Seeing The Past, Present and Future: Macro Trends in Networking and the Role of Software Defined Networking

David Meyer
CTO and Chief Scientist, Brocade
Director, Advanced Technology Center, University of Oregon
Apricot 2013
Singapore
dmm@{brocade.com,uoregon.edu,1-4-5.net,...}
http://www.1-4-5.net/~dmm/talks/apricot2013.pdf

Agenda

- Context: SDN Problem Space and Hypothesis
- (Macro) Trends Inducing an New Landscape
- The Past: How We Got Here
- The Present: What Exactly is the Current State of Affairs?
- The Future: Where's it All Going
- Summary and Q&A if we have time

Danger Will Robinson!!!

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

What I Hope To Achieve

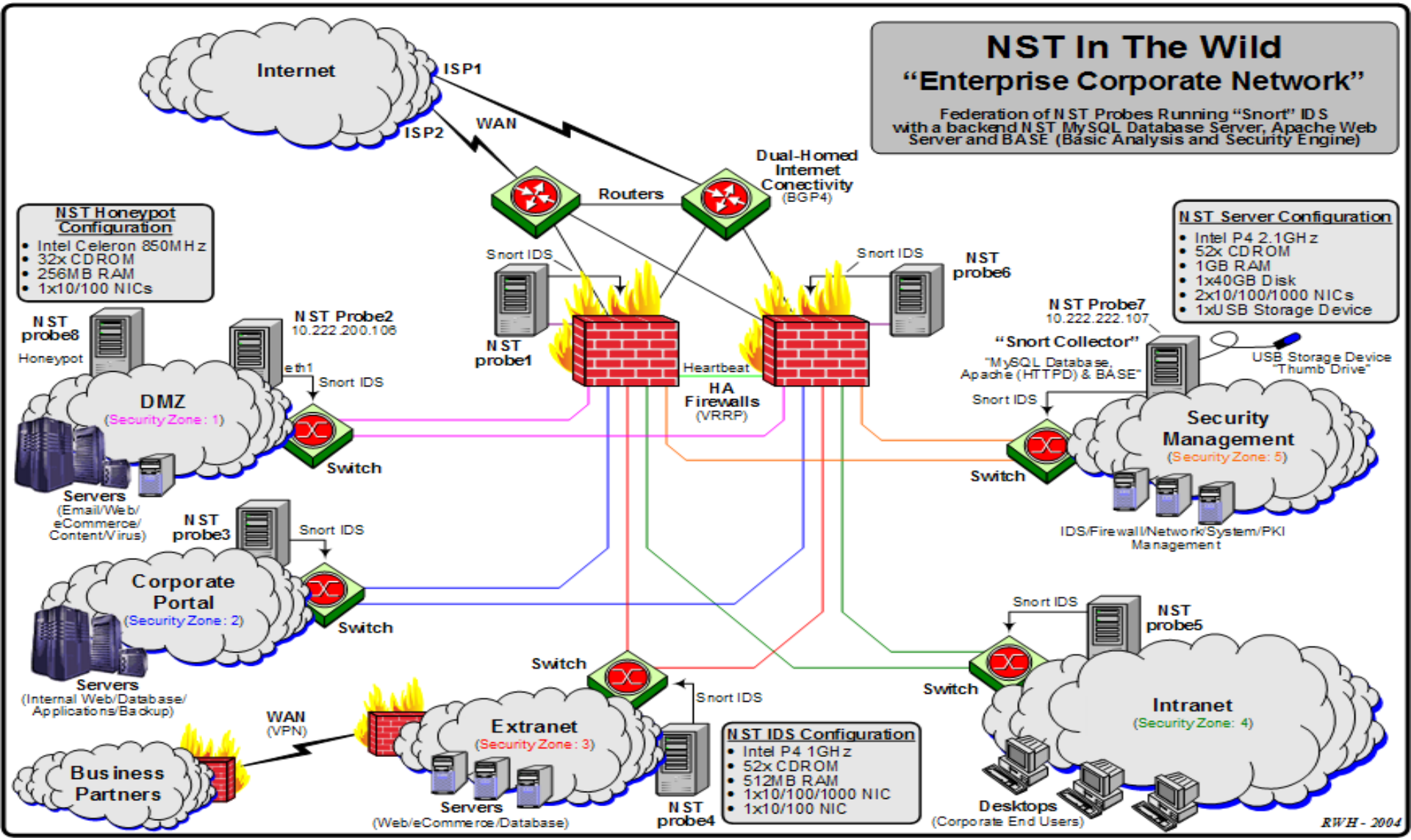
I hope to convince you that uncertainty and volatility are the "coin of the realm" of the future, why this is the case, how SDN (and the rise of software in general) is accelerating this effect, and finally, what we might do to take advantage of it.[1]

1 s/take advantage of/*survive*/ -- @smd

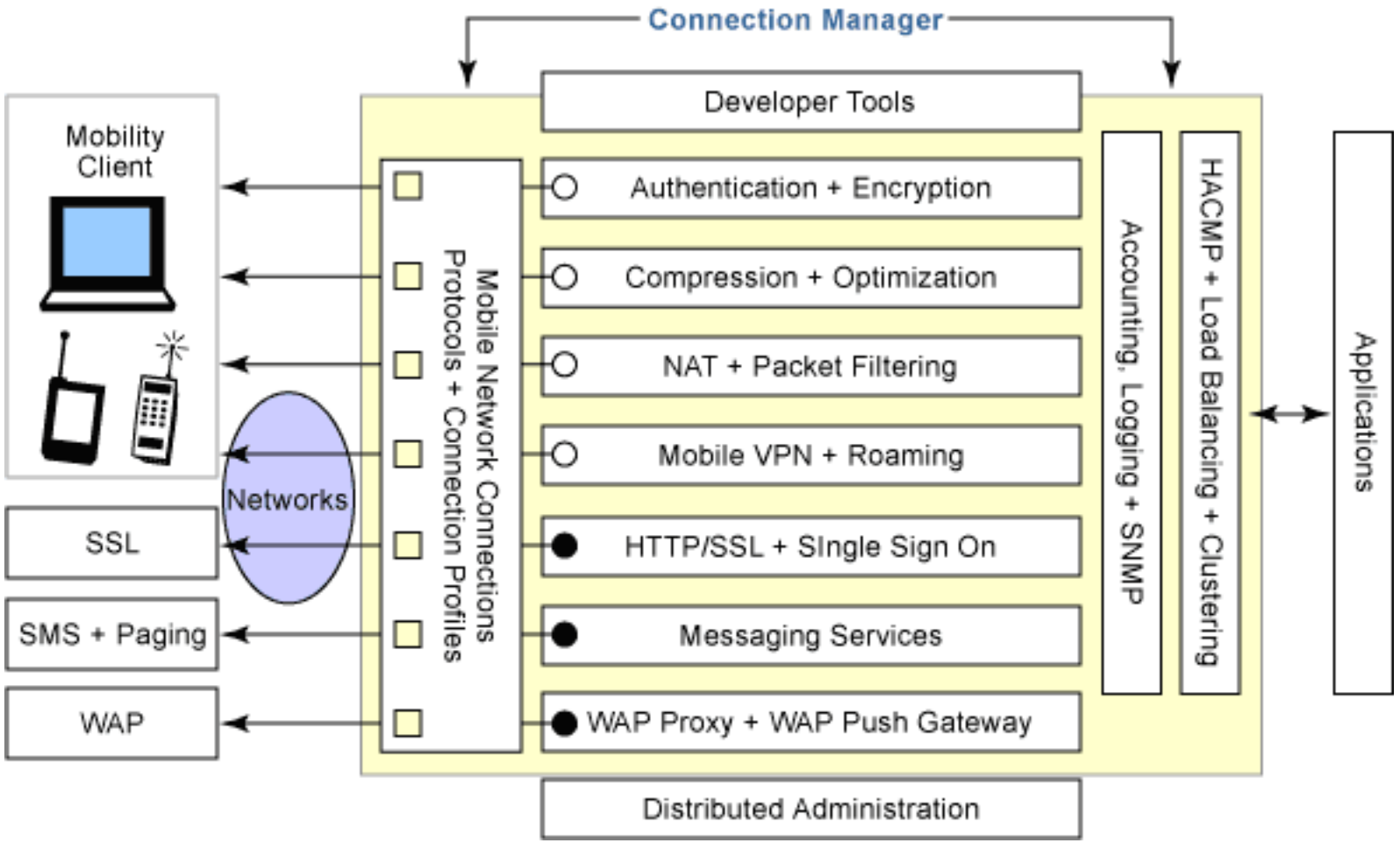
First, What is the SDN Problem Space?

- Network architects, engineers and operators are being presented with the following challenge:
  - ***Provide state of the art network infrastructure and services while minimizing TCO***
- ***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.
  - Is this true? Hold that question…
- *Note future uncertain*: Can't *"skate to where the puck is going to be"* because curve is unknowable (this is a consequence, as we will see, of the "software world" coupled with Moore's law and open-loop control).
  - That is, there is quite a bit of new research that suggests that such uncertainty is inevitable
- So given this hypothesis, what was the problem?

Maybe this is the problem?



