OpenFlow, Software Defined Networking, and Where its All Going

David Meyer
World Telecommunications Congress 2012
March 04 – 07, 2012
Miyazaki, Japan
dmm@cisco.com

Agenda

- What is OpenFlow?
- A Bit on Current Next Gen OpenFlow Thinking
- Brief Overview of OF Standardization Efforts
- SDN History/SDN Controllers (if time)
- Where this is all Going: *The Programmable Network*
- Q&A

Before We Dive In Here…

- Plenty of reasons to be skeptical of OF/SDN, for example…
- The term **SDN** itself means everything to everybody
- **Flow based networking**
  - All kinds of scalability/switch model questions
  - All kinds of business questions
  - Think about where flow based networking can be efficient
- **Centralized Control, even if "logical"**
  - We have a lot of experience in building distributed control planes that scale well, are resilient, have baked in code, …
  - Again, think about where centralized control might be useful, and what scalability issues might arise in controller based architectures
- **How complex does the capability negotiation between the controller and target need to be?**
  - Does the controller need a "per-switch device driver"?
- **OpenFlow is a network element level programming abstraction**
  - Is this the right programmatic level for "network programmability"?
  - In particular, is programming at the forwarding plane level useful or a general purpose paradigm?
- My goal is for you leave this talk with more questions and maybe a few more answers about OF/SDN than when you walked in
  - And a clearer view about where SDN is all going

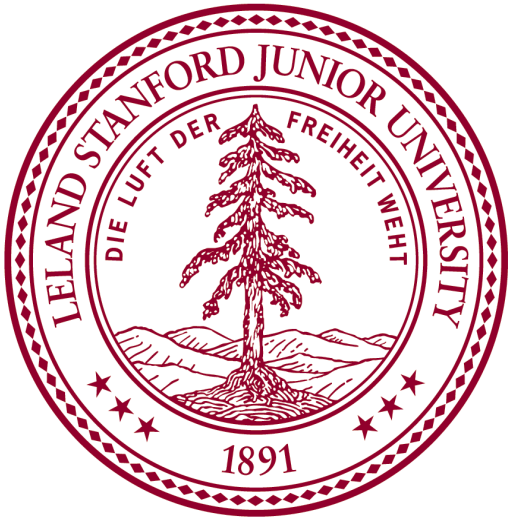
In the Beginning...

Ethane: Addressing the Protection Problem in Enterprise Networks

Martin Casado
Michael Freedman
Glen Gibb
Lew Glendenning
Dan Boneh
Nick McKeown
Scott Shenker
Gregory Watson

Presented By: Martin Casado
PhD Student in Computer Science,
Stanford University

casado@cs.stanford.edu
http://www.stanford.edu/~casado



A Little Later…OpenFlow
(again, with a cast of 2^10s)

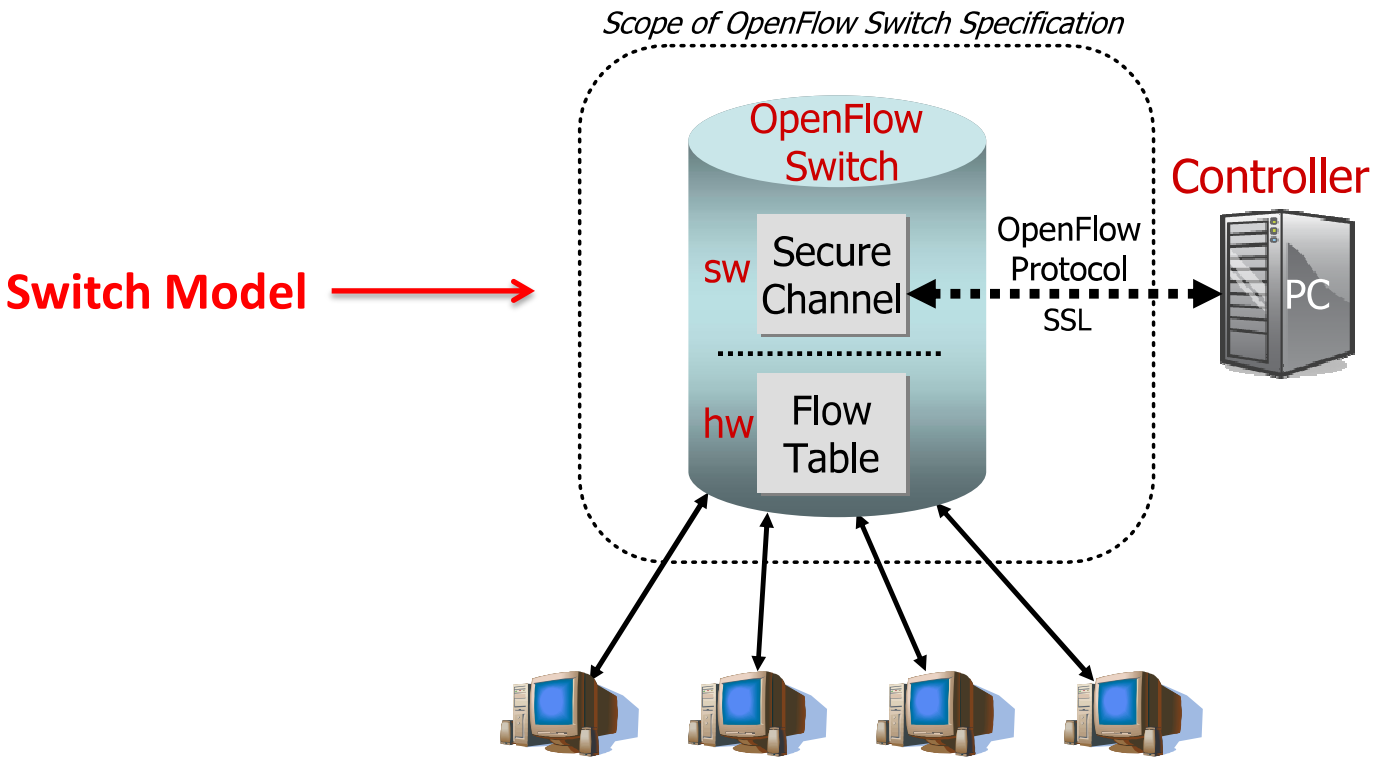
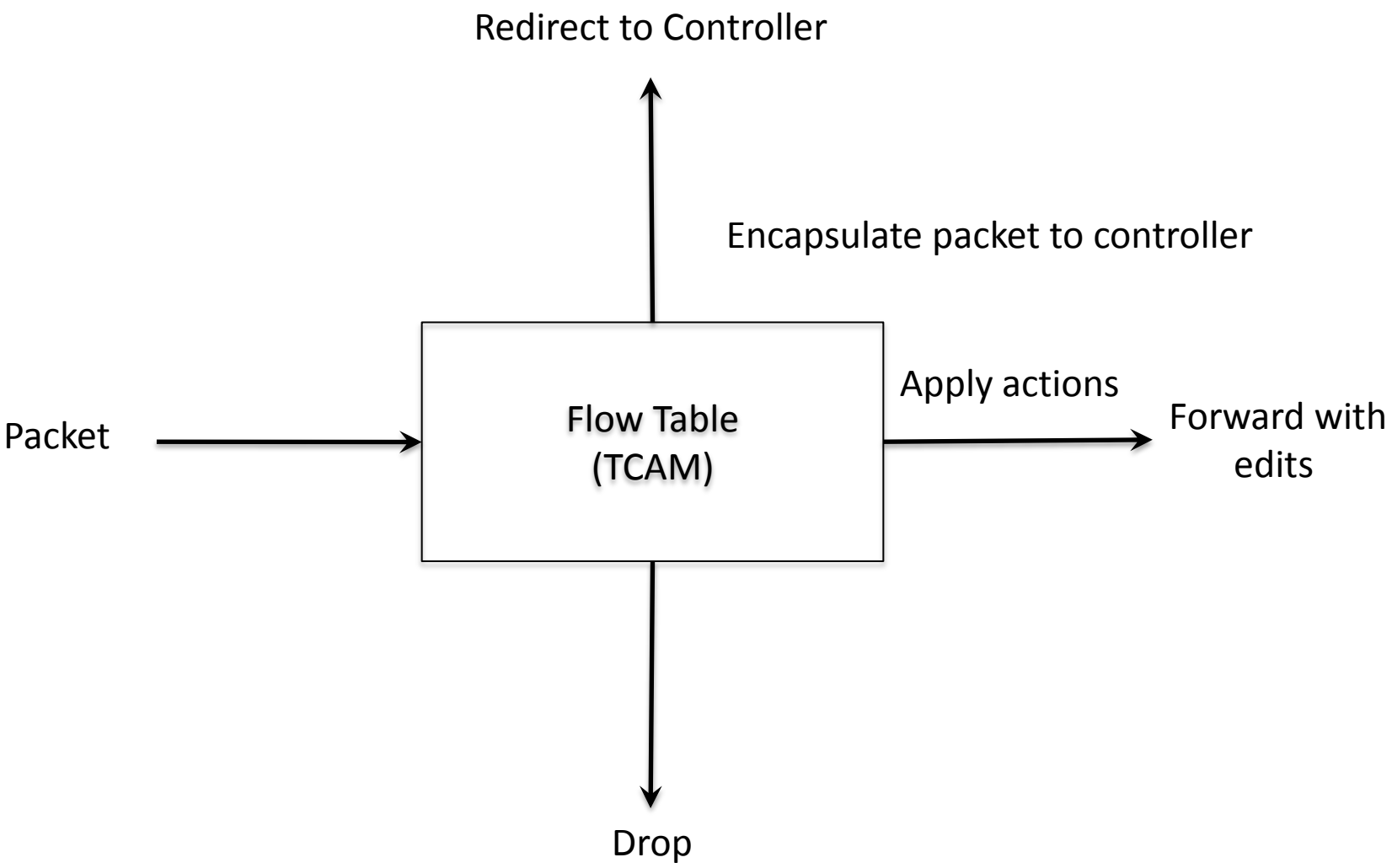


Figure 1: Idealized OpenFlow Switch. The Flow Table is controlled by a remote controller via the Secure Channel.

OpenFlow Switch Model Version 1.0



OpenFlow Switch, v 1.0

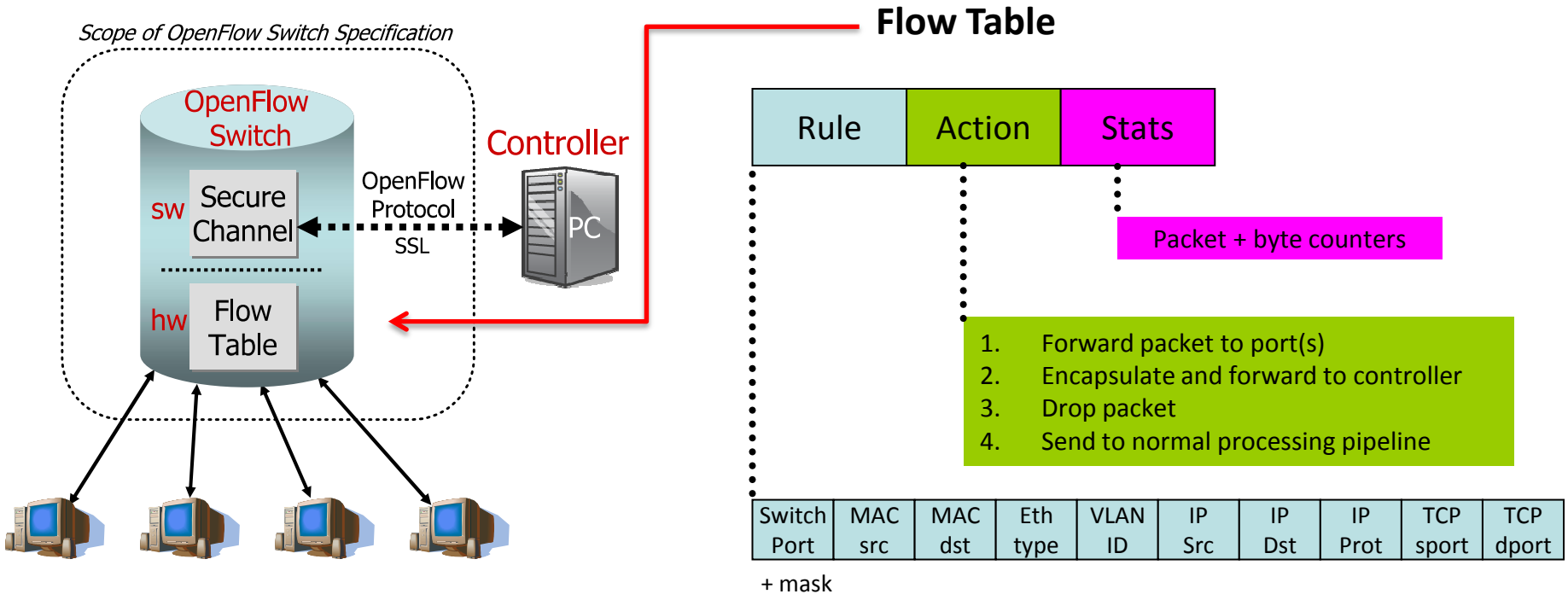


Figure 1: Idealized OpenFlow Switch. The Flow Table is controlled by a remote controller via the Secure Channel.

