And Now For Something Completely Different…

OpenFlow: Today's Reality, Tomorrow's Promise? An Architectural Perspective

David Meyer
CTO and Chief Scientist, Brocade
Director, Advanced Technology Center, University of Oregon
Ethernet Summit 2013
dmm@{brocade.com,uoregon.edu,1-4-5.net,…}

Agenda

- A Brief History of the (OF/SDN) Universe
- What is Today's OpenFlow?
- What is the OpenFlow/SDN Architecture?
- Thinking Architecture: Thin waists and hourglasses
- Tomorrow's OpenFlow/SDN Architecture?
- Summary

Danger Will Robinson!!!

This talk is intended to be controversial/provocative (and maybe a bit "sciencey")

A Brief History of the OF/SDN Universe

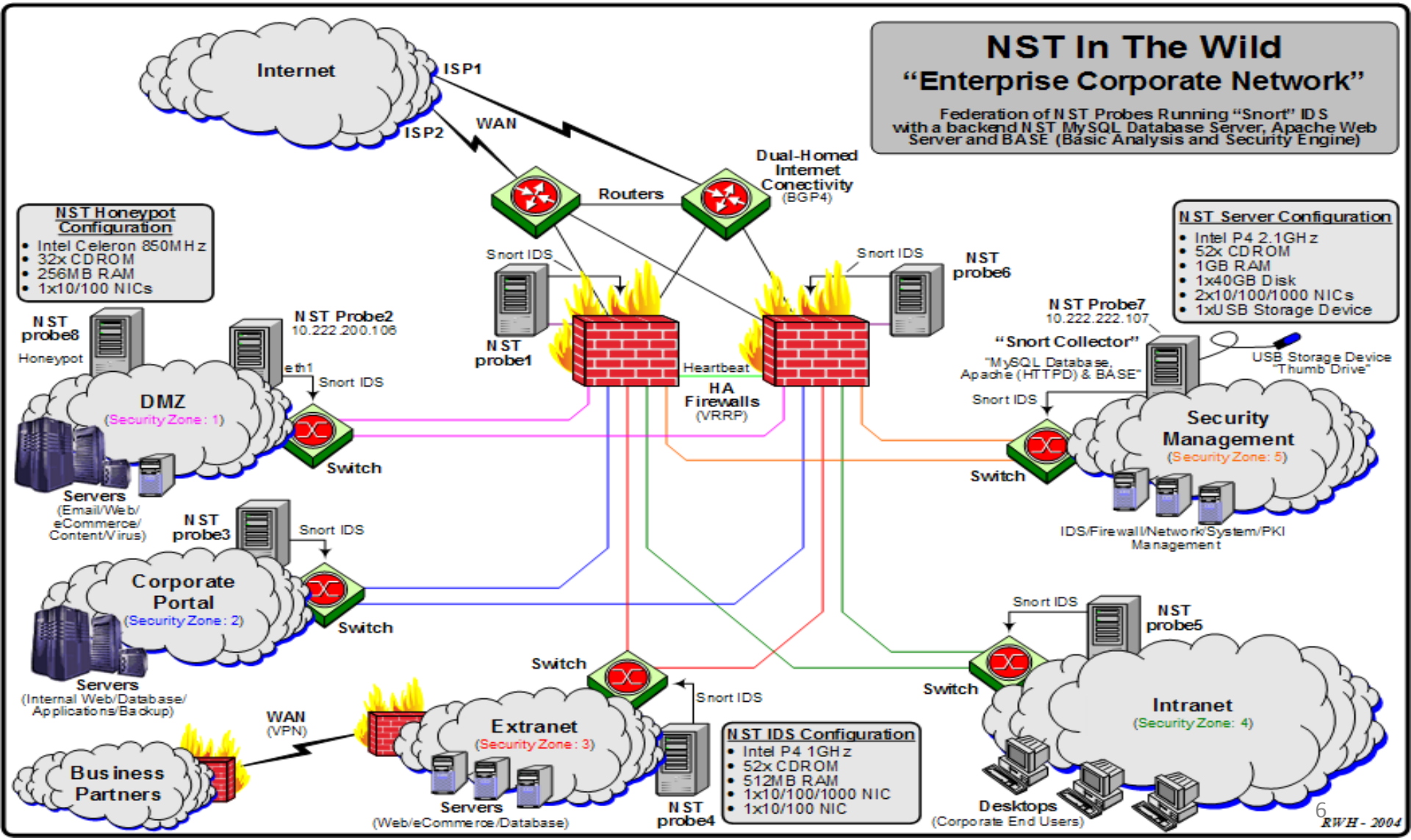
- ***Problem statement:*** Network architects, engineers and operators are being presented with the following challenge:

  - ***Provide state of the art network infrastructure and services while minimizing TCO***

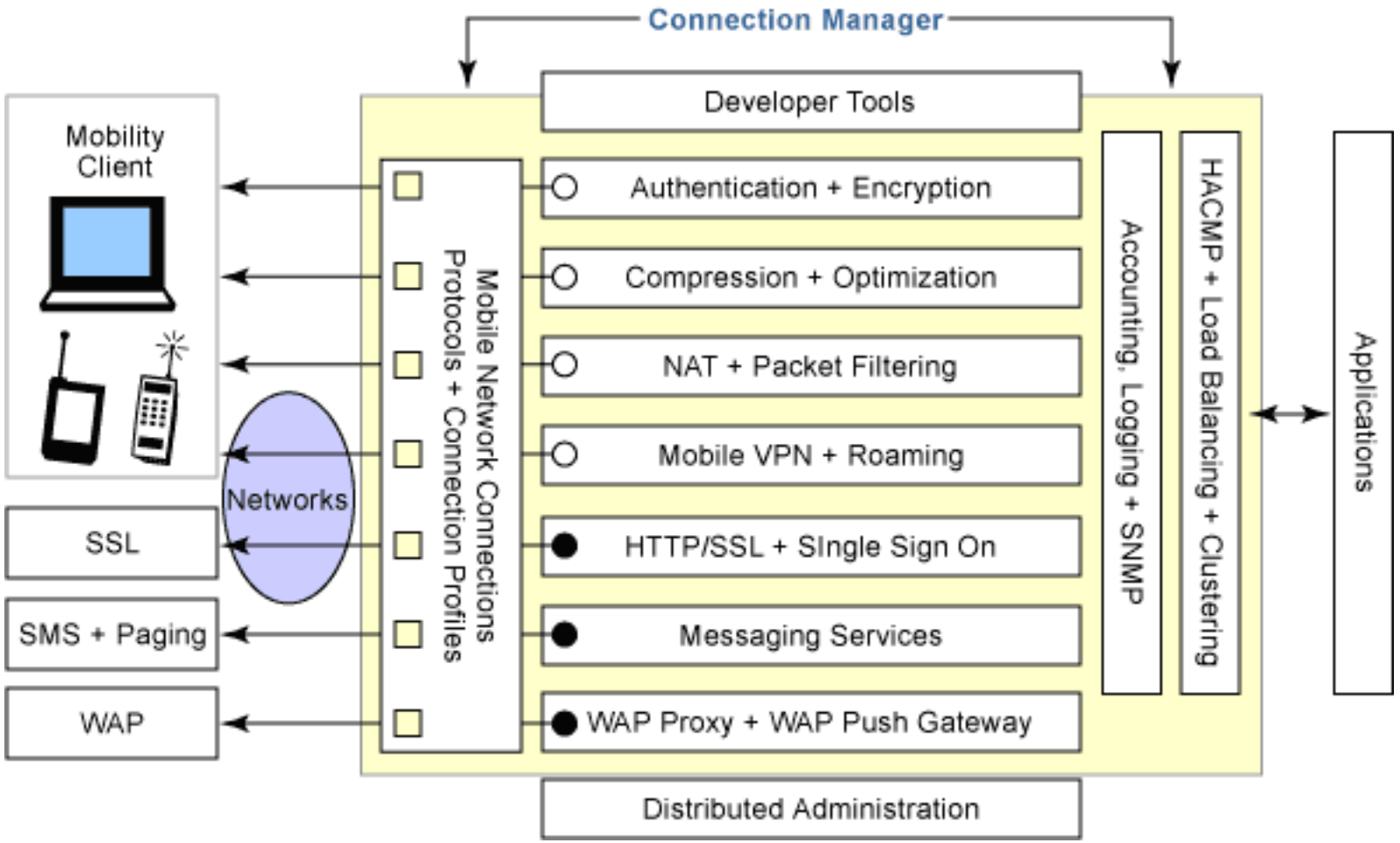
- ***SDN Hypothesis***: It is *the lack of ability to innovate in the underlying network* coupled with the lack of proper network abstractions results in the inability to keep pace with user requirements and to keep TCO under control.
  - Requirements stated informally, out of band, statically, …
  - Better done by machine
    - Obviously we need programmatic automation of config, monitoring, management, …
  - For the most part true, but do we need to change the architecture to solve this?
  - Hold that question…

- So given this hypothesis, what was the problem?

Maybe this is the problem?



Or This?



Many protocols, many touch points, few open interfaces or abstractions,..

Network is Fragile, but is that the problem? BTW, what is fragility/robustness?