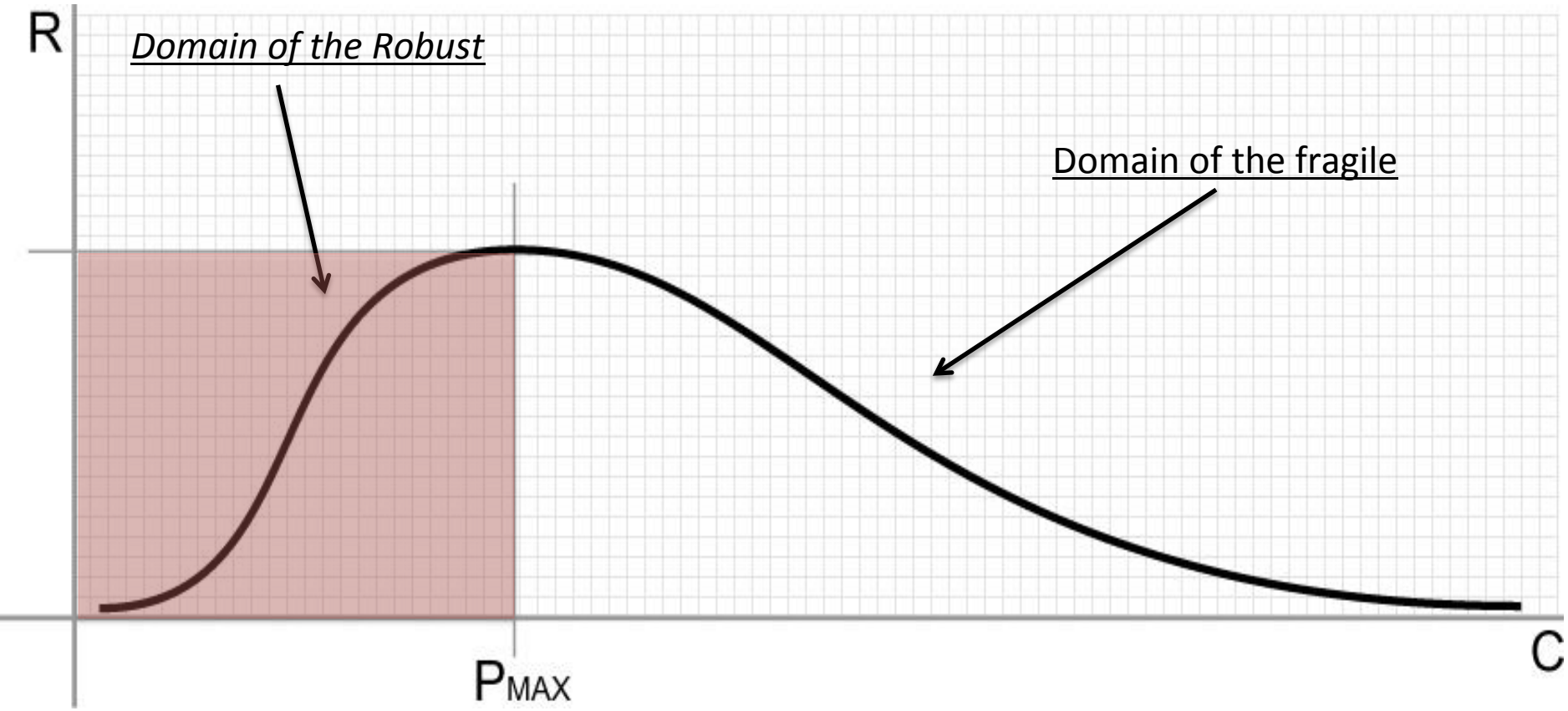
Or This?



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

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

Robustness vs. Complexity
Systems View



Increasing number of policies, protocols, configurations and interactions

Can we characterize the Robust and the Fragile?

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

System Properties as Robustness

- **Reliability** is robustness to component failures
- **Efficiency** is robustness to resource scarcity
- **Scalability** is robustness to changes to the size and complexity of the system as a whole
- **Modularity** is robustness to structure component rearrangements
- **Evolvability** is robustness of lineages to changes on long time scales

Fragility and Scaling

(geeking out for a sec…)

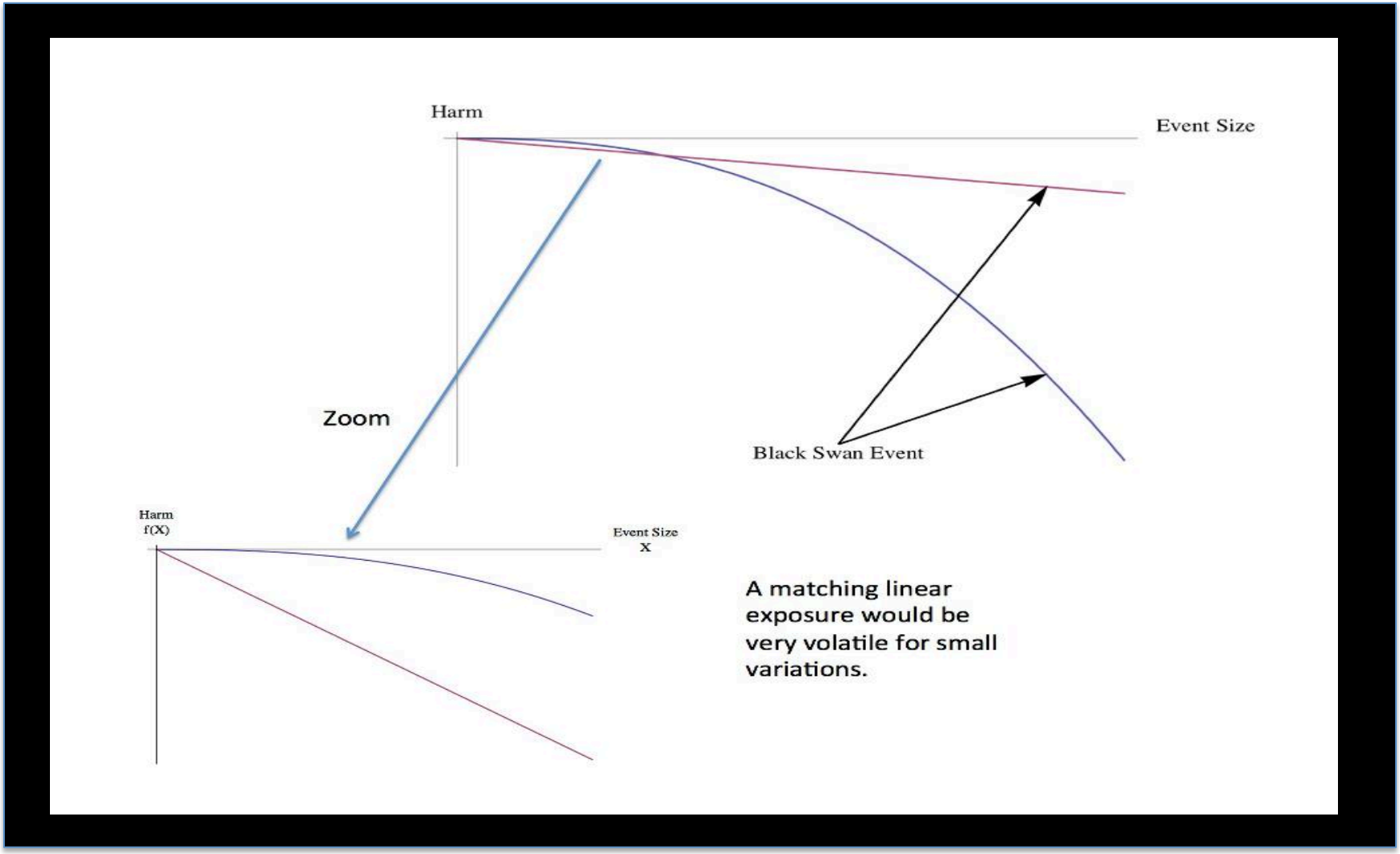
- A bit of a formal description of fragility
  - Let *z* be some stress level, *p* some property, and
  - Let *H(p,z)* be the (negative valued) harm function
  - Then for the fragile the following must hold
    - ***H(p,nz) < nH(p,z) for 0 < nz < K***
    - *K* is the level at which the system collapses (*K-fragility*)
    - This inequality is importantly not mean preserving (Jensen's Inequality)
    - Not mean preserving: ***H(p,(z$_1$ + z$_2$)/2) != (H(p,z$_1$) + H(p,z$_2$))/2***
      - → model error and hence additional uncertainty

- For example, a coffee cup on a table suffers non-linearly more from large deviations (*H(p, nz)*) than from the cumulative effect of smaller events (*nH(p,z)*)
  - So the cup is damaged far more from (i.e., destroyed by) *tail events* than those within a few σ of the mean
  - Too theoretical? Perhaps, but consider: ARP storms, micro-loops, congestion collapse, AS 7007, …
  - BTW, nature requires this property
  - For example, if you jump off something 1 foot high 30 times v/s jumping off something 30 feet high once

- When we say something scales like *O*(n^2), what we mean is the damage to the network has constant acceleration (2) for *weird* enough n (i.e., outside say, 10 σ)
  - That is, you suffer non-linear harm from tail events

What Does The Fragility Curve Look Like?

Non-linear exposure to harmful event → Concavity



Graphic courtesy [Talab2012]

What Is Antifragility?