Header Fields for Matching (v.1.1)

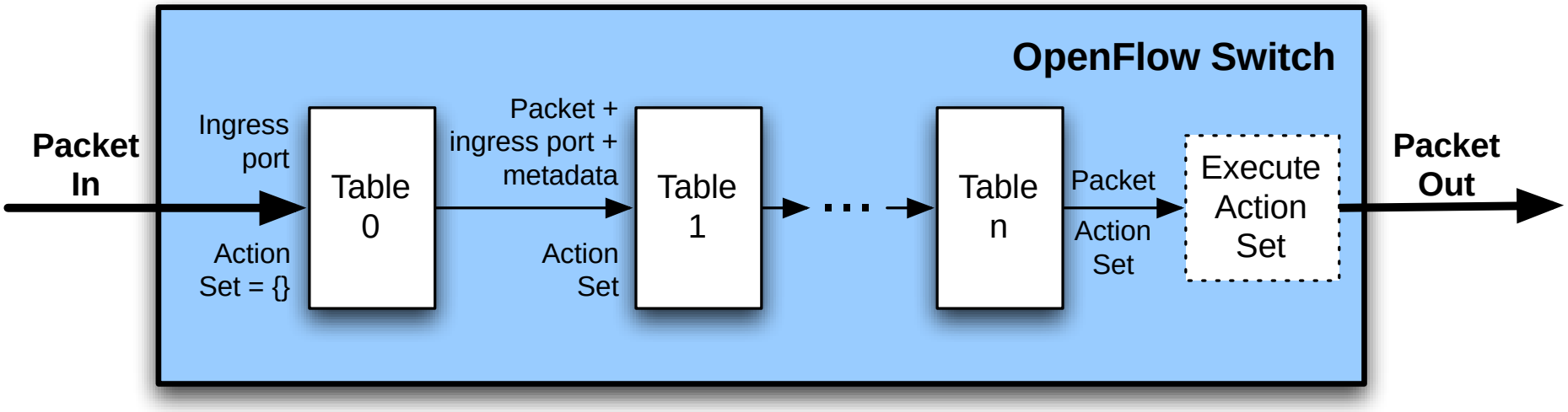
Ingress Port	Metadata	Ether src	Ether dst	Ether type	VLAN id	VLAN priority	MPLS label	MPLS traffic class	IPv4 src	IPv4 dst	IPv4 proto / ARP opcode	IPv4 ToS bits	TCP/ UDP / SCTP src port / ICMP Type	TCP/ UDP / SCTP dst port / ICMP Code

Table 3: Fields from packets used to match against flow entries.

Note that the ability to match over all of the header fields simultaneously essentially "de-layers" the network stack

Why is this important: RYF-Complexity theory states that layering and decentralization are fundamental to providing robust, scalable networks [AldersonDoyle2010]

OpenFlow Version 1.X, X > 0



(a) Packets are matched against multiple tables in the pipeline

{Any,Multi}cast	(1.1)
ECMP	(1.1)
MPLS	(1.1, note push/pop, .1q)
IPv6	(1.2)

1.3 features being currently being considered
-- incremental features, PBB, …
Configuration Protocol under co-development

So What Is OpenFlow?

- ***A Switch Model***
  - Match-Action Tables (MAT)
  - Per-flow counters
- ***An Application Layer Protocol***
  - Binary wire protocol, messages and state machine that allow programming of the MAT
- ***A Transport Protocol***
  - TLS, TCP, ..

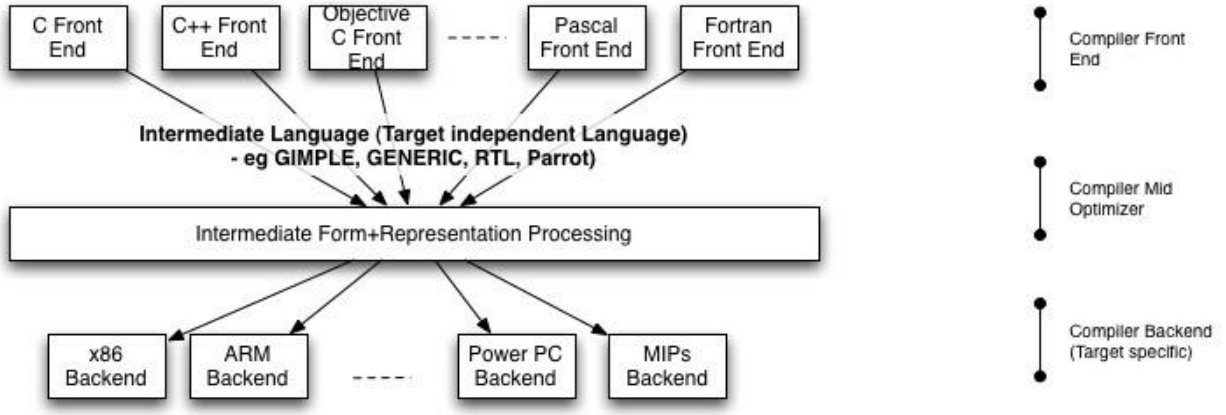
- Note that OF says nothing said about how a given target implements the switch model → OF is an **Abstract Switch Model**
  - However, the model **only deals with the forwarding plane**

Engineering Challenges for the OF Protocol

- **Weak Network Device Abstract Model**
  - For example, it loses information
    - Table Typing
    - Table Relationships
- **Weak Control Plane/Data Plane Interface**
  - Pipeline order → can't express speculative or parallel pipelines
- **Combinatorial State Explosion**
  - Cartesian product of routes*policies
- **Insufficient Network Primitives**
  - Data Plane
  - Replication
  - Rewriting
  - Parsing separated from actions
    - Not particularly h/w friendly
    - Requires assumption of contiguous memory in places
    - Can't name locations in header (consider en/decap)
- …
- Bottom line: Switch model poorly defined →
  - Current versions of OF have various weaknesses
  - Perhaps 1.0 for constrained cases
- Google DIR/HAL designed to address these issues

Next Generation OpenFlow Google DIR Proposal

- DIR is the means of expressing the desired forwarding plane model → Refines/formalizes the “MAT Model”
  - What not how (http://goo.gl/6qwbg)
  - Defines two sets of primitives
  - Core primitives: describe an independent thread of data path functionality, e.g., TABLE_MATCH, METER, QUEUE
  - Control primitives: glue/housekeeping functionality like interaction between any of the independent threads, e.g., EVENT, EVENT_HANDLER, TIMER, WATCH, RAISE, …
  - Build up models from primitives (essentially the LFBs of ForCES)
  - Build device models from models
- Provides a language for describing forwarding pipeline and associated functional blocks
- DIR is exchanged between the Controller and the OF Device
  - DIR is evaluated/compiled controller discovery time…importantly *not* at runtime
- DIR is independent of the target forwarding pipeline
  - The same DIR sent to different devices with similar capabilities produces the same forwarding behavior
- Heavy emphasis on tool chains



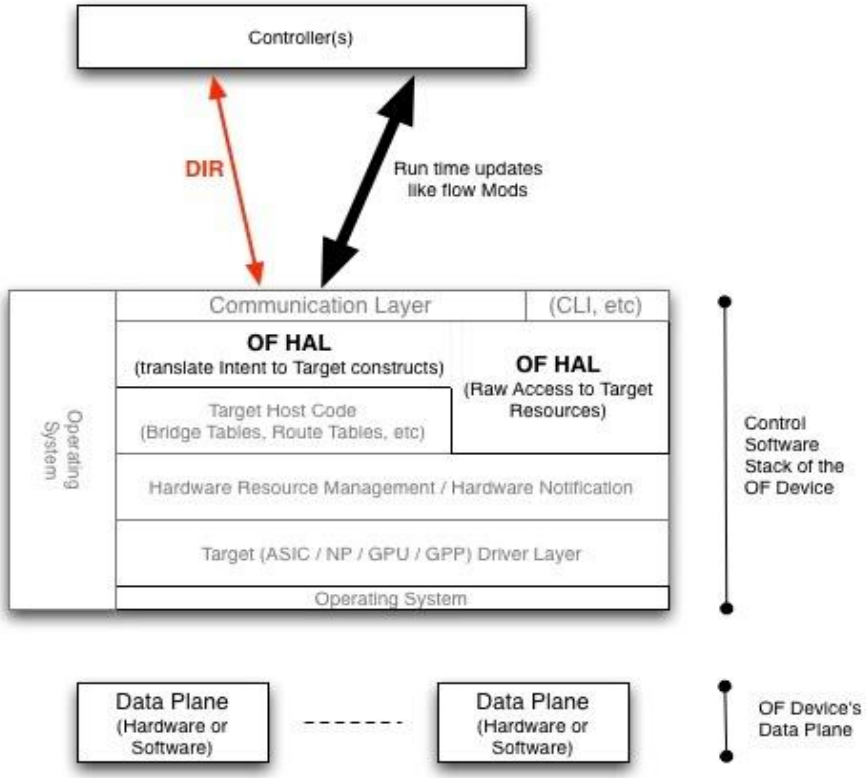
LLVM / GCC Compiler model

Hardware Abstraction Layer (HAL)

HAL Translates Constrained DIR to local platform.

Constrained DIR should be within negotiated capabilities of target platform.

HAL optimizes with well-known module/node names where possible.



OpenFlow Standardization

- Open Networking Foundation
  - http://www.opennetworkingfoundation.org
- Mission is to standardize OF & management APIs
- **Board**: GOOG, FB, MSFT, VZ, DT, Yahoo!, and NTT
- **TAB**: Broadcom, Nicira, GOOG, HP, VZ, CSCO, Stanford
- **Members**: Big Switch Networks, Broadcom, Brocade, Ciena, Cisco, Citrix, Comcast, CompTIA, Dell, Ericsson, Extreme, Force10, Fujitsu, HP, Huawei, IBM, Infoblox, Intel, IP Infusion, Ixia, Juniper, Marvell, Mellanox, Metaswitch, Midokura, NEC, Netgear, Netronome, Nicira, Nokia Siemens, NTT, Plexxi Inc., Pronto/Pica8, Riverbed, Vello Systems, Vmware, …

ONF WGs

- Extensibility
  - Jean Tourrilhes (HP Labs)
- Config-Mgnt
  - Deepak Bansai (Microsoft/AZURE)
- Testing-Interop
  - Michael Haugh (IXIA)/Rob Sherwood (BigSwitch)
- Hybrid
  - Jan Medved (Cisco)
- Marketing-Education
- of-future
  - Curt Beckmann (Brocade)

So, Putting It All Together
The Original SDN Vision was…

- **All About Abstractions**
  - SDN is a bigger concept of which OF can be a component
  - That is, ***Programmable Network > SDN > OpenFlow***

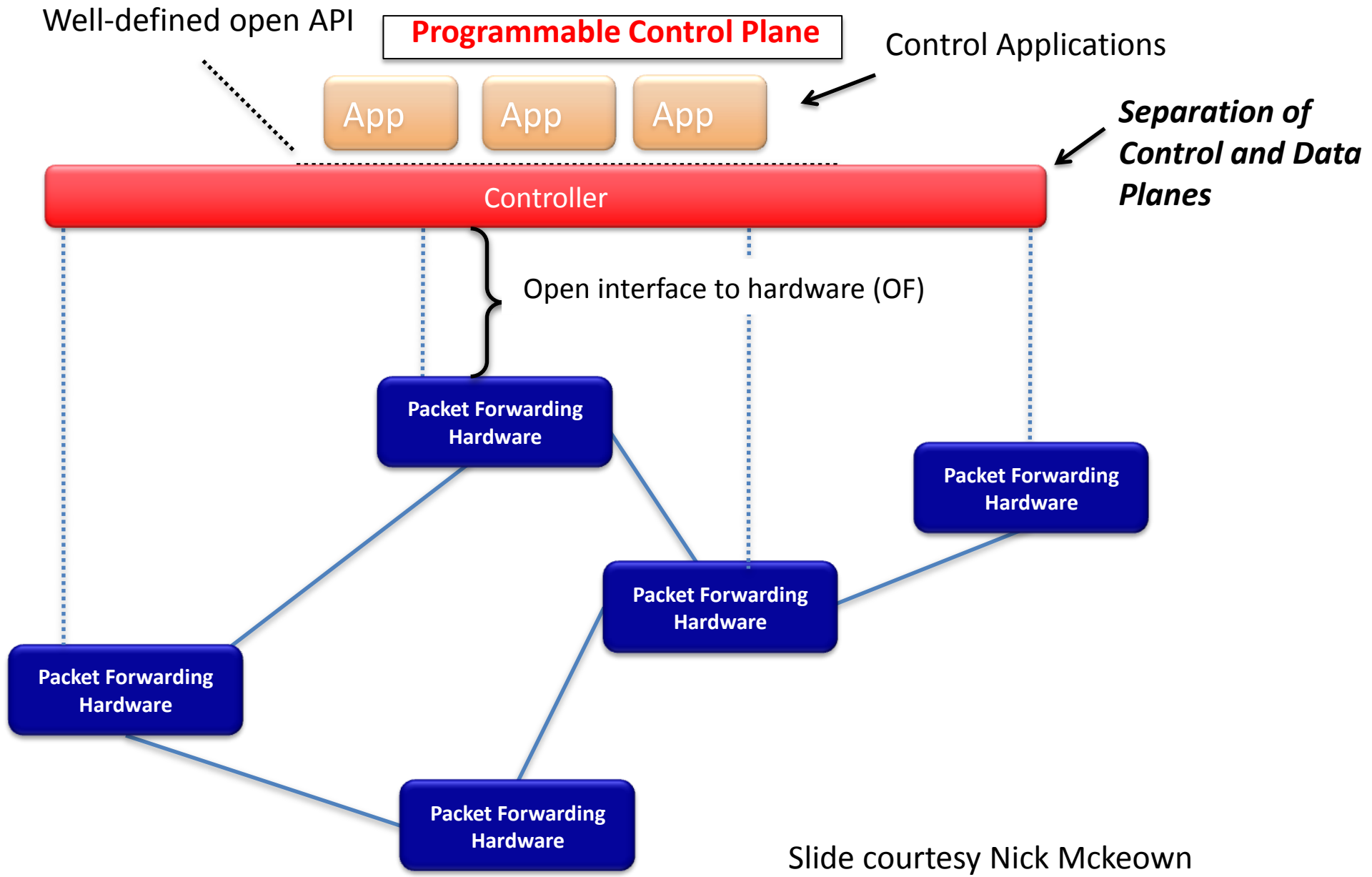
- **Forwarding abstraction**
  - OpenFlow