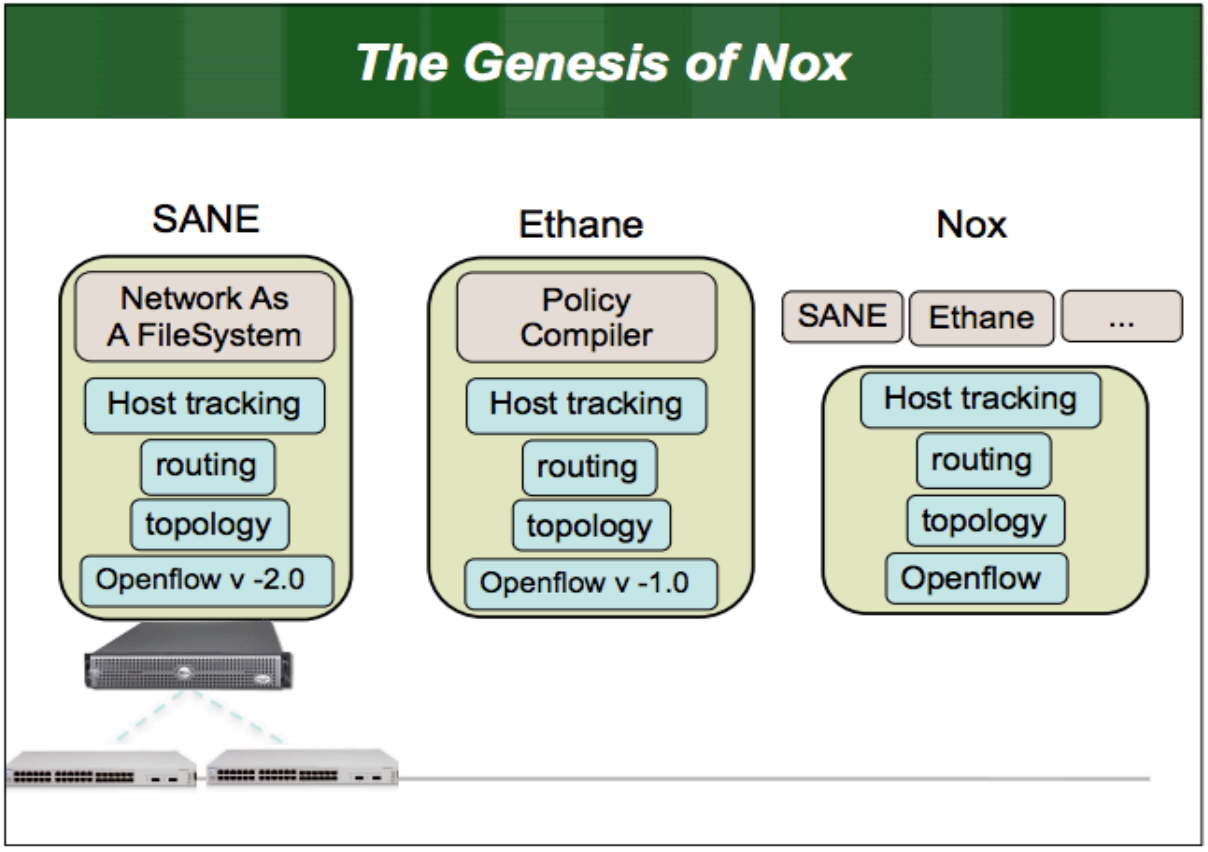
History: How Did We Get Here?

Basically, everything *networking* was too vertically integrated, tightly coupled, non-standard.

Goes without saying that this made the job of the network researcher almost impossible (before the era of overlays).

Question: Does the OF/SDN architecture map to the break down of vertical integration in the compute world?

(in)SANE



Slide courtesy Martin Cassado

OpenFlow Switch, v 1.0

(Gates 104 Crew)

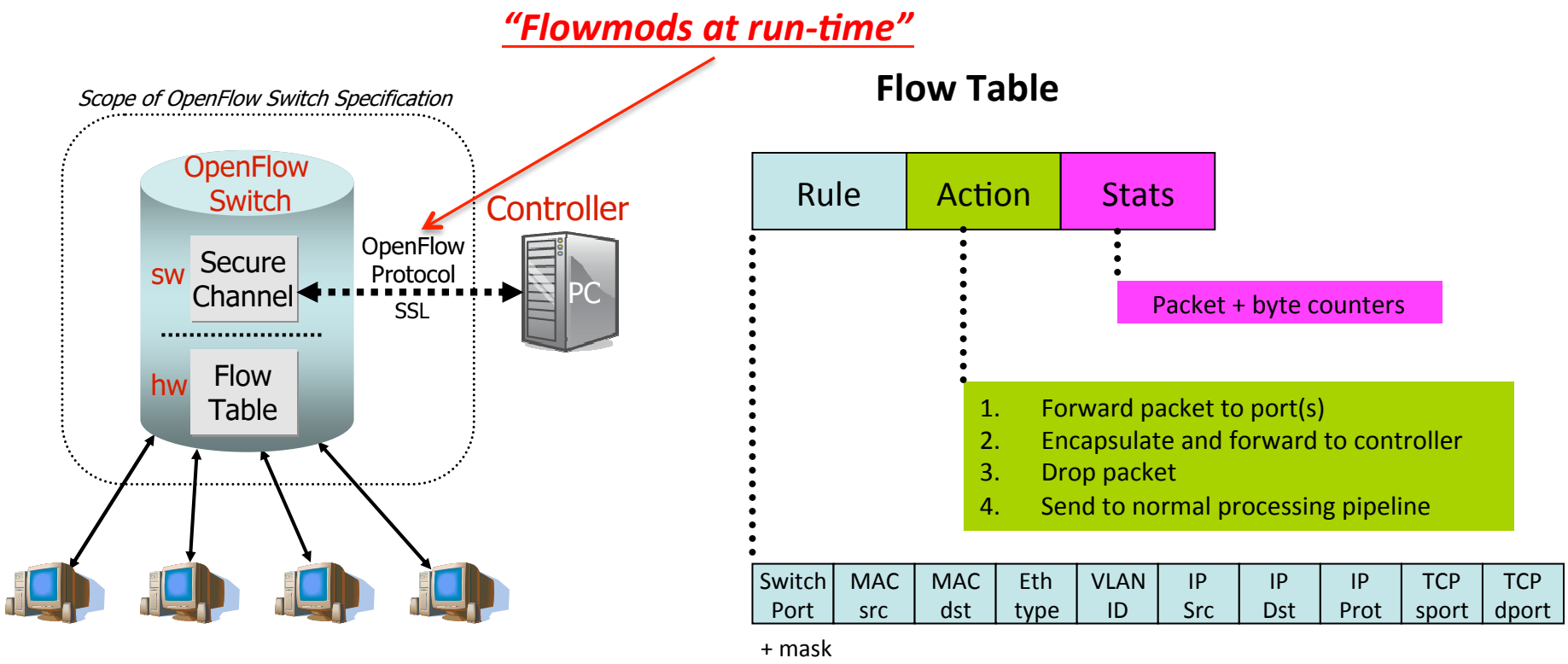
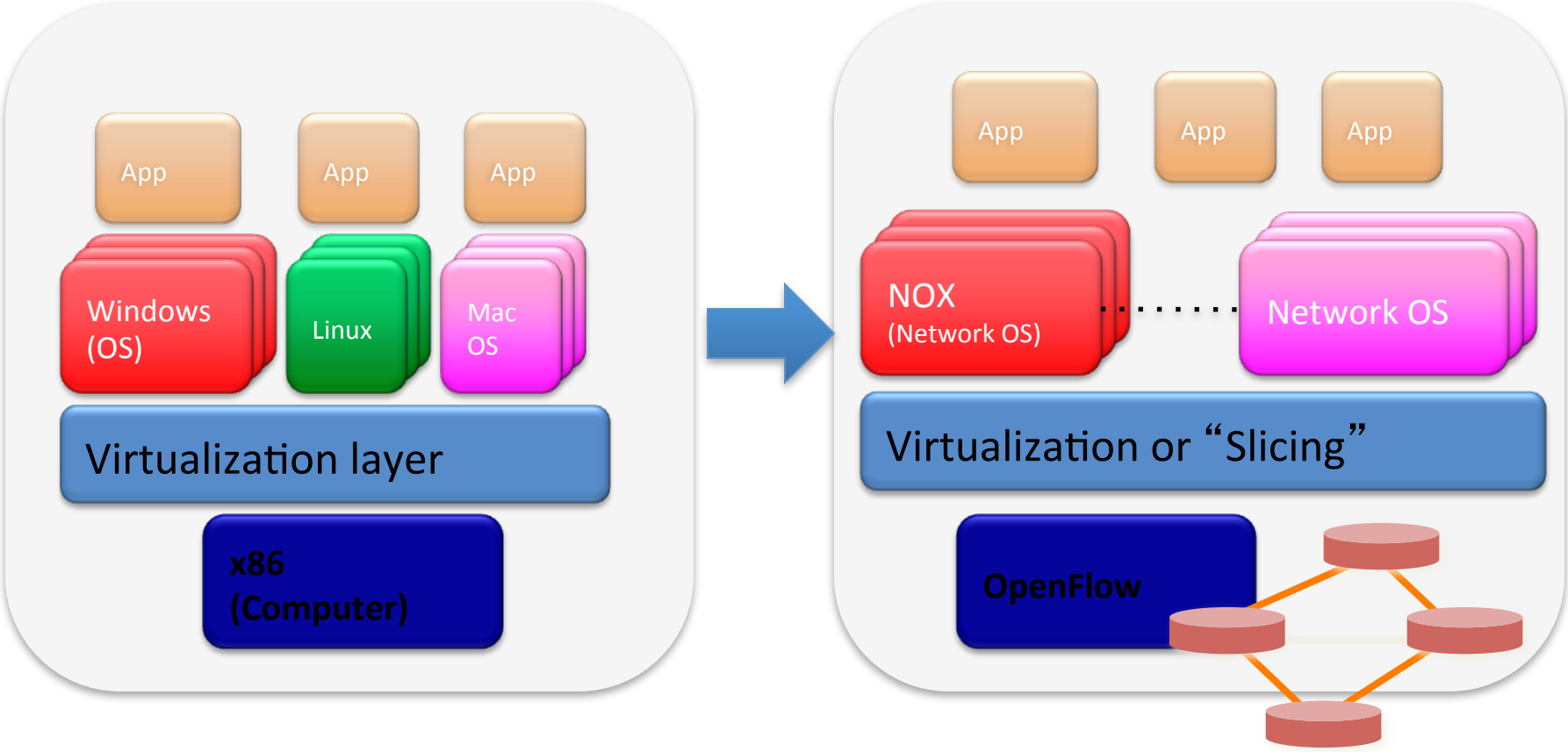


Figure 1: Idealized OpenFlow Switch. The Flow Table is controlled by a remote controller via the Secure Channel.