- Antifragility **is not the opposite** of fragility
  - ***Robustness*** is the opposite of fragility
  - Antifragile systems ***improve*** as a result of [perturbation]

- Metaphors
  - **Fragile**: *Sword of Damocles*
    - Upper bound: No damage
    - Lower bound: Completely destroyed
    - ***The cumulative effect of small perturbations is smaller than the single effect of a large perturbation – dependence on second order effects***
  - **Robust**: *Phoenix*
    - Upper bound == lower bound == no damage
  - **Antifragile**: *Hydra*
    - Lower bound: Robust
    - Upper bound: Becomes better as a result of perturbations (within bounds)

- More detail on this later (if we have time)

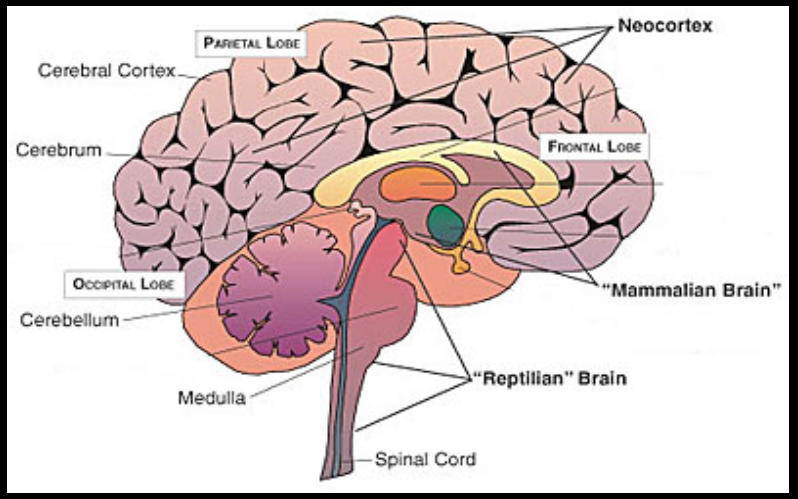
So What Then is *Complexity*?

"In our view, however, complexity is most succinctly discussed in terms of functionality and its robustness. Specifically, *we argue that complexity in highly organized systems arises primarily from* ***design strategies intended to create robustness to uncertainty*** *in their environments and component parts.*" [AldersonDoyle2010]

Back to Macro Trends

The Evolution of Intelligence

Precambrian (Reptilian) Brain to Neocortex → Hardware to Software

HARDWARE	SOFTWARE
	

- Architectural Themes
  - Thin-waist architectures (more on this in a sec)
  - Massively distributed
  - Highly layered with Robust Control loops
  - Component Reuse

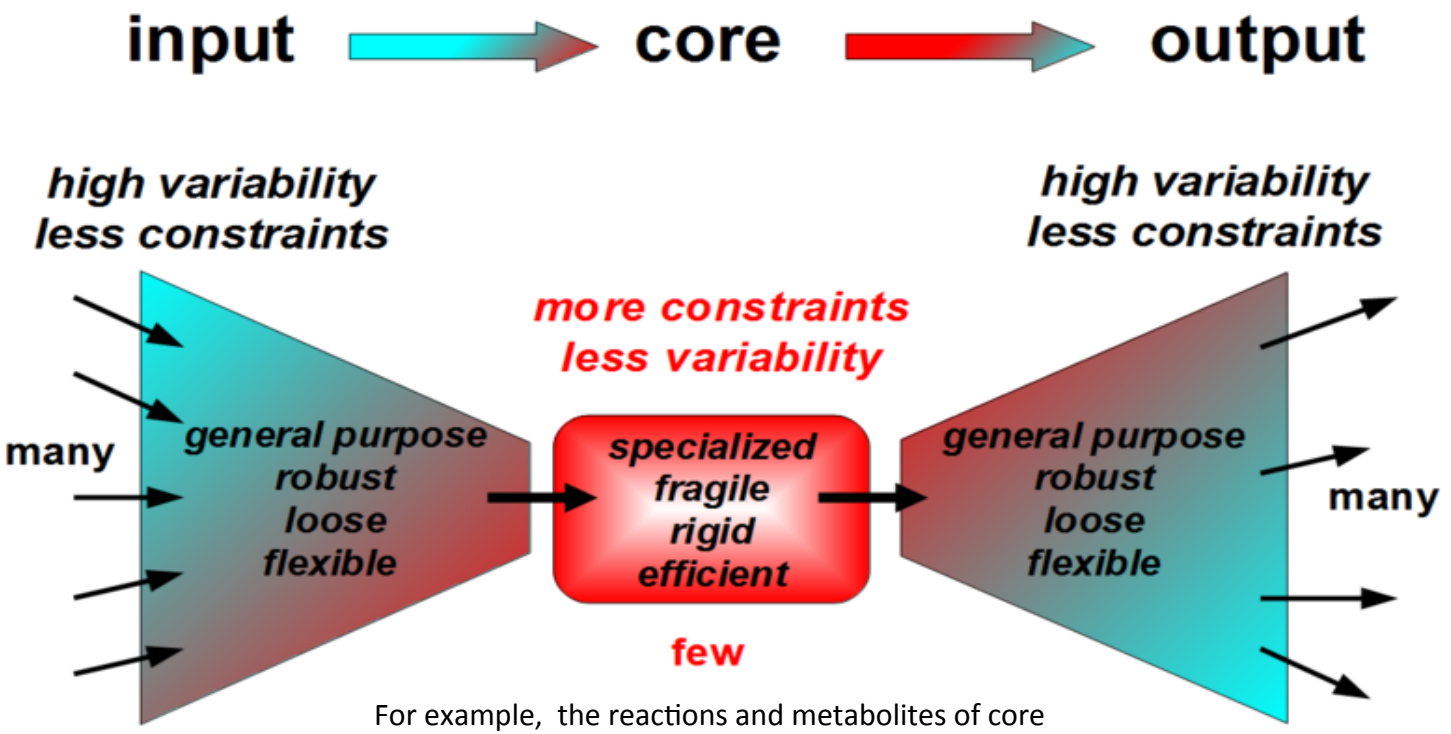
Its all about code

BTW, while we're talking about evolution, the Punctuated Equilibrium model of evolution [Gould & Eldredge1977] depends on the existence of just the kind of *tail events* I described earlier.

Thin Waists 101: The Bowtie Architecture

Idea from biological systems theory

Constraints that Deconstrain

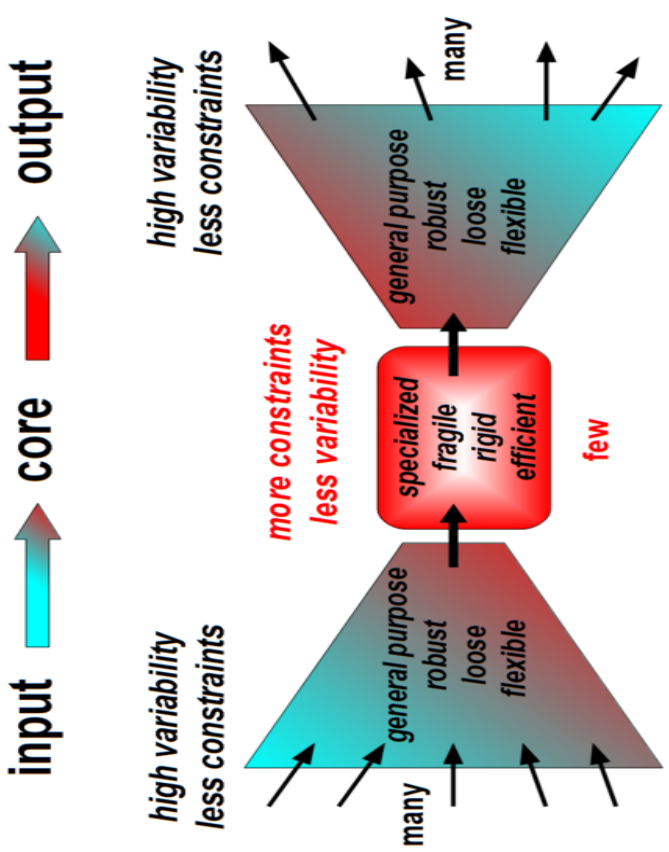


For example, the reactions and metabolites of core metabolism, e.g., ATP metabolism, Krebs/Citric Acid cycle signaling networks, …

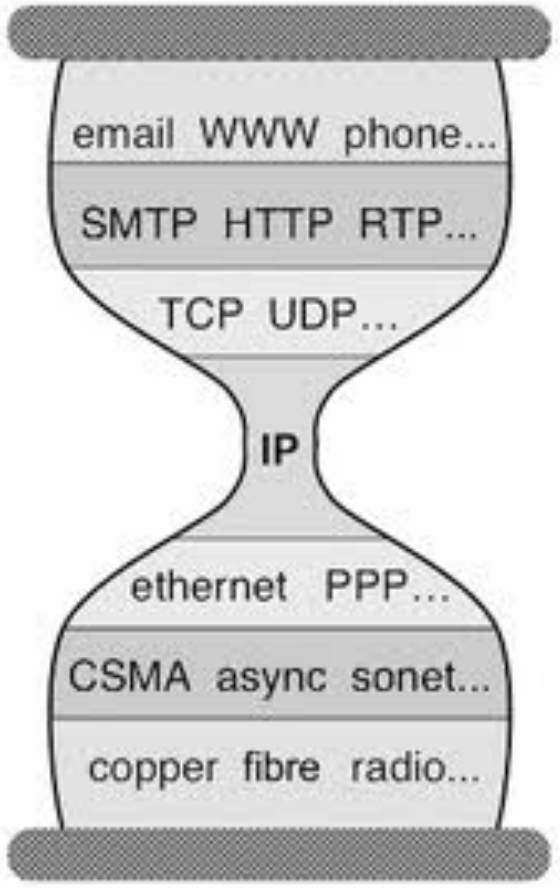
See, e.g., Doyle, et. al., "Architecture, Constraints, and Behavior", http://www.pnas.org/content/108/suppl.3/15624.full

But Wait a Second

Anything Look Familiar?