- **Distributed State Abstraction**
  - Global Network View (Logical and Virtual)
  - Separate State Management from Protocol Design and Implementation

- **Distributed Network Control Abstraction**
  - Network Operating System (NOS)

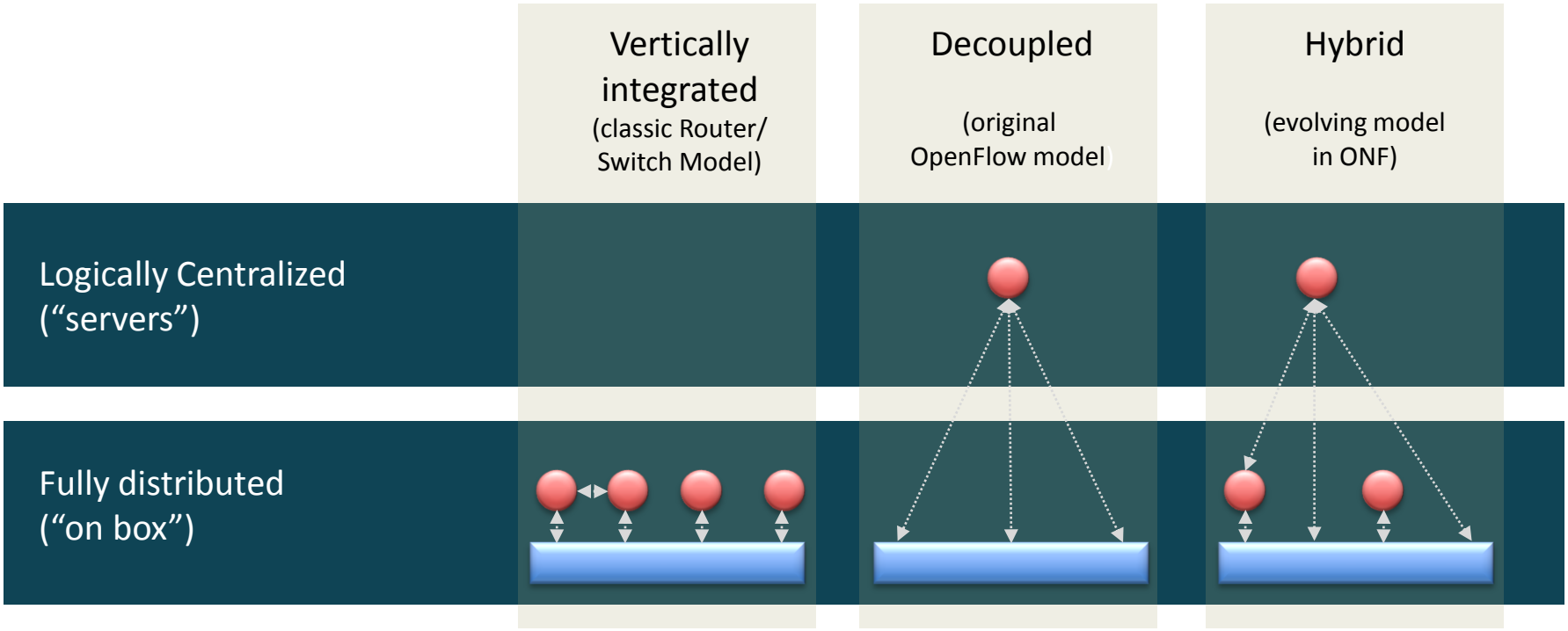
- **Northbound APIs**

SDN v1



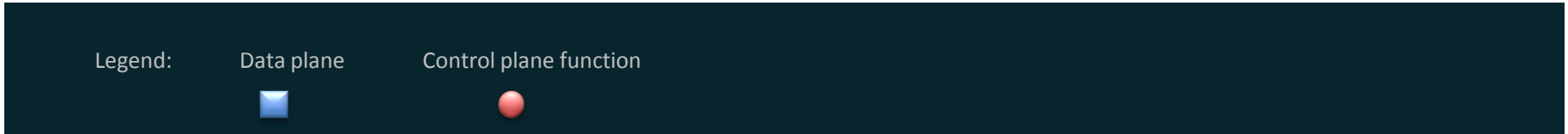
Slide courtesy Nick Mckeown

Cut Another Way: Control Plane Distribution Options



Data Path jointly controlled by standard on-box control plane and centralized off-box controller

Slide courtesy Frank Brockners



BTW, Nothing New Under The Sun…

- ***Separation of control and data planes*** is not a new idea. Examples include:
  - SS7
  - Ipsilon Flow Switching
    - Centralized flow based control, ATM link layer
    - GSMP (RFC 3292)
  - AT&T SDN
    - Centralized control and provisioning of SDH/TDM networks
  - A similar thing happened in TDM voice to VOIP transition
    - Softswitch → Controller
    - Media gateway → Switch
    - H.248 → Device interface
    - Note 2nd order effect: This was really about circuit → packet
  - ForCES
    - Separation of control and data planes
    - RFC 3746 (and many others)

So Here's Another Way to Think About Network Programmability and SDN

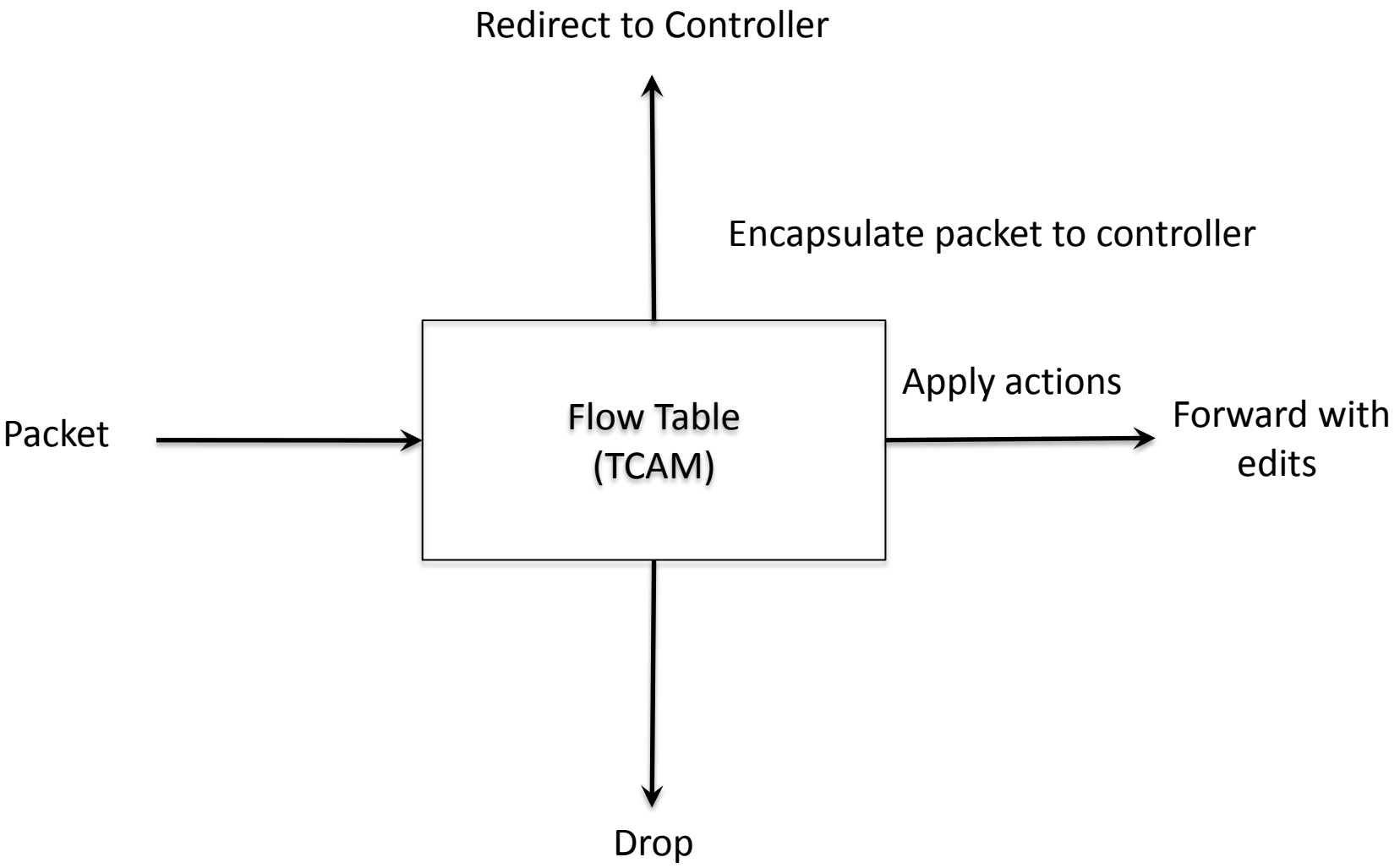
- **Architecture*:*** *Generalized Programmable Network (GPN)*
  - The term "SDN" already too overloaded,….
  - Encompass existing and future control planes
    - And associated features
    - "Hybrid Switch" modes
  - Standardize a diverse set of APIs in addition to OF
    - Such APIs talk to both existing control and data planes

- **Objective:** Enable tighter interaction between applications and the network
  - Inform network of desired behavior
  - Inform application of data intrinsically available in the network
    - Data mining, telemetry, NPS, …
  - Provide greater agility, flexibility, and feature velocity

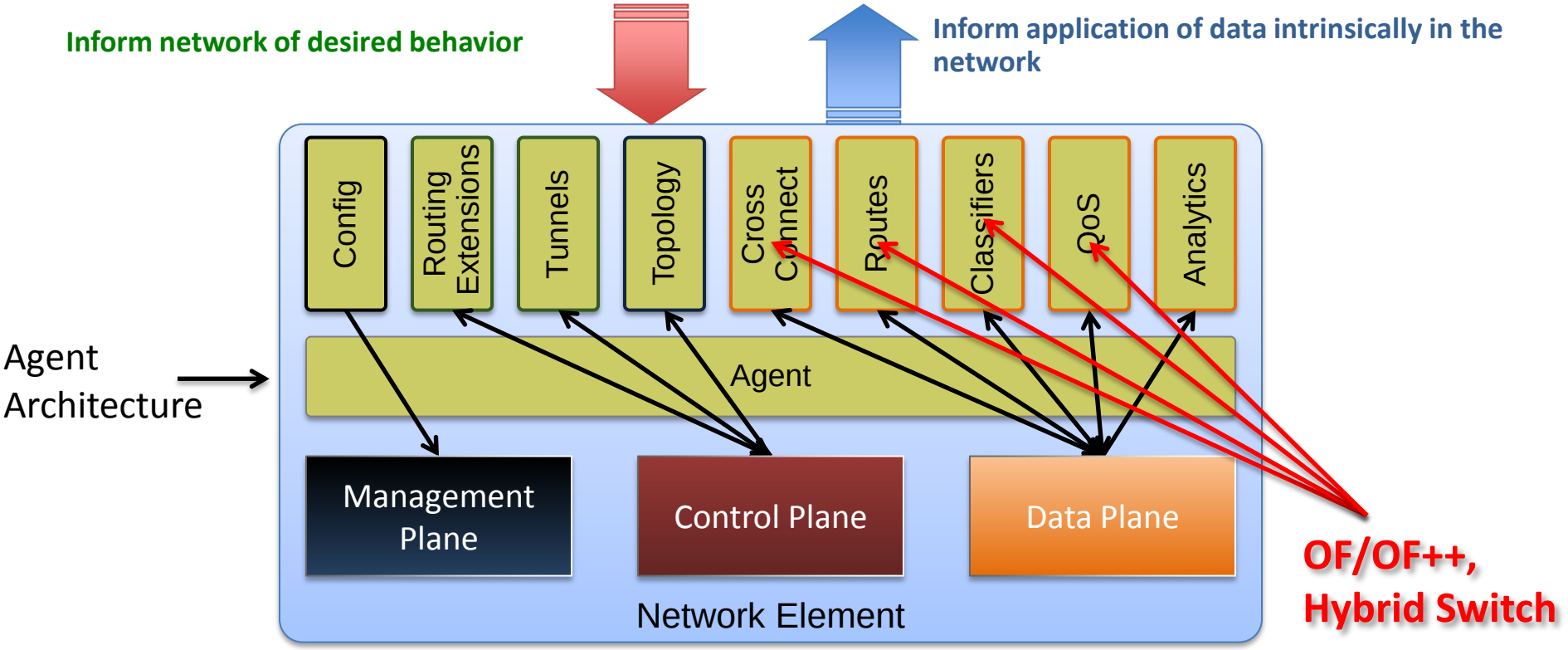
- **Approach:** Define two broad classes of APIs
  - Network APIs
  - Network Element APIs

- Requires a ***Generalized Switch Model***

Recall the OF 1.0 Switch Model (Network Element)



What Do We Really Want From A Network Element?