So question: Is this (architecturally) the same thing as the break-down of vertical integration in the compute world?

Trend (?)



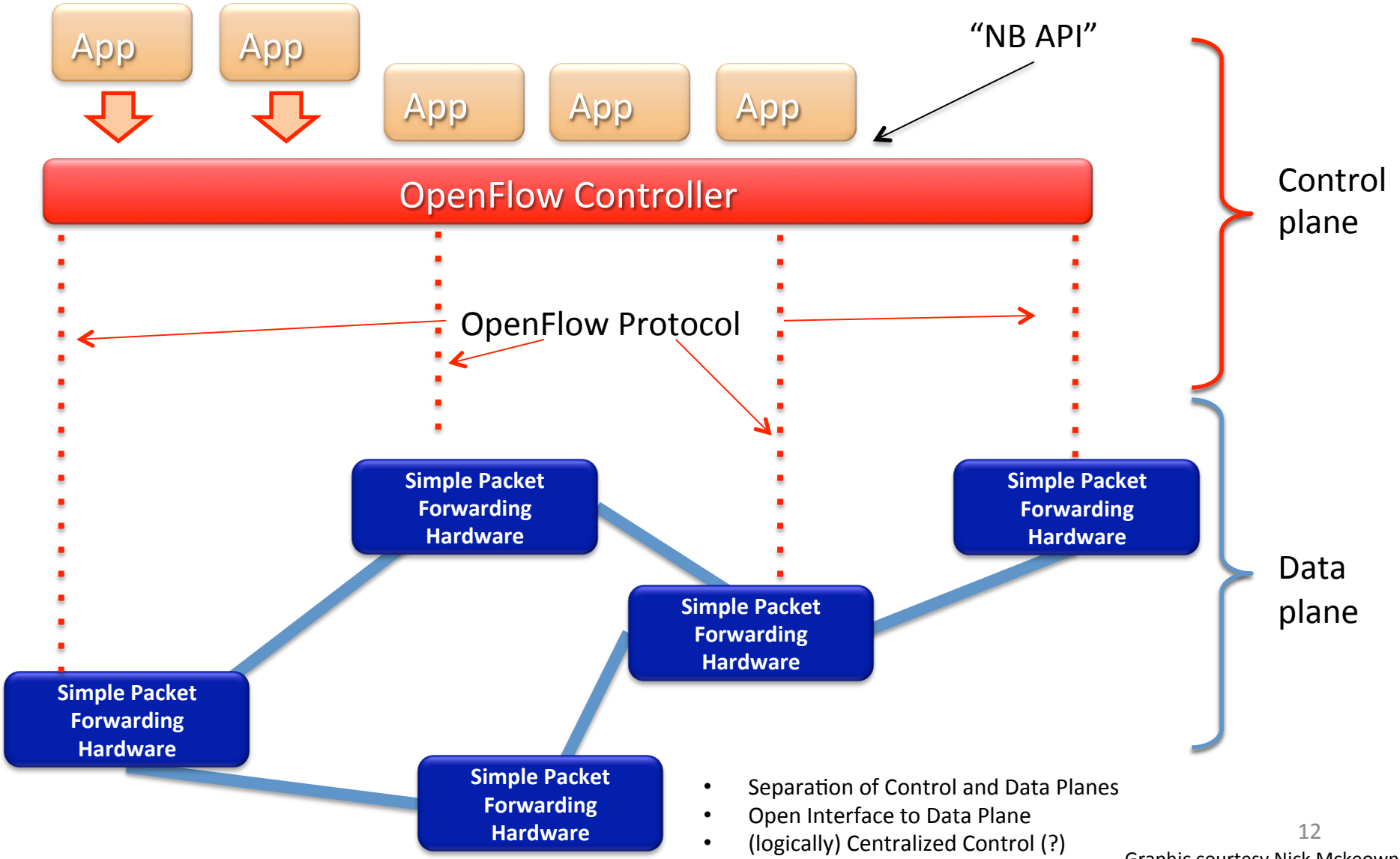
Computer Industry

Network Industry

Simple common stable hardware substrate below+ programmability + strong isolation model + competition above = Result : faster innovation

Slide Courtesy Rob Sherwood

Well…OF/SDN Architecture



- Separation of Control and Data Planes
- Open Interface to Data Plane
- (logically) Centralized Control (?)

Graphic courtesy Nick Mckeown

So Same as What Happened in Compute?

Doesn't Look Like It

Mainframe Business Model

Net Equipment	Commodity Server
Central Logic Manufacture •Proprietary & closely guarded •Single source	**Central Logic Manufacture** •Standard design (x86) •Multiple source •AMD, Intel, Via, …
Finished Hardware Supply •Proprietary & closely guarded •Single source	**Finished Hardware Supply** •Standard design •Multiple source •Dell, SGI, HP, IBM, …
System Software Supply •Proprietary & closely guarded •Single source	**System Software Supply** •Linux (many distros/support) •Windows & other proprietary offerings
Application Stack •Not supported •No programming tools •No 3rd party ecosystem	**Application Stack** •Public/published APIs •High quality prog tools •Rich 3rd party ecosystem

- **Example**:
  - Juniper EX 8216 (used in core or aggregation layers)
  - Fully configured list: $716k w/o optics and $908k with optics
- **Solution**: Merchant silicon, H/W independence, open source protocol/mgmt stack

Graphic courtesy James Hamilton,
http://mvdirona.com/jrh/TalksAndPapers/JamesHamilton_POA20101026_External.pdf

BTW, Logically Centralized?
(hint: logically centralized means distributed)

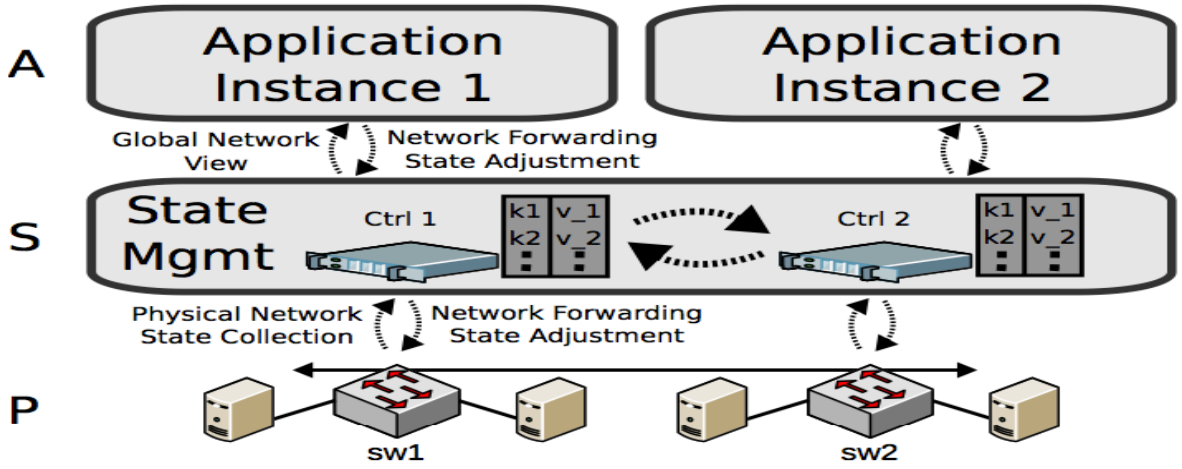


Figure 1: SDN state distribution and management conceptualized in layers: (A)pplication, (S)tate Management, (P)hysical Network

Key Observation: Logically centralized → distributed system → tradeoffs between control plane convergence and state consistency model. Note that this is beyond the *responsiveness* lower bound: *Ω(RTT(switch, controller)+ppt(switch)+ppt(controller))*

Graphic courtesy Levin, D., et. al., "Logically Centralized? State Distribution Trade-offs in Software Defined Networks?", HotSDN 2012, http://conferences.sigcomm.org/sigcomm/2012/paper/hotsdn/p1.pdf