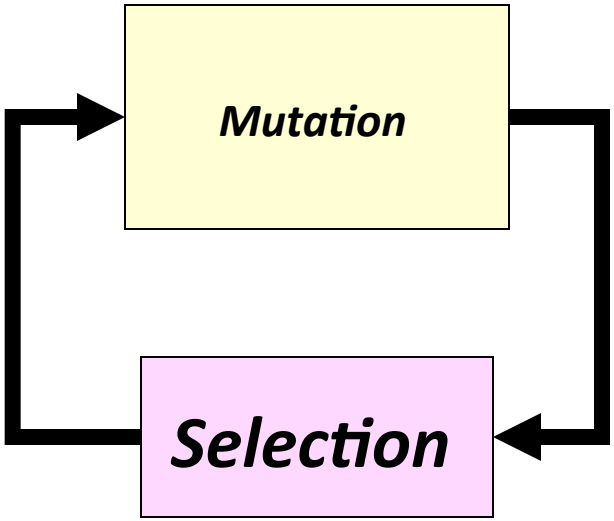


Bowtie Architecture



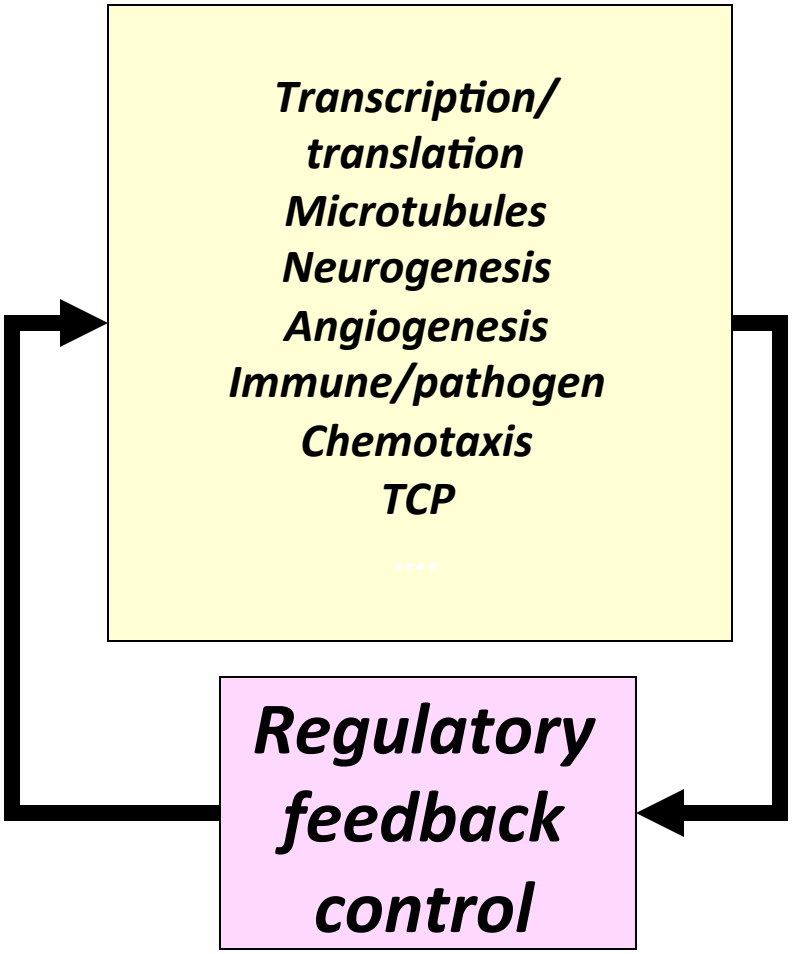
Hourglass Architecture

BTW, there's an apparent paradox → Component behavior ***gratuitously*** uncertain, yet systems have robust performance.



Darwinian evolution uses selection on random mutations to create complexity.

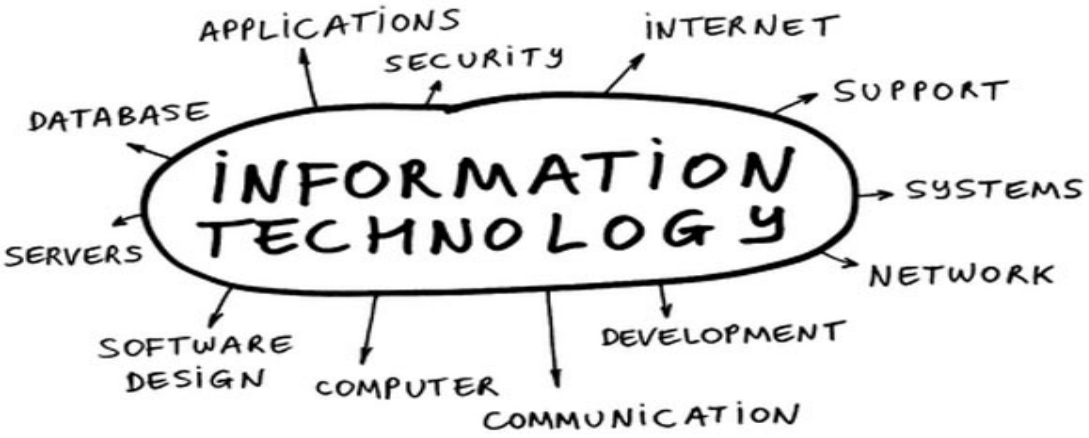
Network folks use what, exactly?



Everything De-silos

Vertical -> Horizontal Integration
Open {APIs, Protocols, Source}
Everything Pluggable
Future is about Ecosystems

Network Centric → IT Centric



- Shift in influence and speed
- Shift in locus of purchasing influence
- Changes in cost structures
  - ETSI NfV, ATIS, IETF, …
- **NetOPs → DevOPs**

Other Important Macro Trends

- Everything Virtualizes
  - Well, we've seen this
- Data Center new "center" of the universe
  - Looks like ~ 40% of all traffic is currently sourced/sinked in a DC
  - Dominant service delivery point
- Integrated orchestration of almost everything
- Bottom Line: Increasing influence of software *everywhere*
  - All integrated with our compute, storage, identities, …
  - Increasing compute, storage, and network "power" → increasing volatility/uncertainty

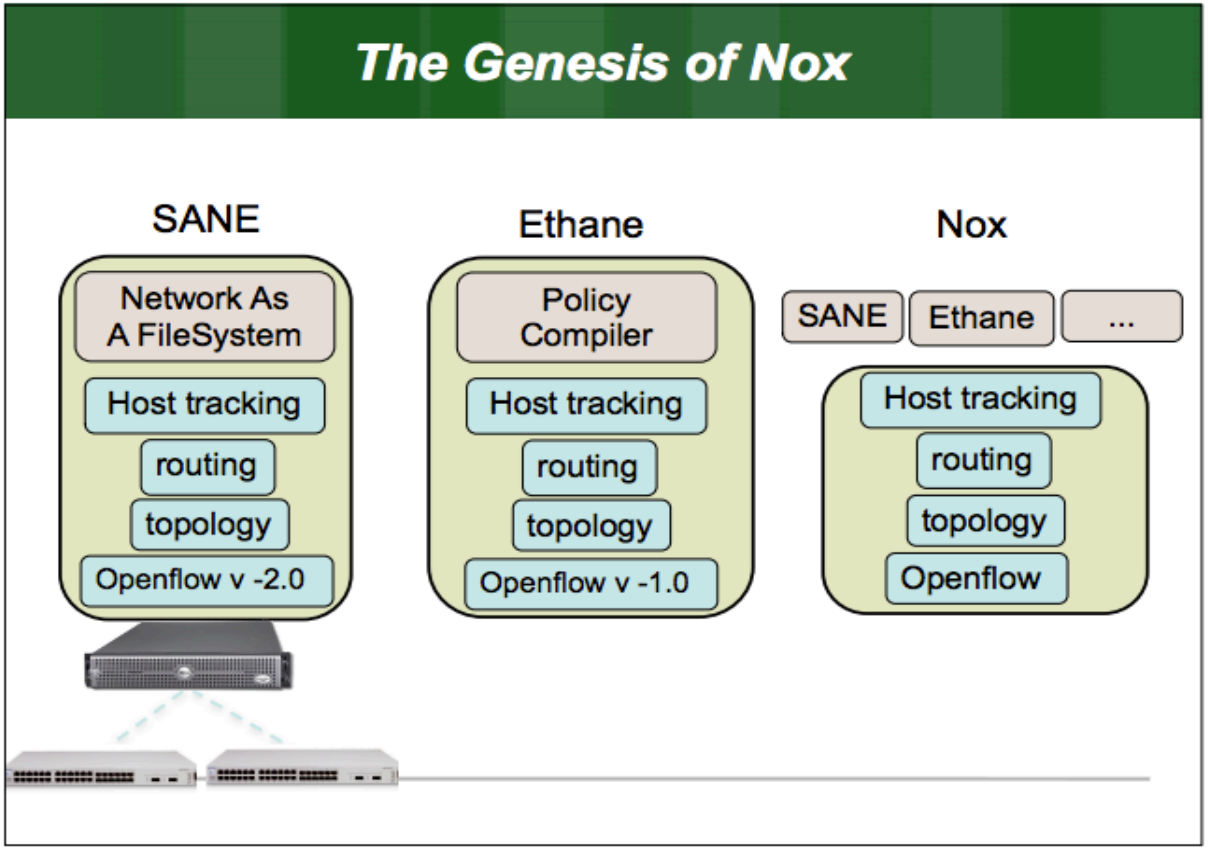
The Past: Ok, How Did We Get Here?