What is a Hybrid Switch?

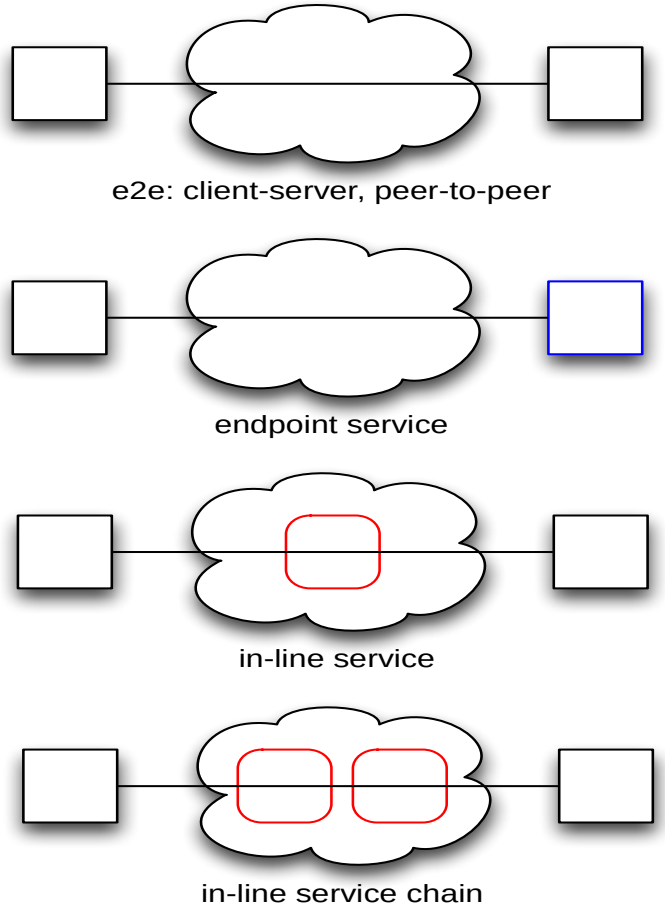
- Abbreviated Hybrid Switch Problem Space
  - Make OF/SDN coexist with existing more general switch/network models
    - **Why is this hard again?**
- Proposed Models
  - *Ships in the Night* and *Integrated Mode*
  - *Integrated Mode*
    - Envision OF as one of many ***APIs*** we can use to build network probramability
- Hybrid Switch WG recently chartered by ONF
  - Jan Medved of Cisco is the Chair
  - Possibly related: "SDNP" activity in IETF
    - But note no "SDNP BOF" at upcoming Paris IETF

A Couple Of Hybrid Switch Use Cases

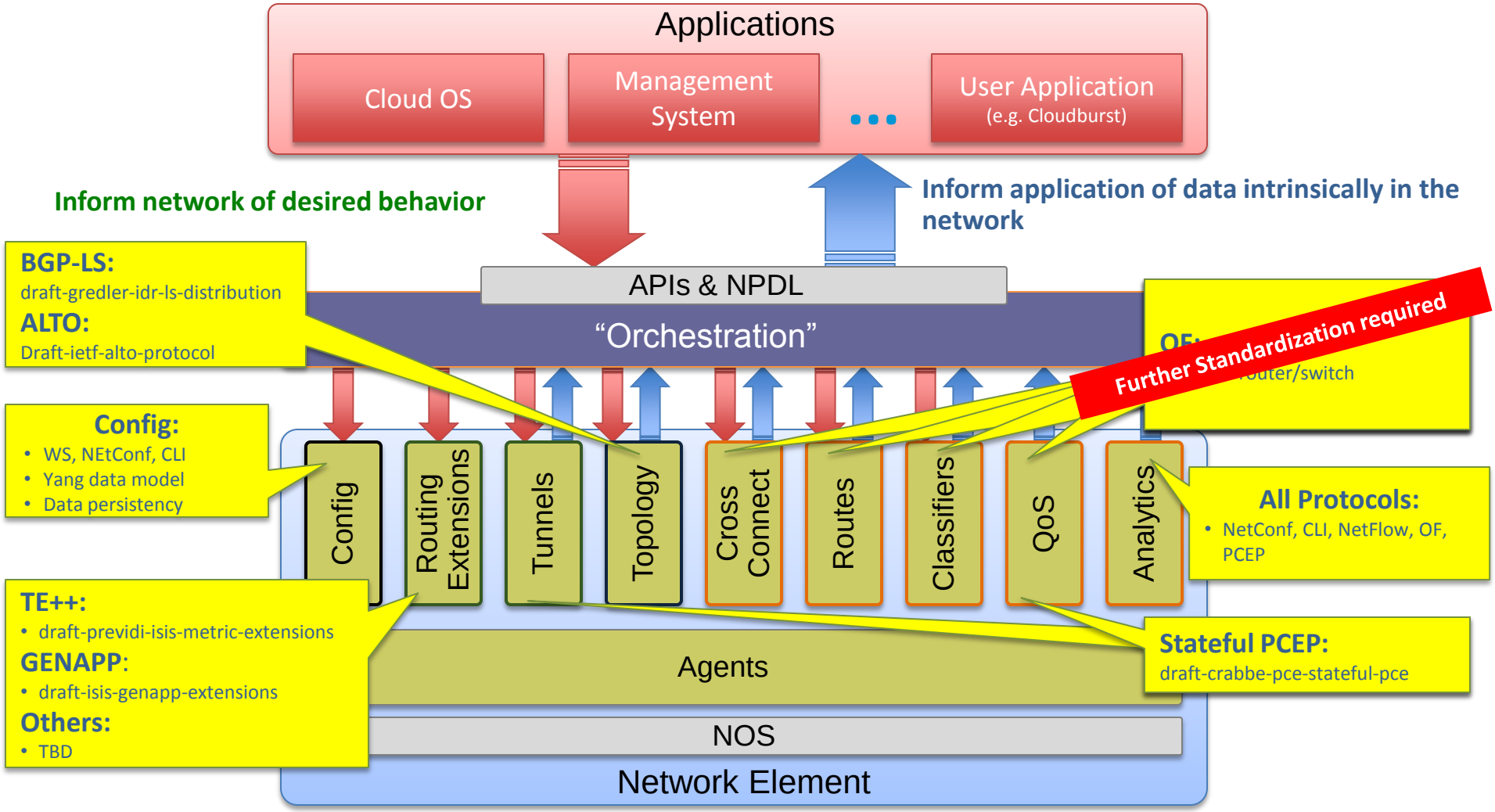
- ***Installing ephemeral routes*** in the RIB
  - Install routes in RIB subject to admin distance or …
  - Moral equivalent of static routes, but dynamic
  - May require changes to the OF protocol/model
- ***Edge classification***
  - Basically use the OF as an API used to install ***ephemeral*** classifiers ***at the edge***
  - Moral equivalent of … 'ip set next-hop <addr>' (PBR)
  - Use case: Service Engineered Paths
    - Program switch edge classifiers to select set of {MPLS, GRE, …} tunnels
    - Core remains the same
- ***Service Chaining***
  - Let's talk a bit more about kinds of service chains…

Programmable Service Chains

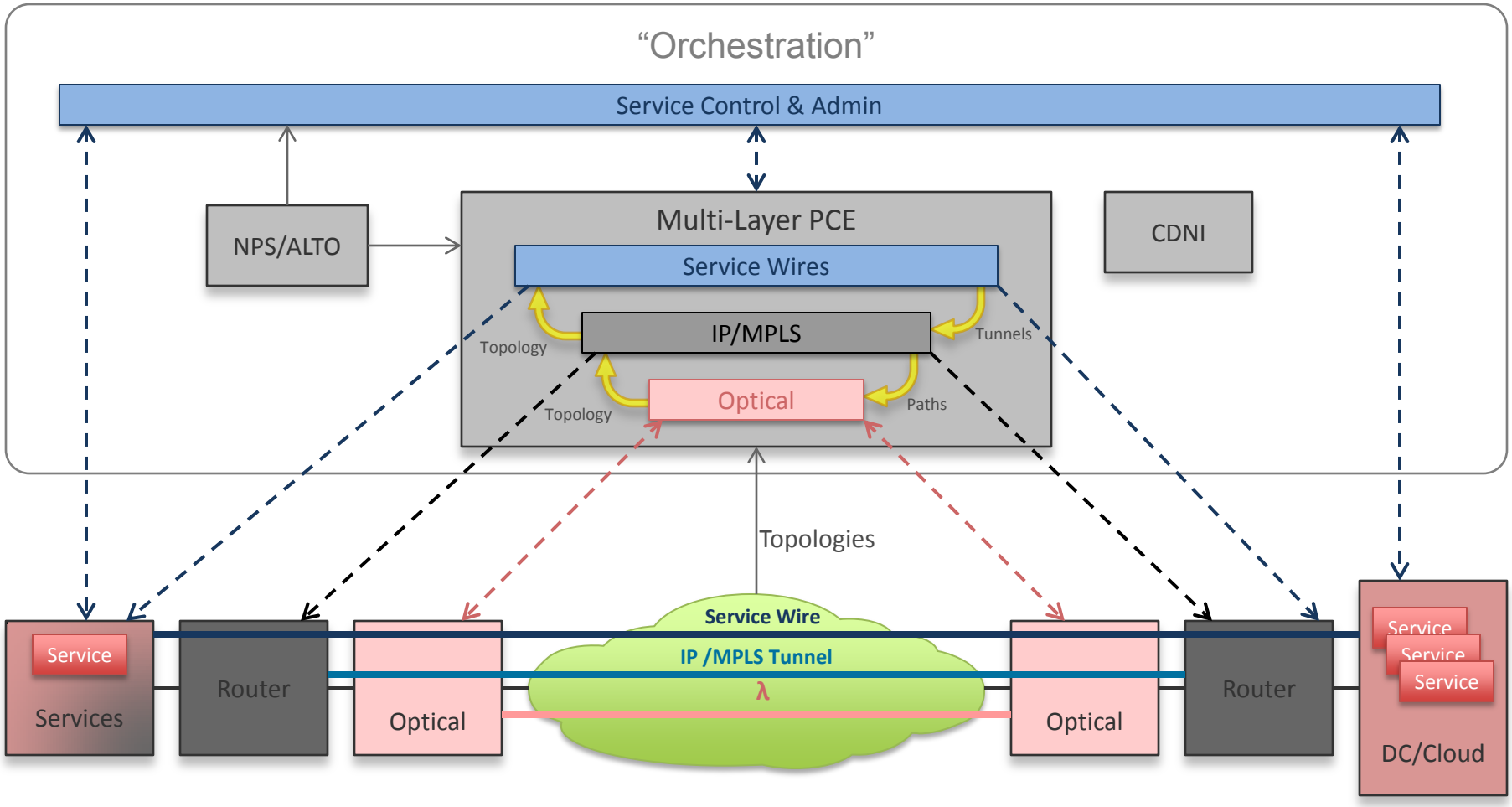
- Basic Use Cases
  - Endpoints vs. In-line services
  - Composite Services / Service Chaining
  - Flow Routing Considerations
    - Fine vs. Coarse Grained Flows
    - Filtering vs. Routing
    - Placement vs. Topology
    - Addressing vs. Flows
- Future/Unknown Use Cases
  - CDN, NDN, Optical xconnect,…



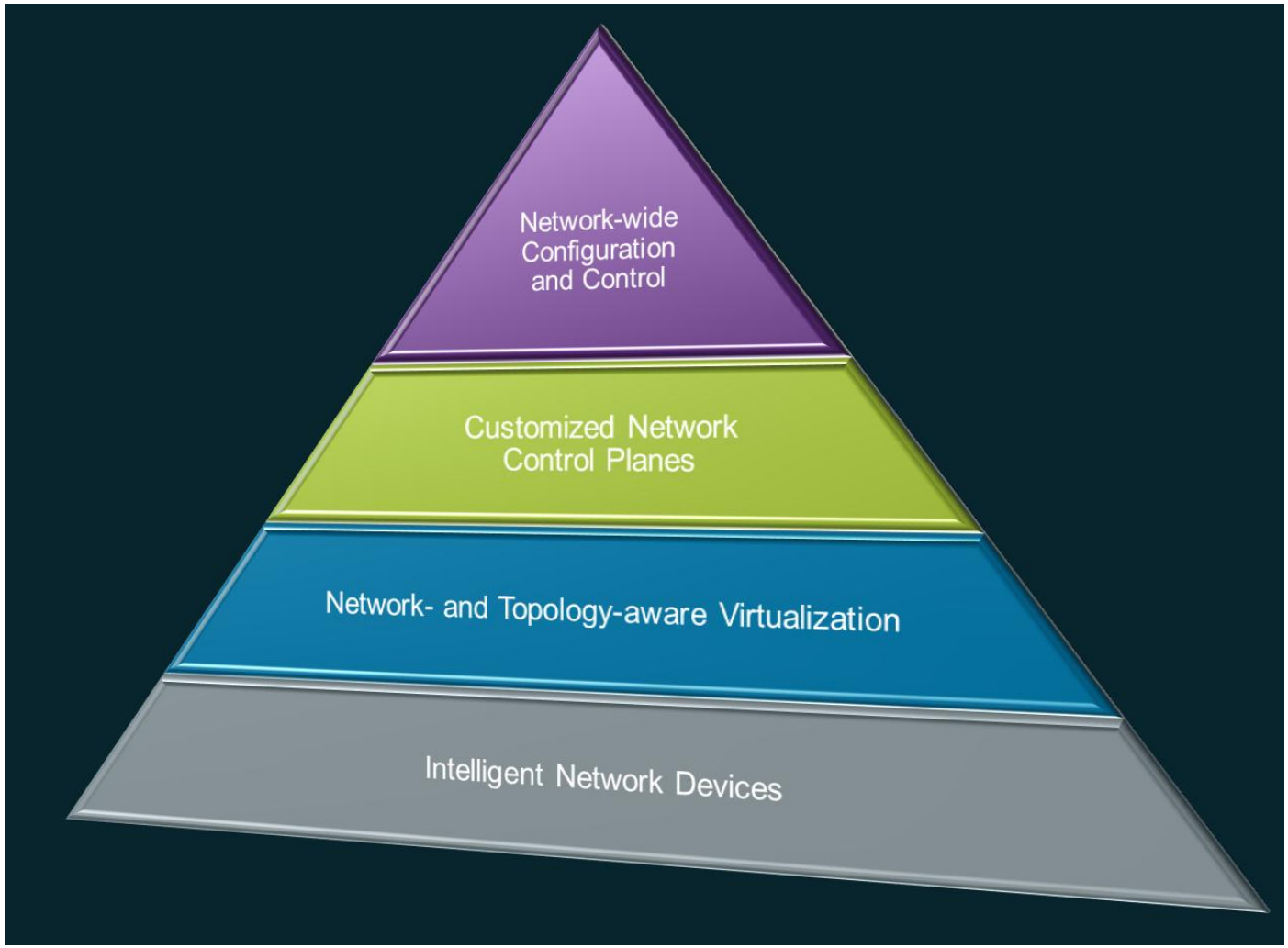
Programmable Network Architecture



Service Provider Network Model



Where this is all going
The Programmable Network



"Orchestration, Configuration, Control"