BTW, Nothing New Under The Sun…

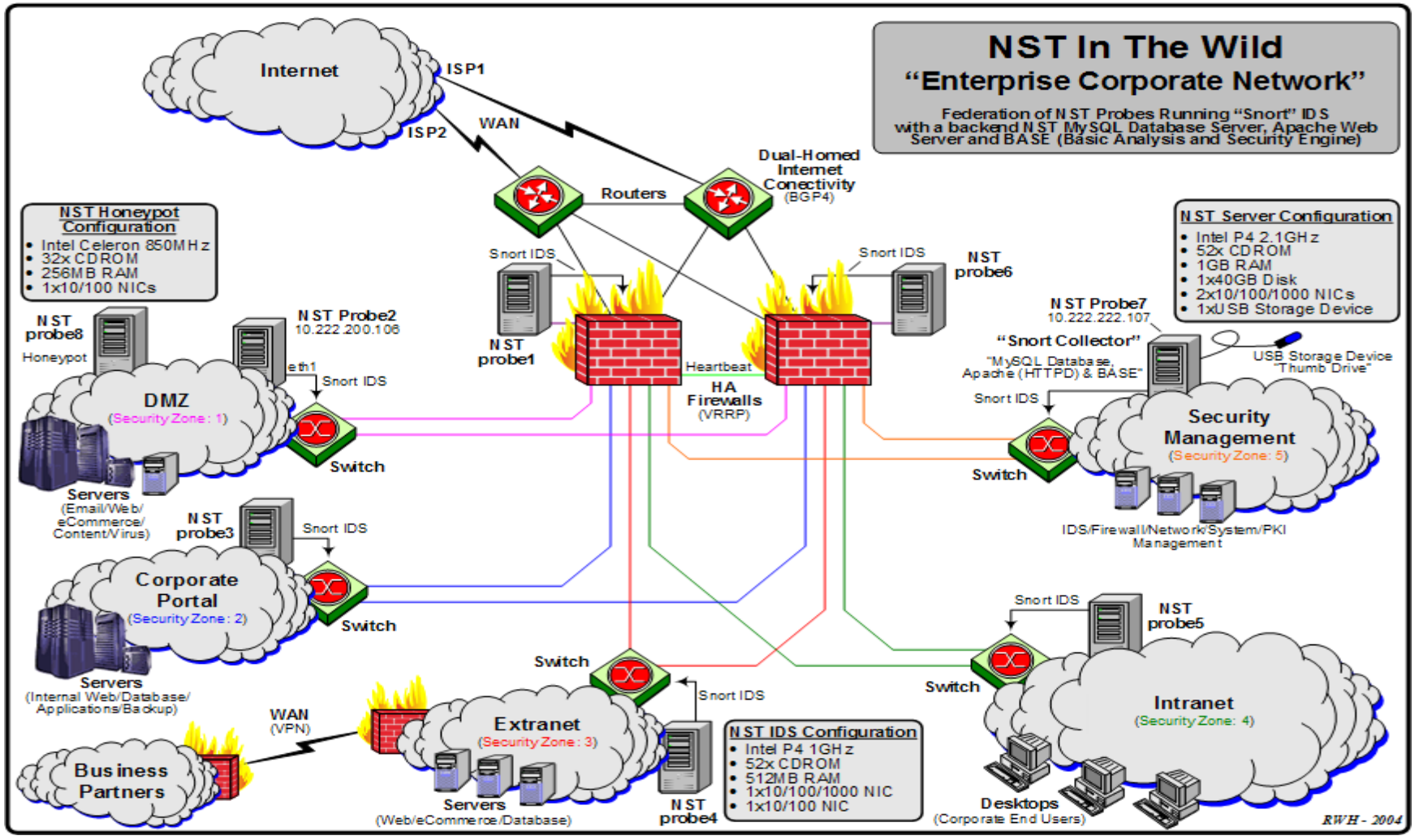
- ***Separation of control and data planes*** is not a new idea. Examples include:
  - SS7
  - Ipsilon Flow Switching
    - Centralized flow based control, ATM link layer
    - GSMP (RFC 3292)
  - AT&T SDN
    - Centralized control and provisioning of SDH/TDM networks
  - A similar thing happened in TDM voice to VOIP transition
    - Softswitch → Controller
    - Media gateway → Switch
    - H.248 → Device interface
    - Note 2nd order effect: This was really about circuit2packet
  - ForCES
    - Separation of control and data planes
    - RFC 3746 (and many others)
  - …

So Reality: What Is OpenFlow?

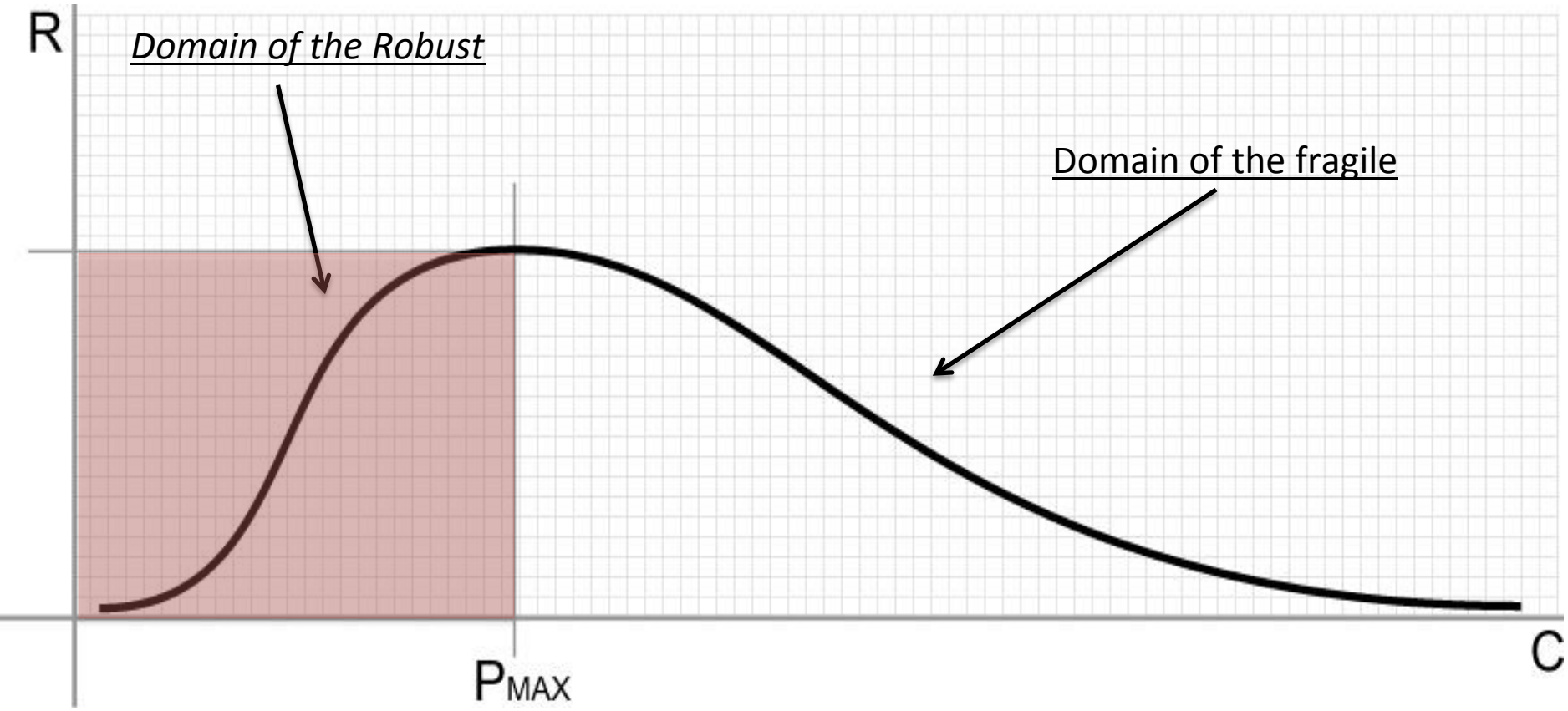
- ***A Switch Model***
  - Match-Action Tables (MAT)
    - How many, How they are connected, table attributes, etc
  - Per-flow counters
- ***An Application Layer Protocol***
  - Binary wire protocol, messages and state machine that allow programming of the MAT
- ***A Transport Protocol***
  - TLS, TCP,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

- OF says nothing about how a given forwarding target implements the switch model
  - → ***OF is an Abstract Switch Model***
  - As we'll see later, I argue that OF is built on the *wrong set of abstractions*

Thinking Architecture/Complexity (Bowties and RYF-Complexity)