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.

Question: What is the relationship between the job of the network researcher and the task of fielding of a production network?

(in)SANE



Salient features: Open interface to forwarding plane, separation of control and data planes

Slide courtesy Martin Cassado

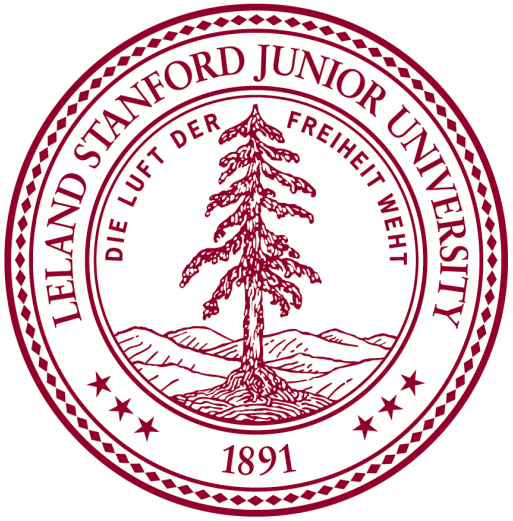
So What was Ethane?

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
(Gates 104 Crew)

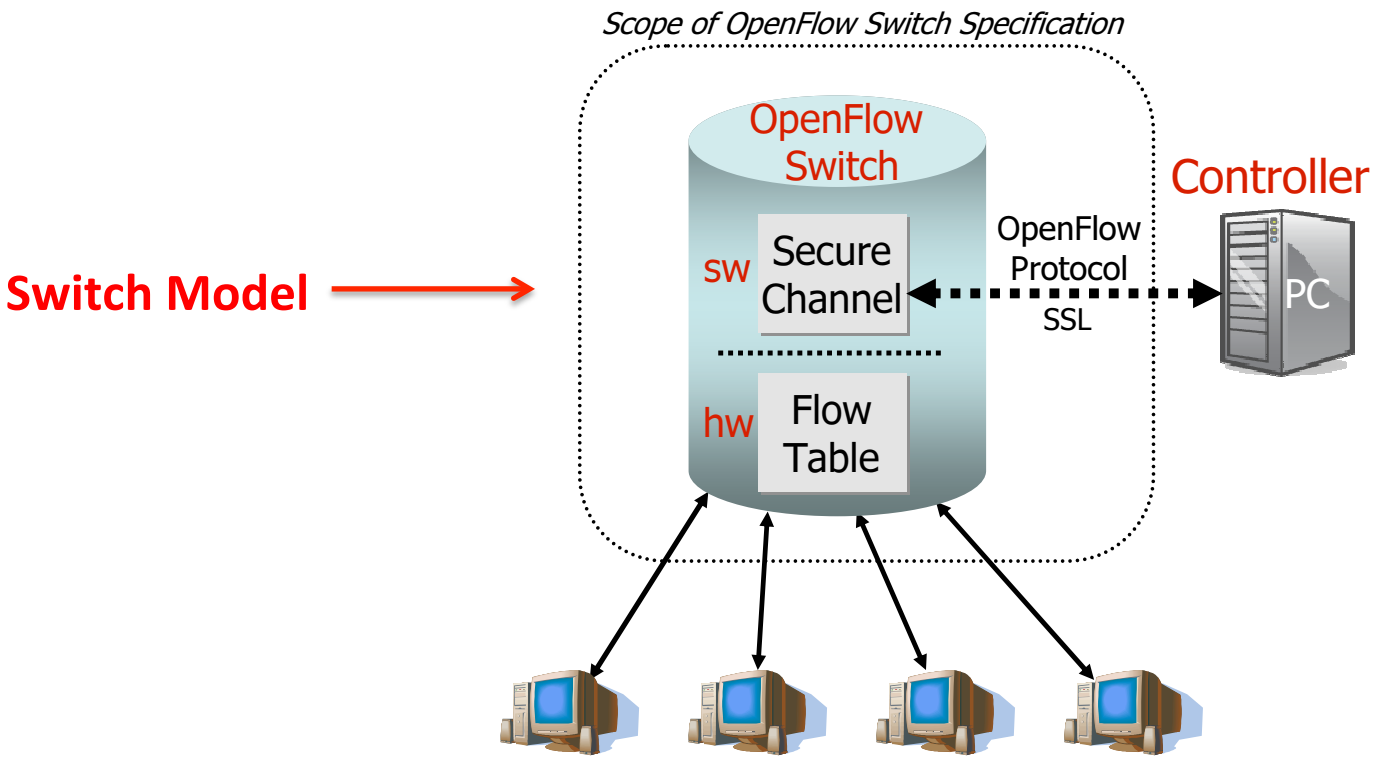


Figure 1: Idealized OpenFlow Switch. The Flow Table is controlled by a remote controller via the Secure Channel.

OpenFlow Switch, v 1.0

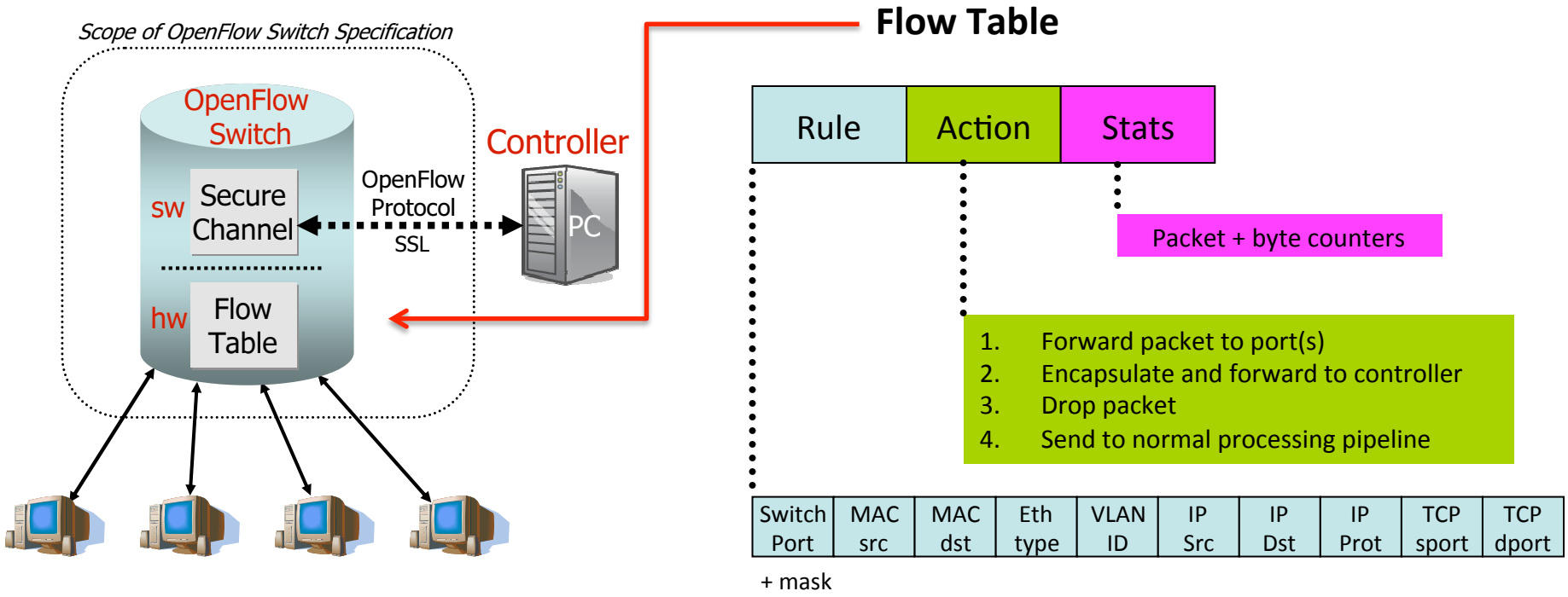


Figure 1: Idealized OpenFlow Switch. The Flow Table is controlled by a remote controller via the Secure Channel.

Again, salient features: Open interface to the forwarding plane, separation of control and data planes, "centralized" control → Great for researchers, but what about production networks?

And BTW, is this (architecturally) the same as the breaking down of vertical integration in the compute world?