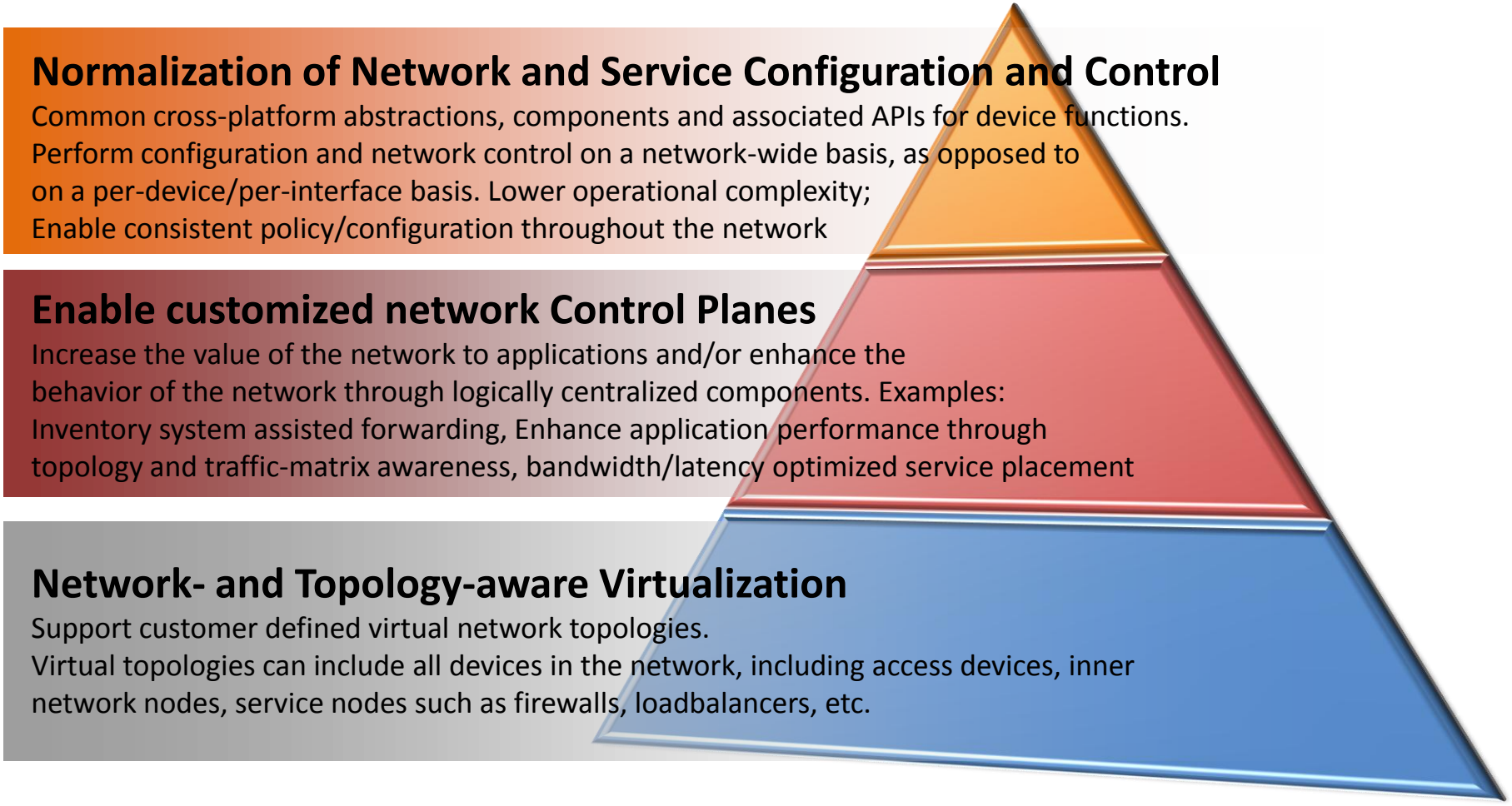
"Programming, Customization"

"Network Hypervisor"

"Efficient Forwarding"

Slide courtesy Frank Brockners

Software Defined Network
"Network Programmability" - Key Value Propositions



Note that the SDN value proposition differs from the "OpenFlow" value proposition. OpenFlow's is focusing on **per-device** level programmability; i.e. creating a (potentially standard) interface to control the forwarding of flows in the device.

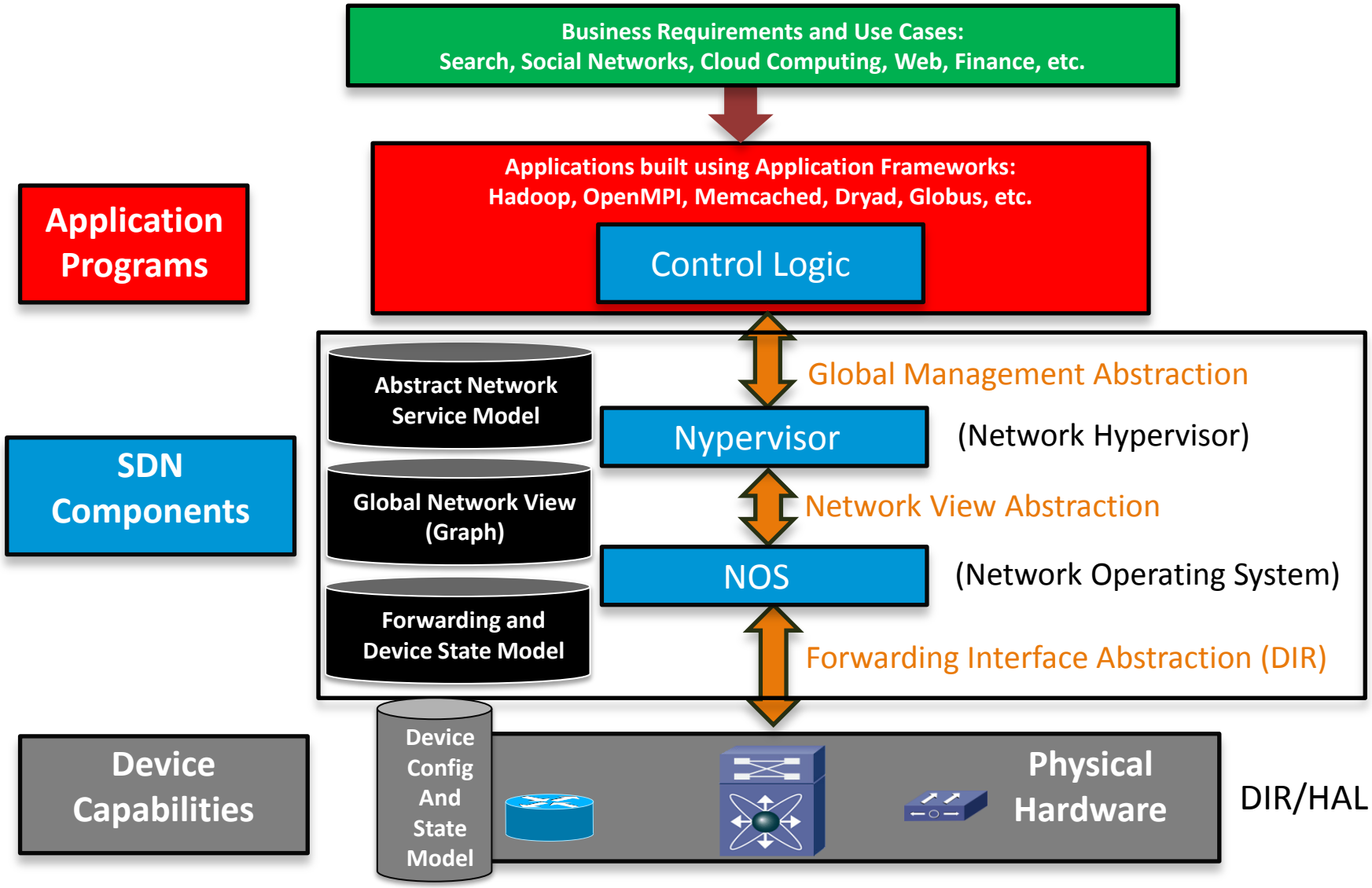
Slide courtesy Frank Brockners

Q&A

Thanks!

Backup Slides

Current OF/SDN Architecture

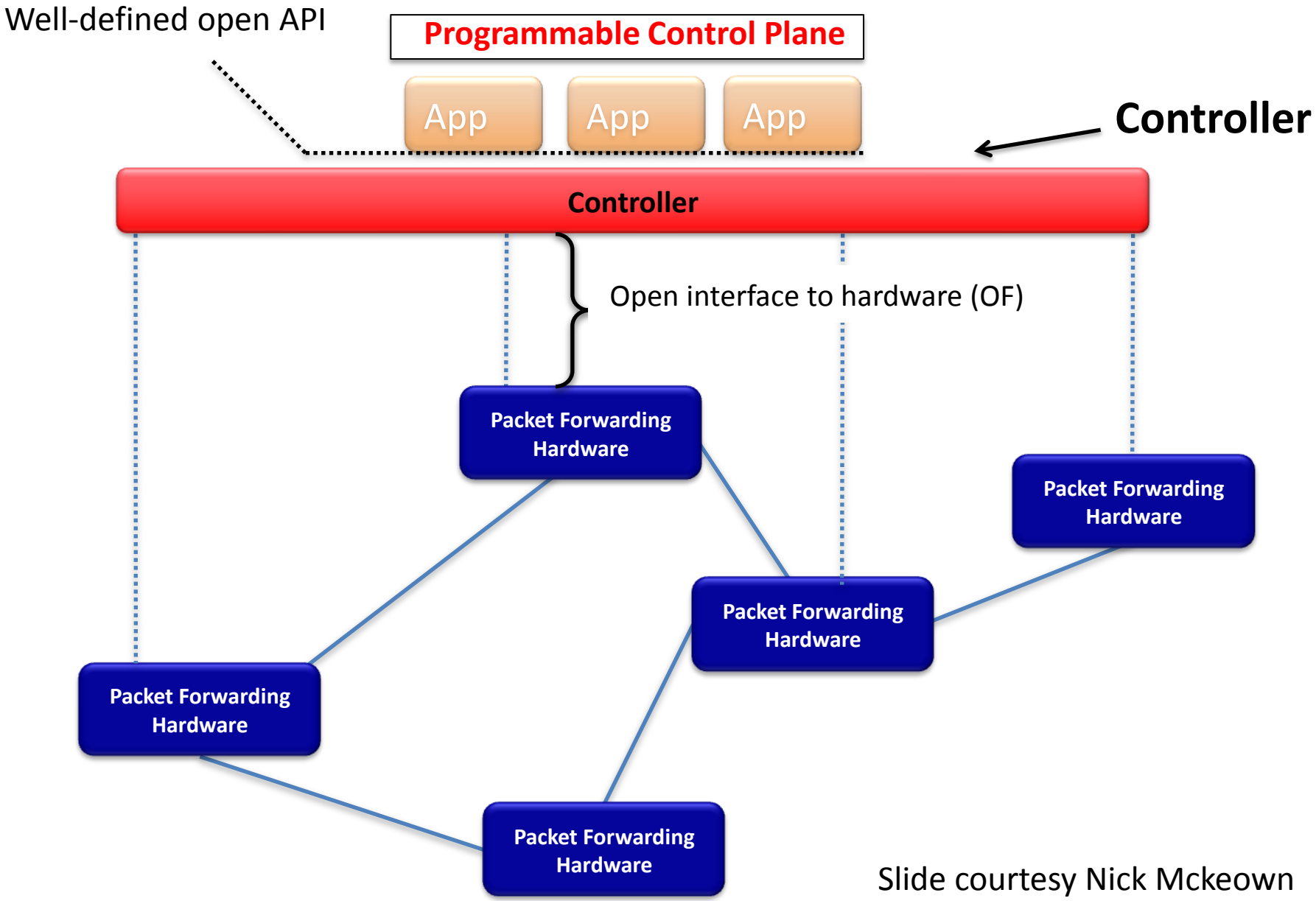


Nothing New Under The Sun…

- Much of the motivation for *this generation's foray into SDN* was grounded in the research community's desire to be able to experiment with new control paradigms.