Robustness vs. Complexity
Systems View



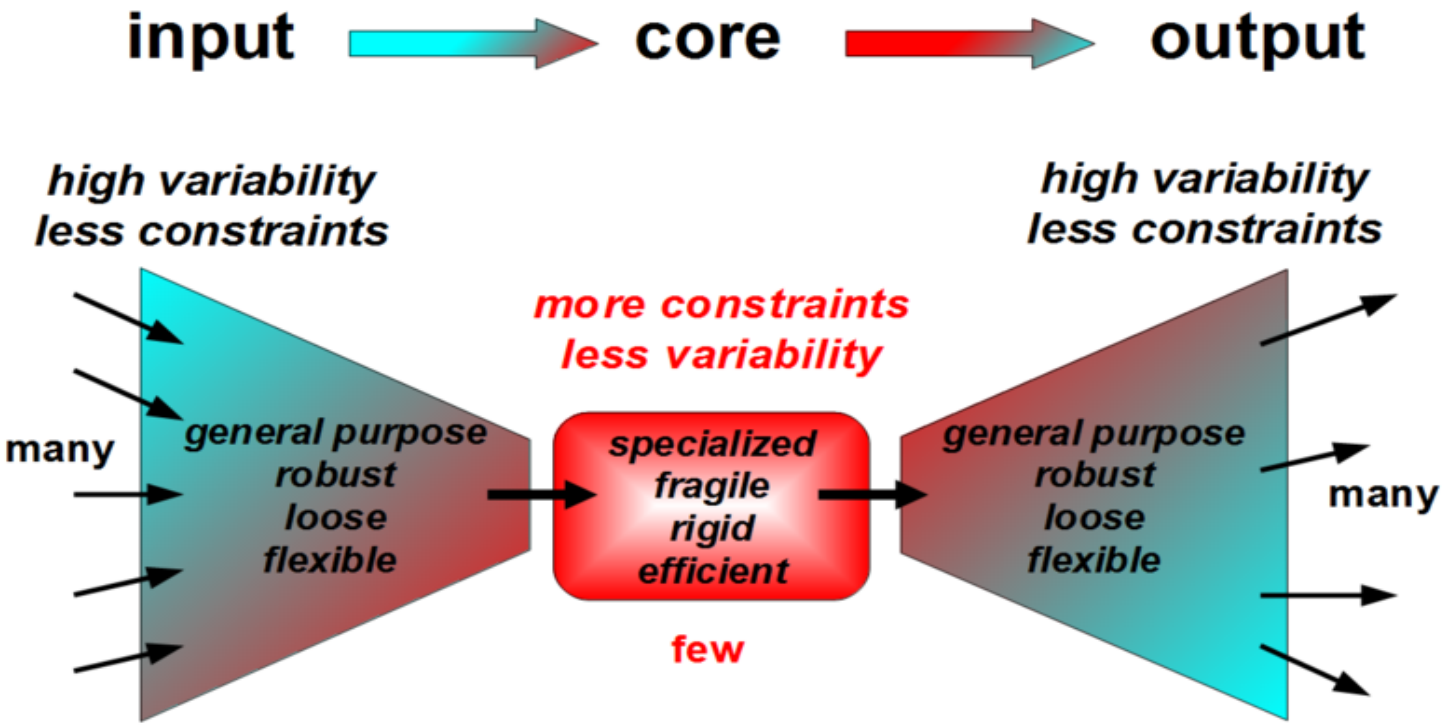
Increasing number of policies, protocols, configurations and interactions

Can we characterize the Robust and the Fragile?

Thin Waists 101: The Bowtie Architecture

Ideas from Systems Biology

Constraints that Deconstrain

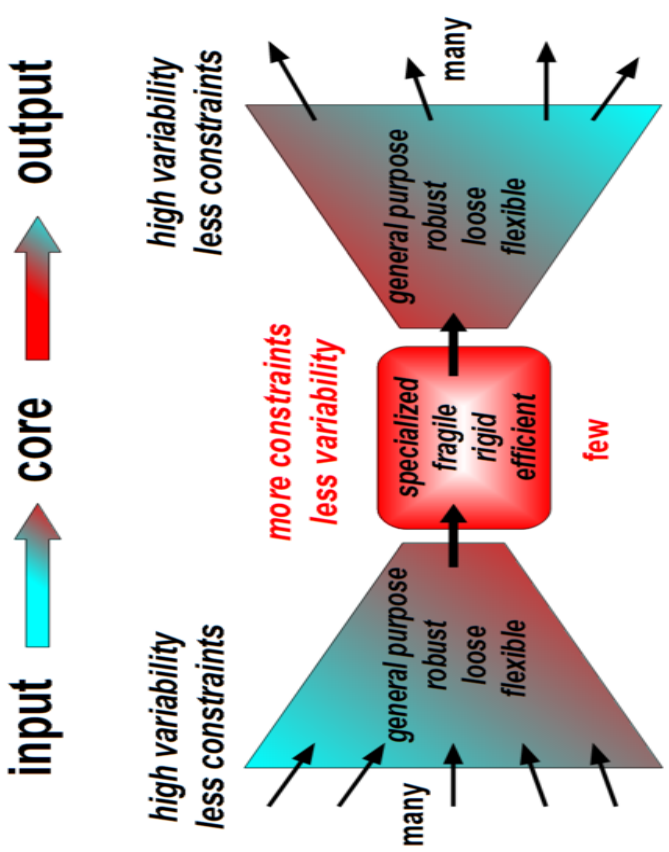


For example, the reactions and metabolites of core metabolism, e.g., ATP metabolism, Krebs/Citric Acid cycle signaling networks, …

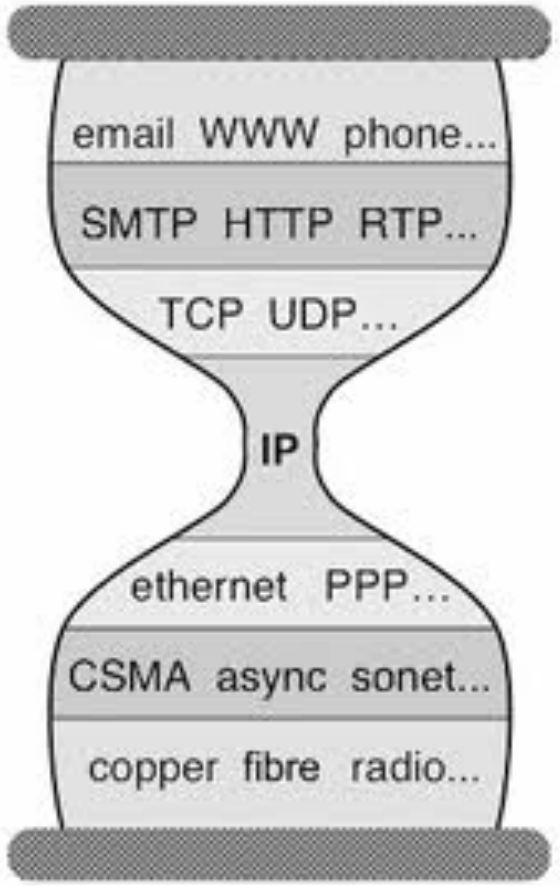
See, e.g., Doyle, et. al., "Architecture, Constraints, and Behavior", http://www.pnas.org/content/108/suppl.3/15624.full

But Wait a Second

Anything Look Familiar?

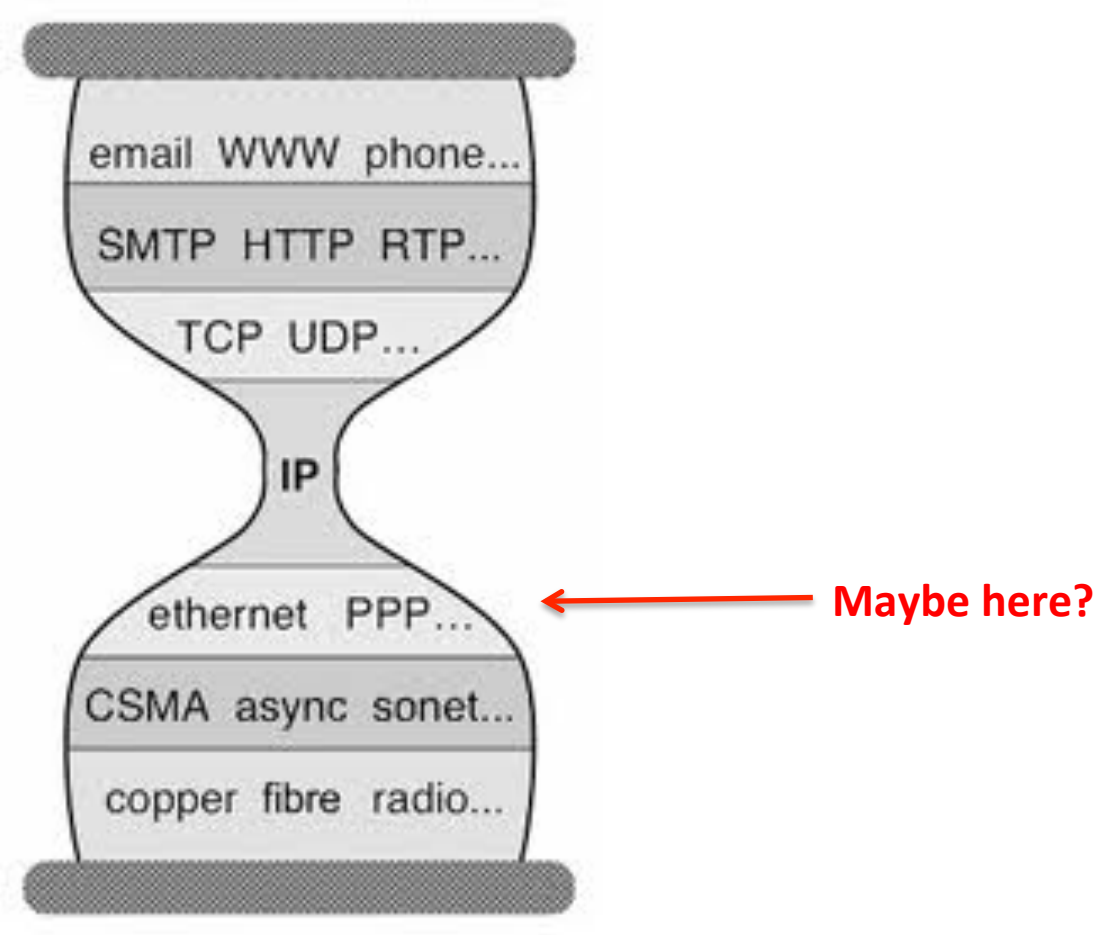


Bowtie Architecture



Hourglass Architecture

So Where Does OF/SDN Fit?