Mainframe Business Model

Net Equipment

Central Logic Manufacture
- Proprietary & closely guarded
- Single source

Finished Hardware Supply
- Proprietary & closely guarded
- Single source

System Software Supply
- Proprietary & closely guarded
- Single source

Application Stack
- Not supported
- No programming tools
- No 3rd party ecosystem

Commodity Server

Central Logic Manufacture
- Standard design (x86)
- Multiple source
  - AMD, Intel, Via, …

Finished Hardware Supply
- Standard design
- Multiple source
  - Dell, SGI, HP, IBM, …

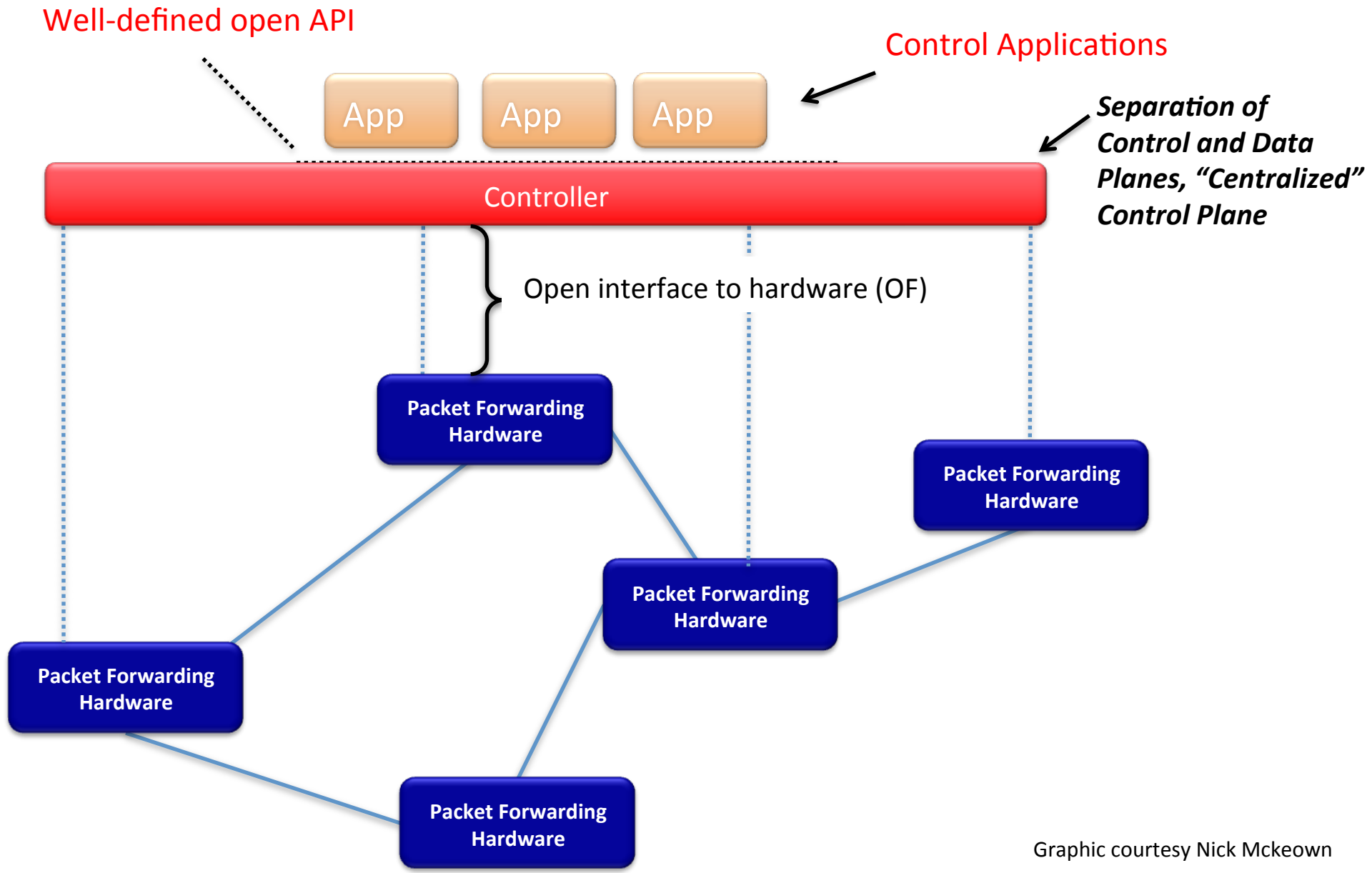
System Software Supply
- Linux (many distros/support)
- Windows & other proprietary offerings

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

Early OF/SDN Architecture



Graphic courtesy Nick Mckeown

Logically Centralized?

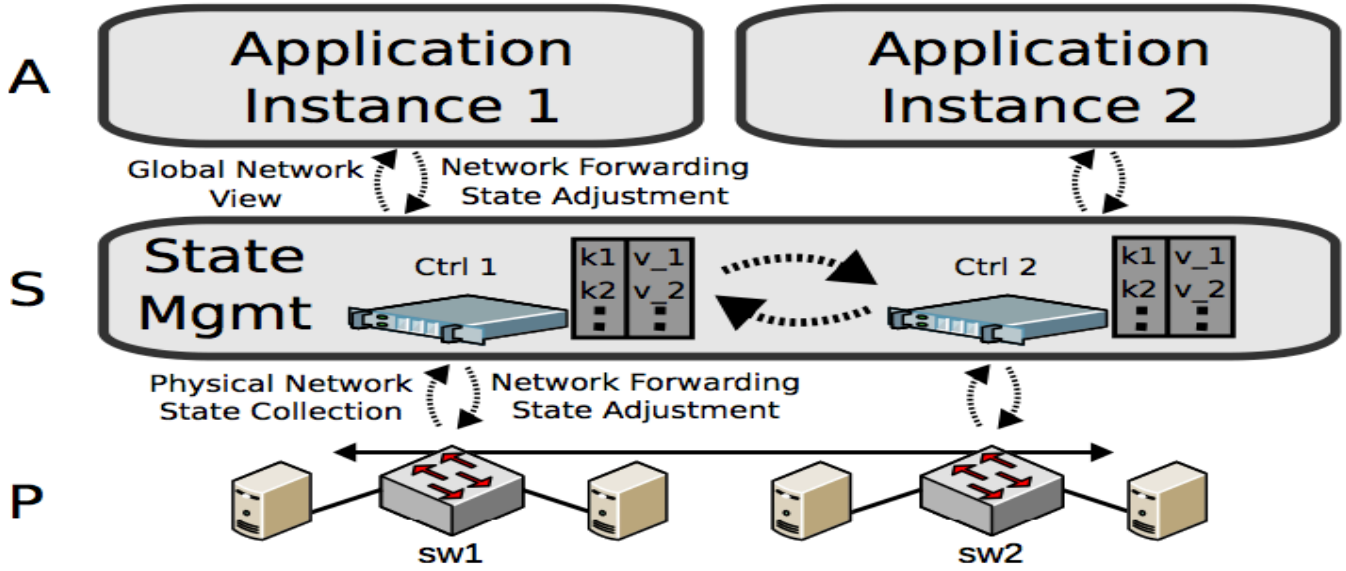


Figure 1: SDN state distribution and management conceptualized in layers: (A)pplication, (S)tate Management, (P)hysical Network

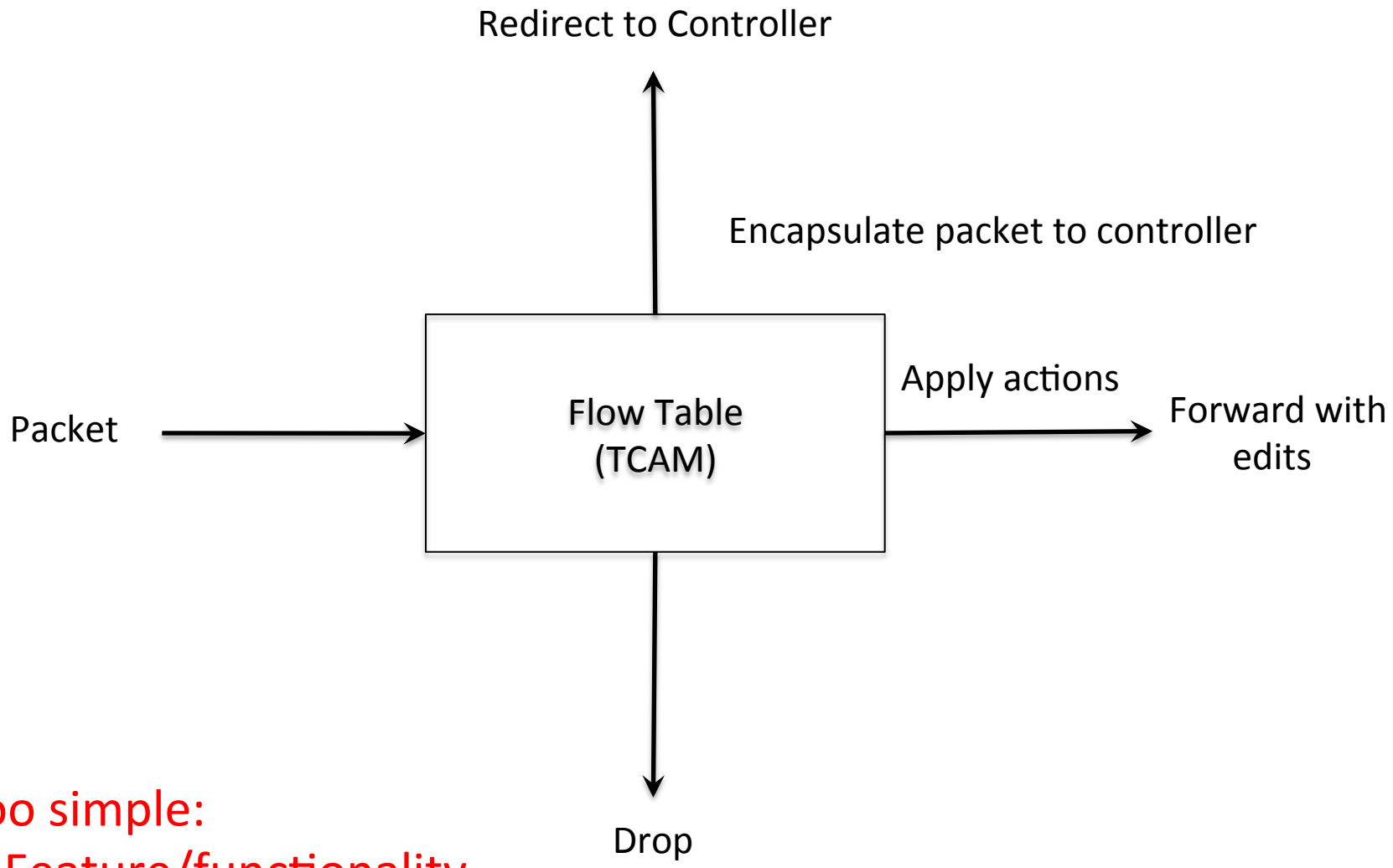
Key Observation: Logically centralized → distributed system → tradeoffs between control plane convergence and state consistency model. And what about the loss of control plane/data plane fate sharing?