- I call it "this generation's foray" because the basic idea, *separation of control and data planes*, is not new. Examples include:
  - Ipsilon Flow Switching
    - Centralized flow based control, ATM link layer
    - GSMP (RFC 3292)
  - AT&T SDN
    - Centralized control and provisioning of SDH/TDM networks
  - A similar thing happened in TDM voice to VOIP transition
    - Softswitch → Controller
    - Media gateway → Switch
    - H.248 → Device interface
  - ForCES
    - Separation of control and data planes
    - RFC 3746 (and many others)

Recall that the Original Model was..



Slide courtesy Nick Mckeown

Controller Diversity

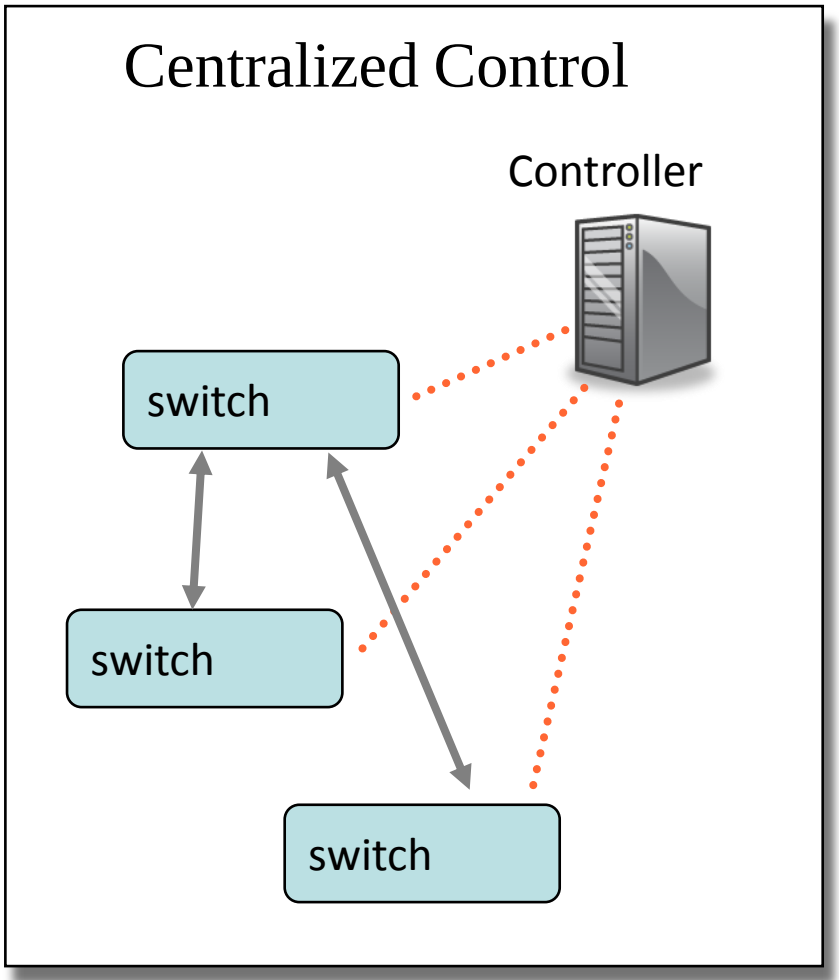
- **Research**
  - RYU
    - NTT
    - http://www.osrg.net/ryu
  - NOX
    - A first generation open source controller
    - http://noxrepo.org/doc/nox-ccr-final.pdf
  - POX
    - "NOX in python" – really more like ONIX
  - Trema
    - Ruby/C controller
    - http://trema.github.com/trema/
  - Floodlight
    - Java based OF controller
    - http://floodlight.openflowhub.org/
  - SNAC
    - Simple Network Access Control
    - http://www.openflow.org/wp/snac/
  - Flowvisor
    - A simple open source "slicing" controller
    - http://www.openflow.org/downloads/technicalreports/openflow-tr-2009-1-flowvisor.pdf
  - Maestro/Beacon
    - Java controllers optimized for multi-core CPUs
    - http://www.cs.rice.edu/~eugeneng/papers/TR10-11.pdf
  - ONIX
    - A second generation infrastructure controller
    - https://www.usenix.org/events/osdi10/tech/full_papers/Koponen.pdf
- **Commercial**
  - Google, NEC, Broadcom, Bigswitch, Nicira, Pica8, …
    - Nicira and Bigswitch recently received new funding
    - Ericsson, Google and Nicira have implemented and deployed ONIX
    - Bigswitch and Pica8 claim that they will open source their controllers (beacon)

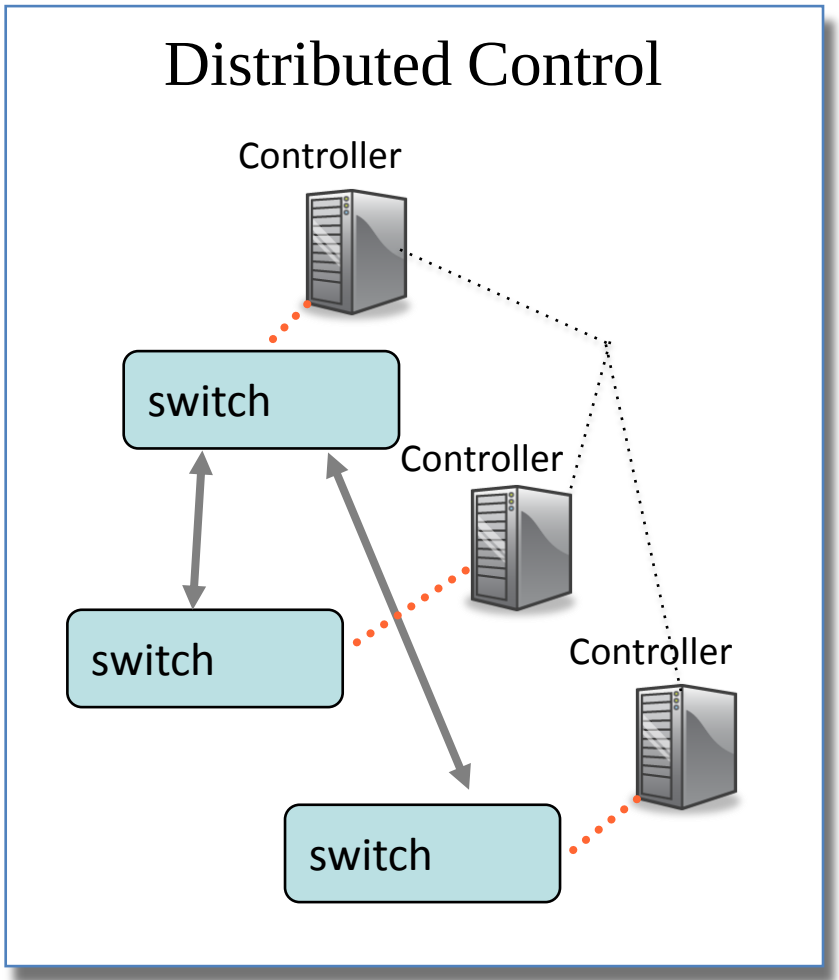
Again, Before You Drink the OF/SDN Koolaid
A Few Issues in Flow Based Controller Design

- Centralized vs. Distributed Controllers
- Flow vs. Aggregated Flows
- Reactive vs. Proactive Flow Setup

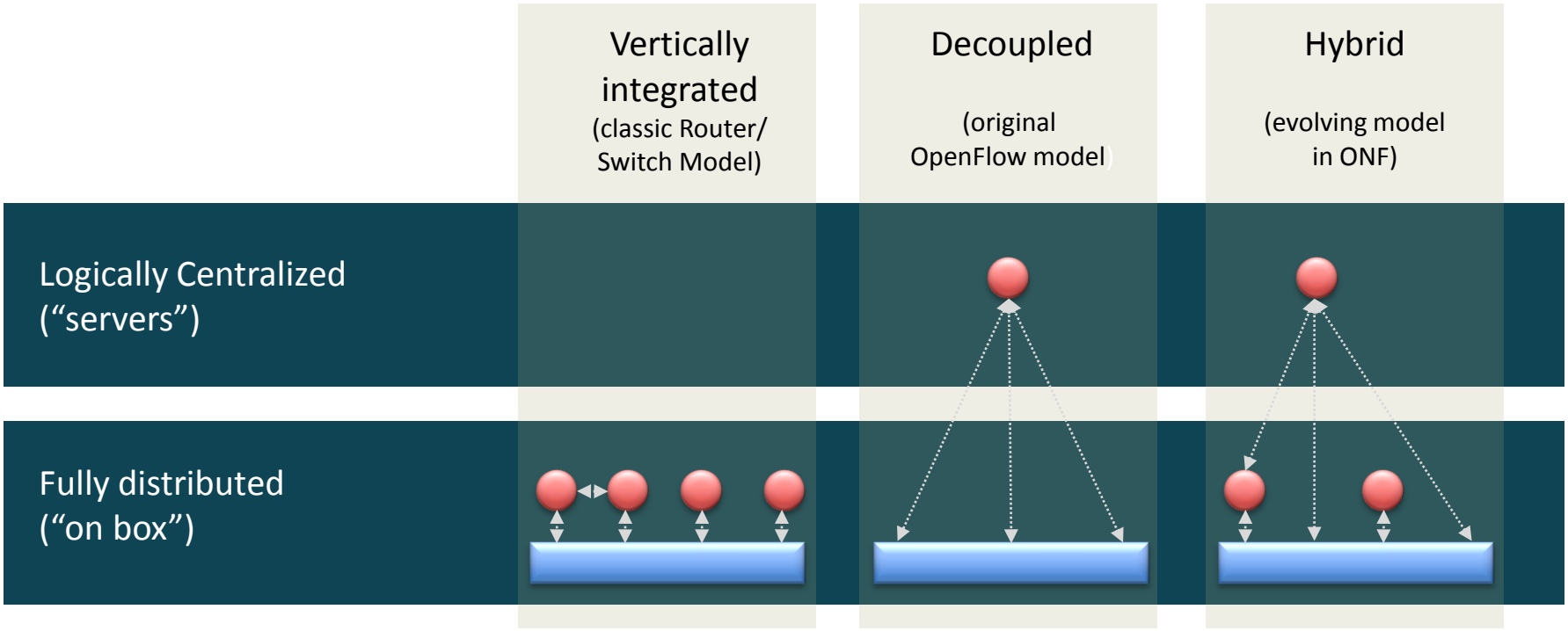
- A Few Scalability Concerns and Open Questions

Early Thinking: Horizontally Integrated Control Centralized vs. Distributed Control





Cut Another Way: Control Plane Distribution Options



Data Path jointly controlled by standard on-box control plane and centralized off-box controller

Slide courtesy Frank Brockners

Legend: Data plane Control plane function

Flow Routing vs. Aggregation

Flow-Based

Every flow is individually set up by controller
Exact-match flow entries
Flow table contains one entry per flow
Good when fine grain control is needed

Aggregated

One flow entry covers large groups of flows
Wildcard flow entries
Flow table contains one entry per category of flows
Good when there are a large number of flows with the same properties

Reactive vs. Proactive

Reactive

First packet of flow triggers controller to insert flow entries ("flow setup packet")
Efficient use of flow table
Every flow incurs additional flow setup time
If control connection lost, switch has limited utility

Proactive

Controller pre-populates flow table in switch
Zero additional flow setup time
Loss of control connection does not disrupt traffic
Essentially requires aggregated (wildcard) rules

A Few Scalability Concerns

- **Controller Scalability**
  - Flow setup scalability
    - MapReduce shuffle phase jobs can produce more than 50K flows/sec
    - Maestro study claimed as many as 100K flows/sec in MSDC setting
    - Obviously punting a "flow setup packet" per flow to a controller isn't going to be viable here
      - And the flow might not live long enough to make it worthwhile
  - What is the required latency and throughput from a transactional perspective?
    - For example, can we detect and respond to data plane state transitions on appropriate timescales?
    - More generally, what are the implications of the *loss of fate-sharing* between the control and data plane?
  - Internal DB scalability, resilience, performance, concurrency control,…
  - …
- **Switch Scalability**
  - Switch CPU and CPU/TCAM bandwidth
    - Many current ToR switches are can install <= 300 flows/sec (PCI limited)
  - Classifier Scalability
    - If classification is implemented by a TCAM, then tag-field bits, number of rows, and power profile are all at a premium
  - TCAM Lookup/Insertion Scalability
    - TCAM designs usually trade off efficient lookups against insertion costs (which can be O(#rows))
  - TCAM optimization
  - Note OF v1.X, X > 0 Lookup Latency
    - Table chaining → lookup no longer O(1)
  - …

