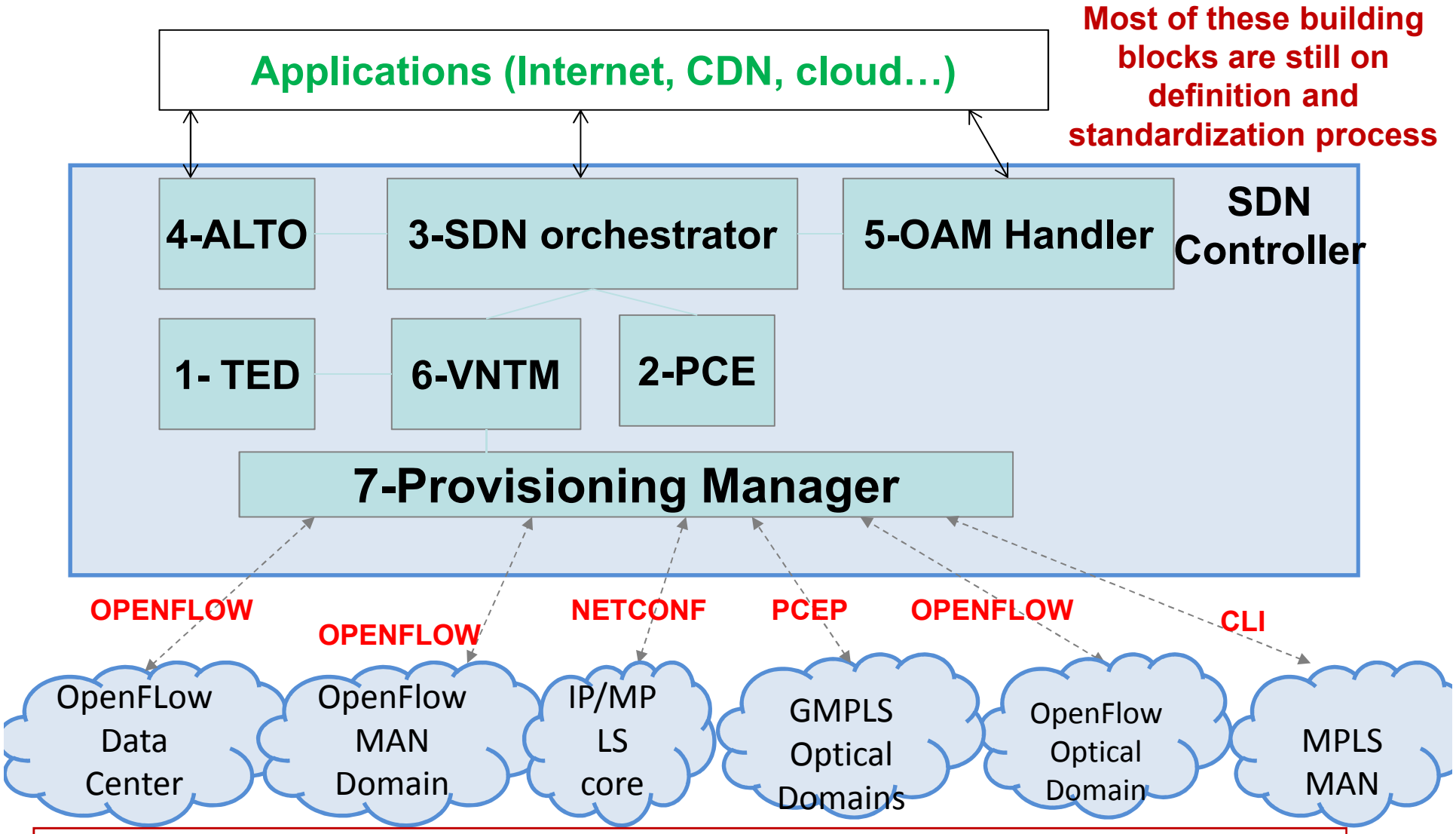
Source: K. Kompella, slides-85-sdnrg-2.pdf

The IRS Architecture



Source: Adrian Farrel

SDN controller based on standard building blocks



E2E networks might be pure OpenFlow based one day, but the migration process will take some time

Application-Based Network Operation

- SDN tools provide high-function, but low granularity
- There is a need to coordinate SDN operation to provide service-level features
- Some components already exist or are proposed
  - Orchestrators
  - OpenFlow Controllers
  - Routing protocols
  - Config daemons
  - IRS Client
  - Virtual Network Topology Manager
- Need a wider architecture to pull the tools together
  - A framework in which the SDN components operate

Source: Adrian Farrel