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 circuit → packet
  - ForCES
    - Separation of control and data planes
    - RFC 3746 (and many others)
  - …

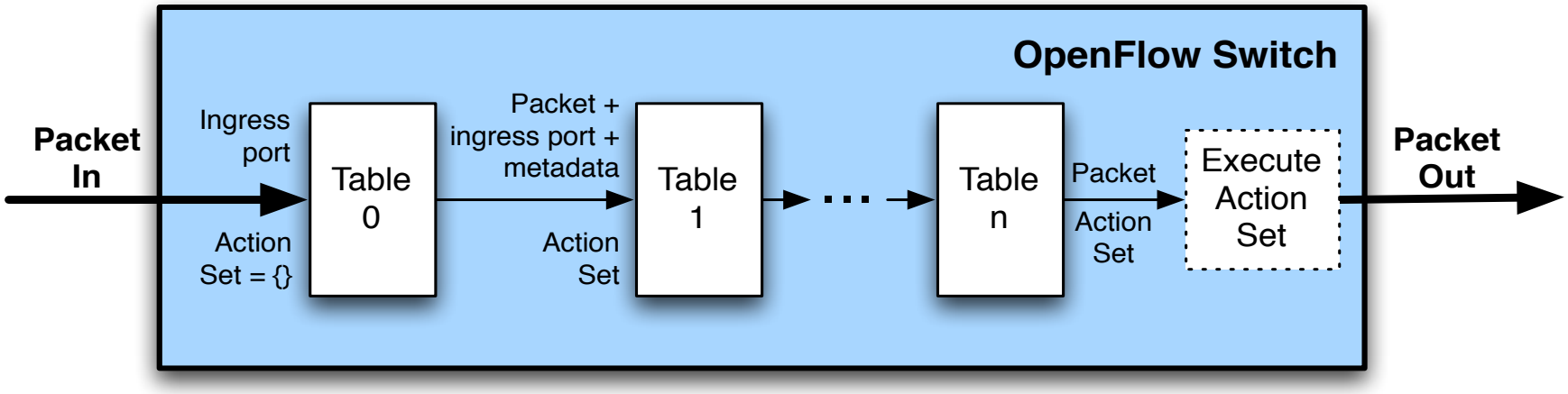
OpenFlow Switch Model Version 1.0



Too simple:

- Feature/functionality
- Expressiveness – consider shared table learning/forwarding bridge

The Present: Current (ONF) SOA



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

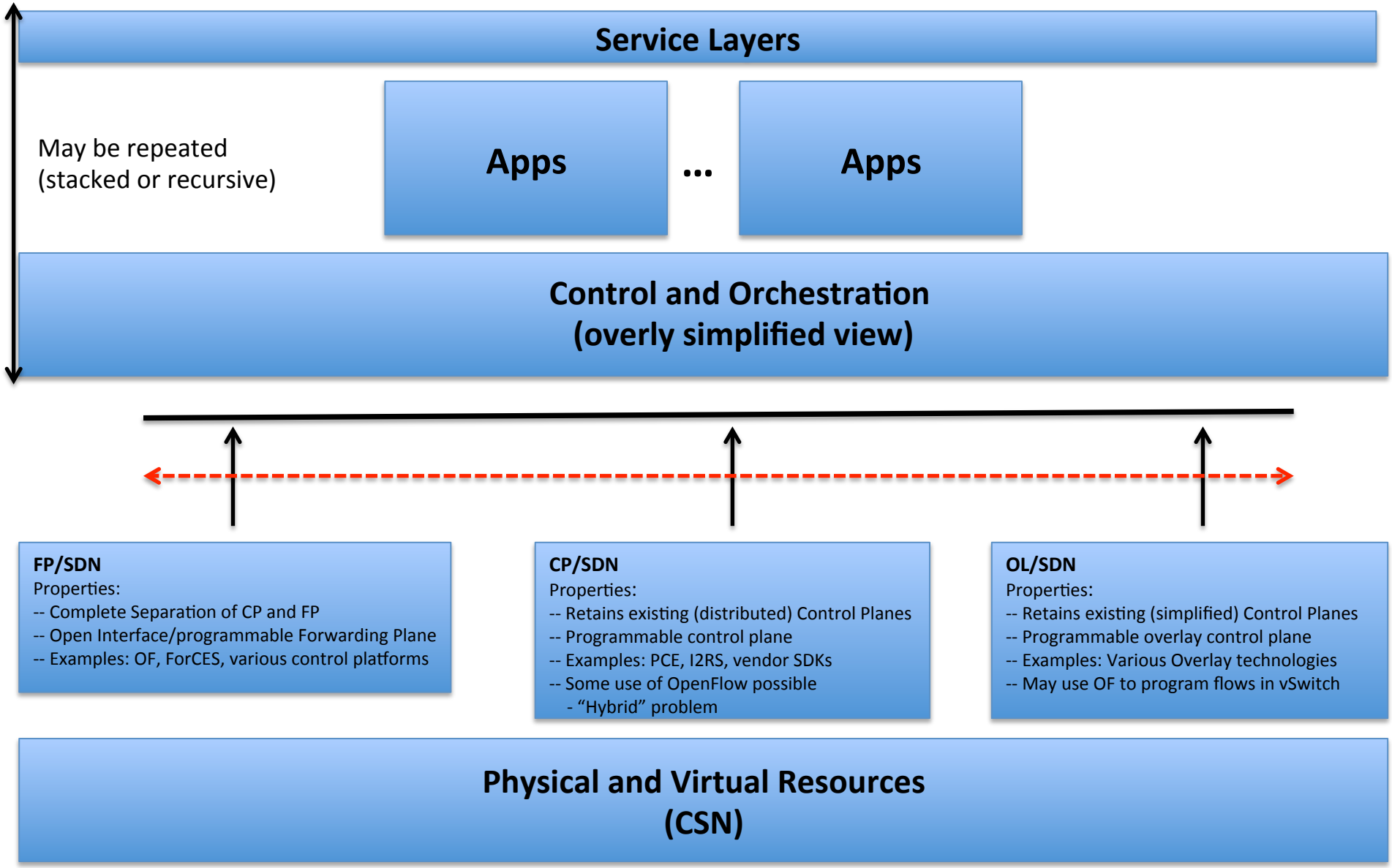
- Why this design? Combinatorics…
- Consider complexity: ~ *O*(n! * $a^{(2^l)}$) paths
  - n = number of tables, a = number of actions, l = width of match fields
- Too Complex:
  - What is a flow?
  - Not naturally implementable on ASIC h/w
  - Breaks new reasoning systems
  - No fixes for the lossy abstractions
  - Architectural questions

Emerging:

- SDN Continuum
- IETF, ETSI, ATIS, …

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

A Simplified View of the *SDN Continuum*

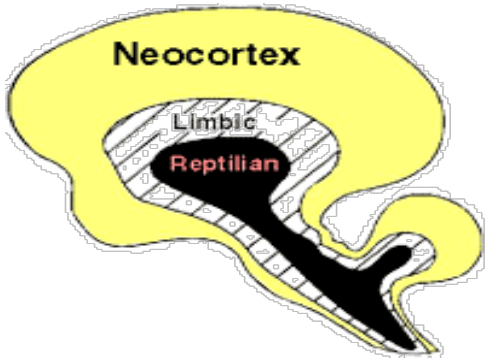


So The Future: Where's it All Going?

But More Seriously….

- High order bit:
  - System(s) we're building are inherently uncertain → cloudy crystal balls
  - Architect for change and rapid evolution – see XP/Agile methodologies for a clue
  - **Increasing roles for s/w and programmability + Moore's law → volatility/uncertainty**
  - Lucky thing for many of us: we work primarily around the narrow waist, most stable place to be
  - "Above the waist" characterized by uncertainty, e.g., http://spotcloud.com/

- Conventional Technology Curves – S & F
  - Moore's Law and the reptilian brain
    - Someone eventually has to forward packets on the wire
  - 400G and 1T in the "near" term
  - Silicon optics, denser core count, ….