Robustness and Fragility

- ***Definition***: A *[property*] of a *[system]* is **robust** if it is *[invariant]* with respect to a *[set of perturbations], up to some limit*

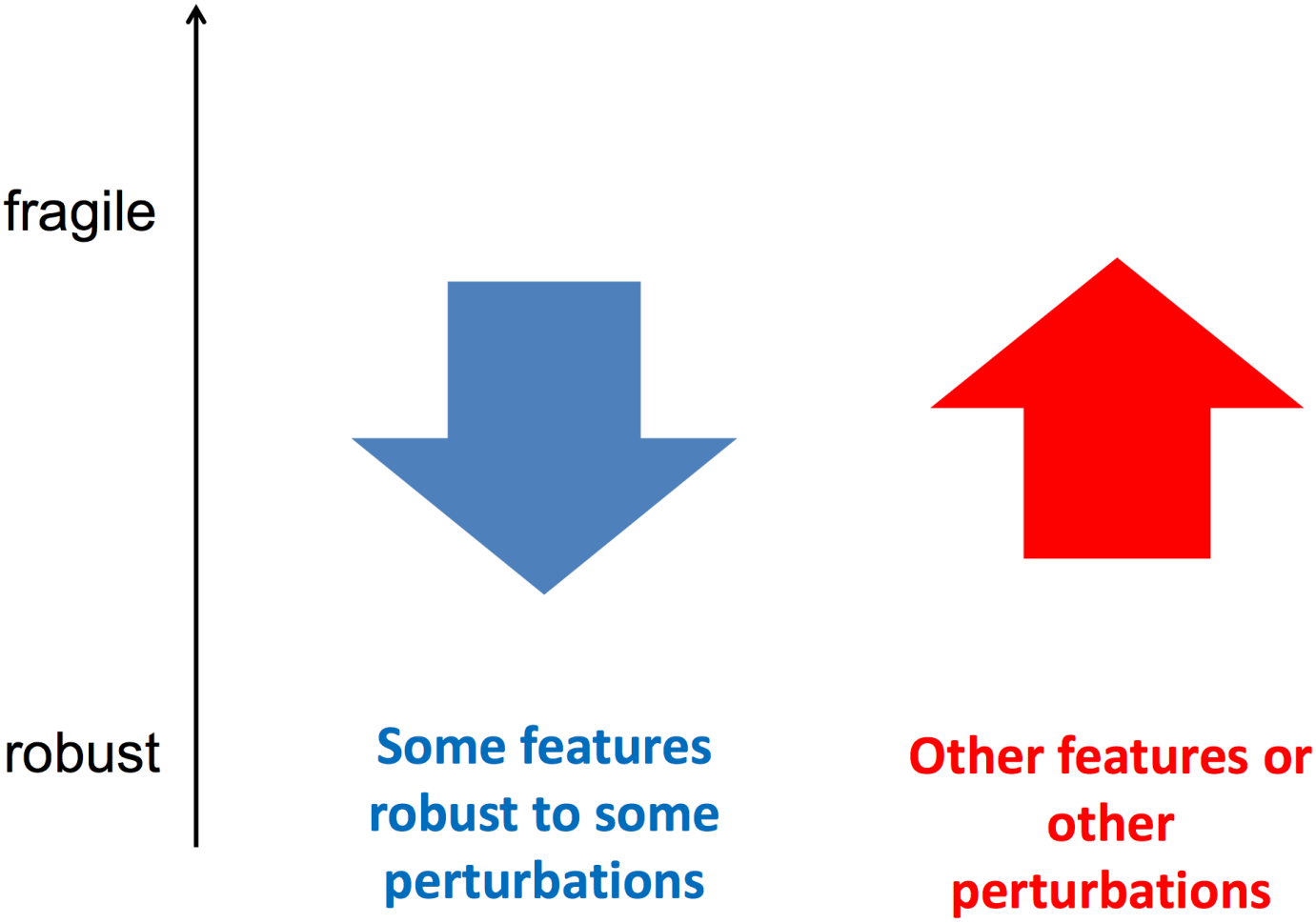
- **Fragility** is the opposite of robustness
  - If you're fragile you depend on 2nd order effects (acceleration)
  - A bit more on this in a sec…

- A system can have a *property* that is *robust* to one set of perturbations and yet *fragile* for a *different property* and/or perturbation → the system is **Robust Yet Fragile (RYF-complex)** [0]
  - Or the system may collapse if it experiences perturbations above a certain threshold (K-fragile)

- Example: A possible ***RYF tradeoff*** is that a system with high efficiency (i.e., using minimal system resources) might be unreliable (i.e., fragile to component failure) or hard to evolve

[0] http://www.istar.upenn.edu/osw/white paper/John Doyle White Paper.pdf

RYF?


fragile
robust
Some features robust to some perturbations
Other features or other perturbations

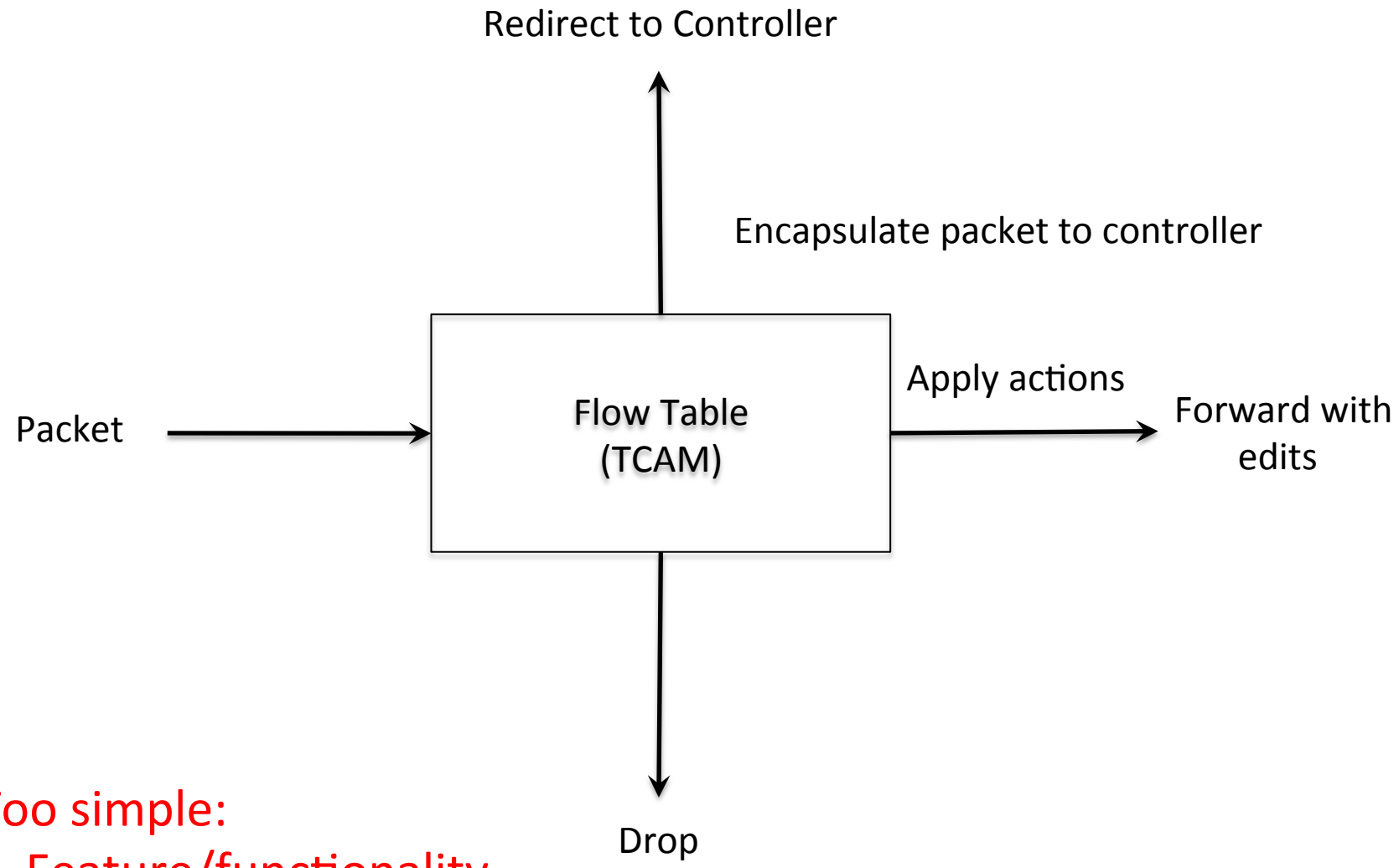
RYF Tradeoffs

Robust		Fragile
Modular	*and*	Distributed
Simple	~~*xor*~~	Complex
Plastic		Frozen
Evolvable		Frozen

Key Architectural Takeaways

- The ***fundamental architectural building*** blocks of systems that scale and are evolvable include
  - Bowties/hourglasses architectures with RYF complexity
  - Massively distributed with robust control loops
  - Highly layered
  - Protocol Based Architecture (PBA)
  - Degeneracy
- OF/SDN?
  - Centralized (at least “logically”)
  - Delayered
  - Non-degenerate
    - In fact, consider NfV in this context…