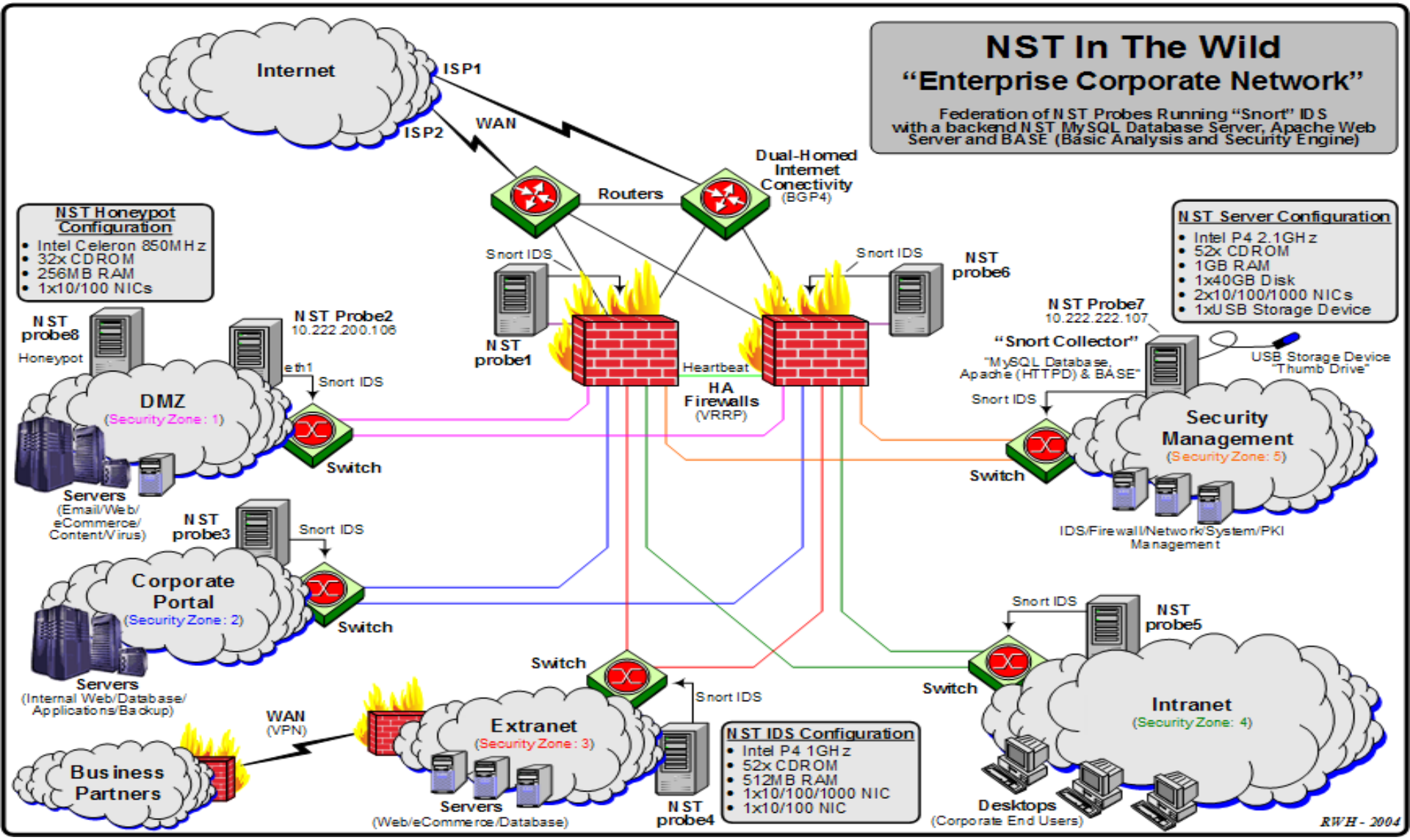
So What Does All This Buy You? (re-Agenda)

- **Problem Space**
  - **Review what problem(s) are we trying to solve?**
- **A Few Use Cases**
- **Reflections on the Promise of OF/SDN**
- **A Few Challenges and Open Questions**

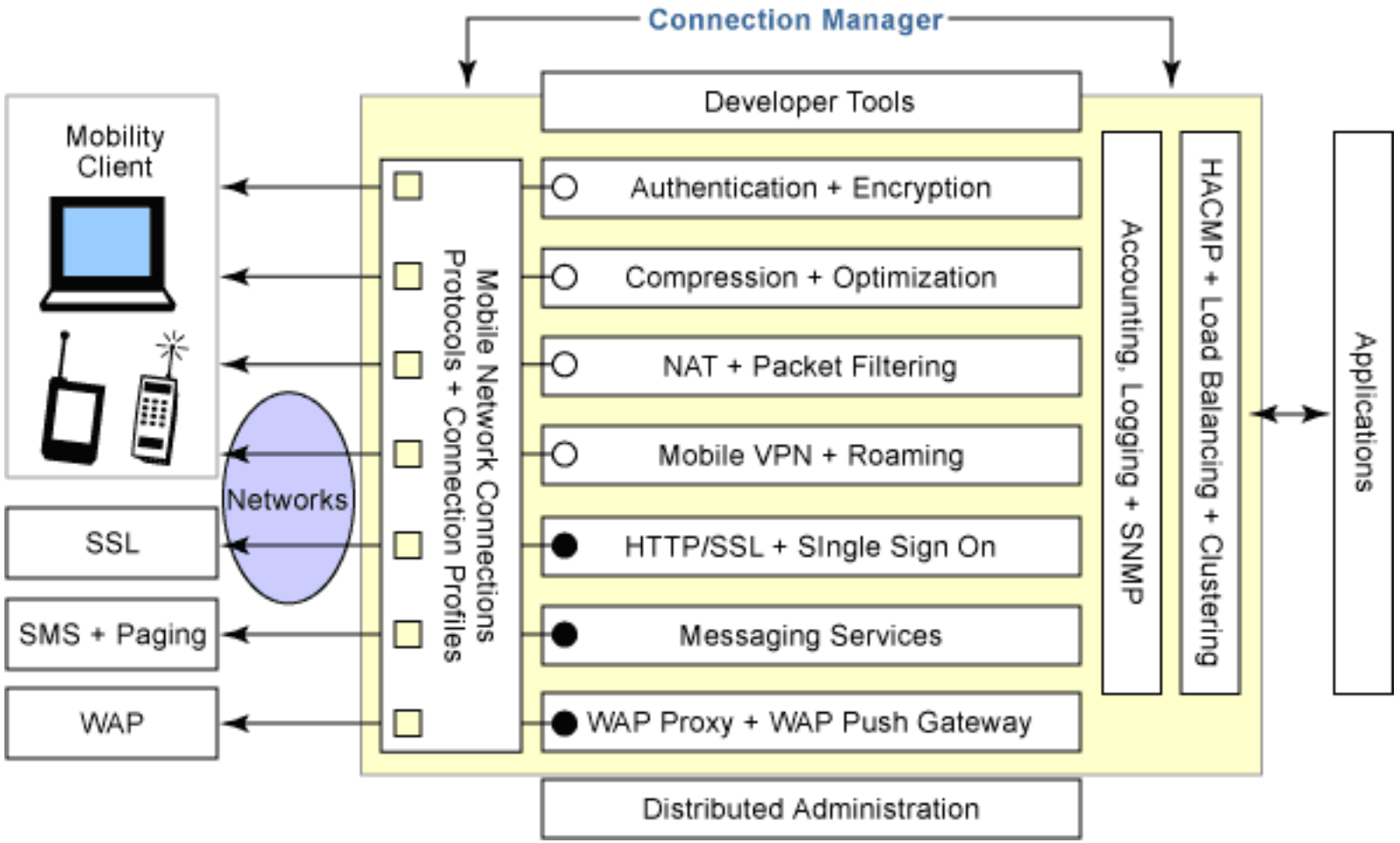
Problem Space

- Network architects, engineers and operators are being presented with the following challenge:
  - ***Provide state of the art network infrastructure and services while minimizing TCO***
- Thesis: It is the lack of ability to innovate in the underlying network coupled with the lack of proper network abstractions results in the inability to keep pace with user requirements and to keep TCO under control
- Surprisingly general problem (or maybe not so surprising…)
  - "I have a new iPhone I want to work on the campus network…"
  - "I need my data centers to have green networking…"
  - → Constant stream of new requirements
- So what's the problem?

Maybe this is the problem?

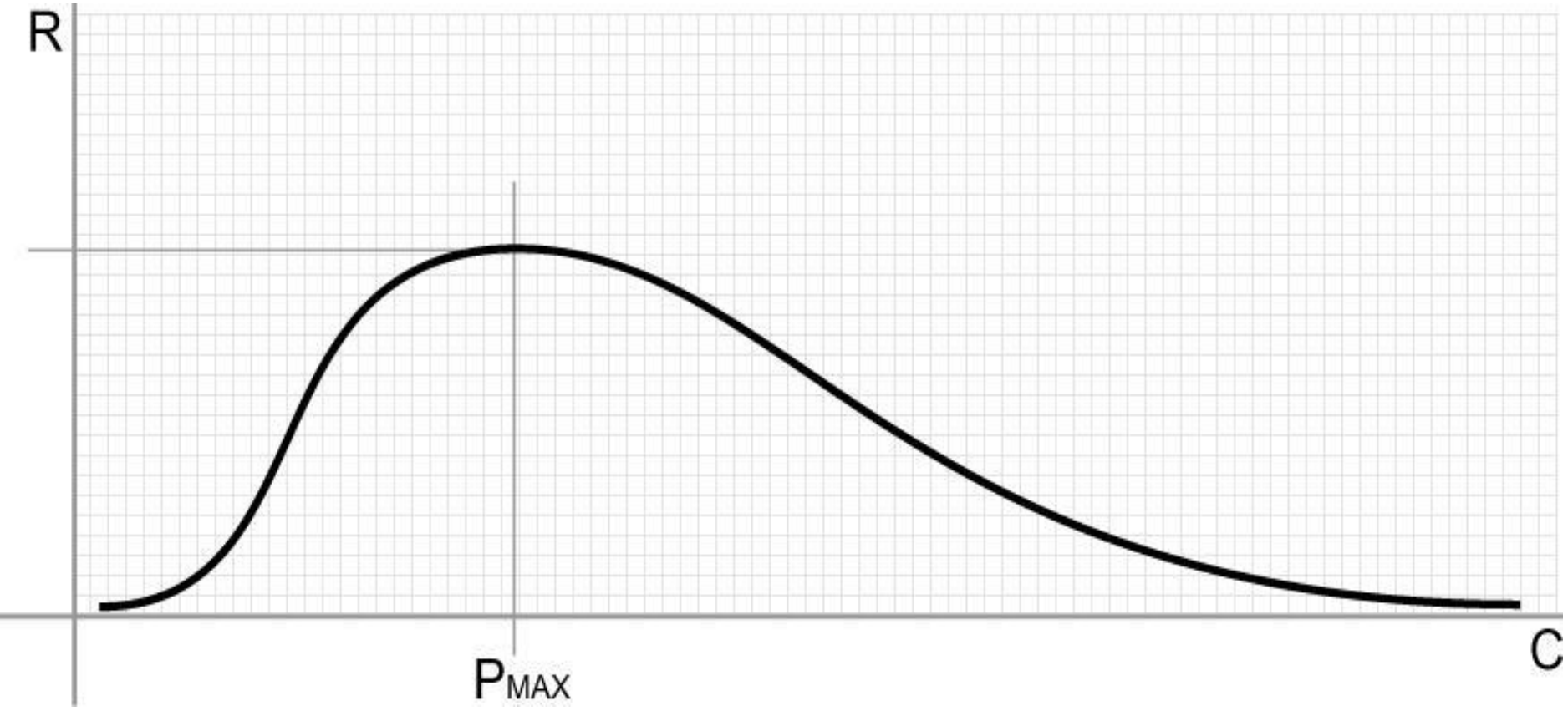


Basically, protocol soup…



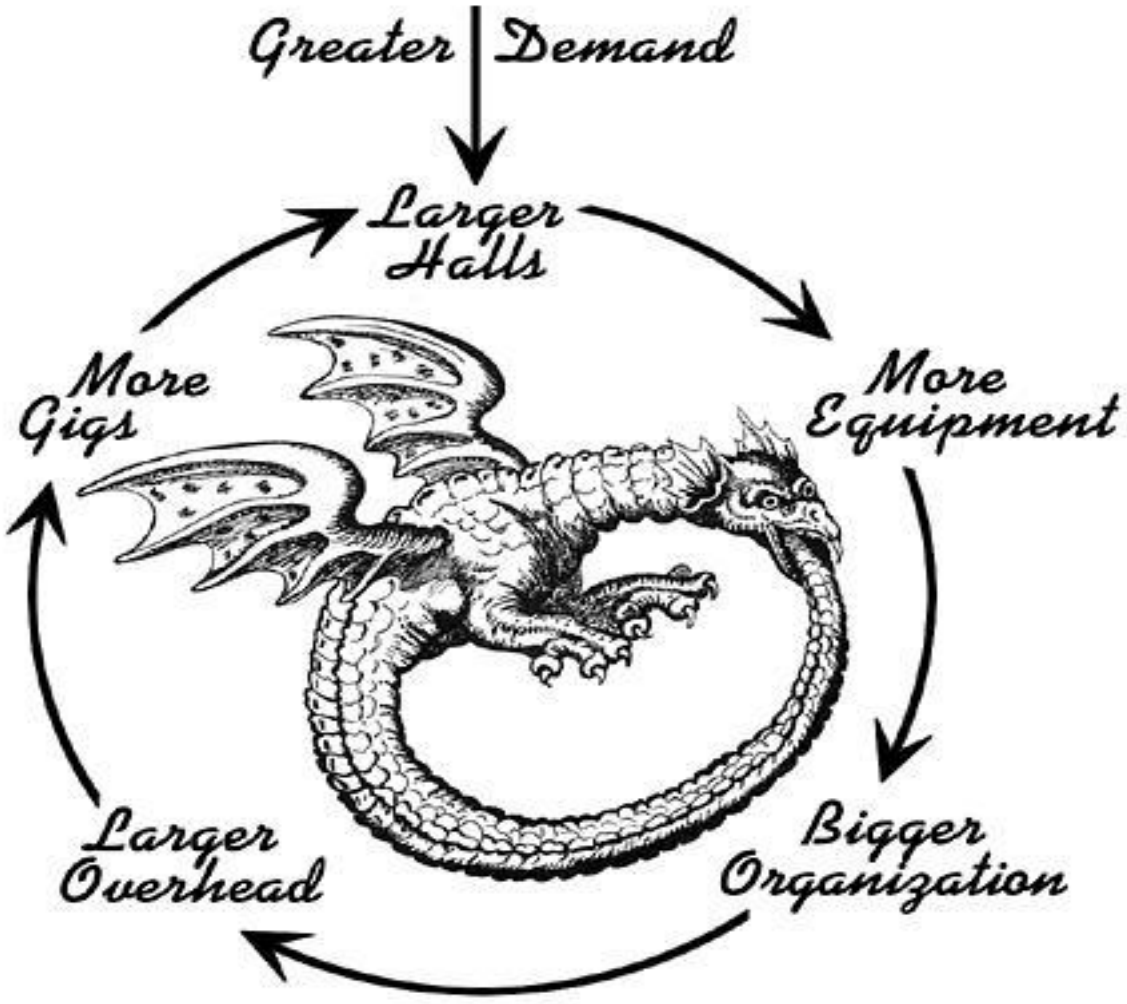
Many protocols, many touch points, few open interfaces or abstractions, … →

Robustness vs. Complexity
"Systems" View



Increasing number of protocols, configurations and interactions

Complexity/Robustness Spirals



See J. Doyle, et. al., "Robustness and the Internet: Theoretical Foundations"

What Tools Does OF/SDN Give Us Address This Growing Complexity?