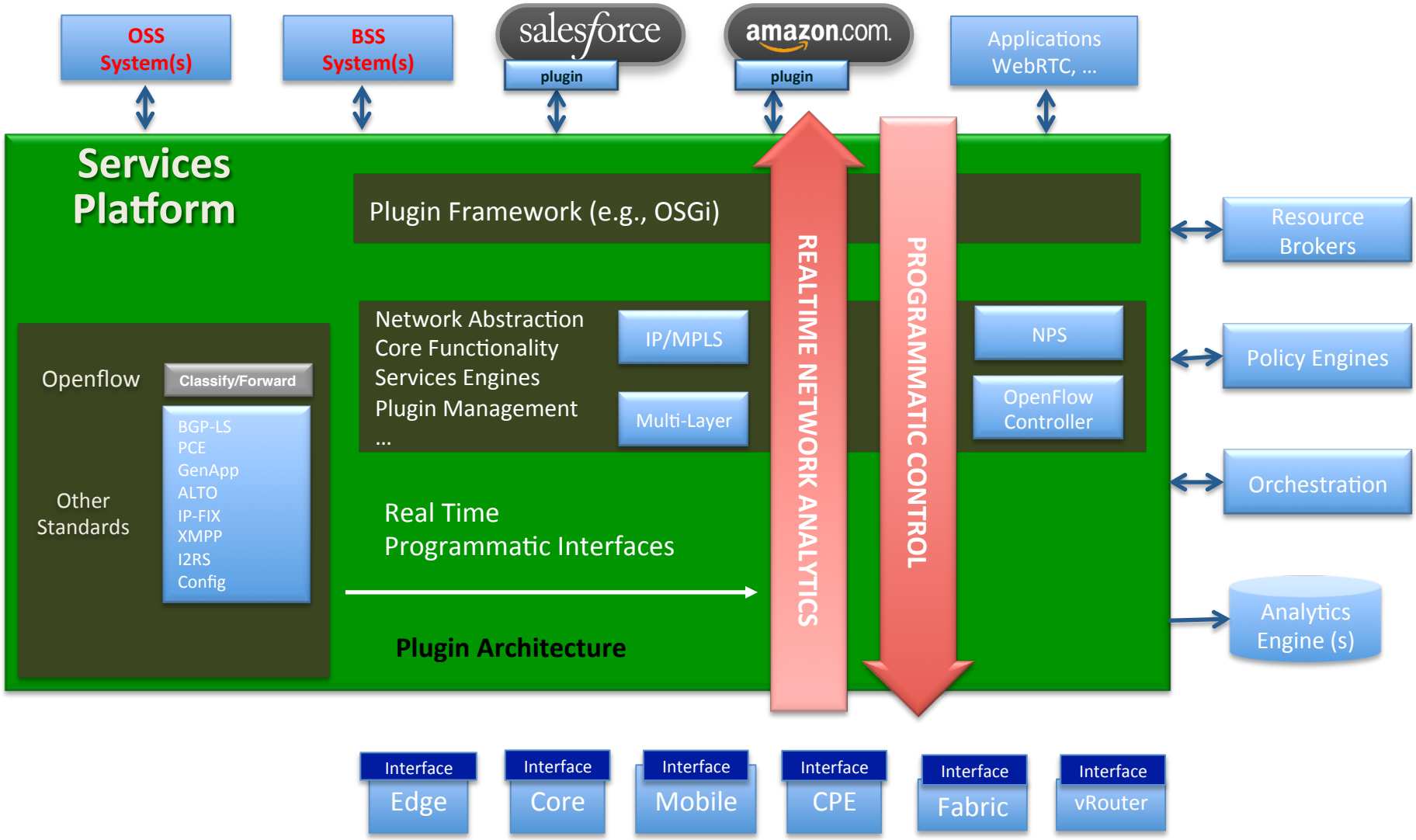


- The future is all about Ecosystems
  - Open Interfaces: Protocols, APIs, Code, Tool Chains
  - Open Control Platforms at every level
  - "Best of Breed" markets
  - ***And again, more volatility/uncertainty injected into system as a whole***

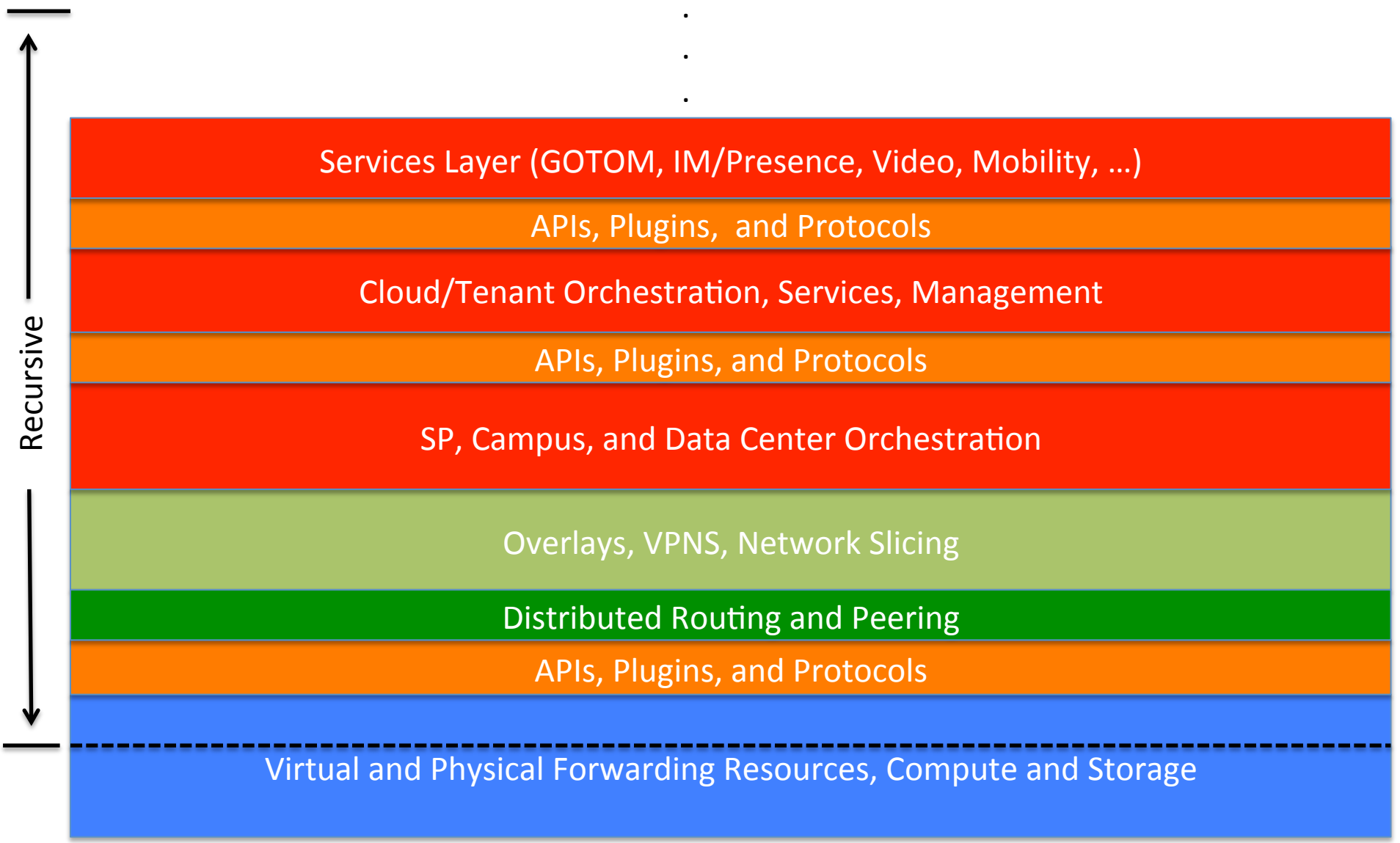
- BTW, open source/open source consortia dominate
  - And what is the role of standards bodies in the age of Open Source?

- So what might such an ecosystem/platform look like?

Ecosystem Platform Schematic



Stack View


Recursive
Services Layer (GOTOM, IM/Presence, Video, Mobility, …)
APIs, Plugins, and Protocols
Cloud/Tenant Orchestration, Services, Management
APIs, Plugins, and Protocols
SP, Campus, and Data Center Orchestration
Overlays, VPNS, Network Slicing
Distributed Routing and Peering
APIs, Plugins, and Protocols
Virtual and Physical Forwarding Resources, Compute and Storage

Summary – What are our Options[1]

- Be conservative with the narrow waist -- constraints that deconstrain
  - We're pretty good at this
  - Reuse parts where possible (we're also pretty good at this; traceroute a canonical example)

- Expect uncertainty and volatility from above
  - Inherent in software, and importantly, in acceleration
    - We know the network is RYF-complex so we know that for H(p,x), the "harm" function, $d^2H(p,x)/dx^2 \neq 0$
    - When you architect for robustness, understand what fragilities have been created
  - → Software (SDN or http://spotcloud.com or …) is inherently non-linear, volatility, and uncertain
    - We need to learn to live with/benefit from the non-linear, random, uncertain

- DevOps

- Develop our understanding bottom up (by "tinkering")
  - Actually an "Internet principle". We learn incrementally…
  - Avoid the top-down (in epistemology, science, engineering,…)
  - Bottom-up v. top-down innovation cycles – cf Curtis Carlson

- Design future software ecosystems to benefit from variability and uncertainty rather than trying to engineer it out (as shielding these systems from the random may actually cause harm)
  - For example, design in ***degeneracy*** -- i.e., "ability of structurally different elements of a system to perform the same function". In other words, design in partial functional overlap of elements capable of non-rigid, flexible and versatile functionality. This allows for evolution *plus* redundancy. Contrast m:n *redundancy* (i.e., we do just the opposite).

1 No pun intended

Q&A

Thanks!