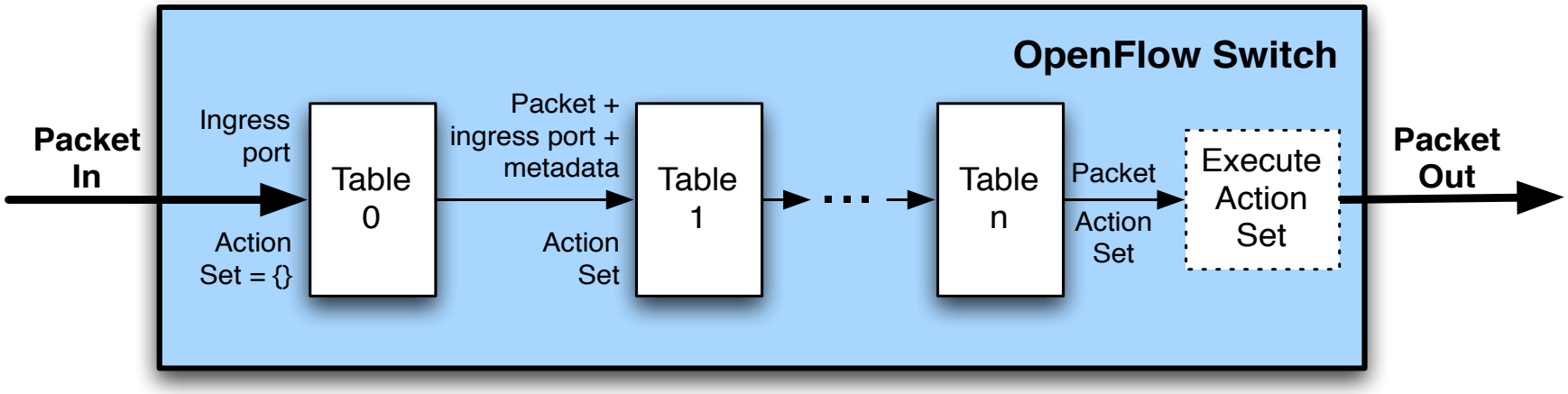
Review: OF Version 1.0



Too simple:

- Feature/functionality
- Expressiveness/Lossy Abstractions

OK, Fast Forward to Today: OF 1.1+



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

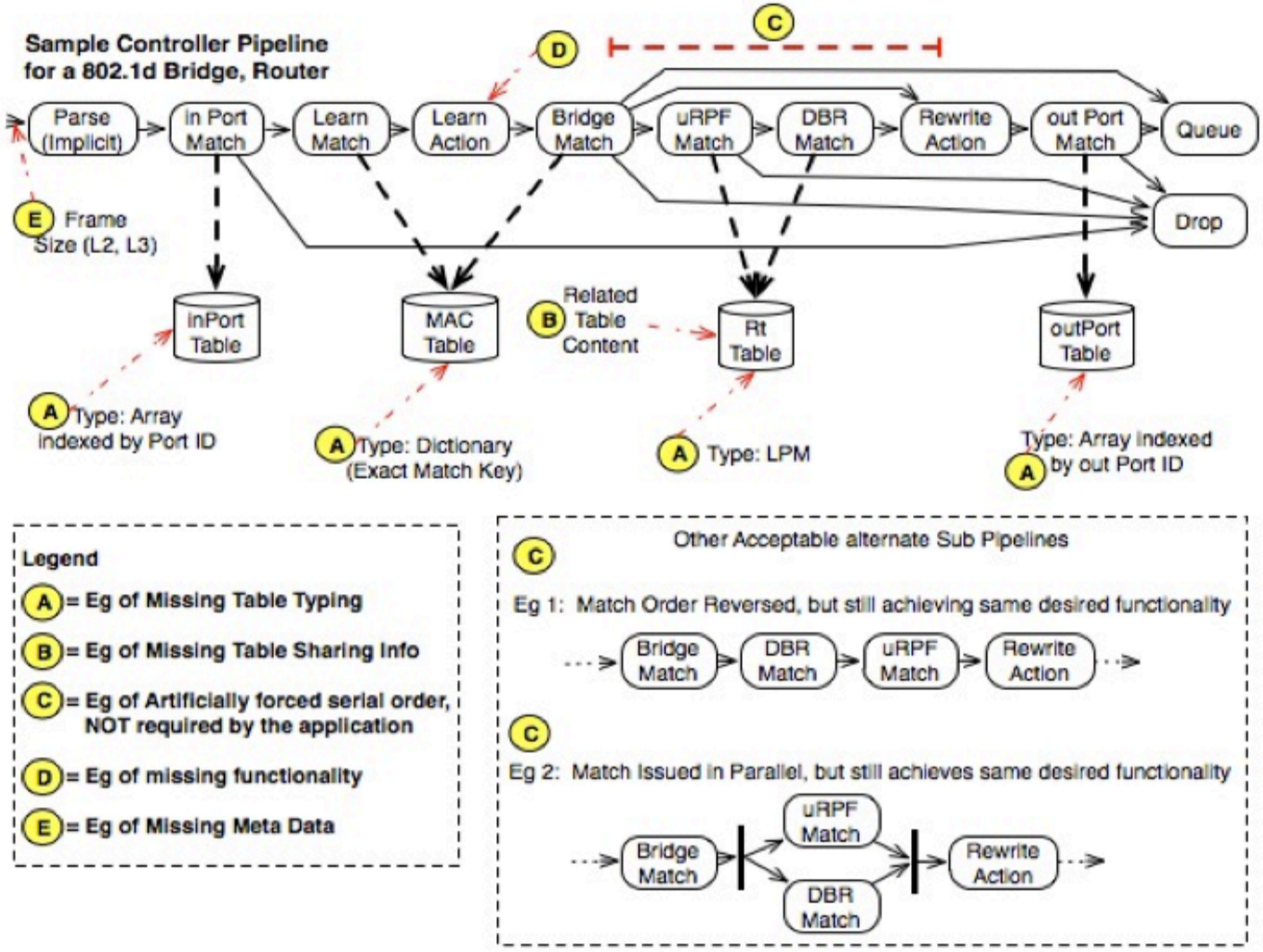
- Why this design?
  - Combinatoric explosion(s) s/a routes*policies in single table
- However, intractable complexity: *O(*n!) paths
  - $c \approx a^{(2^{\wedge}l)} + \alpha$
  - where a = number of actions in a given table, l = width of match field, and
  - α all the factors I didn't consider (e.g., table size, function, group tables)
- Too complex/brittle
  - Algorithmic complexity
  - What is a flow?
  - Not naturally implementable on ASIC h/w
  - Breaks new reasoning systems
  - No fixes for lossy abstractions (next slide)
  - Architectural questions

So question: Is the flow-based abstraction "right" for general network programmability?

Lossy OF Abstractions

- **Problem**: Current versions of OF do not work well on ASIC forwarding targets
- **Why?** Basically, the simple abstraction of setting forwarding table entries looses too much information about the application developer's intent, making it difficult (or impossible) to leverage switch features to deliver scalable and efficient end-to-end implementations
- Issues with the 1.X Abstractions
  - **Information Loss**
    - Controller to switch
  - **Information Leakage**
    - Switch to controller
  - Weak Control Plane to Data Plane Abstraction
  - Combinatorial State Explosion
  - Data Plane Driven Control Events
  - Weak Indirection Infrastructure
  - Time Sensitive Periodic Messaging
  - Multiple Control Engines
  - Weak Extensibility
  - Missing Primitives
- OF 1.X abstractions are “lossy” and as such can’t take advantage of ASIC features
  - As a result a controller must include platform specific code → brittle implementations
  - Specifies too much of “how” the switch should operate (as opposed to “what” it should be doing)

Lossy OF Abstractions

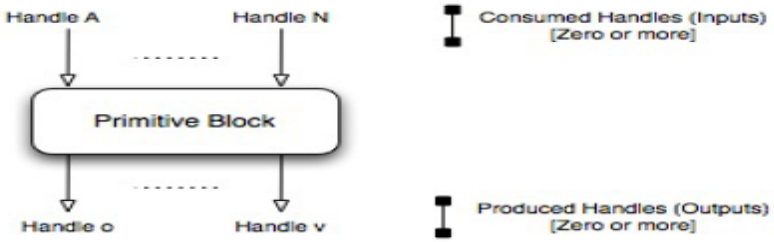


Graphic courtesy Dan Cohn

History of a Solution

Google OF 2.0 Proposal

- Google proposal was a producer/consumer DAG model
  - Just like today's OF 1.1+ only richer
    - OF 1.1+ can only express an implicit linear "DAG" table organization (with conditionals)
    - Google model can express Conditional, Serial, Parallel and Speculative pipelines
    - OF 1.1+ has a restricted set of primitives (basically flow_mods)
  - Producer/Consumer relationship between primitive "blocks"
    - "blocks" built up from rich set of ***primitives*** (example on next slide)
  - Expresses the ordering constrains (if any) to the target
  - Conveys the minimum set of constrains



Example: Shared Table Primitives (pseudo-code)