- **Forwarding abstraction**
  - OpenFlow/Flow Space
- **Distributed State Abstraction**
  - Global Network View (Logical and Virtual)
- **Distributed Network Control Abstraction**
  - Network Operating System (NOS)

These Abstractions Give Us…

- **A unified way of thinking about network "boxes"**
  - Routers, Switches, Firewalls, NATs, Load Balancers,...
  - → forwarding planes
- **A centralized view of the network**
  - Simplified control logic/operations
- **Mechanisms for decoupling policy from configuration**
  - OF/SDN doesn't do this directly, but rather makes such decoupling possible

A Few Use Cases

- **Dynamic Network Access Control**
  - I'll say more about this in a minute
- **Load Balancing**
  - Network
  - Server
  - Latency Equalized Routing
- **Distributed Firewalls and NATs**
- **Network Virtualization**
  - Enterprise Cloud (XaaS)
  - VM Migration and User Mobility
  - Virtual Edge Provisioning
  - "Slicing"
- **Energy-Efficient/Proportional Networking**
- **Adaptive Network Monitoring**
- **Conventional Control Planes**
  - e.g., routeflow project
  - https://sites.google.com/site/routeflow/

So What is The Promise of OF/SDN?

- Consider “Decoupling Policy from Configuration in Campus and Enterprise Networks” by Nick Feamster and colleagues at GA Tech:
  - http://www.gtnoise.net/papers/2010/feamster:lanman2010.pdf

- The study shows how the unified view of network devices coupled with a global network view provided by OF/SDN allows an elegant decoupling of policy and configuration in the context of wireless LAN access control

Typical Registration on a Wireless LAN

During registration, **systems are scanned** for known vulnerabilities. If the scan reveals vulnerabilities, the user is presented with these vulnerabilities and given an opportunity to **update the system**. The **firewall** for the network allows traffic to get to the appropriate update servers for Microsoft and Apple. The **registration VLAN** uses a firewall to block network traffic to unregistered desktops. However, **the firewall allows Web and secure Web** (i.e., port 80 and 443) traffic to pass so that desktop machines can reach update sites. **Various routers and switches are employed to facilitate creating the VLANs necessary for the needed networks**. The local switches determine which VLAN for each machine that joins the network. The **switch will download VLAN maps** periodically from a VMPS. **Unknown MAC addresses are assigned to the unregistered VLAN and known MAC addresses are placed onto the appropriate subnet.** VMPS periodically downloads the VLAN maps from the registration server**. Network security is enforced by creating ARP tables that map each MAC address to it registered IP and pushing that table to each router.**

Rather….Define State Transitions for a Host

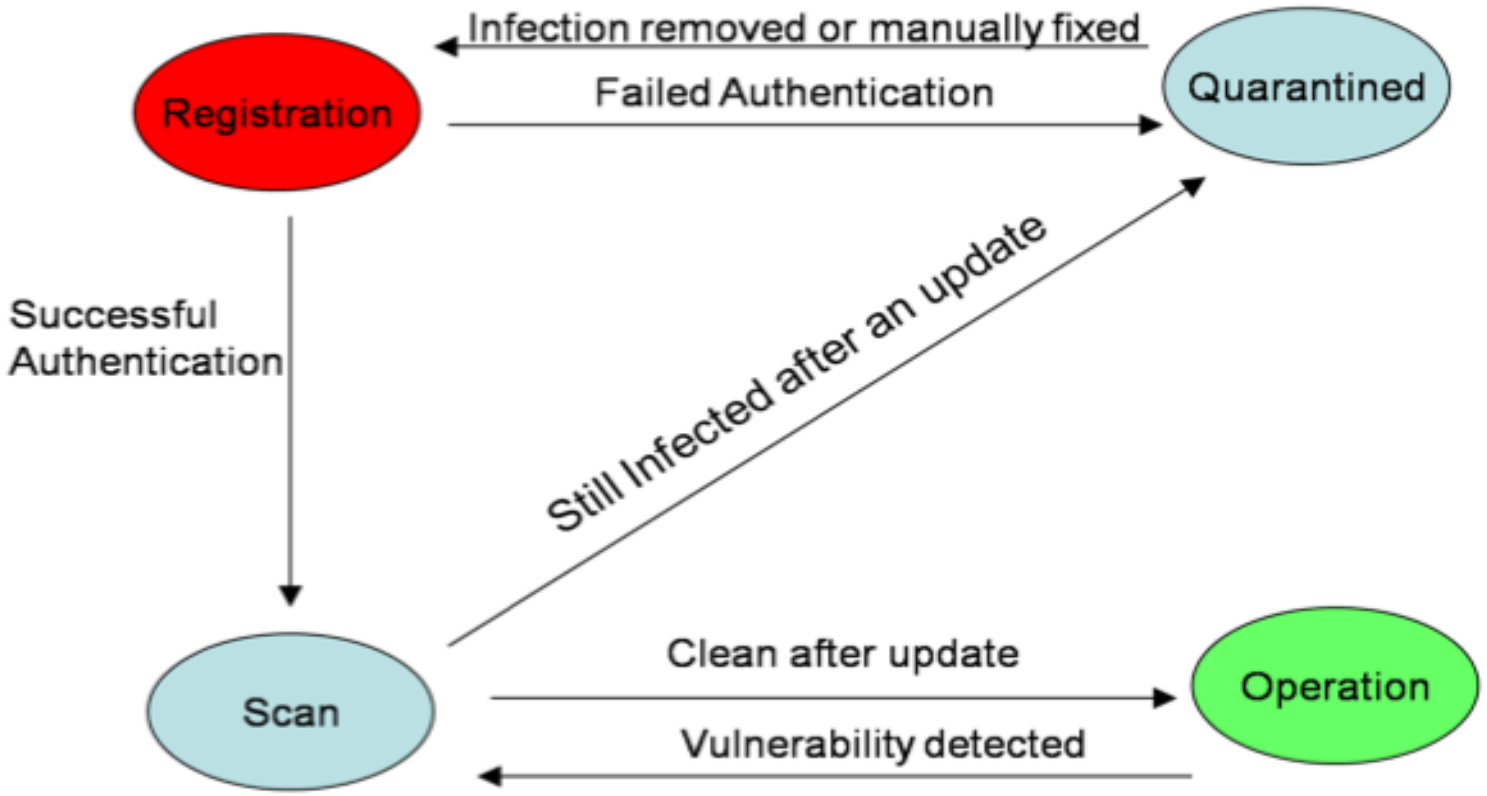


Fig. 2. State transitions for a host. The controller tracks the state of each host and updates the current state according to inputs from external sources (*e.g.*, network monitors).

Decoupling Policy and Configuration

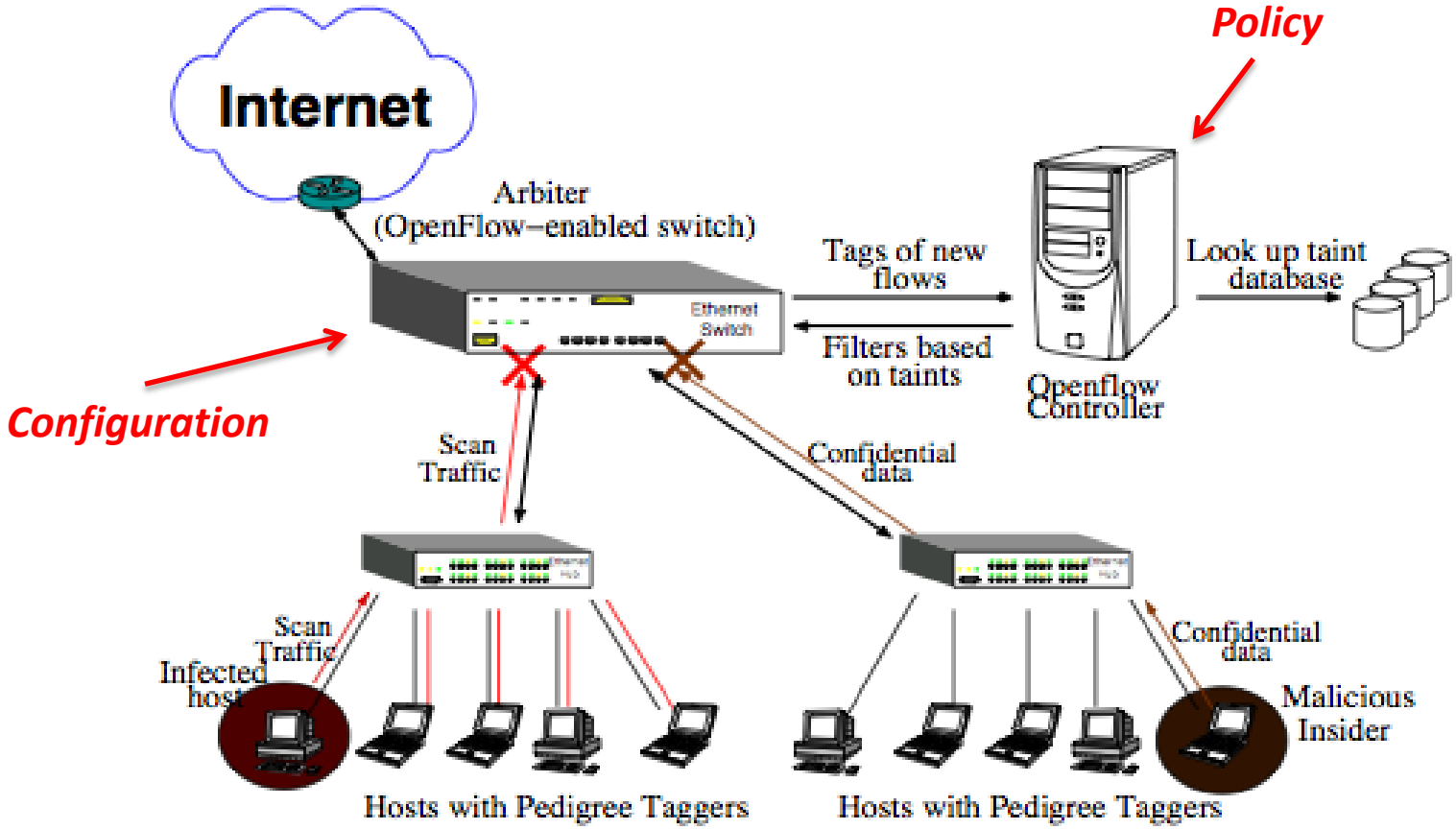


Fig. 5. Pedigree deployment in an enterprise network, where a malicious host is trying to scan its local network.

A Few Challenges/Open Questions

- **Theoretical/Scientific Foundations for SDN**
  - A new way of thinking about networking with a wide variety of intellectual challenges
  - Distributed systems, theory, network architectures, programming languages, …
- **Stabilize/Extend the OpenFlow Protocol**
  - Work well underway in the ONF
- **Programming Languages/Models/Systems**
  - Composition of applications
    - Part of the goal of Frenetic
  - Generic ways to handle streams
    - Functional Reactive Programming promising (e.g. Nettle)
  - Tool chains and offline processes/simulation (e.g., mininet)
- **Hybrid Switch Model**
  - Interaction between on-board control planes and external control planes (why is this hard?)
  - New “Control-Plane/SDN (CP/SDN)” models – I’ll talk more about this on the next slide
- **NOS/Network Hypervisor Scalability**
  - Proactive vs. reactive flow instantiation
  - Transactional throughput
  - Distributed/replicated controller infrastructure
  - Switch Heterogeneity
- **Operational Practices and Tools**

Thanks!