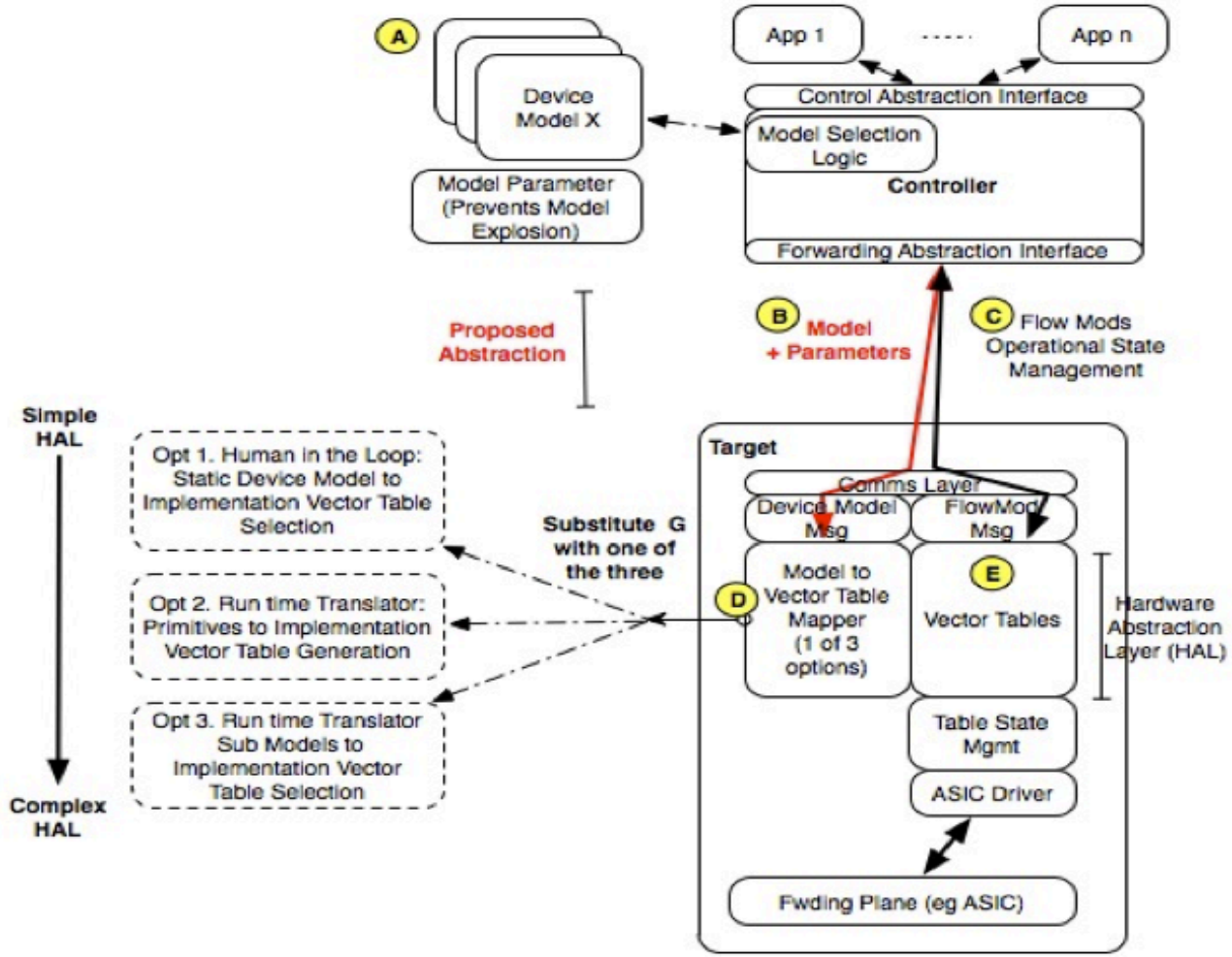
```
primitive TABLEDEF (Name = "MACTable",
            TableShared = TRUE,
             TableType    = ExactMatch,
             SearchKey   = {{(VlanID, 12), (MACAddress, 48)},                  /* bridging */
                            {(PortID, 8), (VlanID, 12), (MACAddress, 48)}},   /* learning  */
             TableRsult    = {BridgingResult,LearningResult},
             TableSize     = MacTableSize);
```

Then to match against such a table for a learning bridge you'd have a primitive operation like:

```
primitive TABLEOPS_MATCH(MatchName = LearningBridgeMatch,
            TABLE = "MACTable",
            consumes(PortID, VLANID, MACSrcAddress),
            SearchKeyMap  =  {((PortID:PortID),
                              (Vlan:VlanID),
                              (MACAddress:MACSrcAddress))},
             produces(LearningHit),
             ResultMap = {(LearningResult:LearningHit)});
```

Putting it All Together

Google OF 2.0 Model



Graphic courtesy Dan Cohn

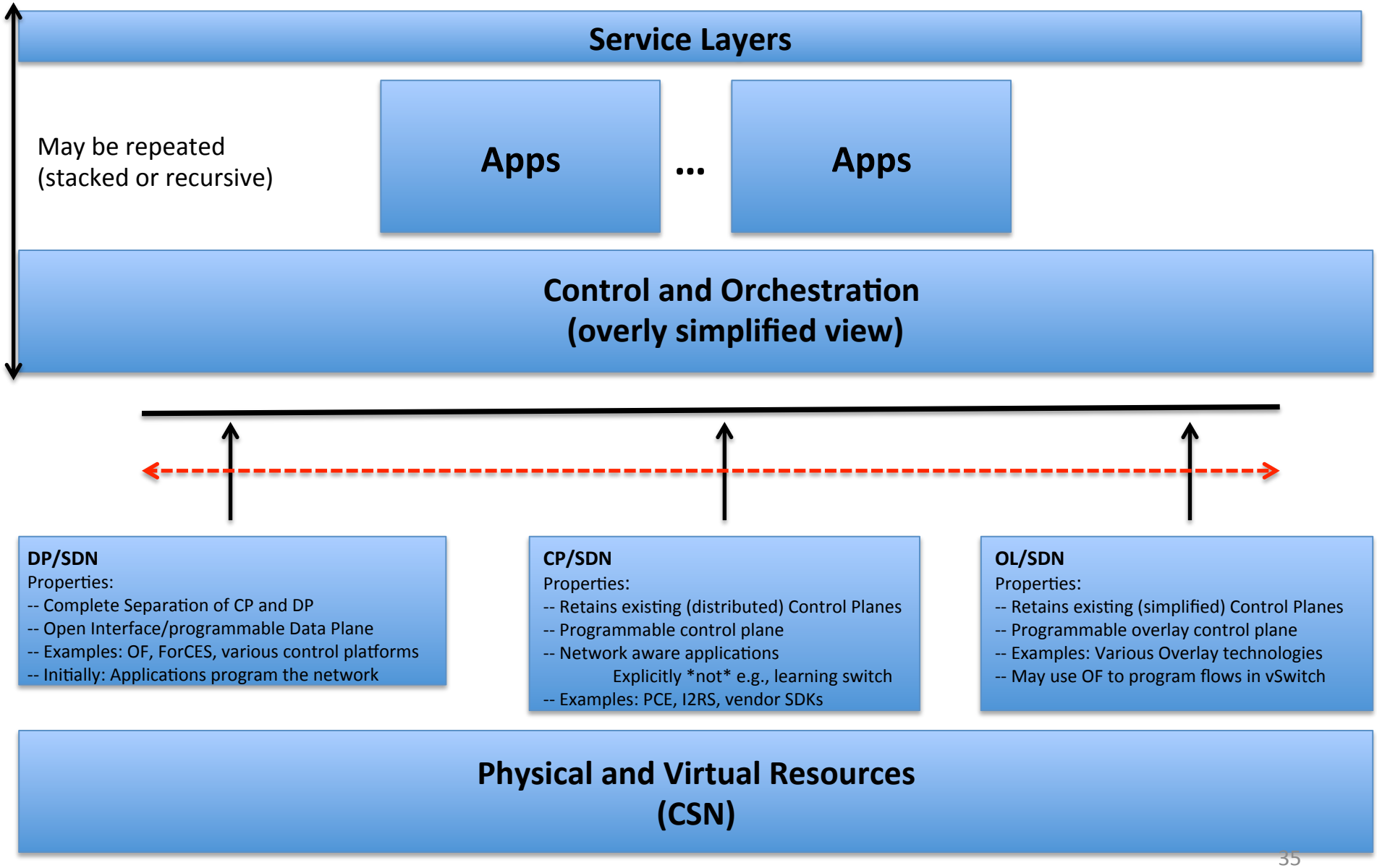
Beautiful idea, but….

- Was seen as too dynamic/futuristic by the ONF TAG
- A "dumbed down" version was proposed
  - Basically less dynamic
    - No runtime composition of models
    - No language to express primitives and their compositions
  - Small set of pre-defined models
  - Was still seen as too ambitious
- Current ONF direction is to add "table typing" to OF 1.3+
  - The capability to constrain a table to be one of exact match, wildcard, LPM, …
  - Along with Table Type Patterns (*TTPs*)
    - A graph plus properties/constraints representing how tables are connected, …
  - This is the work of the Forwarding Abstractions Working Group
- BTW, isn't this converging on ForCES?
  - And why hasn't ForCES succeeded?

Summary

- Only OF 1.0 “widely” implemented
  - *Widely* a relative term
  - A few new software implementations of 1.3
    - e.g., LINC, others
    - See http://www.flowfowarding.org
- OF 1.1+ not widely implemented in h/w
  - Only know of a few implementations
    - Most h/w implementations on NPU-like h/w, so really s/w
- ONF direction “could” yield something
  - “table typing” *could be* a part of a solution
  - Getting the abstractions right is the key
    - And of course, getting the abstractions right is one of the hardest parts of all of this
- Many open questions remain regarding architecture/scalability
  - Both of OF itself and of the OF/SDN architecture
  - Bowties, RYF systems
- What about other network programmability architectures?
  - → Emerging ***SDN Continuum***

Future (now): *The SDN Continuum*



Q&A

Thanks!